Updated Mar 31, 2015

Wizardry IV Return of Wizardry

Messages found in ASCII.KRN

1: **

2: Insert Diskette "A" containing the

3: Save Game you wish to move into

4: Insert Diskette "A" onto which

5: the Save Game is to be moved into

6: Disk Drive 1, then Press ~

7: Move which Save Game (1-8, ~ exits)

8: Move to what Save Game (1-8, ~ exits)

9: Insert your original Diskette "A" into

10: That is not a Diskette "A" - Press ~

12: ^ - Press ~

13: Place SCENARIO Diskette "A"

14: in drive 1, then Press ~

20: ~ For More Wizardry

21: Your Position:

22: Facing ^

23: North.

24: East.

25: South.

26: West.

27: ! East " North # Down.

31: Y

32: N

60: Now loading program segments into

61: extra memory. This improves the

62: performance of the program.

63:

64: You may be asked to insert other

65: Diskettes as the load progresses.

90: As mighty Achilles fell to Paris'

91: arrow, so does Lord Hawkwind fall to

92: the stab of a lowly Dink! Long will

93: Skara Brae be mourning his passing!

100: (Y/N) ?

101: Press ~

102: ^ Got An Item.

103: Will You Wade

104: YN

105: ^ traded an Item!

106: Not Without Payment!

107: Will You Search

108: What Is Your Answer ?

109: Wrong!

110: Correct!

111: You need more Gold

112: Thank You!

113: The Fee is

114: Will You Pay

115: Y

116: N

120: The Pentagram Glows...

121: ...and the Gate Opens!

122: Now you may S)ummon your Monsters!

Some messages had ASCII control characters in them. Before entering them here, I removed the control characters:

14070 to 14073

14082 to 14083

14094 to 14097

14106 to 14109

14118 to 14119

14132 to 14133

14160 to 14161

14170 to 14171

17703

18656

19552

19750

19802

19902

19952

28000 to 28060 are integer values disguised as messages.

See TICKETS and copy protection descriptions.

Other messages from 30000+

The tilde character (~) is displayed as a special "Enter" character similar to a left arrow.

Sometimes a caret is at the beginning of a line. Possibly a concatination character.

Many messages use the caret (^) as a placeholder to insert additional text.

Pound symbol (#), ampersand (&) and percent (%) are similar.

Some messages use the "at" symbol (@) to indicate additional processing.

Some messages use the dollar sign ($) to indicate the response to a riddle.

123: Select Monster Groups by letter (A-L)
124: or Move to N~ext Page or P)rev Page
125: NPS
126: Call Forth 3 Groups of Monsters!
127: Call Forth 2 more Groups!
128: Call Forth 1 more Group!
129: Summon your Monsters!
170: Here at the top of the Ziggurat is a
171: decrepit Temple, crumbling with age
172: and neglect. Inside the Temple is
173: a plain altar. Carved into the top
174: of the altar are three depressions.
177: As you place the stone, a bolt of
178: lightning descends and destroys it!
179: Hmm...the stone somehow gave offense
180: All of the depressions are empty.
181: One of the depressions contains
182: an Item that you have offered.
183: Two of the depressions contain
184: Items that you have offered.
185: You may make an offering of either
186: G)old, an I)tem, or you may l~eave
187: GILYN
188: How about half your Gold (Y/N) ?
189: The ^ vanishes before it lands
190: upon the altar!
191: Gold
192: Item
193: What Item will you offer ?
194: ~ to not make an offering
195: The Item nestles in a depression.
196: It is as if it has become part of
197: the altar.
200: You teleported into rock!
201: You materialized in mid-air above
202: the mouth of the volcano and fell
203: to your doom into molten lava!!
204: You appeared in the Castle moat
205: and have probably drowned!
206: You bounced back to where you were!!
207: Teleport:
208: East = 0 North = 0 Up = 0
209: E)AST/W)EST/N)ORTH/S)OUTH/U)P/D)OWN/T)ELEPORT/Q)UIT
210: Summoning The Guardians!
211: Laying Out Patrol Areas!
212: Setting Up The Traps!
215: As you place the stone, a bolt of
216: lightning descends and destroys you!
217: Gods are not as foolish as you are!!
220: Which Item will you Use ?
221: ~ to not Use an Item
222:
223:
224:
225: Welcome To:
226: The Return Of Werdna!
227:

228: You are Werdna, the Evil Wizard.
229: Years ago you plotted to rule the
230: world. Alas, while unlocking the
231: Amulet's secrets, you were rudely
232: attacked by a party of do-gooder
233: adventurers! They thought you had
234: been slain, but found they could
235: not destroy your body, so it was
236: entombed deep within the earth.
237: Fanatical Guardians and devious
238: Traps surround your hidden Bier as
239: insurance against your return.
245: (AC=
246: , HP=
247:)
250: Press ~
251: Will you Invoke the Special
252: Power of your ^
253: (Y/N) ?
254: Y
255: N
256: Equip Which (#) ?
257: ~ for none
258: Select ^
259: Weapon
260: Armor
261: Cloak
262: Helm
263: Gauntlets
264: Misc. Item
265: * CURSED *
266: Reordering:
269: Weapon
270: It smooths away life's wrinkles.
271: The Ball emits noxious fumes!
272: The fumes repel the insects!
273: Well, The Pin Is PULLED...
274: Zap! The Item Disintegrates!
275: Ahh!! Fresh air!!
276: The pin is already PULLED!
277: You hop high into the air!
280: Press 1-8 to put
281: an Item into the
282: Black Box.
283: Press A-S to get
284: an Item from the
285: Black Box.
286: Press ~ to close
287: the Black Box.
289: The Item Vanishes!
290: Hee! Hee! Hee!
291: Haa! Haa! Haa!
292: Gotcha, Fool!!
293: Clap! Clap!
294: G'wan! Geddoudahere!
295: You hop very high- over the wall!
296: You feel lighter than air!

297: It glows ethereally!
298: Ahhh!! The Curse on Trebor lifts!
299: The Boots rise up and fly away!
300: The Temple is no longer the ruin
301: that it was when you first found
302: it. Now it is bright and whole.
303: Power radiates. The modest altar
304: is resplendent with Gold and
305: Jewels. You feel the Presence!
306:
307: Suspended above the altar are
308: three Magical Swords. A godlike
309: voice, coming from everywhere
310: at once, intones...
311:
312: "Take ye one of these Swords as
313: a reward for restoring unto Me
314: My sacred Temple!"
315:
320: Which Sword will you take ?
321:
322: The G)reen Sword, the B)lue
323: Sword or the A)mber Sword ?
324:
330: GBA
350: The Temple is no longer the ruin
351: that it was when you first found
352: it. Now it is bright and whole.
353: power radiates. The modest altar
354: is now resplendent with gold and
355: jewels. You feel the Presence!
356:
357: Of the Swords, however, there is
358: no sign.
380: *** KABALAAMMMMOOOO ***
381: and if you've counted, this is 1
382: 6 is way off
383: 3 is too small, and 5 too large
384: 4 is the number, the number is 4
402: # Character Name Class AC Hits Status
410: The Chronicles of Hawkwind
411:
412: "A really exciting story,
413: but the ending hasn't been
414: written yet!"
415:
416: Press ~
420: Slowly you sink into a deep slumber.
421:
422: You are sitting beneath a large tree
423: whose wide canopy shelters you.
424:
425: Suddenly, a large White Rabbit wear-
426: ing a waistcoat comes running by. He
427: consults a large pocket watch and
428: exclaims:
429:

430: "I'm late! I'm late!	
431: For the Life of me,	
432: where is the Opening?"	
433:	
434: The image fades and Reality (?) is	
435: once again rampant...	
436:	
437: Press ~	
440: The Fickle Finger of Fate spins...	
441: ...and points at YOU!!!	
442:	
443: Press ~	
500: Entering	
501: The Return Of Werdna	
502: p)LAY GAME/L)EAVE GAME/M)OVE SAVE GAMES	
503: Play which Save Game (1-8) ?	
504: ~ for a New Game, (ESC) exits	
505: Save to which Save Game (1-8) ?	
506: ~ to not Save, (ESC) exits	
510: Oh no!!! The nightmares and the	
511: torments begin again! Next time	
512: I'll be clever enough to escape!	
513:	
514: Press ~ to begin anew	
520: At last you awaken from endless	
521: nightmares and torments, mad with	
522: lust for revenge!	
523:	
524: You want your Amulet back!	
530: But first you have to regain the	
531: powers which have been drained	
532: away and escape from this cursed	
533: prison tomb. Press ~ to begin.	
550: "Tickets...Tickets, Please!	
551: Enter the Validation Code of this	
552: Mordorcharge Card to continue:	
553: Validation Code >	
554: The number you have entered is not	
555: correct. Press ~ to try again.	
600: ~ to leave	
601: LO	
602: Poison	
603: Mage	
604: Prst	
605: Strength	
606: Gold	
607: Lev	
608: I.Q.	
609: Keys	
610: Age	
611: Piety	
612: Vitality	
613: H.P.	
614: A.C.	
615: Agility	
616: Luck	
617: Status	

618: (Poison)
619: You can't cast that
620: Fizzle!
621: Cast on whom (#) ?
622: Oops
623: Now
624: Ok
625: Oops!
626: Not dead
627: Lost
628: Try kadorto
629: Cast what Spell ?
630: >
631: What ?
632: Done!
633: Use Item (#) ?
634: Powerless
635: Not Equipped
636: Drop Item (#) ?
637: Cursed
638: Equipped
639: Dropped
640: Equip Item (#) ?
649: R)EAD/E)QUIP/D)ROP/S)PELL/U)SE/I)EAVE
652: Camp
700: Poison
701: LO
702: An Encounter
703: Stairs Down
704: Stairs Up
705: Take Them (Y/N) ?
706: Y
707: N
708: It's Dark
709: You are in Rock!
710: There are two buttons
711: here marked ^ and #.
712: Press one (~ to leave)
713: Ouch!
714: A Chute!
715: You can't walk through walls!
716: Delay (0-99) >
717: C)amp S)tatus T)ime O)ff A-W-D K
718:
719: Light
720: Protect
721: Identify
722: WK8AL4DR 6XB2OSTCIQ
723: A Pit!
724: Quit This Expedition (Y/N) ?
725: Y
726: N
727: ^ has Died!
728: You are Flying!
729: You levitate over the Pit!
730: You fell down a whole story!
731: You fell down two stories!

	732: Kablam! Land Mine!
	733: You levitate over the Minefield!
	800: Rummaging around in the pile
	801: of Dead, you find...
	802: Press 1-8 to take an Item
	803: ~ when finished stripping bodies
	804: (You can carry ^ more Items)
	805: You can't carry any more Items
	806: (You can carry only 1 more Item)
	900: Press ~
	901: Spells =
	902: Mage
	903: Priest
	904: M)AGE SPELL BOOKS/P)RIEST SPELL BOOKS/I)EAVE
	905: Mage Spells
	906: Priest Spells
	907: Press ~ to leave
	908: Hunting you down!
	909: In the Castle Morgue.
	910: In the
	911: North-
	912: South-
	913: East
	914: West
	915: of this level.
	916: Locate someone
	917: Find whom ? >
	918: ^ is...
	919: Not on this Scenario Diskette!
	920: Identify what
	921: Item (#) ?
	922: ~ to leave
	923: You Identified It!
	924: You don't have a clue!
	1000: A friendly group of
	1001: F)ight or L)eave
	1002: F
	1003: L
	1005: Poison
	1006: LO
	1007: With his dying breath the Herald
	1008: blows upon his long trumpet!
	1010: ~ to leave
	1011: Use Item on person (#) ?
	1012: Use Item on which enemy (#) ?
	1013: Use which Item (#) ?
	1014: ~ to leave
	1015: Spell Points exhausted
	1016: You don't know that Spell
	1017: Cast Spell on person (#) ?
	1018: Cast Spell on which enemy (#) ?
	1019: Spell Name ? >
	1020: Gotcha this time, you Do-Gooder!
	1021: Ambush! Protect me, my Monsters!
	1022: f)IGHT/
	1023: D)ISPELL/
	1024: S)PELL/

	1025: U)SE ITEM/p)ARRY/R)UN						
	1026: UPR						
	1027: 'S OPTIONS						
	1028: Dispell which enemy (#) ?						
	1029: DISPEL						
	1030: Fight which enemy (#) ?						
	1031: FIGHT						
	1032: PARRY						
	1033: SPELL						
	1034: U-ITEM						
	1035: Press ~ to Fight, or						
	1036: T)ake back orders						
	1037: T						
	1038: STEAL						
	1039: AC=						
	1040: You are about to battle						
	1041: *WERDNA*						
	1042: You						
	1043: You						
	1044: ^ are &						
	1045: ^ are Killed!						
	1046: ^ are Decapitated!						
	1047: ^ are not &						
	1048: ^ try to cast a ? but cannot speak!						
	1049: ^ try to cast a ? but nothing happens!						
	1050: ^ cast a ?						
	1051: ^ ? % and miss						
	1052: ^ ? % and hit once for & damage						
	1053: ^ ? % and hit # times for & damage						
	1054: ^ take # damage						
	1055: ^ are completely healed						
	1056: ^ are cured						
	1057: ^ are not cured						
	1060: You are too weak						
	1063: The gods do not hear you!						
	1064: Which boon (#) do you desire ?						
	1065: Cure party						
	1066: Silence enemies						
	1067: Augment magic						
	1068: Teleport enemies						
	1069: Heal party						
	1070: Protect party						
	1071: Raise the dead!						
	1075: No One Expects						
	1076: The Spanish Inquisition!						
	1080: Cast which Spell ?						
	1081: or press ~ to not cast						
	1100: ^ is Drained one level!						
	1101: ^ is Drained # levels!						
	1102: ^ is &						
	1103: Poisoned						
	1104: Paralyzed						
	1105: Stoned						
	1106: Decapitated						
	1107: ^ ? % and misses						
	1108: ^ ? % and hits once for & damage						
	1109: ^ ? % and hits # times for & damage						

	1110: ^ is Killed!						
	1111: ^ is Decapitated!						
	1112: ^ calls for help... but no help comes						
	1113: ^ calls for help... one Monster appears						
	1114: ^ runs away						
	1115: ^ attempts to DISPELL and fails						
	1116: ^ dispells one Monster						
	1117: ^ dispells # Monsters						
	1118: ^ resists the Spell						
	1119: ^ is Killed!						
	1120: ^ takes # damage						
	1121: ^ is &						
	1122: ^ is not &						
	1123: # Hit Points are restored to ^						
	1124: ^ is completely Healed						
	1125: Held						
	1126: Silenced						
	1127: The Spell does not work						
	1128: SLAIN						
	1129: SLEPT						
	1130: Nothing Happens						
	1131: ^ is cured						
	1132: ^ is not cured						
	1133: ^ tries to cast a ? but cannot speak!						
	1134: ^ tries to cast a ? but nothing happens!						
	1135: ^ breathes on the party!						
	1136: ^ casts a ?						
	1137: which is neutralized!						
	1138: ^ Steals an Item and runs away!						
	1139: ^ attempts to Steal an Item but fails						
	1140: # Hit Point is restored to ^						
	1150: bites at						
	1151: claws at						
	1152: tries to gnaw						
	1153: attempts to chew						
	1154: tries to nibble on						
	1155: tries to stab						
	1156: thrusts violently at						
	1157: charges at						
	1158: attempts to slice						
	1159: leaps at						
	1160: attack						
	1290: ^						
	1291: ^						
	1292: ^						
	1293: ^						
	1294: ^						
	1295: ^						
	1300: # ^						
	1301: # ^						
	1302: # ^						
	1303: # ^						
	1304: # ^						
	1305: # ^						
	1310: ^						
	1311: ^						
	1312: ^						

	1313: ^							
	1314: ^							
	1315: ^							
	1320: ^							
	1321: ^							
	1322: ^							
	1323: ^							
	1324: ^							
	1325: ^							
	1330: an ^							
	1331: a ^							
	1332: an ^							
	1333: a ^							
	1334: an ^							
	1335: a ^							
	1340: ^							
	1341: ^							
	1342: ^							
	1343: ^							
	1344: ^							
	1345: ^							
	1350: ^							
	1351: ^							
	1352: ^							
	1353: ^							
	1354: ^							
	1355: ^							
	1360: ^							
	1361: ^							
	1362: ^							
	1363: ^							
	1364: ^							
	1365: ^							
	1370: an ^							
	1371: a ^							
	1372: an ^							
	1373: a ^							
	1374: an ^							
	1375: a ^							
	1380: the ^							
	1381: the ^							
	1382: the ^							
	1383: the ^							
	1384: the ^							
	1385: the ^							
	1390: an ^							
	1391: a ^							
	1392: an ^							
	1393: a ^							
	1394: an ^							
	1395: a ^							
	1400: an ^							
	1401: a ^							
	1402: an ^							
	1403: a ^							
	1404: an ^							
	1405: a ^							

	1410: ^							
	1411: ^							
	1412: ^							
	1413: ^							
	1414: ^							
	1415: ^							
	1420: an ^							
	1421: a ^							
	1422: an ^							
	1423: a ^							
	1424: an ^							
	1425: a ^							
	1430: an ^							
	1431: a ^							
	1432: an ^							
	1433: a ^							
	1434: an ^							
	1435: a ^							
	1440: ^							
	1441: ^							
	1442: ^							
	1443: ^							
	1444: ^							
	1445: ^							
	1450: ^							
	1451: ^							
	1452: ^							
	1453: ^							
	1454: ^							
	1455: ^							
	1460: an ^							
	1461: a ^							
	1462: an ^							
	1463: a ^							
	1464: an ^							
	1465: a ^							
	1470: ^							
	1471: ^							
	1472: ^							
	1473: ^							
	1474: ^							
	1475: ^							
	1480: ^							
	1481: ^							
	1482: ^							
	1483: ^							
	1484: ^							
	1485: ^							
	1490: ^							
	1491: ^							
	1492: ^							
	1493: ^							
	1494: ^							
	1495: ^							
	1500: an ^							
	1501: a ^							
	1502: an ^							

1503: a ^
1504: an ^
1505: a ^
1510: an ^
1511: a ^
1512: an ^
1513: a ^
1514: an ^
1515: a ^
1700: Wizardry is (c)1979-1987 by
1701: Andrew Greenberg, Inc., and
1702: Robert Woodhead, Inc.
1703:
1704: NEC 9801 Pascal is (c)1985 by
1705: the Foretune co., ltd.
1706:
1707: NOTE: if you give characters kanji
1708: names, they will look like tofu on
1709: machines without the kanji rom.
1890: You fall into water and drown!
1900: Congratulations on completing one of
1901: the endings of The Return Of Werdna!
1902:
1903: We would like to find out how well
1904: you did! On the following page will
1905: be displayed a set of code numbers
1906: that measure your performance in the
1907: game. Please call Sir-Tech at the
1908: phone number listed in the document-
1909: ^ation to report these numbers.
1910:
1911: If you do, Sir-Tech will send you a
1912: completion certificate, and you will
1913: be able to obtains some very special
1914: ^mementos of your Victory!
1950: Your three code numbers are:
1951: Press ~ when you have recorded them
2000: Make Scenario Diskette:
2001: Do you have 1) or 2) Disk Drives ?
2002: (Press (ESC) to exit)
2003: 12
2004: Insert Master Diskette into Drive ^,
2005: Insert blank Diskette into Drive ^,
2006: Reading Master Diskette...
2007: Writing to blank Diskette...
2008: Formatting blank Diskette...
2009: Read Error!
2010: Write Error!
2011: "Make" has failed - press ~
2012: Scenario Diskette made - press ~
2013: The Diskette you inserted has Data
2014: on it. If you use it, the Data
2015: will be erased! Use it (Y/N) ?
2016: YN
2017: then press ~
2019: You cannot play without making a
2020: Scenario Diskette!

2021: S)tart Game
2022: M)ake Scenario Diskette
2023: Starting Game:
2024: Select Language:
2025: 1) English
2026: 2) Kana
2027: 3) Kanji
2028: That is not a Master Diskette-Press ~
2029: That is not a Blank Diskette-Press ~
2030: Format error!
2031: # of ^ cycles completed
2032: Insert Scenario Diskette "A"- Press ~
2040: The Return of WERDNA
2041: The Fourth Wizardry Scenario
2042:
2043: Andrew Greenberg,
2044: Robert Woodhead &
2045: Roe R. Adams,III.
2050: Making a Scenario Diskette
2051:
2052: Before you play the game, you MUST
2053: make copies of each of your Master
2054: Diskettes. You can use this util-
2055: ity or any standard copy program.
2056:
2057: Make a copy of Master Diskette "A"
2058: first, then "B", etc.
2059:
2060: At the end of each copy process,
2061: you will be asked to insert YOUR
2062: Scenario Diskette "A", then be re-
2063: turned to the title page. Choose
2064: M)ake Scenario again to copy the
2065: next diskette. Press ~ to begin.
2100: Are you really seeking what is below
2101: life itself? Beware, your choice is
2102: IRREVOCABLE. (A REAL SERIOUS hint!)
2103: ---Make a Backup Save Game Diskette
2104: ^BEFORE attempting this!!!
2105: @Do you really want to complete your
2106: Incantation (Y/N) ?
2110: YN
2120: You have found the Total Oblivion
2121: you have for so long been search-
2122: ing. All traces of your existence
2123: have been obliterated!
2124:
2125: Press ~ to Rest Forever!
2500: "W...E...R...D...N...A..."
2501: "WHERE...ARE...YOU...???"
2502: "I...COME...FOR...YOU..."
2503: "MY...REVENGE...IS...AT...HAND..."
2504: "I...ALMOST...HAVE...YOU..."
2505: "NOW...YOU...ARE...MINE..."
2506: The Ghost of Trebor appears
2507: through the wall. His touch
2508: is chilling unto Death!

	2509: You fell off the Ziggurat!						
	2510: There is a massive explosion,						
	2511: compared to which a TILTOWAIT						
	2512: is a mere HALITO spell!						
	2550: You have found the Bloodstone						
	2600: You have found the Turq.						
	2650: You have found the Amber Dragon						
	5000: NO SPELL						
	5001: *HALITO						
	5002: MOGREF						
	5003: KATINO						
	5004: DUMAPIC						
	5005: *DILTO						
	5006: SOPIC						
	5007: *MAHALITO						
	5008: MOLITO						
	5009: *MORLIS						
	5010: DALTO						
	5011: LAHALITO						
	5012: *MAMORLIS						
	5013: MAKANITO						
	5014: MADALTO						
	5015: *LAKANITO						
	5016: ZILWAN						
	5017: MASOPIC						
	5018: HAMAN						
	5019: *MALOR						
	5020: MAHAMAN						
	5021: TILTOWAIT						
	5022: *KALKI						
	5023: DIOS						
	5024: BADIOS						
	5025: MILWA						
	5026: PORFIC						
	5027: *MATU						
	5028: CALFO						
	5029: MANIFO						
	5030: MONTINO						
	5031: *LOMILWA						
	5032: DIALKO						
	5033: LATUMAPIC						
	5034: BAMATU						
	5035: *DIAL						
	5036: BADIAL						
	5037: LATUMOFIS						
	5038: MAPORFIC						
	5039: *DIALMA						
	5040: BADIALMA						
	5041: LITOKAN						
	5042: KANDI						
	5043: DI						
	5044: BADI						
	5045: *LORTO						
	5046: MADI						
	5047: MABADI						
	5048: LOKTOFEIT						
	5049: *MALIKTO						

	5050: KADORTO						
	9999:						
	10000: Human						
	10001: Elf						
	10002: Dwarf						
	10003: Gnome						
	10004: Hobbit						
	10010: Good						
	10011: Neutral						
	10012: Evil						
	10020: Strength						
	10021: I.Q.						
	10022: Piety						
	10023: Vitality						
	10024: Agility						
	10025: Luck						
	10030: Fighter						
	10031: Mage						
	10032: Priest						
	10033: Thief						
	10034: Bishop						
	10035: Samurai						
	10036: Lord						
	10037: Ninja						
	10040: OK						
	10041: Afraid						
	10042: Asleep						
	10043: Paralyzed						
	10044: Petrified						
	10045: Dead						
	10046: Ashes						
	10047: Lost						
	12000: You have trapped						
	12001: the Wandering Oracle Of Mron!						
	12002:						
	12003: "Al'right, it's a fair cop....						
	12004:						
	12005: Oh.... you just want advice?						
	12006: That's quite different, Mate!						
	12007:						
	12008: Did you bring a Griffin? No, huh,						
	12009: well, I do my best Augury with						
	12010: the entrails of Griffins! Hmm..."						
	12011: @"This will be quite difficult,						
	12012: but for only 2,500 Gold Pieces,						
	12013: perhaps I can go into a trance						
	12014: ^and commune with the gods."						
	12015: No Guarantees, and No Refunds!						
	12050: Pay A) Cash, B) Charge, C) Leave						
	12099: ABC						
	12100: "Be Seeing You!"						
	12101:						
	12102: The Oracle vanishes into thin air!						
	12150: "It is not a wise idea to short-						
	12151: ^change an Oracle, Buddy!"						
	12200: "You don't have a Charge Card"						
	12250: "Hey, this Card is on the Hotlist!						

12251: ^Why, it's Captain Thorin's Card!"
12252: @In a flash of light, Karl Maudlin
12253: steps from a cable car intoning...
12254: "What will you do? What will you
12255: do? I'll tell you what you will
12256: ^do! You won't leave home!"
12298: "I'm sorry, but the number you have
12299: dialed is no longer in service"
12300: THE EGRESS WILL SET YOU FREE!
12301:
12302: YOUR FUTURE IS BLACK;
12303: YOU FEEL BOXED IN!
12304: READ THE ILIAD LATELY?
12305:
12306: "CHOMP, CHOMP...EH, WHAT'S EAST, DOC?"
12307:
12308: SECRETS ABOUND ALL AROUND YOU!
12309: PSST! HAVE YOU MET GLUM YET?
12310: LIVE THE QABALAH!
12311:
12312: THE ANSWER IS CARVED IN STONE.
12313: IT IS RIGHT BEFORE YOUR NOSE!
12314: THE TEMPLE HOLDS AN ANCIENT SECRET.
12315:
12316: HOP HIGH TO ENTER.
12317:
12318: RABBITS ARE SACRED TO THE
12319: THE DREAMPAINTER.
12320: SEEK THE DREAMPAINTER'S SOUL.
12321:
12322: EVERYONE HAS A WEAKNESS!
12323: WHAT IS HIS???
12324: TAKE A STEP TO THE LEFT,
12325: AND A HOP TO THE RIGHT!
12326: GONE TROLLING!
12327:
12328: BEWARE THE GIFTS OF LORD MAYA!
12329:
12330: GET A HANDLE ON THE FORBIDDEN FRUIT!
12331:
12332: ROCKS, MULTILAYERED ROCKS
12333:
12334: HOMER WILL SHOW YOU THE WAY.
12335:
12336: YOU TOO CAN BE SAVED! REPENT YE
12337: SINNER! WASH AWAY THY SINS! REPENT!
12338: DOWN INTO THE BOWELS OF THE EARTH.
12339:
12340: PASSWORD IS YOUR ANCIENT BATTLECRY.
12341:
12342: TO SOAR THE HEIGHTS, YOU MUST
12343: FIRST PLUNGE THE FIERY DEPTHS.
12344: CAREER OPPORTUNITY...EXPERIENCED
12345: MUEZZIN NEEDED...GREAT FRINGES!
12346: CLOUDY TONIGHT,
12347: CHANCE OF RAIN TOMORROW.
12348: CURSED WITH TOO MANY BLESSINGS?

12349: FIND RELIEF IN A BOTTLE!
12350: YOUR FUTURE IS GREY AND HAZY!
12351:
12352: IT'S ALL GREEK TO YOU!
12353: (WELL, MAYBE A LITTLE LATIN)
12354: A TISKET, A TASKET,
12355: TREBOR'S RUMP IS IN A BASKET!!!
12356: FIND THE PATHS OF THE TRUE WAY!
12357:
12358: "SNIFF! SNIFF! WHEW, WHAT A STENCH!"
12359:
12360: LOOK TO THE LYCH-GATE!
12361:
12362: SEEK AMONGST THE HISTORICAL WRITINGS
12363: OF TREBOR'S FOES FOR THE PASSWORD.
12364: OUT TO LUNCH,
12365: CATCH US NEXT TIME!
12366: THE MINUET IN THE MINARET
12367: GOES HAND IN GLOVE!
12368: LOOK AMIDST "THE ROOTS OF THE WORLD"
12369:
12370: "YOU HAVE FORGOTTEN SOMETHING!"
12371:
12372: "NO, NO, IT'S TOO HORRIBLE TO VIEW!
12373: YOU POOR WIZARD, YET MORE SUFFERING.
12374: EVER CONSIDER A DIFFERENT LINE
12375: OF WORK?
12376: THERE IS HOPE IN YOUR FUTURE!!!
12377: NOPE, IT FLICKERED OUT ALREADY.
12378: THE CENOTAPH HIDES THE SECRET WAY.
12379:
12380: FOR TRAVELING THROUGH NOTHING,
12381: SEEK THE NYIN.
12382: HAVE YOU FORGOTTEN SOMETHING?
12383:
13000: Little Old Man
13001: Little Old Men
13002: Dink
13003: A Dink
13004: Fluffy Thing
13005: Fluffy Things
13006: Fuzzball
13007: Fuzzballs
13008: Small Object
13009: Small Objects
13010: Creeping Coin
13011: Creeping Coins
13012: Slime
13013: Slimes
13014: Bubbly Slime
13015: Bubbly Slimes
13016: Small Humanoid
13017: Small Humanoids
13018: Orc
13019: Orcs
13020: Man In Robes
13021: Men In Robes

	13022: Lvl 1 Mage						
	13023: Lvl 1 Mages						
	13024: Gas Cloud						
	13025: Gas Clouds						
	13026: Gas Cloud						
	13027: Gas Clouds						
	13028: Skeleton						
	13029: Skeletons						
	13030: Skeleton						
	13031: Skeletons						
	13032: Corsair						
	13033: Corsairs						
	13034: Garian Raider						
	13035: Garian Raiders						
	13036: Priest						
	13037: Priests						
	13038: Lvl 1 Priest						
	13039: Lvl 1 Priests						
	13040: Weird Humanoid						
	13041: Weird Humanoids						
	13042: Zombie						
	13043: Zombies						
	13044: Small Humanoid						
	13045: Small Humanoids						
	13046: Kobold						
	13047: Kobolds						
	13048: Slime						
	13049: Slimes						
	13050: Creeping Crud						
	13051: Creeping Cruds						
	13052: Strange Plant						
	13053: Strange Plants						
	13054: Crawling Kelp						
	13055: Crawling Kelps						
	13056: Gaunt Figure						
	13057: Gaunt Figures						
	13058: Mummy						
	13059: Mummies						
	13060: Woman In Robes						
	13061: Women In Robes						
	13062: Witch						
	13063: Witches						
	13064: Unseen Entity						
	13065: Unseen Entities						
	13066: Poltergeist						
	13067: Poltergeists						
	13068: Insect						
	13069: Swarm						
	13070: No-See-Um						
	13071: No-See-Um Swarm						
	13072: Scruffy Man						
	13073: Scruffy Men						
	13074: Rogue						
	13075: Rogues						
	13076: Shadowy Figure						
	13077: Shadowy Figures						
	13078: Asher						

	13079: Ashers						
	13080: Large Snake						
	13081: Large Snakes						
	13082: Anaconda						
	13083: Anacondas						
	13084: Shadowy Figure						
	13085: Shadowy Figures						
	13086: Duster						
	13087: Dusters						
	13088: Insect						
	13089: Insects						
	13090: Huge Spider						
	13091: Huge Spiders						
	13092: Priest						
	13093: Priests						
	13094: Lvl 3 Priest						
	13095: Lvl 3 Priests						
	13096: Weird Humanoid						
	13097: Weird Humanoids						
	13098: Rotting Corpse						
	13099: Rotting Corpses						
	13100: Fly						
	13101: Flies						
	13102: Dragon Fly						
	13103: Dragon Flies						
	13104: Unseen Entity						
	13105: Unseen Entity						
	13106: Spirit						
	13107: Spirits						
	13108: Strange Bird						
	13109: Strange Birds						
	13110: Harpy						
	13111: Harpies						
	13112: Bugbear						
	13113: Bugbears						
	13114: Bugbear						
	13115: Bugbears						
	13116: Wererat						
	13117: Wererats						
	13118: Wererat						
	13119: Wererats						
	13120: Man In Armor						
	13121: Men In Armor						
	13122: Ronin						
	13123: Ronins						
	13124: Strange Animal						
	13125: Strange Animals						
	13126: Gaze Hound						
	13127: Gaze Hounds						
	13128: Unseen Entity						
	13129: Unseen Entities						
	13130: Banshee						
	13131: Banshees						
	13132: Unseen Entity						
	13133: Unseen Entity						
	13134: Shade						
	13135: Shades						

	13136: Priest						
	13137: Priests						
	13138: Lvl 5 Priest						
	13139: Lvl 5 Priests						
	13140: Man In Leather						
	13141: Men In Leather						
	13142: Looter						
	13143: Looters						
	13144: Animal						
	13145: Animals						
	13146: Blink Dog						
	13147: Blink Dogs						
	13148: Scruffy Man						
	13149: Scruffy Men						
	13150: Bushwacker						
	13151: Bushwackers						
	13152: Giant Serpent						
	13153: Giant Serpents						
	13154: Moat Monster						
	13155: Moat Monsters						
	13156: Strange Plant						
	13157: Strange Plants						
	13158: Strangler Vine						
	13159: Strangler Vines						
	13160: Amphibian						
	13161: Amphibians						
	13162: Giant Toad						
	13163: Giant Toads						
	13164: Rabbit						
	13165: Rabbits						
	13166: Vorpal Bunny						
	13167: Vorpal Bunnies						
	13168: Slimy Thing						
	13169: Slimy Things						
	13170: Giant Slug						
	13171: Giant Slugs						
	13172: Goblin						
	13173: Goblins						
	13174: Goblin Shaman						
	13175: Goblin Shamans						
	13176: Goblin						
	13177: Goblins						
	13178: Goblin						
	13179: Goblins						
	13180: Strange Bird						
	13181: Strange Birds						
	13182: Cockatrice						
	13183: Cockatrices						
	13184: Ogre						
	13185: Ogres						
	13186: Ogre						
	13187: Ogres						
	13188: Priest						
	13189: Priest						
	13190: Priestess						
	13191: Priestesses						
	13192: Kimonoed Man						

	13193: Kimonoed Men						
	13194: Lvl 3 Samurai						
	13195: Lvl 3 Samurai						
	13196: Unseen Entity						
	13197: Unseen Entities						
	13198: Grave Mist						
	13199: Grave Mists						
	13200: Corsair						
	13201: Corsairs						
	13202: High Corsair						
	13203: High Corsairs						
	13204: Man In Armor						
	13205: Man In Armor						
	13206: Minor Daimyo						
	13207: Minor Daimyos						
	13208: Unseen Entity						
	13209: Unseen Entities						
	13210: Lifestealer						
	13211: Lifestealers						
	13212: Gaunt Figure						
	13213: Gaunt Figures						
	13214: Nightstalker						
	13215: Nightstalkers						
	13216: Unseen Entity						
	13217: Unseen Entities						
	13218: Wight						
	13219: Wights						
	13220: Man In Black						
	13221: Men In Black						
	13222: Master Ninja						
	13223: Master Ninjas						
	13224: Priest						
	13225: Priests						
	13226: Bishop						
	13227: Bishops						
	13228: Wolf						
	13229: Wolves						
	13230: Werewolf						
	13231: Werewolves						
	13232: Goblin						
	13233: Goblins						
	13234: Hobgoblin						
	13235: Hobgoblins						
	13236: Strange Animal						
	13237: Strange Animals						
	13238: Centaur						
	13239: Centaurs						
	13240: Gargoyle						
	13241: Gargoyles						
	13242: Gargoyle						
	13243: Gargoyles						
	13244: Unseen Entity						
	13245: Unseen Entities						
	13246: Ghast						
	13247: Ghasts						
	13248: Dragon						
	13249: Dragons						

	13250: Komodo Dragon						
	13251: Komodo Dragons						
	13252: Animal						
	13253: Animals						
	13254: Hellhound						
	13255: Hellhounds						
	13256: Robed Man						
	13257: Robed Men						
	13258: Priest Of Fung						
	13259: Priests Of Fung						
	13260: Monk						
	13261: Monks						
	13262: Master/Dragons						
	13263: Masters/Dragons						
	13264: Shadowy Figure						
	13265: Shadowy Figures						
	13266: Seraph						
	13267: Seraphim						
	13268: Strange Animal						
	13269: Strange Animals						
	13270: Weretiger						
	13271: Weretigers						
	13272: Insect						
	13273: Insects						
	13274: Boring Beetle						
	13275: Boring Beetles						
	13276: Animal						
	13277: Animals						
	13278: Displacer Beast						
	13279: D'placer Beasts						
	13280: Cave Dweller						
	13281: Cave Dwellers						
	13282: Corrosive Slime						
	13283: Corr. Slimes						
	13284: Dragon						
	13285: Dragons						
	13286: Gas Dragon						
	13287: Gas Dragons						
	13288: Skull						
	13289: Skulls						
	13290: Scryll						
	13291: Scrylls						
	13292: Mottled Figure						
	13293: Mottled Figures						
	13294: Carrier						
	13295: Carriers						
	13296: Man In Armor						
	13297: Men In Armor						
	13298: Myrmidon						
	13299: Myrmidons						
	13300: Strange Animal						
	13301: Strange Animals						
	13302: Gorgon						
	13303: Gorgons						
	13304: Man In Black						
	13305: Men In Black						
	13306: Lvl 6 Ninja						

	13307: Lvl 6 Ninjas						
	13308: Shadowy Figure						
	13309: Shadowy Figures						
	13310: Dark Rider						
	13311: Dark Riders						
	13312: Shadowy Figure						
	13313: Shadowy Figures						
	13314: Doppelganger						
	13315: Doppelgangers						
	13316: Giant Insect						
	13317: Giant Insects						
	13318: Giant Mantis						
	13319: Giant Mantises						
	13320: Glowing Sphere						
	13321: Glowing Spheres						
	13322: Evil Eye						
	13323: Evil Eyes						
	13324: Goblin						
	13325: Goblins						
	13326: Goblin Prince						
	13327: Goblin Princes						
	13328: Monk						
	13329: Monks						
	13330: Master/W. Wind						
	13331: Masters/W. Wind						
	13332: Strange Animal						
	13333: Strange Animals						
	13334: Wyvern						
	13335: Wyverns						
	13336: Dragon						
	13337: Dragon						
	13338: Brass Dragon						
	13339: Brass Dragons						
	13340: Shadowy Figure						
	13341: Shadowy Figures						
	13342: Fiend						
	13343: Fiends						
	13344: Unseen Entity						
	13345: Unseen Entities						
	13346: Will O' Wisp						
	13347: Will O' Wisps						
	13348: Man In Armor						
	13349: Men In Armor						
	13350: Berserker						
	13351: Berserkers						
	13352: Strange Animal						
	13353: Strange Animals						
	13354: Chimera						
	13355: Chimeras						
	13356: Slimy Thing						
	13357: Slimy Things						
	13358: Xeno						
	13359: Xenos						
	13360: Strange Animal						
	13361: Strange Animals						
	13362: Bleeb						
	13363: Bleebs						

	13364: Strange Bird						
	13365: Strange Bird						
	13366: Roc						
	13367: Rocs						
	13368: Man In Armor						
	13369: Men In Armor						
	13370: Major Daimyo						
	13371: Major Daimyos						
	13372: Strange Animal						
	13373: Strange Animals						
	13374: Troll						
	13375: Trolls						
	13376: Man In Armor						
	13377: Men In Armor						
	13378: Champ Samurai						
	13379: Champ Samurai						
	13380: Unseen Entity						
	13381: Unseen Entity						
	13382: Vampire						
	13383: Vampires						
	13384: Unseen Entity						
	13385: Unseen Entities						
	13386: Murphy's Ghost						
	13387: Murphy's Ghosts						
	13388: Strange Animal						
	13389: Strange Animals						
	13390: Manticore						
	13391: Manticores						
	13392: Unseen Entity						
	13393: Unseen Beings						
	13394: Lich						
	13395: Liches						
	13396: Giant						
	13397: Giants						
	13398: Frost Giant						
	13399: Frost Giants						
	13400: Giant						
	13401: Giant						
	13402: Fire Giant						
	13403: Fire Giants						
	13404: Man In Robes						
	13405: Men In Robes						
	13406: Hatamoto						
	13407: Hatamotos						
	13408: Monk						
	13409: Monks						
	13410: Master/Summer						
	13411: Masters/Summer						
	13412: Large Snake						
	13413: Large Snakes						
	13414: Hydra						
	13415: Hydrae						
	13416: Demonic Figure						
	13417: Demonic Figure						
	13418: Succubus						
	13419: Succubi						
	13420: Dragon						

	13421: Dragons						
	13422: Firedrake						
	13423: Firedrakes						
	13424: Dragon						
	13425: Dragons						
	13426: Dragon Zombie						
	13427: Dragon Zombies						
	13428: Giant						
	13429: Giants						
	13430: Cyclops						
	13431: Cyclopes						
	13432: Demonic Figure						
	13433: Demonic Figures						
	13434: Greater Demon						
	13435: Greater Demons						
	13436: Giant						
	13437: Giants						
	13438: Poison Giant						
	13439: Poison Giants						
	13440: Dragon						
	13441: Dragons						
	13442: Gold Dragon						
	13443: Gold Dragons						
	13444: Unseen Being						
	13445: Unseen Beings						
	13446: Maelific						
	13447: Maelifics						
	13448: Unseen Entity						
	13449: Unseen Entitys						
	13450: Vampire Lord						
	13451: Vampire Lords						
	13452: Man In Black						
	13453: Men In Black						
	13454: High Master						
	13455: High Masters						
	13456: Demon						
	13457: Demons						
	13458: Lycurgus						
	13459: Lycurgi						
	13460: Dragon						
	13461: Dragons						
	13462: Black Dragon						
	13463: Black Dragons						
	13464: Cave Dweller						
	13465: Cave Dwellers						
	13466: Foaming Mold						
	13467: Foaming Molds						
	13468: Golem						
	13469: Golems						
	13470: Iron Golem						
	13471: Iron Golems						
	13472: Strange Animal						
	13473: Strange Animals						
	13474: Flack						
	13475: Fleck						
	13476: Demon						
	13477: Demons						

	13478: Entelechy Fuff						
	13479: A Demon Lord						
	14002: A STONE						
	14003: BLOODSTONE						
	14004: A STONE						
	14005: LANDER'S TURQ.						
	14006: A STONE						
	14007: AMBER DRAGON						
	14008: A JEWELED FRUIT						
	14009: HHG AUNTY OCK						
	14010: MOLTING LEATHER						
	14011: WINGED BOOTS						
	14012: A FEATHER						
	14013: DREAMPAINTER KA						
	14014: A GREEN SWORD						
	14015: EAST WIND SWORD						
	14016: A BLUE SWORD						
	14017: WEST WIND SWORD						
	14018: A GOLDEN SWORD						
	14019: DRAGON'S CLAW						
	14020: THE ORANGE ROD						
	14021: HOPALONG CARROT						
	14022: OIL OF OLE'						
	14023: CLEANSING OIL						
	14024: A FORKED STICK						
	14025: WITCHING ROD						
	14026: A WHITE SPHERE						
	14027: AROMATIC BALL						
	14028: THE NYIN						
	14029: VOID TRANSDUCER						
	14030: CLEAR LIGHT						
	14031: KRIS						
	14032: A KEY ON CHAIN						
	14033: INN KEY						
	14034: GLASS SCULPTURE						
	14035: CRYSTAL ROSE						
	14036: A DARK GLOB						
	14037: DAB						
	14038: A SILK CLOTH						
	14039: PENNONCEAUX						
	14040: HAT WITH VISOR						
	14041: MAINTENANCE CAP						
	14042: SWORD						
	14043: LONG SWORD						
	14044: SWORD						
	14045: SHORT SWORD						
	14046: KNOBBED STICK						
	14047: ANOINTED MACE						
	14048: STICK W/CHAIN						
	14049: ANOINTED FLAIL						
	14050: STICK						
	14051: STAFF						
	14052: DAGGER						
	14053: DAGGER						
	14054: SHIELD						
	14055: SMALL SHIELD						
	14056: SHIELD						

	14057: LARGE SHIELD						
	14058: CLOTHING						
	14059: ROBES						
	14060: ARMOR						
	14061: LEATHER ARMOR						
	14062: ARMOR						
	14063: CHAIN MAIL						
	14064: ARMOR						
	14065: BREAST PLATE						
	14066: ARMOR						
	14067: PLATE MAIL						
	14068: HELM						
	14069: HELM						
	14070:						
	14071:						
	14072:						
	14073:						
	14074: SWORD						
	14075: LONG SWORD+1						
	14076: SWORD						
	14077: SHORT SWORD+1						
	14078: KNOBBED STICK						
	14079: MACE+1						
	14080: STAFF						
	14081: STAFF						
	14082:						
	14083:						
	14084: ARMOR						
	14085: LEATHER+1						
	14086: ARMOR						
	14087: CHAIN MAIL+1						
	14088: ARMOR						
	14089: PLATE MAIL+1						
	14090: SHIELD						
	14091: SHIELD+1						
	14092: HOLY RELIQUARY						
	14093: ST. K.A.'S FOOT						
	14094:						
	14095:						
	14096:						
	14097:						
	14098: STICK						
	14099: STAFF+2						
	14100: SWORD						
	14101: DRAGON SLAYER						
	14102: HELM						
	14103: HELM+1						
	14104: AMULET						
	14105: JEWELED AMULET						
	14106:						
	14107:						
	14108:						
	14109:						
	14110: SWORD						
	14111: LONG SWORD+2						
	14112: A CAPE						
	14113: GOOD HOPE CAPE						

	14114: A FURRED CONE						
	14115: MAGICIAN'S HAT						
	14116: BEANIE						
	14117: NOVICE'S CAP						
	14118:						
	14119:						
	14120: GLOVES						
	14121: COPPER GLOVES						
	14122: CONICAL HAT						
	14123: INITIATE TURBAN						
	14124: 3-SIDED CLOTH						
	14125: WIZARD SKULLCAP						
	14126: ARMOR						
	14127: PLATE MAIL+2						
	14128: SHIELD						
	14129: SHIELD+2						
	14130: A CHARGE CARD						
	14131: MORDORCHARGE						
	14132:						
	14133:						
	14134: RING						
	14135: RING						
	14136: SWORD						
	14137: WERE SLAYER						
	14138: SWORD						
	14139: MAGE MASHER						
	14140: KNOBBED STICK						
	14141: MACE						
	14142: STAFF						
	14143: STAFF						
	14144: SWORD						
	14145: BLADE CUSINART'						
	14146: AMULET						
	14147: AMULET						
	14148: ROD						
	14149: ROD						
	14150: CAPE						
	14151: CAPE						
	14152: **USE ME** CAPE						
	14153: CAPE						
	14154: CAPE						
	14155: CAPE						
	14156: AMULET						
	14157: AMULET						
	14158: DIADEM						
	14159: DIADEM						
	14160:						
	14161:						
	14162: DAGGER						
	14163: DAGGER+2						
	14164: DAGGER						
	14165: DAGGER						
	14166: ROBE						
	14167: LICH'S ROBES						
	14168: WHITE CAP						
	14169: SKULL'S CAP						
	14170:						

	14171:						
	14172: GAUNTLETS						
	14173: SILVER GLOVES						
	14174: A YELLOW CARD						
	14175: GETOUT						
	14176: A STONE						
	14177: GOLDEN PYRITE						
	14178: BREATH OF LIFE						
	14179: OXYGEN MASK						
	14180: IRON BOUND BOOK						
	14181: CHRONICLES OF H						
	14182: ARMOR						
	14183: LORD'S GARB						
	14184: WEAPON						
	14185: MURAMASA BLADE						
	14186: WEAPON						
	14187: SHURIKEN						
	14188: ARMOR						
	14189: CHAIN						
	14194: RING						
	14195: RING						
	14196: RING						
	14197: RING						
	14198: RING						
	14199: RING						
	14200: A HAIR REMOVER						
	14201: ADEPT BALDNESS						
	14202: TALE OF MADNESS						
	14203: ARABIC DIARY						
	14204: WIRED BONES						
	14205: DEMONIC CHIMES						
	14206: CHARRED TALLOW						
	14207: BLACK CANDLE						
	14208: A WEIGHTY CUBE						
	14209: BLACK BOX						
	14210: HOLY RELIQUARY						
	14211: ST.TREBOR RUMP						
	14212: HOLY RELIQUARY						
	14213: BISH'S TONGUE						
	14214: HOLY RELIQUARY						
	14215: ST. RIMBO DIGIT						
	14216: GWILYM'S ARROW						
	14217: ARROW						
	14218: A GOLD BALL						
	14219: ORB						
	14220: A HORN						
	14221: RALLYING HORN						
	14222: A WAX SEAL						
	14223: SIGNET RING						
	14224: SHINY GAUNTLET						
	14225: MYTHRIL GLOVE						
	14226: HOLY RELIQUARY						
	14227: HOLY LIMP WRIST						
	14228: CLOAK						
	14229: TWILIGHT CLOAK						
	14230: CLOAK						
	14231: SHADOW CLOAK						

14232: DUNCE CAP

14233: CONE

14234: CLOAK

14235: DARKNESS CLOAK

14236: CLOAK

14237: NIGHT CLOAK

14238: CLOAK

14239: ENTROPY CLOAK

15000: With a feeling of triumph, You exclaim

15001: "Free at last! Now beware, Jailers,

15002: for I come to wreak my revenge and

15003: ^reclaim the Amulet!"

15050: Before you, in the secret Inner

15051: Chamber of the Temple, looms a

15052: gargantuan stone statue of the

15053: ^Dreampainter covered in writing.

15100: "So far, so good," You muse. Now for

15101: the first step in Your Master Plan of

15102: World Domination (Mark II). "Hmmmmm,

15103: which way seems best, right or left?"

15200: "I am the Guardian of the Inner Way.

15201: Return thee to thy prison or die, Oh

15202: has-been Wizard. Do not try to dis-

15203: ^turb the lives of honest folk!"

15400: "I am the Guardian of the Middle Way.

15401: Cease thy wanderings, and return to

15402: ^your cell.

15403:

15404: This be your final warning, Evil One!"

15600: "I am the Guardian of the Outer Way.

15601: ^Suffer and die for your Infamy!"

15800: "I am the Guardian of the Pyramid of

15801: Entrapment. Go back to the torment

15802: of your dreams, Werdna. For but one

15803: ^step away lies Death!"

16000: As you kick the thick dust of the

16001: corridor with your sandal, you hear

16002: something go flying. Searching, you

16003: find a green stone streaked with red

16004: ^veins of a mysterious mineral.

16200: Dinner is over. The room is empty.

16201: The small chest on the table yields

16202: a midnight blue stone streaked with

16203: ^twinkling stars!

16400: The Catacombs ---

16401:

16402: --- Where the dead live again!

16550: Aha! A Secret Passage! The shrill

16551: screams of the Dead are muffled

16552: here. The floor bears the marks of

16553: ^the passage of many feet.

16800: Enter and Abandon All Hope!

16801: (A traditional Intourist greeting)

16850: Dante was here!

16900: Bring your Marshmallows?

16950: The heat is getting quite unbearable.

16951: The Glow ahead is intense. What can

16952: ^be causing it?
16953: @A monstrous statue rises up from the
16954: sullenly glowing ground and bars your
16955: path! Flee while there is still time!
17200: It's the Abyss!
17201: May Kadorto have Mercy on your Soul!
17202: Go Back! Go Back!
17300: It's the Abyss. Boy have you really
17301: bought the farm this time. Retreat!
17400: You plummet into Hell, and the flames
17401: lick towards you. Slowly, your fall
17402: begins to check itself as your Boots
17403: of Levitation begin to arrest the
17404: powerful attraction of the infernal
17405: regions! The heat from the flames
17406: becomes more intense, until it is
17407: ^almost unendurable!
17408: @Just as your fall is arrested and you
17409: begin to rise again, you are able to
17410: reach out and pluck the Jeweled Fruit
17411: ^from the Tree of Liquid Fire!
17412:
17413: @Upward you rise, until the shattered
17414: Gates of Hell are once again visible
17415: ^through the flames and smoke!
17416: @From far below, a mocking voice calls
17417: out to you: "You are getting warmer,
17418: but you aren't nearly hot enough for
17419: ^my tastes! HaHaHah!"
17420: @Suddenly, a huge jet of flame washes
17421: up and over you, immolating you in an
17422: instant! There is an instant of pure
17423: ^agony, and then you feel no more!
17700: Read the other sign. It's right
17701: behind you!
17702:
17703:
17800: Welcome to
17801:
17802: The Death of a Thousand Cuts!
17850: Have you forgotten something?
17900: Wow! A beautiful piece of amber!
17901: Embedded within it is the image
17902: of a Dragon with wings outspread!
17950: The explosions have exposed a Golden
17951: ^Vein running through the Cavern!
17952: Fabulous wealth is here for the tak-
17953: ing. You spot a small nugget and
17954: ^hastily grab it!
18100: You stand before the very Gates
18101: ^of Hell itself!
18102:
18103: The flames are so intense that
18104: you must shield your eyes from
18105: ^their infernal brilliance!
18106:
18107: You may offer B)ribes, U)se an

18108: ^Item or (Hint!) R~un away!
18149: BUR
18150: A hideous wail of many tormented souls
18151: screams forth from the Chimes, shaking
18152: ^the very foundations of the Gates!
18200: Each gutteral word you utter from the
18201: Diary takes form and hurls itself
18202: ^upon the Gates like a battering ram!
18250: A brilliant light, like a small nova,
18251: issues forth from the Black Candle,
18252: and melts the weakened Gates. They
18253: slowly slump to the ground to form a
18254: ^pool of glowing metal!
18255:
18256: The Gates of Hell are open!!
18257:
18258: Do you dare to enter (Y/N) ?
18300: You enter and fall into the Abyss!
18301:
18302: **** Aaaaarrrrrrrgggggghhhhhhh ****
18303:
18304: It burns, by the gods, it burns!!!
18305:
18306: @The fires of Hell lick and scorch
18307: your flesh for what seems to be an
18308: eternity. Finally, the torment is
18309: at an end. You begin to sink into
18310: oblivion, grateful to whatever god
18311: still listens to your prayers.
18312: @As your eyes close for the last
18313: time, you see a Jeweled Fruit
18314: hanging from a Tree of Liquid Fire.
18315: then all is black. Perhaps this is
18316: the True Death, the Giver of Peace?
18317: @
18318:
18319:
18320: Wrong Again!
18350: Wait! The fires of Hell are not burn-
18351: ing you! Searching through your gear,
18352: you see that the Dreampainter's Ka
18353: is absorbing the terrific heat of the
18354: conflagration, shielding you.
18355:
18356: @As you continue to fall, you pass the
18357: Tree of Liquid Fire, with its precious
18358: fruit hanging daintily upon a branch.
18359: @
18360: Deeper and deeper into the Abyss you
18361: descend, witnessing sights which make
18362: your own evil atrocities look like the
18363: ^minor peccadilloes.
18364:
18365: @The sights you see become too depraved
18366: even for you! Your mind is twisted by
18367: the demonic visions! It's torture, a
18368: torture worse than the fires of Hell!!

18369:
18370: Slowly, with demented care, the gods
18371: ^dismember your mind!
18400: Wait! The fires of Hell are not burn-
18401: ing you and your fall is slowing down!
18402: searching through your gear, you see
18403: that the Dreampainter's Ka is absorb-
18404: ing the terrific heat and shielding
18405: you, and your Boots of Levitation are
18406: ^checking your descent.
18407:
18408: @Just as your fall is arrested and you
18409: begin to rise again, you are able to
18410: reach out and pluck the Jeweled Fruit
18411: ^from the Tree of Fire!
18412:
18413: Upward you rise, until you pass back
18414: ^through the shattered Gates of Hell!
18415:
18416: @From far below, a mocking voice calls
18417: out to you: "Have you forgotten some-
18418: ^thing? HaHaHaHaHa!"
18419: @Slowly, the Gates of Hell reform
18420: ^from the puddle of slag.
18450: You throw your offering through the
18451: Gates of Hell, only to see the Gold
18452: ^Coins melt and vaporize!
18460: Nothing happens!
18500: You are at ground level, craning your
18501: neck to look upward at a magnificent
18502: Ziggurat. It stretches upward almost
18503: ^as far as the eye can see.
18550: You are on a first level ledge of an
18551: extremely Art-Deco Ziggurat. A close
18552: examination of the faded cartouches
18553: and paintings leads you to the belief
18554: ^that this is the fabled lost
18555:
18556: "Temple of the Dreampainter"
18557:
18558: which was dedicated to one of the
18559: ancient creators of this world, whom
18560: ^men later called gods.
18600: A small sign on the left wall
18601: reads as follows:
18602:
18603: ** PRIEST'S HOLE **
18604: ** For Emergency Use Only! **
18650: This Way To The Scenic Vista!
18651: (Children under the age of 90
18652: must be accompanied by
18653: Parent, Guardian, Priest
18654: or Demon -- NO EXCEPTIONS!)
18655:
18656:
18700: The view from these heights is truly
18701: breathtaking. Concentric rings of the

18702: Temple stretch out below. Looking
18703: outward, you see a small Orange Rod
18704: that drifts gently around the Temple
18705: in the mild breeze, occasionally just
18706: wafting close enough to the Temple
18707: to be tantalizingly out of reach.
18750: This really is a Scenic Vista!!
18751: The ceiling is so close that
18752: you can almost touch it. Look-
18753: ing upwards, you can see stairs
18754: from the top of the Temple that
18755: lead upwards through the roof
18756: ^of the level.
18757:
18758: Don't get too near the edge, it
18759: ^looks like a long fall!
18800: Nothing happens!
18850: It is generally regarded by most Sages
18851: that taking the "Holy Hand Grenade of
18852: Aunty Ock" into Hell is a bad move.
18853: in the past, you have not been muchly
18854: impressed with the opinions of Sages,
18855: but in this case they are DEAD ON!
18856:
18857: * * * * K A B O O O O O M * * * *
18858:
18859: @In an explosion that makes one of your
18860: finest TILTOWAITS look like a wet fire
18861: cracker, the potent Holy Relic makes a
18862: very loud noise. It also wrecks large
18863: areas of the Abyss (blowing it to Hell
18864: ^and gone, naturally!)
18865:
18866: @Oh yes, just in case it has not dawned
18867: ^upon you---Back to your Bier you go.
18900: The Gates of Hell slowly reassemble
18901: ^themselves from the pool of slag.
18902:
18903: From far away, from far below, you
18904: hear a chorus of demonic laughter!
18950: A Big Blimp is hovering here. Along
18951: its side is rippling this message:
18952:
18953: @Once in a Millennium sale...Lych-gates
18954: 1/3 off...Never know when you will be
18955: needing one! Find yours at Boltac's...
18956: Once in a Millennium sale...Lych-gates
18957: ^1/3 off.....
19000: Enter the Realm of the
19001: Whirling Dervish
19050: What just happened here?
19100: While swooping around the temple, you
19101: are able to pluck the Orange Rod out
19102: ^of the sky.
19150: Oops! You can't carry any more Items!
19200: The Item was placed in the Black Box!
19250: You can't carry any more Items, and

19251: The Black Box is full. Too bad!
19300: You have obtained ^.
19350: Sigh! There you were, with a chance
19351: to get a pretty nasty artifact, and
19352: you didn't have any space to carry
19353: it with! I thought that you were a
19354: ^Master Wizardry Adventurer!
19400: Let me get this straight. After all
19401: the rigamarole you had to go through
19402: to get into Hell, you went there and
19403: had no room to pick up the Item you
19404: ^worked so hard to get to? Sigh!
19450: Running around with every Item slot
19451: full is not considered a smart ploy
19452: ^by most Master Adventurers.
19500: Searching slowly in the dusty floor,
19501: you discover an ancient Pentagram
19502: carved into the rock. Being careful
19503: that you do not step inside the
19504: Circle, you raise your arms and
19505: ^speak the Words of Summoning!
19550: This Way To The Egress
19551:
19552:
19600: The cage door slams shut behind you.
19601: Looks like you have found a new way
19602: ^to pass eternity.
19650: Behold - The Maze of Wandering!
19651:
19652:
19653: Watch where you step!
19700: Upward from this level there is
19701: ^no turning back!
19702: @Have you forgotten something?
19750: EGRESS
19800: This Way To The Egress
19801:
19802:
19850: A floating sign hovers here, pulsing
19851: with this message:
19852:
19853: Welcome to The Cosmic Cube!
19854: Spend a lifetime or two having fun!
19855: (No Deposit, No Return)
19856: @Ponder upon the ancient mystery of
19857: The Lady Or The Tigers if you wish
19858: to tread the hidden ways of the Cube.
19859: @Before you are three doors:
19860:
19861: Upon the left door is written:
19862: A tiger is in the right room.
19863:
19864: Upon the center door is written:
19865: A tiger is in this room.
19866:
19867: Upon the right door is written:
19868: A lady is in this room.

19869: @No room is empty, but only one door
19870: tells the truth. Choose wisely, find
19871: the lady, and you will take the first
19872: step upon the Golden Path. Choose
19873: falsely and you will roam The Cosmic
19874: ^Cube for all eternity!
19900: This Way To The Egress
19901:
19902:
19950: This Way To The Egress
19951:
19952:
20000: You stumble over the long dead body
20001: of some unfortunate explorer. His
20002: outstretched hand has scrawled a
20003: ^few words in the dust:
20004:
20005: "Beware---The Cosmic Cube!
20006:
20007: ---The Red S...."
20008: @Around his neck is a strange object
20009: that looks like the Death Mask of a
20010: Demon. Gingerly, searching for the
20011: traps you have come to expect, you
20012: ^remove the object.
20013:
20014: In the distance, the silent screams
20015: ^of the Butterflies draw nearer!
20100: The Creatures of Light
20101: and Darkness await you!
20150: The door to the Inn is locked.
20151: without the Large Brass Key
20152: ^you cannot gain admittance.
20200: "Oh, I say, this is a Meeting of
20201: the Order of the Laurel!"
20202: @Inside a lovely gazebo amidst the
20203: flowers of the Inner Court sit the
20204: Masters and Mistresses of the
20205: ^Laurel, all Peers of the Realm:
20206: @Baroness Kathryn Goodwyn, Sigismund
20207: Vasa Care, Johannes von Nurenstein,
20208: Gabreilla Maddelena Pisano, Arwen
20209: Evaine Merch Gwynth, and Salaamall-
20210: ^ah the Corpulent.
20250: "Thou hast done nothing to advance
20251: the Arts in the Realm. The Guild
20252: of Limners especially needs thy
20253: aid. Do not return until thou hast
20254: ^achieved some skill or craft!"
20300: "Oh, thou hast the Daub of Puce.
20301:
20302: The Guild of Limners will be most
20303: happy! They have been searching
20304: for the ancient recipe for that
20305: exact Royal shade for many years
20306: so that they might touch up the
20307: old painting of Trebor. No other

20308: shade of deep red would do for
20309: the nose, for sure! Advance now
20310: and accept the Accolade of the
20311: ^Laurels!"
20350: "Hi there, this is a meeting of
20351: the Order of the Tyger's Cubs!!"
20352: @"I am Cormac Kyle of Skara Brae.
20353: These are my friends & companions:
20354: Patri ibn Cariadoc, Krisha Von
20355: Halstern, Alison Von Halstern
20356: Mitchell of Clan Mitchell, and
20357: ^Luke Maximillian. Come on in!"
20400: "We have lost our new Page School
20401: ^ banner. Have you seen it?"
20402: "We would really like some help,
20403: @"but do not mess with us, as we
20404: are Royal Pages, I hope you know!"
20450: "Thanks for returning our banner
20451: to us. Here is a ball as a just
20452: reward. Remember, above all else
20453: ^have fun!!!"
20500: The Von Halstern Chivalry:
20501:
20502: Sir Manfred Von Halstern, Sir Magnus
20503: Bloodaxe, Sir Hrymgnar Aeulfson,
20504: Duke Sir Hanno Von Halstern, Master
20505: Feral, and Sir Vykor stand aside for
20506: ^the Noble Werdna!
20550: This is the Captains' Council.
20551:
20552: "We see that you have obtained a
20553: Royal Pardon! You have been an
20554: exceptional adversary, and you
20555: have managed to defeat us all
20556: at one time or another. For a
20557: fee of 1,000,000 GP, (consider
20558: it Weregild) we will bend knee
20559: to you and pledge our Fealty.
20560: After all, everyone likes to
20561: ^back a winner!"
20600: Will you pay
20650: You don't have 1,000,000 GP!
20700: It is said that "the longest part
20701: of any journey is the first step."
20702:
20703: Watch where you step or your trip
20704: may last an eternity!
21000: Daylight! You are actually above the
21001: ground. Ahead of you is the Castle!!
21002: @A sign on the Castle wall reads:
21003: No Flying Allowed Near The Castle.
21004: No Moat Diving From The Ramparts.
21005:
21006: This Means YOU!!!
21050: The Edge of Town
21100: The Training Grounds
21150: Welcome (?) to Boltac's Trading Post

21200: The Adventurer's Inn

21201: 1/2 Star - Mychelyn Guide

21202: Visa, Mordor Charge accepted

21250: Gilgamesh's Tavern

21251: "Potent Potables"

21252: Step in and step up to the bar!

21300: The Temple of Cant

21301: We support the Fundamental Freedom

21302: to worship the God of *OUR* choice.

21350: * * THE GREAT GOD KADORTO * *

21351: All in his presence must

21352: abase themselves!

21400: The tall Donjon rises before you.

21401: It is the Citadel of the Castle.

21450: So this is the fabulous Boltac's.

21451: Look at all the toys. Too bad he

21452: won't sell to the likes of you,

21453: ^even if you paid in Mythril!

21454: @Hmmm... isn't that the legendary

21455: Lych-gate of the Archmage Phred?

21456: It has been rumored capable of

21457: razing Castles when in the hands

21458: of one who is truly Evil. <Grin>

21500: Try to steal the Lych-gate

21550: You have interrupted a meeting of

21551: The Ladies of the Rose.

21552:

21553: Seated around a table sipping tea

21554: are their Graces Khadijah of House

21555: Hakim, Wanda Von Halstern, Diana

21556: Alene, and Kunegunda Henschel Von

21557: Schattenberg, and their Excellen-

21558: cies Ecaterina Amber of Tospenwood

21559: ^and Mara Tudora Kolarova.

21600: "Begone, Oh unchivalrous cur, and

21601: ^take thy stench with thee!"

21650: "Begone, Oh unchivalrous cur. No

21651: matter how sweet thou might seem,

21652: ^we know thee by thy deeds!"

21700: "Tales of thy chivalrous action

21701: towards thy most bitter enemy,

21702: Trebor, have reached us. Yet,

21703: bring not thy foul stench amongst

21704: ^women of quality!"

21750: "Tales of thy chivalrous action

21751: towards thy most bitter enemy,

21752: Trebor, have reached us, and thou

21753: hast also bathed away the stench

21754: ^of thy past."

21755: @"Advance and receive our Token of

21756: Grace. Keep it on thy person

21757: at all times, for it is a Royal

21758: Pardon for thy crimes, and none

21759: shall bother thee whilst thee

21760: ^wear it!"

21800: Herein meets

21801: The Order of the Pelican!

21802: @Master El of the Two Knives is
21803: addressing the Order. Other
21804: Masters and Mistresses here,
21805: all Peers of the Realm, are:
21806: Aravis Katheryn Delclare, Jae-
21807: lla of Armida, Alisoun Maccoul
21808: of Elphane, Steffan ap Cenydd
21809: of Silverwing, and Anne of
21810: ^Hatfield.
21850: "Get out! Thou hast done no
21851: ^service for this Realm!"
21900: Advance and take this Cap, that
21901: all might know of the service ye
21902: have performed for the Realm by
21903: restoring the fabled Temple of
21904: the Dreampainter to its full
21905: ^glory!"
21950: Just as you are about to dive into
21951: the beautifully clear water, the
21952: lifeguard comes running over and
21953: grabs you by the neck and seat.
21954: @As you sail through the air, you
21955: hear, "Begone, thou art much too
21956: soiled and stained to bathe in
21957: The Royal Pool. Do not come back
21958: until thy hast cleansed thyself!"
22000: Halt! No further may thee travel
22001: unless thou hast decided to seek
22002: ^thy destiny in the Amulet!
22050: Is this thy wish
22100: Then thou hast to get past us,
22101: ^for we guard this Portal!
22150: In this study meets the Council of
22151: Barons. Seated in front of the
22152: fireplace are the most powerful
22153: Barons of the Realm: Sir Patri du
22154: Chat Gris, Baroness Elspeth Keyf
22155: of Neddingham, Baron Yosef Alaric,
22156: Baron Eofn ap Erwyn, Baron Leonard
22157: the Younger, and Baron Algernon
22158: ^Hartesmond.
22159: @A Page comes running in with a cup
22160: ^of Mead and a large parchment.
22161:
22162: "This document pledges you to
22163: reinstate our traditional Rights
22164: and Privileges stolen by Trebor!"
22180: Will you sign it
22200: "In return for your pledge, we in
22201: turn give our Oath of Loyalty
22202: and support. Take this Ring of
22203: Good Counsel and wear it as a
22204: ^token of our mutual Fealty!"
22250: "Then to Hell with you and your
22251: infernal arrogance," proclaims
22252: Baroness Elspeth, as she sends
22253: you on your way with a wave of

22254: ^her hand!
22300: Your Maintenance Cap instills in
22301: you the knowledge you need to
22302: ^repair the fountain!
22350: Before you is a small grotto,
22351: filled with sparkling water. A
22352: large number of clawprints can
22353: be seen around the pool, but
22354: there is no creature in sight.
22355: @You have found the Ron Wartow Not-
22356: Yet-Memorial Wading Pool, which is
22357: dedicated to the Elite who jump to
22358: brilliant, logical conclusions, but
22359: alas, are wrong! Ponder this from
22360: "the Wartowliedtod" before bathing:
22361: @"Oh, will he ever return?
22362: ^ Will he ever return?
22363: ^ His fate is still unlearned!"
22400: As you desperately paddle out of
22401: the Pool, your hand falls upon a
22402: Forked Stick that is floating in
22403: the pool, apparently unharmed by
22404: ^the Acid.
22450: Before you is a small Pool of
22451: Shimmering Liquid, surrounded
22452: by a grove of Majestic Oaks.
22550: You have found one of the many
22551: majestic fountains that are
22552: scattered about the Castle. A
22553: ^sign next to this one reads:
22554:
22555: "Fountain Closed For Repairs."
22556: " No Wading At Any Time. "
22557: "Keep Off The Grass As Well!"
22600: You have found one of the many
22601: majestic fountains that are
22602: scattered about the Castle. A
22603: ^sign next to this one reads:
22604:
22605: "Fountain Open For Public Bathing."
22606: " Lifeguard Must Be On Duty! "
22607:
22608: There is no lifeguard in sight...
22650: As you bathe in this Pool, you feel
22651: strangely refreshed. You emerge in
22652: the very pink of good health. Your
22653: Monster allies take one look at the
22654: new YOU, sparkling clean, smelling
22655: of wild flowers and with a blush on
22656: your cheeks, and flee in terror!!!!
22700: Arggggghh!! This Pool is full of
22701: Acid, leached from the Oaks. IT
22702: ^BURNS!
22750: As you bathe in this Pool, you are
22751: enveloped by a feeling of forebod-
22752: ing. Suddenly, the weight of your
22753: past Evil deeds descends upon your

22754: ^stooped shoulders again!
22800: As you bathe yourself in the Pool,
22801: it is as if the Evil of your Soul
22802: ^has been somewhat cleansed!
22850: A pleasant dip, but nothing seems
22851: ^to have happened...
22900: A Hoplite advances upon you. Stops.
22901: Salutes. He yells into your ear in
22902: ^bad Attic Latin...
22903:
22904: "Halt, and give the Password, Sir!"
22950: $TREBOR SUX
23000: Everywhere there is a brilliance that
23001: is intolerable to look upon. From
23002: ^somewhere comes a deep voice...
23003:
23004: "I am Attainment. I am that part of
23005: the body that is above all the rest,
23006: yet beyond mere touch. What am I?"
23050: $CRANIUM
23051: $BRAIN
23100: Feeling like you are floating in a
23101: vast sky, you enter this room of
23102: pure soft blue. The being dressed
23103: in a diaphanous blue kirtle speaks
23104: ^to you...
23105:
23106: "I am Devotion. I am also that part
23107: of the face upon which the Moon doth
23108: ^rise. What am I?"
23150: $LEFT CHEEK
23200: Blood-red crimson is the color of this
23201: room. Nothing is heard, but then the
23202: being dressed in crimson robes writes
23203: ^in fiery letters in the air...
23204:
23205: "I am the Virtue of Silence, and the
23206: Vice of Avarice. I am that part of
23207: the face upon which the sun rises each
23208: ^day. What am I?"
23250: $RIGHT CHEEK
23300: Somber is this room. Everywhere a
23301: deep violet absorbs all. The being
23302: ^here intones...
23303:
23304: "I am Obedience, yet I also beat the
23305: Drum of Bigotry. I hold the shield
23306: ^that protects. What am I?"
23350: $LEFT ARM
23351: $LEFT HAND
23400: Orange are the walls and orange is the
23401: brigandine of the being who greets you
23402: ^in this room...
23403:
23404: "I am Courage, but I am also Cruelty.
23405: I hold the sword, yet I can also hold
23406: ^the whip. What am I?"

23450: $RIGHT ARM
23451: $RIGHT HAND
23500: This entire room seems to be carved
23501: out of a single huge slab of clear
23502: rose-pink quartz. The being in the
23503: room wears armor of the same trans-
23504: ^lucent color...
23505:
23506: "I am Devotion To Great Work, but
23507: as Pride I can ruin all. Within me
23508: beats the great heart, the mighty
23509: ^engine. What am I?"
23550: $BREAST
23551: $CHEST
23600: Amber glow the walls here. Amber too
23601: is the houppelande of the being
23602: ^waiting here...
23603:
23604: "I am Unselfishness, yet I am Lust. I
23605: am the foundation of the body upon
23606: ^which the main rests. What am I?"
23650: $HIPS
23700: Regalness pervades this room. The
23701: walls and pillars are of a violet-
23702: purple. The being here is swathed
23703: ^in the Royal Cloth of Tyre...
23704:
23705: "I am Truthfulness on one hand, and
23706: Falsehood on the other. I symbolize
23707: the parts of the body that are its
23708: ^main support. What am I?"
23750: $LEGS
23800: The walls of this room are all indigo.
23801: Before you stands a being wrapped in a
23802: ^dazzling indigo toga...
23803:
23804: "I am Independence yet I am Idleness.
23805: I symbolize the part that fuels the
23806: ^the body with energy. What am I?"
23850: $STOMACH
23900: You are in a yellow room. Before you
23901: stands a being dressed from head to
23902: ^toe in deepest yellow...
23903:
23904: "I am Discrimination at my best, and
23905: Avarice at my worst. I symbolize the
23906: part of the body upon which all the
23907: ^rest stands. What part am I?"
23950: $FEET
24000: Beyond this doorway lies the Great
24001: Void, the absence of all. Only a
24002: Great Seeker may venture into this
24003: endless emptiness. Unless you are
24004: properly prepared, you will not
24005: survive. Enter if you have the
24006: means to cross the Void, otherwise
24007: be content in the Mysteries that

24008: you have learned so far. Are you
24009: ^ready (Y/N)?
24050: YN
24100: The answer to the Greatest Question
24101: is also the simplest. Upon what
24102: Paths have you trod? Where are you?
24150: $TREE OF LIFE
24200: All the colors of the Universe
24201: radiate off the walls and floor
24202: of this room. The being here is
24203: dressed in a miparti...
24204:
24205: "I am Knowledge. as a Seeker of
24206: Truth, I present you with this
24207: gift. With it, you can cut the
24208: Veils of Illusion. Now go. You
24209: ^have a destiny to fulfill!
24250: Assembled before you are the great
24251: Dukes of the Realm: Duke Akbar ibn
24252: Murad Al-ben Muhammed ibm Hakim,
24253: Duke Siegfried Von Halstern, Duke
24254: Cariadoc of the Bow, Duke Vissevald
24255: Selkriksson, Barak Duke Hasdrubal
24256: ^and Duke Ronald Wilmot.
24300: Duke Akbar, as Senior Duke present,
24301: ^challenges you:
24302:
24303: "We note that thou comest before us
24304: bearing a God-given Sword and the
24305: support of the Peers, Mercenaries
24306: and Barons of the Realm. Thou hast
24307: also lain the tormented spirit of
24308: ^our late mad Overlord to rest."
24309: @"Each of us has been Champion of
24310: the Realm and has won countless
24311: victories upon the Field of Honor.
24312: Art thou willing to prove thy cour-
24313: age and nobility by accepting single
24314: combat from each of us here assemb-
24315: led? Remember, by the rules of
24316: the list, thou can use neither thy
24317: allies nor thy Arts Magical in this
24318: contest--- May God Defend The Right!
24340: Do thee accept our challenge
24350: "By thy willingness to accept the
24351: challenge, in conditions most un-
24352: favorable to thyself, thou hast
24353: proven thy willingness to live
24354: (and die) by the Chivalric Codes!"
24355: @Looking around at his fellow Dukes,
24356: Duke Akbar continues, "because of
24357: all that thou hast become, and for
24358: the many fine deeds thou hast done
24359: for the good of the Realm, we have
24360: decided to offer to thee Rulership
24361: of all of our lands and estates
24362: in the hope that thou wilt govern

24363: justly, and reunite our strife-torn
24364: Dominions. We would have an end to
24365: this constant warring between our
24366: people and the many Monsters that
24367: dwell in the Planes adjoining ours.
24368: @Thou art our best hope to stride the
24369: Worlds. Wilt thee give up thy Quest
24370: for the mystical Amulet and accept
24371: the title of Overlord with all its
24372: power and responsibility? Consider
24373: our offer carefully, for only once
24374: will it be offered! Wilt thee assume
24375: now the Throne and take the Crown?"
24390: Assume the Throne
24400: "Then may the gods support thee in
24401: ^thy Quest. Good luck to thee!"
24450: "Craven coward, thou shalt die like
24451: ^the cur thou art! To arms!!"
24500: The assembled Peers and Commoners
24501: shout their approval!
24502:
24503: "Long live The Overlord Werdna!!"
24504:
24505: "Vivant!! Vivant!! Vivant!!"
24506: @Then a hauntingly familiar voice
24507: ^whispers in your ear,
24508:
24509: "Remember - a favor owed...someday...
24510: Be Seeing You! <Cackle> <Cackle>"
24511: @ ** Future Historical Footnote **
24512:
24513: Under your benign and enlightened
24514: Rulership, the Realm will enter a
24515: ^Golden Age of Peace and Plenty.
24516:
24517: The evil acts of the past will be
24518: forgiven, and your people will
24519: ^surely remember you fondly!
24520:
24521: But even so, as the years pass,
24522: ^you will always wonder...
24523: @
24524:
24525:
24526:
24527:
24528: "Have You Forgotten Something?"
24550: You see an old Witch stirring a
24551: large cauldron:
24552:
24553: "So nice of you to wander by (Heh,
24554: Heh, Heh), my fine fellow. Are you
24555: ^enjoying your travels? <Snicker>"
24556: @"A man with your aspirations
24557: should definitely have some of
24558: my famous 'Blue Blood Special'.
24559: If you have the makings, I'll

24560: ^be glad to brew you a batch!"
24600: "Let's see now what parts
24601: of the recipe you have.."
24650: You need ^
24651: You have ^
24652: Camphor
24653: Rabbit's Fur
24654: Fe s-sub-2
24655: Tannic Acid
24656: Spanish Unguent
24657: a Blender
24700: "My my! You will have to go find
24701: some more ingredients before I
24702: ^can help you."
24703: @"Be Seeing You! <Cackle> <Cackle>"
24750: "Well, you have been an industr-
24751: ious young man! You have all of
24752: ^the ingredients of my recipe!
24753:
24754: "I'll be happy to make my brew,
24755: but naturally, I will require a
24756: token of payment. Not money, of
24757: course...just a favor at a later
24758: ^date. <Cackle>"
24790: Accept the deal
24800: "Bubble, bubble, toil and trouble,
24801: Cauldron boil and cauldron bubble!"
24802:
24803: "Oh, don't mind the theatrics, it
24804: is all part of the union contract.
24805: We have to keep up our image, you
24806: know. Just hold on a minute and I
24807: will get your 'Blue Blood Special'
24808: ^out of the refrigerator..."
24809: @"Be Seeing You! <Cackle> <Cackle>"
24850: Aha! A hidden room jutting out of
24851: the Donjon's side. What a great
24852: view of the town from here. In the
24853: ^room is a wizened old scholar...
24854: @"Greetings, Werdna. I am Master
24855: Bertram, Curator of the Overlord's
24856: Treasure Room. I have an item that
24857: you dearly require. Tell me what
24858: ^it is, and it shall be yours!"
24900: $MYTHRIL GAUNTLET
24901: $MYTHRIL GLOVE
24950: You are truly to be counted
24951: amongst the Wise.
25000: Guarding the doorway into the
25001: Inner Sanctum of the Temple is
25002: Lord Hawkwind of Skara Brae,
25003: Elfin Ninja--the last member of
25004: the Softalk All-Stars. He ob-
25005: serves you with calm disdain.
25006:
25007: "Thou shalt not pass this way,
25008: ^this day or ever!"

	25040: LORD HAWKWIND noticeth it not!					
	25050: LORD HAWKWIND ^!					
	25051: Laughs					
	25052: Chuckles					
	25053: Snickers					
	25054: Reads "Pikestaff"					
	25055: Answers His Mail					
	25056: Files His Nails					
	25057: Sends Out For Sushi					
	25058: Ignores You					
	25059: Pets His Dragons					
	25060: Pays For The Sushi					
	25061: Feeds Sushi To His Dragons					
	25062: Pretends To Take A Nap					
	25063: Really Falls Asleep (From Boredom)					
	25064: Wakes Up					
	25065: Brushes His Teeth					
	25066: Brews Tea					
	25067: Oils His Blade					
	25068: Says His Morning Prayers					
	25069: Petitions To Offer You Up As A Sacrifice					
	25070: Fills Out Sacrificial Order Form In Triplicate					
	25071: Gets Approval For Your Sacrifice					
	25072: Binds And Gags You					
	25073: Puts You On The Sacrificial Altar					
	25074: Anoints You With Sacrificial Oils					
	25075: Sacrifices You To Kadorto					
	25100: Parry					
	25101: Bind					
	25102: Place					
	25103: Anoint					
	25104: Sacrif					
	25150: Kadorto is pleased that the pious					
	25151: Lord Hawkwind has offered up a					
	25152: Sacrifice, but considers you to be					
	25153: a very poor substitute for a Goat!					
	25154: VL					
	25200: Amid the clutter of empty Mead bot-					
	25201: tles and wine flasks pitched up on					
	25202: the roof of Gilgamesh's Tavern, you					
	25203: ^discover a packrat's nest.					
	25240: Rummage in the nest					
	25250: Trash, trinkets, toys, tribbles,					
	25251: Great Uncle Ted, a Black Hole, an					
	25252: old copy of Terror of Tiny Town,					
	25253: ^lots of junk!					
	25254: @Hey, wait a minute! A Black Hole?					
	25255: Hmm, something is pulsating within!					
	25290: Reach in and grab for it					
	25300: Good idea! You could have lost					
	25301: a hand that way!					
	25350: The nest has been disturbed and					
	25351: there is nothing left in it but					
	25352: Great Uncle Ted, who always was					
	25353: ^the last to leave!					
	25400: "Cream! Vanishing Cream! Get your					
	25401: Vanishing Cream here! It's going					

25402: fast! Only a few jars left! Only
25403: 50,000 GP a jar! Truly the buy of
25404: a lifetime - You'll curse the day
25405: ^you didn't get some!"
25440: Buy Cream before it vanishes
25441:
25500: In the corner are scattered some
25501: small White Spheres. They look
25502: ^somewhat like marbles.
25540: Pick up a marble
25541:
25600: "Psst...Hay Meester...You wanna
25601: buy some you know what? Cheap!
25602: ^Only 100 GP!"
25640: Buy some "You Know What"
25641: d
25700: "What do you seek the most,
25701: Oh Pilgrim?"
25750: $AMULET
25800: Relax...Sit back...Close your eyes.
25801: You are trying too hard.
25802:
25803: Let the Energies that surround you
25804: here resonate within your body.
25805: @Reflect upon each of the Oracle's
25806: visions--one holds the answer.
25850: You need to reflect more before you
25851: ^may seek this Path!
26000: Before you looms the giant statue of
26001: the Almighty Kadorto. You are not
26002: ^even as tall as his little toe!
26003:
26004: As you crane your neck backwards and
26005: look up, you see the object of your
26006: long search. The mystical Amulet is
26007: dangling from the closed fist of the
26008: ^statue!
26009: @"At last!" you cry, but your joy is
26010: short-lived. There is no way that you
26011: ^can see to reach the Amulet!
26012: @Fighting back a growing feeling of
26013: desperation, you try everything you
26014: can think of to get to the Amulet.
26015: Every spell you know is useless! You
26016: rummage through your items, trying
26017: each in turn, hoping that one will
26018: ^be of aid!
26050: Alas, nothing seems to work. the
26051: Priests of the Temple laugh at you
26052: openly. Says one, "Oh fool, we of
26053: Kadorto let all make attempts at
26054: possessing the Amulet, so sure are
26055: we that only a God could take it!
26056: ^Now begone! Try again tomorrow!
26100: What luck! Your Holy Limp Wrist
26101: Reliquary casts a DIALKO spell!
26102: With a feeling of triumph, you

26103: watch as the DIALKO takes effect.
26104: The hand of the statue softens and
26105: opens, freeing the Amulet. To your
26106: amazement, the DIALKO spreads out.
26107: @The newly awakened God yawns and
26108: peers down at you with a look of
26109: ^utter contempt.
26110:
26111: "Insect! So you want this pretty
26112: bauble? You have no idea what the
26113: powers of this Amulet are, or what
26114: its real purpose is. Here! Catch
26115: ^it!"
26116: @Kadorto throws the Amulet at you
26117: with a disdainful flip of his huge
26118: ^wrist!
26150: You reach out and catch the Amulet!
26151:
26152: You have it in your hand!
26153:
26154: Wait a moment..
26155:
26156: You have it in your hand!
26157: YOUR BARE HAND!
26158:
26159: * * p o o f* *
26200: You reach out and catch the Amulet!
26201:
26202: You have it in your hand!
26203:
26204: The Mythril Gauntlet protects you
26205: ^from its raging energies!
26206: @"Hmmmm...," intones Kadorto, "Think
26207: you're clever, don't you? Well, to
26208: keep the Amulet, you will have to
26209: defeat me!"
26210:
26211: "Tell me, are you a God?"
26212: @Thinking quickly, you reply, "Yes!",
26213: which seems to be the right answer.
26214:
26215: #"Then die like a God!"
26250: Kadorto looks you over closely, and
26251: then declares:
26252:
26253: "You come before Me armed with mere
26254: toys! Hah! Hah! Hah! Back to your
26255: ^Bier old fool!"
26300: You draw your Green Sword and begin
26301: the battle. Old forgotten powers
26302: awaken and the sword leaps to attack
26303: the pompous Kadorto. The sword par-
26304: ries thunderbolt after thunderbolt
26305: while you bide your time looking for
26306: ^an opening...
26307: @There it is! A quick thrust to the
26308: ankle! Kadorto looks puzzled as a

	26309: thread of green lightning snakes up					
	26310: his legs. In a flash, he becomes a					
	26311: ^towering statue of green stone!					
	26312: @Fascinated beyond all caution, you					
	26313: reach out to feel the statue. The					
	26314: entire statue crumbles to dust when					
	26315: you touch it! The room is filled					
	26316: with choking dust. You try to run					
	26317: outside, but you can't hold your					
	26318: breath long enough. As you inhale					
	26319: the dust, you hear a smug laugh and					
	26320: Kadorto's voice whispers "Gotcha!!"					
	26321: @You feel yourself stretching and					
	26322: growing rapidly. You are very diz-					
	26323: zy... the room is spinning around!!					
	26324: You sit down. Ah, that feels much					
	26325: better! You try to stand. What???					
	26326: You can't move! You can see and					
	26327: ^hear, but you cannot move!					
	26328: @The frightened Priests return in the					
	26329: morning and find that a miracle has					
	26330: happened! A somewhat different Kad-					
	26331: orto, all green, sits on the Throne,					
	26332: ^the Amulet clutched in his fist!					
	26333: @There you sit, playing God, aware					
	26334: of each slow second's passing,					
	26335: hoping that some greedy fool will					
	26336: come and try to steal the Amulet,					
	26337: releasing you. Centuries pass,					
	26338: and you realize that you got what					
	26339: you always wanted, the adulation					
	26340: and worship of many people. How					
	26341: ironic...					
	26342: #But you always have this nagging					
	26343: ^doubt.. You always wonder..					
	26344: #Have You Forgotten Something?					
	26350: You draw your Blue Sword and begin the					
	26351: battle. Ancient forces are pulled in-					
	26352: to the battle, and your sword begins					
	26353: to glow fiercely. Kadorto lunges down					
	26354: at you, but you leap nimbly aside. His					
	26355: blow makes rubble of the marble floor!					
	26356: Faster and faster your spinning blade					
	26357: weaves a deadly pattern in your hands!					
	26358: @You leap high onto the foot of the					
	26359: Throne. Kadorto is just recovering					
	26360: from his attack and is still bent					
	26361: over. On its own volition, the sword					
	26362: licks out and touches Kadorto's chest!					
	26363: ^The sword scores a Critical Hit!					
	26364: @Kadorto utters a strange gurgle, some-					
	26365: what like a laugh, then with a shimmer					
	26366: ^of distorting light, he vanishes!					
	26367: @The Priests enter and proclaim you the					
	26368: new God! "What size and shape would					
	26369: you like, Oh God?", they ask. "My own					
	26370: will be sufficient!", you reply. The					

26371: ^Priests are not overly impressed.
26372: @"How quaint," says one, glancing up at
26373: the empty Throne, "At least Kadorto
26374: knew how to look like a God. Well, we
26375: will do the best with what we have.
26376: Wait until you see the new robes we
26377: will design for you, the Ceremonies
26378: and Processionals! We will take good
26379: care of you, Oh Werdna! Your every
26380: ^wish is our command!"
26381: @The years pass quickly as you settle
26382: into the God business. Using the pow-
26383: er of the Amulet, you raise huge temp-
26384: les, spacious retreats, and luxurious
26385: ^monasteries for your loyal Priests.
26386: @Oh, the people grumble under the bur-
26387: den of their tithing. Perhaps you are
26388: pushing things a little. "But no," a
26389: priest whispers into your ear, "Yours
26390: is the greater glory. The people love
26391: You for it, and You must guide them."
26392: #But yet, even though you are a God,
26393: ^now and then, you wonder...
26394: #Have You Forgotten Something?
26400: You draw your Dragon's Claw. It hums
26401: with anticipation. Kadorto laughs at
26402: ^you, "What will you do with that?"
26403: @Actually, you are not quite sure, but
26404: it has sustained you through many
26405: trials, lending you its strength and
26406: energy. Kadorto sends down a pillar
26407: of flame... and the blade absorbs it!
26408: He throws a bolt of lightning, only to
26409: see the blade cleave it in half!
26410: @You feel filled with energy!! You let
26411: fly with a mighty blow, and your sword
26412: strikes true, slicing deep into the
26413: big toe of Kadorto! You feel his life
26414: force flowing into you through the
26415: sword. As it does, you begin to grow
26416: and he begins to shrink! Finally, you
26417: are the tall God, and he is the puny
26418: mortal. "Thank you, free at last", he
26419: ^croaks as he expires!
26420: @You laugh as the Priests scurry around
26421: removing the remains. Finally, they
26422: all assemble in front of you, abasing
26423: themselves and raising their voices
26424: ^upward in supplication:
26425: @"All pray to you, Oh Great God Werdna.
26426: We rejoice the weak pacifist Kadorto
26427: has been defeated! Take up the sacred
26428: Amulet. Lead us, oh mighty one, into
26429: glorious battle. Make the world trem-
26430: ble at your every step! Let us fill
26431: the altars with sacrifices, and the
26432: ^temple with gold and slaves!"

26433: @The years pass by in a blur of fire,
26434: blood and destruction. Large areas of
26435: the world lie desolate. Your Priests
26436: bloat you with sacrifices and praise.
26437: After this, there are other Planes to
26438: conquer, other Planes whose energy can
26439: feed your lust for power! You have
26440: all your desires, all your dreams ful-
26441: filled! But yet.....every once in a
26442: ^long while, you wonder...
26443: #Have You Forgotten Something?
26450: You draw your Dagger of Clear Light,
26451: and face Kadorto unafraid, for you are
26452: whole and secure in your Knowledge of
26453: The Tree of Life. Kadorto stops his
26454: ^laughing when he sees your blade.
26455: #"No, no, not that!", he cries!
26456: @The Kris blazes forth. In its clear
26457: penetrating light, no lie or illusion
26458: can remain. Kadorto's motions become
26459: jerky, and smoke begins to pour out of
26460: his knees and elbows. His head pops
26461: open, and a singed high Priest climbs
26462: ^out of a concealed control cabin.
26463: @Now you see the real truth! Kadorto is
26464: a fake, an invention of the Priests,
26465: a cunning device to perpetuate their
26466: social position and control over men!
26467: @You laugh at the ridiculous sight of a
26468: high Priest with his robes smoldering.
26469: It feels good to be alive.You take out
26470: the Amulet, and in the light of the
26471: Kris, you see it for what it really
26472: is: a dangerous trap for those unwary,
26473: ^a joke of the Gods.
26474: @For it is neither Good or Evil, but
26475: fashioned out of pure chaos. You vow
26476: to "return it" to its Makers, and you
26477: are sure the Kris of Truth will aid
26478: you. But that can wait for tomorrow.
26479: ^Today is to be enjoyed!
26480: @Outside, into the beautiful sunshine,
26481: you walk. Feeling at last free and
26482: alive. You have returned to the world.
26483: You look back at the temple for a
26484: ^moment, and wonder...
26485: #Have You Forgotten Something?
26486: @You laugh, for you know that you have
26487: not. You are master of your fate, and
26488: the winding paths of the Tree Of Life
26489: illuminate the shape of your destiny!
26490: @* * * Congratulations * * *
26491:
26492: You have returned. You have found the
26493: path of fulfillment! Give us a call at
26494: (315)393-6633 and tell us about it!
26495:

	26496: @The authors of this program would like							
	26497: to congratulate you on your skill. In							
	26498: recognition of your ability, we confer							
	26499: ^upon you the rank of:							
	26500:							
	26501: Wizardry Grandmaster Adventurer							
	26502: #PS: Have You Forgotten Something?							
	26503: #PPS: Don't worry, you haven't.							
	26504: @PPPS: Don't tell anyone else how to							
	26505: win. Let it be as much a challenge							
	26506: ^to them as it was to you!							
	26507: @PPPPS: This really is							
	26508: The End Of The Game!							
	26509: #PPPPPS: Trust Us!							
	26510: #PPPPPPS: Really!							
	26511: #PPPPPPPS: You have done it all!							
	26512: #"Be Seeing You! <grin>"							
	26550: * * * Congratulations * * *							
	26551:							
	26552: You have completed							
	26553: #The Return Of Werdna!							
	26554: @The authors of this program would like							
	26555: to congratulate you on your skill. In							
	26556: recognition of your ability, we confer							
	26557: ^upon you the rank of:							
	26558:							
	26559: Wizardry Master Adventurer							
	26560: @We hope you are satisfied with							
	26561: your fate. If not, then consid-							
	26562: er that anything is possible...							
	26563: #...Even if highly improbable!							
	26564: @It is those "what if" nexuses							
	26565: that change our destinies.							
	26566: Freedom of choice is one of							
	26567: the great gifts of Lord Maya!							
	26568: Suppose you had chosen to							
	26569: take a different Item, or							
	26570: tread yet another Path? How							
	26571: altered would be your fate?							
	26572: Go and find your Dreams!							
	26573: @But remember to only seek							
	26574: what you will be happy to							
	26575: ^find! <grin>							
	26576: #Be Seeing You!							
	28000	2	60	0				
	28001	4	56	4	187	8		
	28002	4	62	4	13	7		
	28003	4	169	4	39	2		
	28004	4	187	4	42	2		
	28005	4	195	4	77	2		
	28006	4	2	5	61	0		
	28007	4	106	5	98	4		
	28008	4	236	5	207	0		
	28009	4	249	5	2	4		
	28010	4	20	6	240	4		
	28011	4	52	6	70	7		
	28012	4	66	6	210	7		

	28013	4	71	6	193	1		
	28014	4	133	6	203	7		
	28015	4	176	6	51	2		
	28016	4	199	6	154	1		
	28017	4	217	6	197	5		
	28018	4	221	6	107	0		
	28019	4	90	8	57	3		
	28020	4	146	8	171	1		
	28021	4	171	8	0	0		
	28022	4	245	8	160	5		
	28023	4	253	8	64	6		
	28024	4	9	9	229	2		
	28025	4	34	9	199	0		
	28026	4	58	9	60	3		
	28027	4	73	9	191	1		
	28028	4	91	9	197	0		
	28029	4	133	9	231	4		
	28030	4	147	9	20	2		
	28031	4	166	9	98	6		
	28032	4	176	9	192	2		
	28033	4	210	10	161	1		
	28034	4	228	10	42	6		
	28035	4	240	10	9	8		
	28036	4	252	10	142	5		
	28037	4	76	11	19	7		
	28038	4	81	11	56	4		
	28039	4	94	11	74	7		
	28040	4	103	11	93	6		
	28041	4	106	11	254	8		
	28042	4	125	11	205	6		
	28043	4	174	11	52	9		
	28044	4	198	11	70	4		
	28045	4	216	11	134	1		
	28046	4	65	12	217	2		
	28047	4	171	12	164	6		
	28048	4	206	12	103	5		
	28049	4	231	12	231	0		
	28050	4	41	13	81	6		
	28051	4	86	13	209	4		
	28052	4	135	13	152	3		
	28053	4	210	13	56	6		
	28054	4	219	13	80	6		
	28055	4	3	14	182	4		
	28056	4	195	14	248	4		
	28057	4	232	14	133	7		
	28058	4	12	15	95	7		
	28059	4	28	15	87	3		
	28060	4	156	15	82	3		
	29000: WERDNA							
	29001: holding the short end of the stick!							
	29002: KADORTO							
	29003: waiting for you!							
	29004: ADAMS							
	29005: grinning sardonically!							
	29006: WOODHEAD							
	29007: taking a well-deserved vacation!							
	29008: GREENBERG							

	29009: counting royalties in the tropics!						
	29010: KILROY						
	29011: (or rather, was) here!						
	29012: WARTOW						
	29013: saying "I knew that!"						
	30000:						
	30001:						
	30002:						
	30010:						
	30050:						
	30051:						
	30052:						
	30053:						
	30054:						
	30100:						
	30101:						
	30102:						
	30103:						
	30104:						
	30105:						
	30110:						
	30111:						
	30120:						
	30121:						
	30122:						
	30123:						
	30124:						
	30125:						
	30126:						
	30150:						
	30151:						
	30152:						
	30153:						
	30154:						
	30155:						
	30156:						
	30157:						
	30158:						
	30159:						
	30160:						
	30161:						
	30162:						
	30200:						
	30201:						
	30202:						
	30220:						
	30221:						
	30222:						
	30223:						
	30224:						
	30225:						
	30250:						
	30251:						
	30252:						
	30260:						
	30261:						
	30262:						

	30263:							
	30264:							
	30265:							
	30266:							
	30267:							
	30268:							
	30269:							
	30270:							
	30271:							
	30272:							
	30273:							
	30274:							
	30300:							
	30301:							
	30302:							
	30303:							
	30304:							
	30305:							
	30306:							
	30307:							
	30308:							
	30309:							
	30310:							
	30311:							
	30312:							
	30313:							
	30350:							
	30351:							
	30352:							
	30353:							
	30354:							
	30355:							
	30356:							
	30400:							
	30401:							
	30402:							
	30403:							
	30451:							
	30452:							
	30453:							
	30454:							
	30455:							
	30456:							
	30457:							
	30480:							
	30481:							
	30482:							
	30483:							
	30490:							
	30491:							
	30500:							
	30501:							
	30502:							
	30503:							
	30504:							
	30505:							
	30506:							

	30507:							
	30550:							
	30551:							
	30552:							
	30600:							
	30601:							
	30602:							
	30603:							
	30604:							
	30605:							
	30606:							
	30650:							
	30651:							
	30652:							
	30653:							
	30654:							
	30655:							
	30656:							
	30657:							
	30658:							
	30659:							
	30660:							
	30700:							
	30701:							
	30702:							
	30750:							
	30751:							
	30752:							
	30753:							
	30851:							
	30852:							
	30860:							
	30861:							
	30862:							
	30863:							
	30864:							
	30950:							
	30951:							
	30952:							
	30953:							
	30954:							

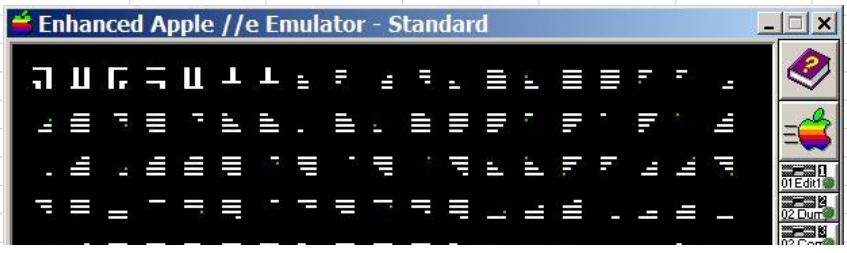
200.CHARSET contents

Chars 0..127

Chars 128..255 are blank (?)

Chars 256..767

Chars 768..1023




02 Com
w01 wiz2
1
2
1
2
H

SYSTEM.RELOC

VIRTUAL		DISK1		DISK2		DISK3		DISK4		DISK5	
0	1	0	1	0	1	0	1	0	1	0	1
2	85	2	85								
86	132			2	48	2	48	2	48	2	48
133	141	86	94								
142	152	95	105	49	59	49	59	49	59	49	59
153	158	106	111								
159	181			60	82	60	82	60	82	60	82
182	200	112	130	83	101	83	101	83	101	83	101
201	208	131	138								
209	224			102	117	102	117	102	117	102	117
225	227	139	141	118	120	118	120	118	120	118	120
228	229	142	143	121	122	121	122	121	122		
230	230	144	144	123	123	123	123	123	123	121	121
231	232	145	146	124	125	124	125	124	125		
233	233	147	147	126	126	126	126	126	126	122	122
234	234	148	148	127	127	127	127	127	127		
235	235	149	149	128	128	128	128				
236	236	150	150			129	129	128	128		
237	237	151	151	129	129	130	130	129	129	123	123
238	238	152	152	130	130						
239	239	153	153	131	131			130	130		
240	240	154	154	132	132					124	124
241	241			133	133			131	131		
242	243			134	135	131	132				
244	244			136	136						
245	245										
246	249	155	158								
250	251										
252	253									125	126
254	265							132	143		
266	269									127	130
270	271										
272	275							144	147		
276	276							148	148	131	131
277	277					133	133			132	132
278	278							149	149		
279	280										
281	282			137	138						
283	285										
286	292			139	145	134	140	150	156	133	139
293	299	159	165	146	152	141	147	157	163	140	146
300	300			153	153						
301	313	166	178	154	166	148	160	164	176	147	159
314	314	179	179								
315	315	180	180							160	160
316	339	181	204	167	190	161	184	177	200	161	184
340	352			191	203						
353	356	205	208	204	207	185	188	201	204	185	188
357	357					189	189	205	205		
358	360							206	208		
361	361					190	190	209	209		
362	362							210	210	189	189
363	367							211	215		
368	368			208	208	191	191	216	216		

	369	369			209	209			217	217		
	370	370			210	210	192	192	218	218		
	371	371	209	209	211	211	193	193	219	219	190	190
	372	380									191	199
	381	381					194	194			200	200
	382	382					195	195				
	383	383					196	196			201	201
	384	385									202	203
	386	387					197	198			204	205
	388	403									206	221
	404	406	210	212	212	214	199	201	220	222	222	224
	407	407			215	215	202	202				
	408	408	213	213	216	216	203	203	223	223	225	225
	409	414	214	219								
	415	415			217	217			224	224	226	226
	416	416			218	218	204	204				
	417	417					205	205				
	418	418			219	219	206	206	225	225	227	227
	419	419			220	220			226	226	228	228
	420	420			221	221						
	421	421			222	222			227	227	229	229
	422	437			223	238						
	438	438			239	239	207	207				
	439	460					208	229				
	461	461			240	240	230	230	228	228	230	230
	462	466			241	245						
	467	467			246	246	231	231				
	468	481					232	245				
	482	482			247	247	246	246				
	483	487			248	252						
	488	488			253	253	247	247				
	489	489					248	248				
	490	490					249	249	229	229		
	491	504							230	243		
	505	507							244	246	231	233
	508	509									234	235
	510	511					250	251			236	237
	512	513							247	248		
	514	523					252	261				
	524	533			254	263						
	534	535									238	239
	536	537							249	250		
	538	539									240	241
	540	541					262	263				
	542	549			264	271	264	271	251	258	242	249
	550	555	220	225	272	277	272	277	259	264	250	255

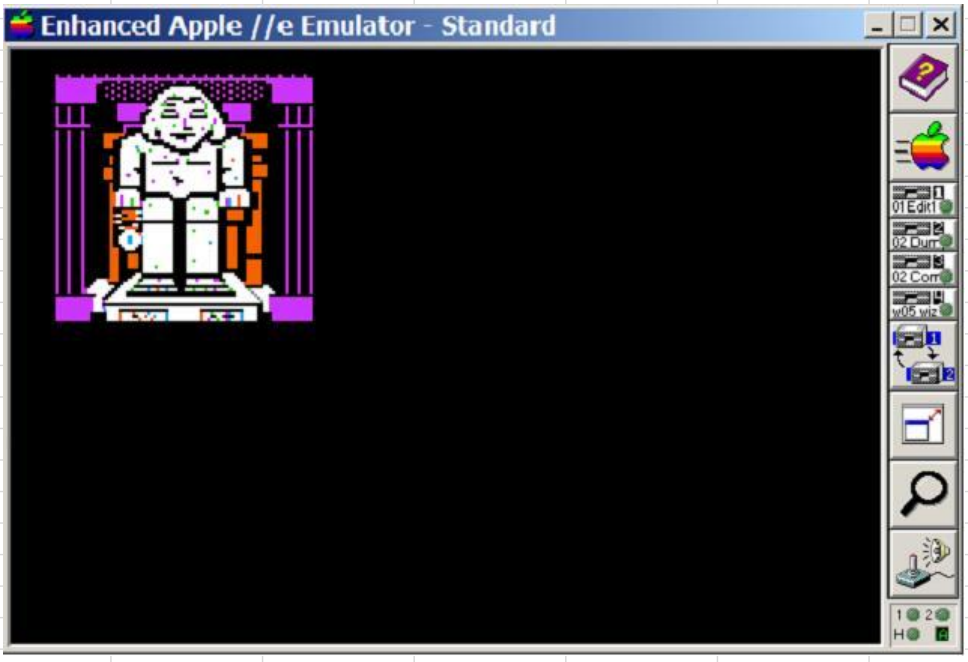
In the first part of 200.MONSTERS are 24 bit mapped pics of the monsters.
There are many more pics in the file(see below). 60 Blocks and 2 "pics" per block with some gaps.

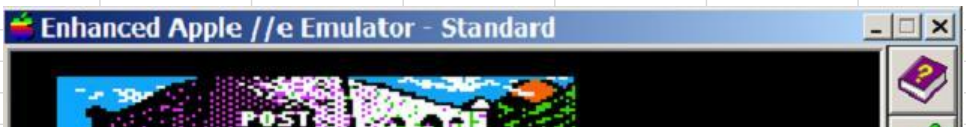
```
MONSTERS: PACKED ARRAY[ 0..23] OF
                PACKED RECORD
                        PICS:   PACKED ARRAY[ 0..4] OF PACKED ARRAY[ 0..5} OF
                                                PACKED ARRAY[ 0..7] OF 0..255;
                        PAD :   PACKED ARRAY[ 0..15] OF 0..255;
                END;
```

24 monsters
5 rows per monster
6 columns per monster
8 bytes per hi-res "box"
Each byte has value 0 to 255.

Each monster occupies 30 hi-res "boxes". 6 across and 5 down. Each box has 8 values representing 8 x 7 dots on the screen.

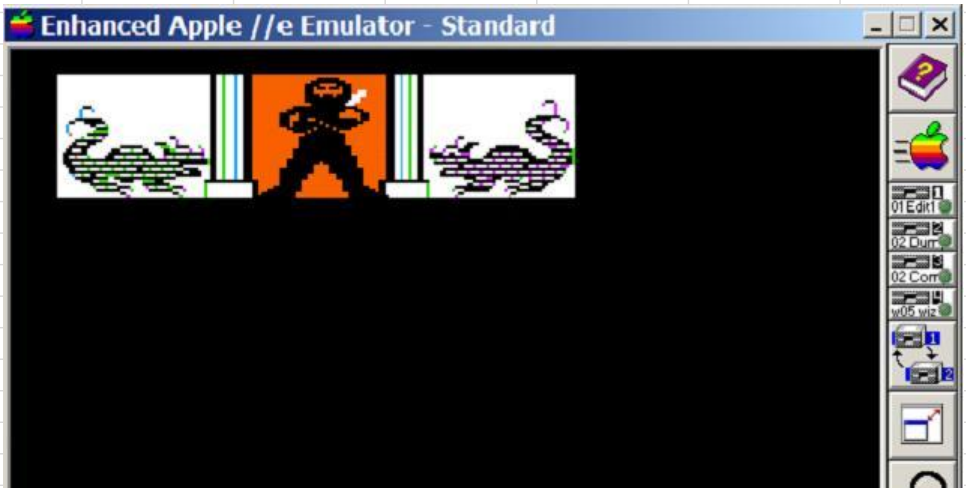

Enhanced Apple //e Emulator - Standard


Enhanced Apple //e Emulator - Standard

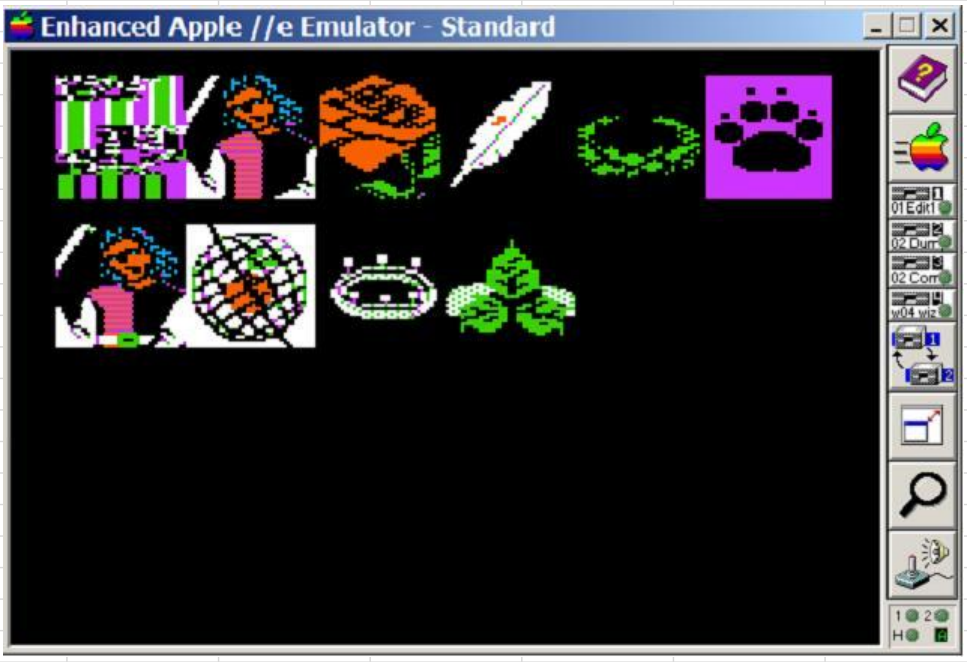

Enhanced Apple //e Emulator - Standard


NO
BILLS

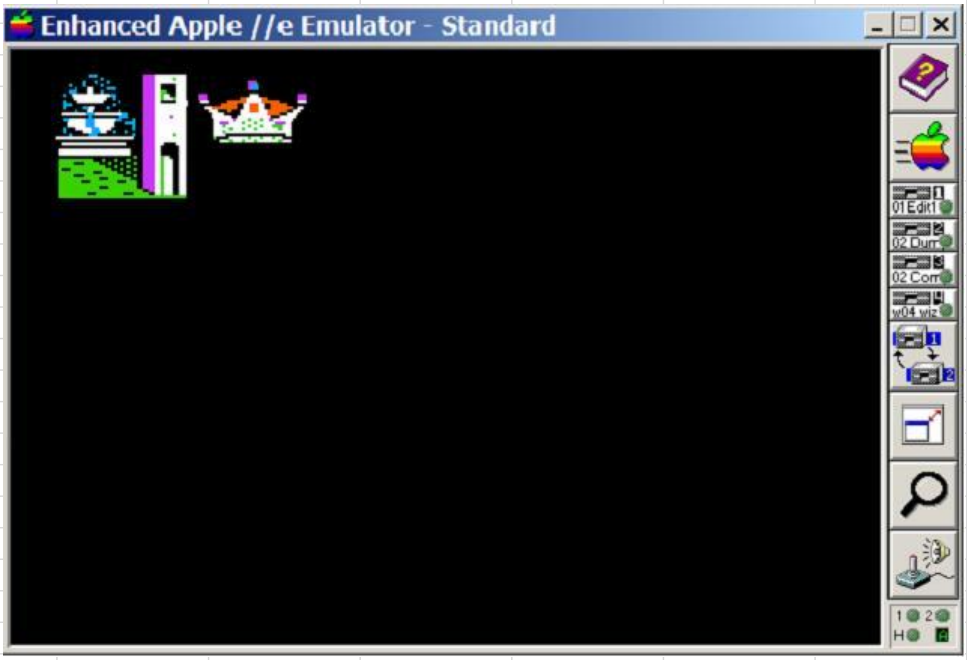

Enhanced Apple //e Emulator - Standard
ROOMS
STABLES
TO THE
ORDERS


Enhanced Apple //e Emulator - Standard

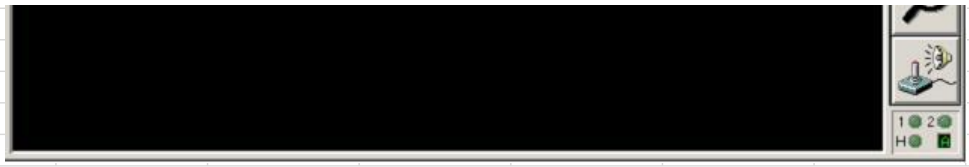

Enhanced Apple //e Emulator - Standard


Enhanced Apple //e Emulator - Standard


Enhanced Apple //e Emulator - Standard


Enhanced Apple //e Emulator - Standard


Enhanced Apple //e Emulator - Standard
CLUB
DEAD


1 2
H

Updated Mar 31, 2015

```
*800LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL
0800-   00          BRK
0801-   EA          NOP             NOP first time, then RTS
0802-   A9 60       LDA   #$60      60 === RTS
0804-   8D 01 08    STA   $0801     Only come thru here once
0807-   A5 2B       LDA   $2B       Slot# * 16
0809-   85 FB       STA   $FB       $FB := Slot # * 16
080B-   4A          LSR
080C-   4A          LSR
080D-   4A          LSR
080E-   4A          LSR
080F-   09 C0       ORA   #$C0
0811-   85 F3       STA   $F3       $F3 := $Cx  (x=Boot slot)
0813-   8D 34 08    STA   $0834

0816-   EE 35 08    INC   $0835     Begin Loop
0819-   AD 35 08    LDA   $0835     Sector #
081C-   C9 0C       CMP   #$0C      Read 11 more sectors yet?  $0900..$13FF
081E-   B0 0F       BCS   $082F     Yes, $082F
0820-   C9 08       CMP   #$08      Convert to PASCAL interleave
0822-   90 02       BCC   $0826
0824-   E9 08       SBC   #$08
0826-   2A          ROL
0827-   85 3D       STA   $3D       Sector # for I/O
0829-   20 32 08    JSR   $0832
082C-   4C 16 08    JMP   $0816

082F-   4C CD 08    JMP   $08CD     JMP after reading 11 more sectors
0832-   4C 5C 00    JMP   $005C     $Cx5C  Firmware READ SECTOR routine.
0835-   00          BRK             Sector #
--------------------------------------------------------------------------------
0836-   0D 53 59    ORA   $5953     "SYSTEM.INTERP"
0839-   53          ???
083A-   54          ???
083B-   45 4D       EOR   $4D
083D-   2E 49 4E    ROL   $4E49
0840-   54          ???
0841-   45 52       EOR   $52
0843-   50 00       BVC   $0845
--------------------------------------------------------------------------------
0845-   50 4C       BVC   $0893     "PLEASE RESTART FROM ANOTHER DISKETTE
0847-   45 41       EOR   $41
0849-   53          ???
084A-   45 20       EOR   $20
084C-   52          ???
084D-   45 53       EOR   $53
084F-   54          ???
0850-   41 52       EOR   ($52,X)
0852-   54          ???
0853-   20 46 52    JSR   $5246
```

```
0856-  4F          ???
0857-  4D 20 41    EOR   $4120
085A-  4E 4F 54    LSR   $544F
085D-  48          PHA
085E-  45 52       EOR   $52
0860-  20 44 49    JSR   $4944
0863-  53          ???
0864-  4B          ???
0865-  45 54       EOR   $54
0867-  54          ???
0868-  45 00       EOR   $00
-------------------------------------------------------------------------------
086A-  43          ???           "CAN NOT RUN THE PROGRAM!"
086B-  41 4E       EOR   ($4E,X)
086D-  20 4E 4F    JSR   $4F4E
0870-  54          ???
0871-  20 52 55    JSR   $5552
0874-  4E 20 54    LSR   $5420
0877-  48          PHA
0878-  45 20       EOR   $20
087A-  50 52       BVC   $08CE
087C-  4F          ???
087D-  47          ???
087E-  52          ???
087F-  41 4D       EOR   ($4D,X)
0881-  21 00       AND   ($00,X)
-------------------------------------------------------------------------------
0883-  57          ???           "WIZARDRY REQUIRES 64 K-BYTES MEMORY"
0884-  49 5A       EOR   #$5A
0886-  41 52       EOR   ($52,X)
0888-  44          ???
0889-  52          ???
088A-  59 20 52    EOR   $5220,Y
088D-  45 51       EOR   $51
088F-  55 49       EOR   $49,X
0891-  52          ???
0892-  45 53       EOR   $53
0894-  20 36 34    JSR   $3436
0897-  20 4B 2D    JSR   $2D4B
089A-  42          ???
089B-  59 54 45    EOR   $4554,Y
089E-  53          ???
089F-  20 4D 45    JSR   $454D
08A2-  4D 4F 52    EOR   $524F
08A5-  59 00
-------------------------------------------------------------------------------
               A0  EOR   $A000,Y
08A8-  00          BRK
-------------------------------------------------------------------------------
08A7-  A0 00       LDY   #$00      "CAN NOT RUN THE PROGRAM!"
08A9-  B9 6A 08    LDA   $086A,Y
```

```
08AC-   F0 08       BEQ   $08B6
08AE-   09 80       ORA   #$80
08B0-   99 88 04    STA   $0488,Y
08B3-   C8          INY
08B4-   D0 F3       BNE   $08A9
08B6-   A0 00       LDY   #$00      "WIZARDRY REQUIRES 64 K-BYTES MEMORY"
08B8-   B9 83 08    LDA   $0883,Y
08BB-   F0 08       BEQ   $08C5
08BD-   09 80       ORA   #$80
08BF-   99 03 05    STA   $0503,Y
08C2-   C8          INY
08C3-   D0 F3       BNE   $08B8
08C5-   A6 2B       LDX   $2B
08C7-   BD 88 C0    LDA   $C088,X   Turn drive off
08CA-   4C CA 08    JMP   $08CA     Infinite loop
--------------------------------------------------------------------------------
                                    After first 12 sectors read
08CD-   A2 00       LDX   #$00
08CF-   AD B3 FB    LDA   $FBB3  Type of APPLE
08D2-   C9 06       CMP   #$06   IIe?
08D4-   D0 14       BNE   $08EA    No, $08EA  (II, II+)
08D6-   E8          INX
08D7-   AD C0 FB    LDA   $FBC0  IIc?
08DA-   F0 0C       BEQ   $08E8    Yes, $08E8 (IIc)
08DC-   C9 EA       CMP   #$EA   IIe vanilla?
08DE-   F0 0A       BEQ   $08EA    Yes, $8EA  (IIe)
08E0-   38          SEC            No, IIe enhanced, IIe Option Card, or newer
08E1-   20 1F FE    JSR   $FE1F  Newer?
08E4-   B0 03       BCS   $08E9    No, $8E9
08E6-   A2 42       LDX   #$42     Yes        (IIGS)
08E8-   E8          INX
08E9-   E8          INX
-------------------------------
                                  X identifies Apple:
                                    0: II, II+
                                    1: IIe
                                    2: IIe enhanced or IIe Option Card
                                    3: IIc
                                   44: IIGS (?)
08EA-   8A          TXA
08EB-   2C 61 C0    BIT   $C061   (Open Apple key?) (Enhanced keyboard?)
08EE-   10 0B       BPL   $08FB     Not pressed, $08FB  (JMP taken APPLEWIN)
08F0-   2C 62 C0    BIT   $C062   (Closed Apple key?)
08F3-   10 06       BPL   $08FB     Not pressed, $08FB
08F5-   A5 FE       LDA   $FE     BOTH Apple Keys pressed? (?)
                                  Small bug(?) $FE not previously init'd (!?)
08F7-   09 80       ORA   #$80    (could also be both BUTTONs on Paddles (?))
08F9-   85 FE       STA   $FE     $FE (high bit) = ???
08FB-   85 FE       STA   $FE     $FE (01 or 02 in AppleWin, +high bit)
                                  **********************************
08FD-   AD 83 C0    LDA   $C083   $D000-$DFFF, Bank 2
```

```
0900-   AD 83 C0    LDA   $C083   Write Enable
                                  ************************************
0903-   A9 DE       LDA   #$DE
0905-   8D 00 D0    STA   $D000
0908-   A2 BC       LDX   #$BC
090A-   8E 00 E0    STX   $E000
090D-   CD 00 D0    CMP   $D000   Check memory
0910-   D0 95       BNE   $08A7   Error, give bad news
0912-   EC 00 E0    CPX   $E000   Check memory
0915-   D0 90       BNE   $08A7   Error, give bad news
                                  ************************************
0917-   AD 82 C0    LDA   $C082   Enable ROM, Write protect RAM
                                  ************************************
091A-   A9 30       LDA   #$30    $CA.CB := $0830   "SYSTEM.INTERP"
091C-   85 CA       STA   $CA
091E-   A9 08       LDA   #$08
0920-   85 CB       STA   $CB
0922-   A9 00       LDA   #$00    $C8.C9 := $0C00   Pascal Directory
0924-   85 C8       STA   $C8
0926-   A9 0C       LDA   #$0C
0928-   85 C9       STA   $C9

092A-   A0 10       LDY   #$10
092C-   B1 C8       LDA   ($C8),Y  Directory File Count
092E-   AA          TAX
092F-   10 0B       BPL   $093C    Advance $C8.C9 by #$1A, and DEX

0931-   A0 05       LDY   #$05     Search directory for "SYSTEM.INTERP"
0933-   C8          INY
0934-   B1 CA       LDA   ($CA),Y
0936-   F0 29       BEQ   $0961      Found "SYSTEM.INTERP", GOTO $0961
0938-   D1 C8       CMP   ($C8),Y
093A-   F0 F7       BEQ   $0933

093C-   18          CLC              Advance $C8.C9 by #$1A, and DEX
093D-   A5 C8       LDA   $C8          #$1A is directory entry length
093F-   69 1A       ADC   #$1A         i.e., advance to next entry in DIR
0941-   85 C8       STA   $C8
0943-   90 02       BCC   $0947
0945-   E6 C9       INC   $C9
0947-   CA          DEX
0948-   10 E7       BPL   $0931    JMP $0931
-------------------------------------------------------------------------------
094A-   A0 00       LDY   #$00     "PLEASE RESTART FROM ANOTHER DISKETTE"
094C-   B9 45 08    LDA   $0845,Y
094F-   F0 08       BEQ   $0959
0951-   09 80       ORA   #$80
0953-   99 82 04    STA   $0482,Y
0956-   C8          INY
0957-   D0 F3       BNE   $094C
0959-   A6 2B       LDX   $2B
```

```
095B-   BD 88 C0    LDA   $C088,X  Motor Off
095E-   4C 5E 09    JMP   $095E    Infinite Loop
-------------------------------------------------------------------------------

  To here after finding "SYSTEM.INTERP" in the Directory

0961-   A0 00       LDY   #$00     First Block of SYSTEM.INTERP
0963-   B1 C8       LDA   ($C8),Y
0965-   AA          TAX            Block (Track/Sector) to read
0966-   18          CLC            $F4.F5 := First Block + #$18 =
0967-   69 18       ADC   #$18       INTERP Last Block
0969-   85 F4       STA   $F4      Hard-coded, 24 blocks = 48 sectors
096B-   C8          INY              for $D000..$FFFF
096C-   B1 C8       LDA   ($C8),Y  $F4.F5 is transferred to $34.35 at
096E-   A8          TAY              start of INTERP
096F-   69 00       ADC   #$00
0971-   85 F5       STA   $F5      $F4.F5 := First Block + #$18 = Last Block
0973-   98          TYA            Test "length" of file SYSTEM.INTERP
0974-   4A          LSR            Too big?
0975-   D0 D3       BNE   $094A       Yes, Error, give bad news
0977-   88          DEY          DEY (???)
0978-   8A          TXA          First Block (T/S) of SYSTEM.INTERP
0979-   6A          ROR
097A-   4A          LSR
097B-   29 FE       AND   #$FE
097D-   85 5A       STA   $5A     Desired track (in 1/2 track inc (?))
097F-   8A          TXA
0980-   29 07       AND   #$07
0982-   0A          ASL
0983-   8D 35 08    STA   $0835   Sector to read
0986-   A2 30       LDX   #$30    Hard-coded ($D000..$FFFF)
0988-   86 47       STX   $47     $47 := #$30  # of sectors to read
098A-   A9 00       LDA   #$00
098C-   8D 78 04    STA   $0478   Current Track

098F-   A9 EF       LDA   #$EF    #EFD8 motor on time
0991-   85 59       STA   $59
0993-   A9 D8       LDA   #$D8
0995-   85 58       STA   $58
0997-   20 5C 0B    JSR   $0B5C   SEEKABS ($5A = Desired Track in 1/2 track inc)
099A-   AD 78 04    LDA   $0478
099D-   4A          LSR
099E-   85 41       STA   $41     Track # (for $C65C)
09A0-   A9 56       LDA   #$56
09A2-   20 A8 FC    JSR   $FCA8   Wait

09A5-   AD 35 08    LDA   $0835   Sector to read
09A8-   C9 08       CMP   #$08    Pascal interleave
09AA-   90 02       BCC   $09AE
09AC-   E9 08       SBC   #$08
09AE-   2A          ROL
```

```
09AF-   85 3D       STA   $3D     Sector # (for $C65C)
09B1-   A6 2B       LDX   $2B
09B3-   20 32 08    JSR   $0832   Read Sector    (to $26.27 for $C65C)
09B6-   C6 47       DEC   $47     Read all sectors?
09B8-   F0 15       BEQ   $09CF     Yes, $09CF
09BA-   EE 35 08    INC   $0835   Increment Sector #
09BD-   AD 35 08    LDA   $0835
09C0-   C9 10       CMP   #$10
09C2-   90 E1       BCC   $09A5   Read another sector
09C4-   A9 00       LDA   #$00
09C6-   8D 35 08    STA   $0835   Start next track at sector 0
09C9-   E6 5A       INC   $5A     Desired Track (in 1/2 track increment)
09CB-   E6 5A       INC   $5A
09CD-   B0 C0       BCS   $098F   Move to new track

09CF-   A6 2B       LDX   $2B
09D1-   BD 88 C0    LDA   $C088,X  Motor Off
--------------------------------------------------------------------------------
  48 sectors starting at beginning of SYSTEM.INTERP read into memory
  $1400..43FF.  Moved to $D000-FFFF.  Last 4 sectors go elsewhere.

09D4-   A9 43       LDA   #$43   $C8.C9 --> $4300
09D6-   85 C9       STA   $C9
09D8-   A9 FF       LDA   #$FF
09DA-   85 CB       STA   $CB    $CA.CB --> $FF00
09DC-   A9 00       LDA   #$00
09DE-   85 C8       STA   $C8
09E0-   85 CA       STA   $CA
                                 ************************************
09E2-   AD 83 C0    LDA   $C083   Second Bank Memory in $D000-$DFFF
09E5-   AD 83 C0    LDA   $C083   Write Enable
                                 ************************************
09E8-   A2 30       LDX   #$30
09EA-   A0 00       LDY   #$00

09EC-   B1 C8       LDA   ($C8),Y  Re-Locate $1400..$43FF (SYSTEM.INTERP)
09EE-   91 CA       STA   ($CA),Y  To:       $D000..$FFFF
09F0-   88          DEY
09F1-   D0 F9       BNE   $09EC

09F3-   C6 C9       DEC   $C9
09F5-   C6 CB       DEC   $CB
09F7-   CA          DEX
09F8-   D0 F2       BNE   $09EC
--------------------------------------------------------------------------------

                                  Still here after relocating code.
09FA-   20 C4 0A    JSR   $0AC4    AUX Memory Bank stuff

09FD-   A6 F3       LDX   $F3      F3 === Boot slot (e.g., $C6)
09FF-   AD 78 04    LDA   $0478    Current Track
```

```
0A02-   9D B8 03    STA   $03B8,X  $47E for Slot 6 boot (Scratchpad for slot 6)
0A05-   A9 00       LDA   #$00
0A07-   85 FA       STA   $FA      FA := 0
0A09-   8D 75 FC    STA   $FC75    FC75 := 0
0A0C-   A5 FB       LDA   $FB      Slot# * 16 (Boot device, e.g., $60)
0A0E-   8D 67 FC    STA   $FC67    FC67 := Boot slot * 16
0A11-   A9 DE       LDA   #$DE
0A13-   85 F6       STA   $F6      F6 := #$DE    $BCDE
0A15-   A9 BC       LDA   #$BC
0A17-   85 F7       STA   $F7      F7 := #$BC
                                   ************************************
0A19-   AD 80 C0    LDA   $C080    Select 2nd Bank, Write Protect, Read RAM
                                   ************************************
0A1C-   4C 00 D0    JMP   $D000    Start executing SYSTEM.INTERP
-------------------------------------------------------------------------------
                                 A1F..AC3 moved to $40..E4
                                   in each AUX z-page (see $0B1C).

                                 See $E9D8 and $0CB0 for JSR $0040.

                                 Y = AUX bank section (03, 04, or 01)
                                     03 = $D000..DFFF, Bank 1
                                     04 = $D000..FFFF, Bank 2
                                     01 = $0200..BFFF

                                 X = 00 (Read AUX) or 01 (Write AUX)
                                     $E985, $E988, $E994, or $E9A0

                                 $0CAA.0CAB
                                    DEST (Read) / SRC  (Write)
                                 $0CAE.0CAF
                                    SRC  (Read) / DEST (Write)

                                 $0CAC.0CAD
                                    Length in bytes
[$0040]
0A1F-   EA          NOP
0A20-   C0 03       CPY   #$03
0A22-   90 10       BCC   $0A34    Y = 01?  Yes, $0A34 ($0055)
0A24-   D0 08       BNE   $0A2E        $0200..$BFFF ($5F blocks)

                                 Y = 03
                                     $C000..$CFFF (08 blocks)
0A26-   AD 8B C0    LDA   $C08B    Read/Write RAM, Bank 1
0A29-   AD 8B C0    LDA   $C08B    Read/Write RAM, Bank 1
0A2C-   B0 06       BCS   $0A34    Branch always to 0A34

                                 Y = 04
                                     $D000..$FFFF ($18 blocks)
0A2E-   AD 83 C0    LDA   $C083    Read/Write RAM, Bank 2
0A31-   AD 83 C0    LDA   $C083    Read/Write RAM, Bank 2
```

```
[$0055]
0A34-   8A          TXA           Y := X == Read/Write flag
0A35-   A8          TAY

                                  Calc $22.23 (end addr for xfer)
0A36-   AE AD 0C    LDX   $0CAD     X := $0CAD (length (high byte))
0A39-   AD AA 0C    LDA   $0CAA     $22 := $0CAA
0A3C-   85 22       STA   $22
0A3E-   18          CLC
0A3F-   8A          TXA
0A40-   6D AB 0C    ADC   $0CAB
0A43-   85 23       STA   $23       $23 := $0CAD + $0CAB

                                  $22.23 + $CAC is DEST (Read) / SRC (Write)

                                  Calc $20.21 (end addr for xfer)
0A45-   AD AE 0C    LDA   $0CAE     $20 := $0CAE
0A48-   85 20       STA   $20
0A4A-   18          CLC
0A4B-   8A          TXA
0A4C-   6D AF 0C    ADC   $0CAF
0A4F-   85 21       STA   $21       $21 := $0CAD + $0CAF

                                  $20.21 + $CAC is SRC (Read) / DEST (Write)

0A51-   E8          INX           X := X + 1 (e.g., length of $0180 = 2 xfers)
0A52-   98          TYA           A := Y == Read/Write flag
0A53-   F0 20       BEQ   $0A75   Read from AUX memory

                                  Write to AUX memory
0A55-   AC AC 0C    LDY   $0CAC
0A58-   8C 05 C0    STY   $C005   Write AUX Memory
0A5B-   F0 07       BEQ   $0A64
0A5D-   B1 22       LDA   ($22),Y  From:  $22.23 -->
0A5F-   91 20       STA   ($20),Y  To:    $20.21 -->
0A61-   88          DEY
0A62-   D0 F9       BNE   $0A5D
0A64-   C6 23       DEC   $23
0A66-   C6 21       DEC   $21
0A68-   CA          DEX           X := X - 1
0A69-   D0 F2       BNE   $0A5D

                                  DONE:
0A6B-   8D 02 C0    STA   $C002   Read Main Memory
0A6E-   8D 04 C0    STA   $C004   Write Main Memory
0A71-   AD 80 C0    LDA   $C080   Read Ram, No Write, Bank 2
0A74-   60          RTS           RTS
--------------------------------
                                  Read from AUX memory
0A75-   AC AC 0C    LDY   $0CAC
```

```
0A78-   8C 03 C0    STY   $C003   Read AUX Memory
0A7B-   F0 0E       BEQ   $0A8B
0A7D-   88          DEY
0A7E-   F0 07       BEQ   $0A87
0A80-   B1 20       LDA   ($20),Y  From:  $20.21 -->
0A82-   91 22       STA   ($22),Y  To:    $22.23 -->
0A84-   88          DEY
0A85-   D0 F9       BNE   $0A80
0A87-   B1 20       LDA   ($20),Y
0A89-   91 22       STA   ($22),Y
0A8B-   88          DEY
0A8C-   C6 21       DEC   $21
0A8E-   C6 23       DEC   $23
0A90-   CA          DEX           X := X - 1
0A91-   D0 ED       BNE   $0A80
0A93-   F0 D6       BEQ   $0A6B    JMP DONE
-------------------------------------------------------------------------------
                                AUX RAM STUFF
                                 (see $0ACD, FE=01 or 02, Apple IIe or Enhanced)

                                Check for AUX RAM
                                Return:  Carry = 1 NOT FOUND
                                                 0 FOUND

                                 See Extended 80-Column Text Card Supplement.pdf
                                   for description of $800 same as $C00.
                                   "Sparse memory mapping".  See p. 36 - 39.

[$00]
0A95-   8D 05 C0    STA   $C005  Write Aux Memory
0A98-   8D 03 C0    STA   $C003  Read Aux Memory
0A9B-   A9 EE       LDA   #$EE
0A9D-   8D 00 08    STA   $0800  Write to  $0800 (could be $0C00)
0AA0-   8D 80 05    STA   $0580  Write to   $0580 (primary text screen)
0AA3-   AC 00 0C    LDY   $0C00  Read from $0C00 (could be $0800 address)
0AA6-   8C 81 05    STY   $0581  Write to   $0581 (primary text screen)
0AA9-   CD 00 0C    CMP   $0C00  Sparse memory mapping?  ($0C00 same as $0800)
0AAC-   D0 0E       BNE   $0ABC    No, $0ABC (Aux Memory Exists, not just 1K)

[$19]
0AAE-   0E 00 0C    ASL   $0C00  Change $0C00 in Aux Memory, and SEC
0AB1-   AD 00 08    LDA   $0800  A := #$EE from Second Text Page in Aux Memory
0AB4-   8D 00 06    STA   $0600  Write #$EE to Primary Text Screen
0AB7-   CD 00 0C    CMP   $0C00  Compare with $0C00 (#$EE or changed?)
0ABA-   F0 01       BEQ   $0ABD  if = now, then Aux Memory doesnt exist (C = 1)
0ABC-   18          CLC
0ABD-   8D 04 C0    STA   $C004  Write Main Memory
0AC0-   8D 02 C0    STA   $C002  Read Main Memory

[$2E]
0AC3-   60          RTS          RTS
```

```
--------------------------------------------------------------------------------
                                    $FE = 01 for IIe
0AC4-   A6 FE       LDX   $FE       $FE = 02 for Enhanced IIe
0AC6-   D0 05       BNE   $0ACD       $FE > 0?  Yes, 0ACD
0AC8-   A2 00       LDX   #$00      FE = 00?
0ACA-   4C 4B 0B    JMP   $0B4B       Yes, $0B4B (No Aux memory)
--------------------------------
0ACD-   78          SEI             Apple IIe or Enhanced IIe
0ACE-   A2 2F       LDX   #$2F
0AD0-   BD 95 0A    LDA   $0A95,X   Move code from $0A95..0AC3 to $00..$2E
0AD3-   95 00       STA   $00,X
0AD5-   CA          DEX
0AD6-   10 F8       BPL   $0AD0
0AD8-   20 00 00    JSR   $0000     Execute code moved to $00.$2E
0ADB-   B0 EB       BCS   $0AC8    If no Aux memory GOTO $0AC8
--------------------------------

  These references have code almost identical to Wizardry IV code:

    http://apple2online.com/web_documents/ae_z-ram_ii_user__s_manual.pdf

    http://www.apple-iigs.info/doc/fichiers/ramworks3.pdf

  RamWorks RAM uses C073 exclusivly where others act on $C07x (any in range).
--------------------------------
                                Find all valid auxiliary memory banks

0ADD-   8D 09 C0    STA   $C009     Use Aux Memory $00..$1FF (zpage, stack, $D000)
0AE0-   A0 7F       LDY   #$7F      Write "bank number" to each bank
0AE2-   8C 73 C0    STY   $C073     Select bank #Y
0AE5-   84 00       STY   $00       Save bank# in the bank
0AE7-   98          TYA
0AE8-   49 FF       EOR   #$FF
0AEA-   85 01       STA   $01       Store another self-check byte
0AEC-   88          DEY
0AED-   10 F3       BPL   $0AE2

0AEF-   A9 00       LDA   #$00      Read "bank number" from each bank
0AF1-   A8          TAY             Y := 0, then 1, 2, 3, ...
0AF2-   AA          TAX             X := 0, then 3, 6, 9, ...

0AF3-   8C 73 C0    STY   $C073     FindThem   Select bank #Y
0AF6-   C4 00       CPY   $00
0AF8-   D0 35       BNE   $0B2F     Check bank number, BNE to NotOne
0AFA-   98          TYA
0AFB-   49 FF       EOR   #$FF
0AFD-   C5 01       CMP   $01       Check another self-check byte
0AFF-   D0 2E       BNE   $0B2F     BNE to NotOne

                                    Found a valid bank!
0B01-   98          TYA               A := bank number
```

```
0B02-   9D 00 02    STA   $0200,X    200: AA AA AA BB BB BB CC CC CC ....
0B05-   9D 01 02    STA   $0201,X         00 00 00 00 (for AppleWin)
0B08-   9D 02 02    STA   $0202,X
0B0B-   A9 03       LDA   #$03       300: 03 04 01 03 04 01 03 04 01 .... 05
0B0D-   9D 00 03    STA   $0300,X         03 04 01 05 (for AppleWin)
0B10-   A9 04       LDA   #$04
0B12-   9D 01 03    STA   $0301,X
0B15-   A9 01       LDA   #$01
0B17-   9D 02 03    STA   $0302,X
0B1A-   8A          TXA               Save index to store next found bank
0B1B-   48          PHA

0B1C-   A2 A5       LDX   #$A5     xfer code from A1F..AC3 to $40..E4
0B1E-   BD 1E 0A    LDA   $0A1E,X    (in this Aux Mem bank's z-page)
0B21-   95 3F       STA   $3F,X
0B23-   CA          DEX
0B24-   D0 F8       BNE   $0B1E

0B26-   68          PLA               Restore index for next found bank
0B27-   18          CLC               Increment to next possible 64K bank
0B28-   69 03       ADC   #$03
0B2A-   AA          TAX
0B2B-   E0 40       CPX   #$40        Max of 64 x 64K banks
0B2D-   B0 03       BCS   $0B32       Too many?  Yes, 0B32, else find another one

                                   NotOne:
0B2F-   C8          INY               Next bank #
0B30-   10 C1       BPL   $0AF3       Try to find it
---------------------------------
0B32-   A9 00       LDA   #$00      Reset to video bank (Bank #0)
0B34-   8D 73 C0    STA   $C073
0B37-   8D 08 C0    STA   $C008     Use Main $00..$1FF  (zpage, stack, $D000)
0B3A-   8A          TXA
0B3B-   A8          TAY

0B3C-   B9 00 03    LDA   $0300,Y   Transfer bank information to
0B3F-   99 E4 FB    STA   $FBE4,Y     $FBE4 table
0B42-   B9 00 02    LDA   $0200,Y     $FC24 table
0B45-   99 24 FC    STA   $FC24,Y   For AppleWin IIe there is 1 bank (Y = 3)
0B48-   88          DEY
0B49-   10 F1       BPL   $0B3C

0B4B-   86 FC       STX   $FC       X = 3 * num of banks (X could be 0)
0B4D-   A9 05       LDA   #$05       end sentinel
0B4F-   9D E4 FB    STA   $FBE4,X
0B52-   A9 00       LDA   #$00
0B54-   9D 24 FC    STA   $FC24,X
0B57-   60          RTS             RTS to $09FD

                                   For AppleWin and IIe:
                                     FBE4: 03 04 01 05
```

```
                                        FC24: 00 00 00 00
--------------------------------------------------------------------------------
0B58-   00          BRK
0B59-   00          BRK
0B5A-   00          BRK
0B5B-   00          BRK
--------------------------------------------------------------------------------

  SEEKABS

SEEK

0B5C-   A5 5A       LDA   $5A     Desired Track
0B5E-   CD 78 04    CMP   $0478   Current Track
0B61-   F0 53       BEQ   $0BB6
0B63-   A9 00       LDA   #$00    Halftracks moved count
0B65-   85 5B       STA   $5B

  Move 1/2 DOS track distance at a time

0B67-   AD 78 04    LDA   $0478
0B6A-   85 5C       STA   $5C     Save prior 1/2 track before move
0B6C-   38          SEC
0B6D-   E5 5A       SBC   $5A
0B6F-   F0 33       BEQ   $0BA4   Same track?  Yes, $0BA4
0B71-   B0 07       BCS   $0B7A
0B73-   49 FF       EOR   #$FF
0B75-   EE 78 04    INC   $0478   Current Track after arm moves
0B78-   90 05       BCC   $0B7F
0B7A-   69 FE       ADC   #$FE
0B7C-   CE 78 04    DEC   $0478
0B7F-   C5 5B       CMP   $5B
0B81-   90 02       BCC   $0B85
0B83-   A5 5B       LDA   $5B
0B85-   C9 0C       CMP   #$0C
0B87-   B0 01       BCS   $0B8A
0B89-   A8          TAY
0B8A-   38          SEC
0B8B-   20 A8 0B    JSR   $0BA8
0B8E-   B9 C8 0B    LDA   $0BC8,Y
0B91-   20 B7 0B    JSR   $0BB7    MSWAIT
0B94-   A5 5C       LDA   $5C
0B96-   18          CLC
0B97-   20 AB 0B    JSR   $0BAB
0B9A-   B9 D4 0B    LDA   $0BD4,Y
0B9D-   20 B7 0B    JSR   $0BB7    MSWAIT
0BA0-   E6 5B       INC   $5B
0BA2-   D0 C3       BNE   $0B67

  Done moving arm
```

```
0BA4-   20 B7 0B    JSR   $0BB7    MSWAIT
0BA7-   18          CLC             C=0 === Phase Off and RTS back

0BA8-   AD 78 04    LDA   $0478    Track (1/2 track for phasing)
0BAB-   29 03       AND   #$03
0BAD-   2A          ROL
0BAE-   05 2B       ORA   $2B
0BB0-   AA          TAX
0BB1-   BD 80 C0    LDA   $C080,X  Phase (0, 1, 2, 3) On / Off
0BB4-   A6 2B       LDX   $2B
0BB6-   60          RTS
--------------------------------
0BB7-   A2 11       LDX   #$11     Arm Move Delay
0BB9-   CA          DEX
0BBA-   D0 FD       BNE   $0BB9
0BBC-   E6 49       INC   $49
0BBE-   D0 02       BNE   $0BC2
0BC0-   E6 48       INC   $48
0BC2-   38          SEC
0BC3-   E9 01       SBC   #$01
0BC5-   D0 F0       BNE   $0BB7
0BC7-   60          RTS
--------------------------------
0BC8-   01 30       ORA   ($30,X)   Arm Move Delay Table
0BCA-   28          PLP
0BCB-   24 20       BIT   $20
0BCD-   1E 1D 1C    ASL   $1C1D,X
0BD0-   1C          ???
0BD1-   1C          ???
0BD2-   1C          ???
0BD3-   1C          ???
0BD4-   70 2C       BVS   $0C02
0BD6-   26 22       ROL   $22
0BD8-   1F          ???
0BD9-   1E 1D 1C    ASL   $1C1D,X
0BDC-   1C          ???
0BDD-   1C          ???
0BDE-   1C          ???
0BDF-   1C          ???
--------------------------------
0BE0-   31 32       AND   ($32),Y
0BE2-   2F          ???
0BE3-   32          ???
0BE4-   32          ???
0BE5-   2F          ???
0BE6-   38          SEC
0BE7-   37          ???
0BE8-   30 30       BMI   $0C1A
0BEA-   30 30       BMI   $0C1C
0BEC-   30 30       BMI   $0C1E
0BEE-   34          ???
```

```
0BEF-   30 00       BMI   $0BF1
0BF1-   00          BRK
0BF2-   00          BRK
0BF3-   00          BRK
0BF4-   00          BRK
0BF5-   00          BRK
0BF6-   00          BRK
0BF7-   00          BRK
0BF8-   00          BRK
0BF9-   00          BRK
0BFA-   00          BRK
0BFB-   00          BRK
0BFC-   00          BRK
0BFD-   00          BRK
0BFE-   01 00       ORA   ($00,X)   01 === DISK #1 identifier

  ------------------------------ Track $00 Sector $0B Pascal Directory ---------

0C00-   00          BRK
0C01-   00          BRK
0C02-   06 00       ASL   $00
0C04-   00          BRK
0C05-   00          BRK
0C06-   07          ???
0C07-   57          ???             "WIZBOOT"
0C08-   49 5A       EOR   #$5A
0C0A-   42          ???
0C0B-   4F          ???
0C0C-   4F          ???
0C0D-   54          ???
0C0E-   00          BRK
0C0F-   08          PHP
0C10-   15 00       ORA   $00,X
0C12-   00          BRK
0C13-   00          BRK
0C14-   BB          ???
0C15-   AE 00 00    LDX   $0000
0C18-   00          BRK
0C19-   00          BRK

0C1A-   06 00       ASL   $00
0C1C-   0A          ASL
0C1D-   00          BRK
0C1E-   05 00       ORA   $00
0C20-   0C          ???
0C21-   53          ???             "SYSTEM.RELOC"
0C22-   59 53 54    EOR   $5453,Y
0C25-   45 4D       EOR   $4D
0C27-   2E 52 45    ROL   $4552
0C2A-   4C 4F 43    JMP   $434F
0C2D-   00          BRK
```

```
0C2E-  40          RTI
0C2F-  B2          ???
0C30-  00          BRK
0C31-  02          ???
0C32-  0A          ASL
0C33-  AF          ???

0C34-  0A          ASL
0C35-  00          BRK
0C36-  24 00       BIT   $00
0C38-  02          ???
0C39-  00          BRK
0C3A-  0D 53 59    ORA   $5953     "SYSTEM.INTERP"
0C3D-  53          ???
0C3E-  54          ???
0C3F-  45 4D       EOR   $4D
0C41-  2E 49 4E    ROL   $4E49
0C44-  54          ???
0C45-  45 52       EOR   $52
0C47-  50 40       BVC   $0C89
0C49-  B2          ???
0C4A-  00          BRK
0C4B-  02          ???
0C4C-  CB          ???
0C4D-  AE

          24 00    LDX   $0024
0C50-  2C 00 05    BIT   $0500
0C53-  00          BRK
0C54-  0B          ???
0C55-  32          ???            "200.CHARSET"
0C56-  30 30       BMI   $0C88
0C58-  2E 43 48    ROL   $4843
0C5B-  41 52       EOR   ($52,X)
0C5D-  53          ???
0C5E-  45 54       EOR   $54
0C60-  00          BRK
0C61-  00          BRK
0C62-  40          RTI
0C63-  B2          ???
0C64-  00          BRK
0C65-  02          ???
0C66-  A9 AF       LDA   #$AF
0C68-  2C 00 44    BIT   $4400
0C6B-  00          BRK
0C6C-  05 00       ORA   $00
0C6E-  0B          ???
0C6F-  57          ???
0C70-  45 52       EOR   $52
0C72-  44          ???
0C73-  4E 41 2E    LSR   $2E41
```

```
0C76-  44          ???
0C77-  41 54       EOR   ($54,X)
0C79-  41 00       EOR   ($00,X)
0C7B-  00          BRK
0C7C-  40          RTI
0C7D-  B2          ???
0C7E-  00          BRK
0C7F-  02          ???
0C80-  19 AF 44    ORA   $44AF,Y
0C83-  00          BRK
0C84-  46 00       LSR   $00
0C86-  05 00       ORA   $00
0C88-  0B          ???
0C89-  4B          ???
0C8A-  41 4E       EOR   ($4E,X)
0C8C-  41 2E       EOR   ($2E,X)
0C8E-  4B          ???
0C8F-  45 59       EOR   $59
0C91-  4D 41 50    EOR   $5041
0C94-  00          BRK
0C95-  00          BRK
0C96-  40          RTI
0C97-  B2          ???
0C98-  00          BRK
0C99-  02          ???
0C9A-  F5 AE       SBC   $AE,X
0C9C-  46 00       LSR   $00
0C9E-  E2          ???
0C9F-  00          BRK
0CA0-  02          ???
0CA1-  00          BRK
0CA2-  0D 53 59    ORA   $5953
0CA5-  53          ???
0CA6-  54          ???
0CA7-  45 4D       EOR   $4D
0CA9-  2E 50 41    ROL   $4150
0CAC-  53          ???
0CAD-  43          ???
0CAE-  41 4C       EOR   ($4C,X)
0CB0-  40          RTI
0CB1-  B2          ???
0CB2-  00          BRK
0CB3-  02          ???
0CB4-  BB          ???
0CB5-  AE E2 00    LDX   $00E2
0CB8-  1E 01 05    ASL   $0501,X
0CBB-  00          BRK
0CBC-  0C          ???
0CBD-  32          ???
0CBE-  30 30       BMI   $0CF0
0CC0-  2E 4D 4F    ROL   $4F4D
```

	0CC3-	4E 53 54	LSR	$5453			
	0CC6-	45 52	EOR	$52			
	0CC8-	53	???				
	0CC9-	00	BRK				
	0CCA-	40	RTI				
	0CCB-	B2	???				
	0CCC-	00	BRK				
	0CCD-	02	???				
	0CCE-	7B	???				
	0CCF-	A8	TAY				
	0CD0-	1E 01 25	ASL	$2501,X			
	0CD3-	01 04	ORA	($04,X)			
	0CD5-	00	BRK				
	0CD6-	09 4D	ORA	#$4D			
	0CD8-	41 5A	EOR	($5A,X)			
	0CDA-	45 2E	EOR	$2E			
	0CDC-	49 4E	EOR	#$4E			
	0CDE-	46 4F	LSR	$4F			
	0CE0-	72	???				
	0CE1-	87	???				
	0CE2-	00	BRK				
	0CE3-	00	BRK				
	0CE4-	40	RTI				
	0CE5-	B2	???				
	0CE6-	00	BRK				
	0CE7-	02	???				
	0CE8-	7B	???				
	0CE9-	A8	TAY				
	0CEA-	25 01	AND	$01			
	0CEC-	9B	???				
	0CED-	01 05	ORA	($05,X)			
	0CEF-	00	BRK				
	0CF0-	09 41	ORA	#$41			
	0CF2-	53	???				
	0CF3-	43	???				
	0CF4-	49 49	EOR	#$49			
	0CF6-	2E 4B 52	ROL	$524B			
	0CF9-	4E 72 87	LSR	$8772			
	0CFC-	00	BRK				
	0CFD-	00	BRK				
	0CFE-	40	RTI				
	0CFF-	B2	???				
	0D00-	00	BRK				
	0D01-	02	???				
	0D02-	BB	???				
	0D03-	AE 9B 01	LDX	$019B			
	0D06-	9D 01 05	STA	$0501,X			
	0D09-	00	BRK				
	0D0A-	0A	ASL				
	0D0B-	41 53	EOR	($53,X)			
	0D0D-	43	???				

	0D0E-	49 49	EOR	#$49			
	0D10-	2E 48 55	ROL	$5548			
	0D13-	46 46	LSR	$46			
	0D15-	87	???				
	0D16-	00	BRK				
	0D17-	00	BRK				
	0D18-	40	RTI				
	0D19-	B2	???				
	0D1A-	00	BRK				
	0D1B-	02	???				
	0D1C-	BB	???				
	0D1D-	AE 9D 01	LDX	$019D			
	0D20-	2D 02 05	AND	$0502			
	0D23-	00	BRK				
	0D24-	0D 53 43	ORA	$4353			
	0D27-	45 4E	EOR	$4E			
	0D29-	41 52	EOR	($52,X)			
	0D2B-	49 4F	EOR	#$4F			
	0D2D-	2E 44 41	ROL	$4144			
	0D30-	54	???				
	0D31-	41 40	EOR	($40,X)			
	0D33-	B2	???				
	0D34-	00	BRK				
	0D35-	02	???				
	0D36-	BB	???				
	0D37-	AE 2D 02	LDX	$022D			
	0D3A-	31 02	AND	($02),Y			
	0D3C-	05 00	ORA	$00			
	0D3E-	0A	ASL				
	0D3F-	41 53	EOR	($53,X)			
	0D41-	43	???				
	0D42-	49 49	EOR	#$49			
	0D44-	2E 53 45	ROL	$4553			
	0D47-	43	???				
	0D48-	53	???				
	0D49-	87	???				
	0D4A-	00	BRK				
	0D4B-	00	BRK				
	0D4C-	40	RTI				
	0D4D-	B2	???				
	0D4E-	00	BRK				
	0D4F-	02	???				
	0D50-	BB	???				
	0D51-	AE 31 02	LDX	$0231			
	0D54-	3B	???				
	0D55-	02	???				
	0D56-	03	???				
	0D57-	00	BRK				
	0D58-	0C	???				
	0D59-	41 53	EOR	($53,X)			
	0D5B-	43	???				

	0D5C-	49 49	EOR	#$49			
	0D5E-	33	???				
	0D5F-	30 2E	BMI	$0D8F			
	0D61-	54	???				
	0D62-	45 58	EOR	$58			
	0D64-	54	???				
	0D65-	00	BRK				
	0D66-	40	RTI				
	0D67-	B2	???				
	0D68-	00	BRK				
	0D69-	02	???				
	0D6A-	7A	???				
	0D6B-	AF	???				
	0D6C-	3B	???				
	0D6D-	02	???				
	0D6E-	40	RTI				
	0D6F-	02	???				
	0D70-	05 00	ORA	$00			
	0D72-	0B	???				
	0D73-	50 4C	BVC	$0DC1			
	0D75-	41 59	EOR	($59,X)			
	0D77-	45 52	EOR	$52			
	0D79-	2E 49 4E	ROL	$4E49			
	0D7C-	46 4F	LSR	$4F			
	0D7E-	00	BRK				
	0D7F-	00	BRK				
	0D80-	40	RTI				
	0D81-	B2	???				
	0D82-	00	BRK				
	0D83-	02	???				
	0D84-	BB	???				
	0D85-	AE 40 02	LDX	$0240			
	0D88-	41 02	EOR	($02,X)			
	0D8A-	05 00	ORA	$00			
	0D8C-	08	PHP				
	0D8D-	49 53	EOR	#$53			
	0D8F-	2E 41 50	ROL	$5041			
	0D92-	50 4C	BVC	$0DE0			
	0D94-	45 00	EOR	$00			
	0D96-	72	???				
	0D97-	87	???				
	0D98-	00	BRK				
	0D99-	00	BRK				
	0D9A-	40	RTI				
	0D9B-	B2	???				
	0D9C-	00	BRK				
	0D9D-	02	???				
	0D9E-	99 AE 41	STA	$41AE,Y			
	0DA1-	02	???				
	0DA2-	42	???				
	0DA3-	02	???				

	0DA4-	05 00	ORA	$00				
	0DA6-	08	PHP					
	0DA7-	4C 49 4E	JMP	$4E49				
	0DAA-	45 53	EOR	$53				
	0DAC-	2E 32 34	ROL	$3432				
	0DAF-	00	BRK					
	0DB0-	72	???					
	0DB1-	87	???					
	0DB2-	00	BRK					
	0DB3-	00	BRK					
	0DB4-	40	RTI					
	0DB5-	B2	???					
	0DB6-	00	BRK					
	0DB7-	02	???					
	0DB8-	99 AE 42	STA	$42AE,Y				
	0DBB-	02	???					
	0DBC-	43	???					
	0DBD-	02	???					
	0DBE-	05 00	ORA	$00				
	0DC0-	09 48	ORA	#$48				
	0DC2-	41 53	EOR	($53,X)				
	0DC4-	2E 43 41	ROL	$4143				
	0DC7-	43	???					
	0DC8-	48	PHA					
	0DC9-	45 48	EOR	$48				
	0DCB-	45 00	EOR	$00				
	0DCD-	00	BRK					
	0DCE-	40	RTI					
	0DCF-	B2	???					
	0DD0-	00	BRK					
	0DD1-	02	???					
	0DD2-	99 AE 43	STA	$43AE,Y				
	0DD5-	02	???					
	0DD6-	44	???					
	0DD7-	02	???					
	0DD8-	05 00	ORA	$00				
	0DDA-	0A	ASL					
	0DDB-	48	PHA					
	0DDC-	41 53	EOR	($53,X)				
	0DDE-	2E 53 54	ROL	$5453				
	0DE1-	52	???					
	0DE2-	4F	???					
	0DE3-	50 53	BVC	$0E38				
	0DE5-	87	???					
	0DE6-	00	BRK					
	0DE7-	00	BRK					
	0DE8-	40	RTI					
	0DE9-	B2	???					
	0DEA-	00	BRK					
	0DEB-	02	???					
	0DEC-	99 AE 44	STA	$44AE,Y				

```
0DEF-  02          ???
0DF0-  45 02       EOR   $02
0DF2-  05 00       ORA   $00
0DF4-  0B          ???
0DF5-  48          PHA
0DF6-  41 53       EOR   ($53,X)
0DF8-  2E 4B 52    ROL   $524B
0DFB-  4E 53 52    LSR   $5253
0DFE-  43          ???
0DFF-  48          PHA
0E00-  00          BRK
0E01-  00          BRK
0E02-  40          RTI
0E03-  B2          ???
0E04-  00          BRK
0E05-  02          ???
0E06-  5A          ???
0E07-  AE 45 02    LDX   $0245
0E0A-  4D 02 03    EOR   $0302
0E0D-  00          BRK
0E0E-  0E 57 45    ASL   $4557
0E11-  52          ???
0E12-  53          ???
0E13-  43          ???
0E14-  52          ???
0E15-  49 50       EOR   #$50
0E17-  54          ???
0E18-  2E 54 45    ROL   $4554
0E1B-  58          CLI
0E1C-  54          ???
0E1D-  B2          ???
0E1E-  00          BRK
0E1F-  02          ???
0E20-  FA          ???
0E21-  AE 4D 02    LDX   $024D
0E24-  00          BRK
0E25-  08          PHP
0E26-  05 00       ORA   $00
0E28-  0F          ???
0E29-  57          ???
0E2A-  45 52       EOR   $52
0E2C-  44          ???
0E2D-  4E 41 2E    LSR   $2E41
0E30-  31 31       AND   ($31),Y
0E32-  2E 31 31    ROL   $3131
0E35-  2E 38 37    ROL   $3738
0E38-  00          BRK
0E39-  02          ???
0E3A-  BB          ???
0E3B-  AE 4D 02    LDX   $024D
0E3E-  00          BRK
```

	0E3F-	08	PHP				
	0E40-	05 00	ORA	$00			
	0E42-	00	BRK				
	0E43-	57	???				
	0E44-	45 52	EOR	$52			
	0E46-	44	???				
	0E47-	4E 41 2E	LSR	$2E41			
	0E4A-	31 31	AND	($31),Y			
	0E4C-	2E 31 31	ROL	$3131			
	0E4F-	2E 38 37	ROL	$3738			
	0E52-	00	BRK				
	0E53-	02	???				
	0E54-	BB	???				
	0E55-	AE A2 01	LDX	$01A2			
	0E58-	A5 01	LDA	$01			
	0E5A-	02	???				
	0E5B-	00	BRK				
	0E5C-	0D 53 48	ORA	$4853			
	0E5F-	4F	???				
	0E60-	57	???				
	0E61-	44	???				
	0E62-	49 43	EOR	#$43			
	0E64-	54	???				
	0E65-	2E 43 4F	ROL	$4F43			
	0E68-	44	???				
	0E69-	45 40	EOR	$40			
	0E6B-	B2	???				
	0E6C-	00	BRK				
	0E6D-	02	???				
	0E6E-	D8	CLD				
	0E6F-	AE A5 01	LDX	$01A5			
	0E72-	AD 01 03	LDA	$0301			
	0E75-	00	BRK				
	0E76-	0D 53 48	ORA	$4853			
	0E79-	4F	???				
	0E7A-	57	???				
	0E7B-	44	???				
	0E7C-	49 43	EOR	#$43			
	0E7E-	54	???				
	0E7F-	2E 54 45	ROL	$4554			
	0E82-	58	CLI				
	0E83-	54	???				
	0E84-	40	RTI				
	0E85-	B2	???				
	0E86-	00	BRK				
	0E87-	02	???				
	0E88-	D8	CLD				
	0E89-	AE 00 02	LDX	$0200			
	0E8C-	1E 02 03	ASL	$0302,X			
	0E8F-	00	BRK				
	0E90-	0A	ASL				

	0E91-	49 50	EOR	#$50			
	0E93-	52	???				
	0E94-	4F	???				
	0E95-	43	???				
	0E96-	2E 54 45	ROL	$4554			
	0E99-	58	CLI				
	0E9A-	54	???				
	0E9B-	87	???				
	0E9C-	00	BRK				
	0E9D-	00	BRK				
	0E9E-	40	RTI				
	0E9F-	B2	???				
	0EA0-	00	BRK				
	0EA1-	02	???				
	0EA2-	5B	???				
	0EA3-	AE 1E 02	LDX	$021E			
	0EA6-	38	SEC				
	0EA7-	02	???				
	0EA8-	02	???				
	0EA9-	00	BRK				
	0EAA-	0F	???				
	0EAB-	4F	???				
	0EAC-	4C 44 45	JMP	$4544			
	0EAF-	49 4E	EOR	#$4E			
	0EB1-	54	???				
	0EB2-	45 52	EOR	$52			
	0EB4-	50 2E	BVC	$0EE4			
	0EB6-	43	???				
	0EB7-	4F	???				
	0EB8-	44	???				
	0EB9-	45 00	EOR	$00			
	0EBB-	02	???				
	0EBC-	5B	???				
	0EBD-	AE 38 02	LDX	$0238			
	0EC0-	3C	???				
	0EC1-	02	???				
	0EC2-	02	???				
	0EC3-	00	BRK				
	0EC4-	0D 4E 45	ORA	$454E			
	0EC7-	57	???				
	0EC8-	45 42	EOR	$42			
	0ECA-	4F	???				
	0ECB-	4F	???				
	0ECC-	54	???				
	0ECD-	2E 43 4F	ROL	$4F43			
	0ED0-	44	???				
	0ED1-	45 40	EOR	$40			
	0ED3-	B2	???				
	0ED4-	00	BRK				
	0ED5-	02	???				
	0ED6-	5B	???				

	0ED7-	AE 3C 02	LDX	$023C			
	0EDA-	4C 02 03	JMP	$0302			
	0EDD-	00	BRK				
	0EDE-	0B	???				
	0EDF-	49 4D	EOR	#$4D			
	0EE1-	41 43	EOR	($43,X)			
	0EE3-	52	???				
	0EE4-	4F	???				
	0EE5-	2E 54 45	ROL	$4554			
	0EE8-	58	CLI				
	0EE9-	54	???				
	0EEA-	00	BRK				
	0EEB-	00	BRK				
	0EEC-	40	RTI				
	0EED-	B2	???				
	0EEE-	00	BRK				
	0EEF-	02	???				
	0EF0-	5B	???				
	0EF1-	AE 4C 02	LDX	$024C			
	0EF4-	67	???				
	0EF5-	02	???				
	0EF6-	02	???				
	0EF7-	00	BRK				
	0EF8-	0F	???				
	0EF9-	4E 45 57	LSR	$5745			
	0EFC-	45 49	EOR	$49			
	0EFE-	4E 54 45	LSR	$4554			
	0F01-	52	???				
	0F02-	50 2E	BVC	$0F32			
	0F04-	43	???				
	0F05-	4F	???				
	0F06-	44	???				
	0F07-	45 00	EOR	$00			
	0F09-	02	???				
	0F0A-	6B	???				
	0F0B-	AE 67 02	LDX	$0267			
	0F0E-	99 02 03	STA	$0302,Y			
	0F11-	00	BRK				
	0F12-	0B	???				
	0F13-	55 4E	EOR	$4E,X			
	0F15-	49 54	EOR	#$54			
	0F17-	31 33	AND	($33),Y			
	0F19-	2E 54 45	ROL	$4554			
	0F1C-	58	CLI				
	0F1D-	54	???				
	0F1E-	00	BRK				
	0F1F-	00	BRK				
	0F20-	40	RTI				
	0F21-	B2	???				
	0F22-	00	BRK				
	0F23-	02	???				

	0F24-	6B	???				
	0F25-	AE AF 02	LDX	$02AF			
	0F28-	B3	???				
	0F29-	02	???				
	0F2A-	02	???				
	0F2B-	00	BRK				
	0F2C-	0C	???				
	0F2D-	4E 45 57	LSR	$5745			
	0F30-	42	???				
	0F31-	4F	???				
	0F32-	4F	???				
	0F33-	54	???				
	0F34-	2E 43 4F	ROL	$4F43			
	0F37-	44	???				
	0F38-	45 00	EOR	$00			
	0F3A-	40	RTI				
	0F3B-	B2	???				
	0F3C-	00	BRK				
	0F3D-	02	???				
	0F3E-	6B	???				
	0F3F-	AE B3 02	LDX	$02B3			
	0F42-	CD 02 02	CMP	$0202			
	0F45-	00	BRK				
	0F46-	07	???				
	0F47-	4E 45 57	LSR	$5745			
	0F4A-	54	???				
	0F4B-	45 53	EOR	$53			
	0F4D-	54	???				
	0F4E-	00	BRK				
	0F4F-	00	BRK				
	0F50-	72	???				
	0F51-	87	???				
	0F52-	00	BRK				
	0F53-	00	BRK				
	0F54-	40	RTI				
	0F55-	B2	???				
	0F56-	00	BRK				
	0F57-	02	???				
	0F58-	6B	???				
	0F59-	AE CD 02	LDX	$02CD			
	0F5C-	E9 02	SBC	#$02			
	0F5E-	03	???				
	0F5F-	00	BRK				
	0F60-	0C	???				
	0F61-	49 4C	EOR	#$4C			
	0F63-	4F	???				
	0F64-	43	???				
	0F65-	41 4C	EOR	($4C,X)			
	0F67-	53	???				
	0F68-	2E 54 45	ROL	$4554			
	0F6B-	58	CLI				

	0F6C-	54	???				
	0F6D-	00	BRK				
	0F6E-	40	RTI				
	0F6F-	B2	???				
	0F70-	00	BRK				
	0F71-	02	???				
	0F72-	7B	???				
	0F73-	AE E9 02	LDX	$02E9			
	0F76-	03	???				
	0F77-	03	???				
	0F78-	02	???				
	0F79-	00	BRK				
	0F7A-	0C	???				
	0F7B-	49 4E	EOR	#$4E			
	0F7D-	54	???				
	0F7E-	45 52	EOR	$52			
	0F80-	50 31	BVC	$0FB3			
	0F82-	2E 43 4F	ROL	$4F43			
	0F85-	44	???				
	0F86-	45 00	EOR	$00			
	0F88-	40	RTI				
	0F89-	B2	???				
	0F8A-	00	BRK				
	0F8B-	02	???				
	0F8C-	9B	???				
	0F8D-	AE 03 03	LDX	$0303			
	0F90-	1D 03 02	ORA	$0203,X			
	0F93-	00	BRK				
	0F94-	05 54	ORA	$54			
	0F96-	45 53	EOR	$53			
	0F98-	54	???				
	0F99-	31 00	AND	($00),Y			
	0F9B-	00	BRK				
	0F9C-	00	BRK				
	0F9D-	00	BRK				
	0F9E-	72	???				
	0F9F-	87	???				
	0FA0-	00	BRK				
	0FA1-	00	BRK				
	0FA2-	40	RTI				
	0FA3-	B2	???				
	0FA4-	00	BRK				
	0FA5-	02	???				
	0FA6-	9B	???				
	0FA7-	AE 1D 03	LDX	$031D			
	0FAA-	27	???				
	0FAB-	03	???				
	0FAC-	03	???				
	0FAD-	00	BRK				
	0FAE-	0D 57 49	ORA	$4957			
	0FB1-	5A	???				

	0FB2-	47	???				
	0FB3-	52	???				
	0FB4-	41 50	EOR	($50,X)			
	0FB6-	48	PHA				
	0FB7-	2E 54 45	ROL	$4554			
	0FBA-	58	CLI				
	0FBB-	54	???				
	0FBC-	40	RTI				
	0FBD-	B2	???				
	0FBE-	00	BRK				
	0FBF-	02	???				
	0FC0-	9B	???				
	0FC1-	AE 27 03	LDX	$0327			
	0FC4-	3B	???				
	0FC5-	03	???				
	0FC6-	03	???				
	0FC7-	00	BRK				
	0FC8-	0F	???				
	0FC9-	4D 4F 4E	EOR	$4E4F			
	0FCC-	53	???				
	0FCD-	54	???				
	0FCE-	47	???				
	0FCF-	52	???				
	0FD0-	41 50	EOR	($50,X)			
	0FD2-	48	PHA				
	0FD3-	2E 54 45	ROL	$4554			
	0FD6-	58	CLI				
	0FD7-	54	???				
	0FD8-	00	BRK				
	0FD9-	02	???				
	0FDA-	9B	???				
	0FDB-	AE 3B 03	LDX	$033B			
	0FDE-	59 03 03	EOR	$0303,Y			
	0FE1-	00	BRK				
	0FE2-	0C	???				
	0FE3-	4E 45 57	LSR	$5745			
	0FE6-	42	???				
	0FE7-	4F	???				
	0FE8-	4F	???				
	0FE9-	54	???				
	0FEA-	2E 54 45	ROL	$4554			
	0FED-	58	CLI				
	0FEE-	54	???				
	0FEF-	00	BRK				
	0FF0-	40	RTI				
	0FF1-	B2	???				
	0FF2-	00	BRK				
	0FF3-	02	???				
	0FF4-	9B	???				
	0FF5-	AE 59 03	LDX	$0359			
	0FF8-	74	???				

```
0FF9-  03          ???
0FFA-  02          ???
0FFB-  00          BRK
0FFC-  0B          ???
0FFD-  49 4E       EOR   #$4E
0FFF-  54          ???
1000-  45 52       EOR   $52
1002-  50 2E       BVC   $1032
1004-  43          ???
1005-  4F          ???
1006-  44          ???
1007-  45 00       EOR   $00
1009-  00          BRK
100A-  6C B4 00    JMP   ($00B4)
100D-  02          ???
100E-  BB          ???
100F-  AE C1 03    LDX   $03C1
1012-  D7          ???
1013-  03          ???
1014-  03          ???
1015-  00          BRK
1016-  0B          ???
1017-  49 4C       EOR   #$4C
1019-  4F          ???
101A-  43          ???
101B-  41 54       EOR   ($54,X)
101D-  2E 54 45    ROL   $4554
1020-  58          CLI
1021-  54          ???
1022-  00          BRK
1023-  00          BRK
1024-  40          RTI
1025-  B2          ???
1026-  00          BRK
1027-  02          ???
1028-  9B          ???
1029-  AE 0B 04    LDX   $040B
102C-  29 04       AND   #$04
102E-  03          ???
102F-  00          BRK
1030-  0E 49 53    ASL   $5349
1033-  54          ???
1034-  41 4E       EOR   ($4E,X)
1036-  44          ???
1037-  41 52       EOR   ($52,X)
1039-  44          ???
103A-  2E 54 45    ROL   $4554
103D-  58          CLI
103E-  54          ???
103F-  B2          ???
1040-  00          BRK
```

	1041-	02	???				
	1042-	9B	???				
	1043-	AE 83 04	LDX	$0483			
	1046-	9F	???				
	1047-	04	???				
	1048-	03	???				
	1049-	00	BRK				
	104A-	0A	ASL				
	104B-	49 49	EOR	#$49			
	104D-	4E 49 54	LSR	$5449			
	1050-	2E 54 45	ROL	$4554			
	1053-	58	CLI				
	1054-	54	???				
	1055-	87	???				
	1056-	00	BRK				
	1057-	00	BRK				
	1058-	40	RTI				
	1059-	B2	???				
	105A-	00	BRK				
	105B-	02	???				
	105C-	AB	???				
	105D-	AE 9F 04	LDX	$049F			
	1060-	A3	???				
	1061-	04	???				
	1062-	02	???				
	1063-	00	BRK				
	1064-	0A	ASL				
	1065-	49 49	EOR	#$49			
	1067-	4E 49 54	LSR	$5449			
	106A-	2E 43 4F	ROL	$4F43			
	106D-	44	???				
	106E-	45 87	EOR	$87			
	1070-	00	BRK				
	1071-	00	BRK				
	1072-	40	RTI				
	1073-	B2	???				
	1074-	00	BRK				
	1075-	02	???				
	1076-	AB	???				
	1077-	AE BD 04	LDX	$04BD			
	107A-	D7	???				
	107B-	04	???				
	107C-	02	???				
	107D-	00	BRK				
	107E-	0C	???				
	107F-	49 4E	EOR	#$4E			
	1081-	54	???				
	1082-	45 52	EOR	$52			
	1084-	50 37	BVC	$10BD			
	1086-	2E 43 4F	ROL	$4F43			
	1089-	44	???				

	108A-	45 00	EOR	$00				
	108C-	40	RTI					
	108D-	B2	???					
	108E-	00	BRK					
	108F-	02	???					
	1090-	AB	???					
	1091-	AE D7 04	LDX	$04D7				
	1094-	F1 04	SBC	($04),Y				
	1096-	02	???					
	1097-	00	BRK					
	1098-	05 54	ORA	$54				
	109A-	45 53	EOR	$53				
	109C-	54	???					
	109D-	37	???					
	109E-	00	BRK					
	109F-	00	BRK					
	10A0-	00	BRK					
	10A1-	00	BRK					
	10A2-	72	???					
	10A3-	87	???					
	10A4-	00	BRK					
	10A5-	00	BRK					
	10A6-	40	RTI					
	10A7-	B2	???					
	10A8-	00	BRK					
	10A9-	02	???					
	10AA-	AB	???					
	10AB-	AE 87 05	LDX	$0587				
	10AE-	95 05	STA	$05,X				
	10B0-	03	???					
	10B1-	00	BRK					
	10B2-	0D 4E 45	ORA	$454E				
	10B5-	57	???					
	10B6-	4D 45 53	EOR	$5345				
	10B9-	53	???					
	10BA-	47	???					
	10BB-	2E 54 45	ROL	$4554				
	10BE-	58	CLI					
	10BF-	54	???					
	10C0-	40	RTI					
	10C1-	B2	???					
	10C2-	00	BRK					
	10C3-	02	???					
	10C4-	AB	???					
	10C5-	AE 95 05	LDX	$0595				
	10C8-	BB	???					
	10C9-	05 03	ORA	$03				
	10CB-	00	BRK					
	10CC-	0C	???					
	10CD-	49 54	EOR	#$54				
	10CF-	41 42	EOR	($42,X)				

	10D1-	4C 45 53	JMP	$5345			
	10D4-	2E 54 45	ROL	$4554			
	10D7-	58	CLI				
	10D8-	54	???				
	10D9-	00	BRK				
	10DA-	EA	NOP				
	10DB-	A4 00	LDY	$00			
	10DD-	02	???				
	10DE-	BB	???				
	10DF-	AE 83 06	LDX	$0683			
	10E2-	9B	???				
	10E3-	06 03	ASL	$03			
	10E5-	00	BRK				
	10E6-	0D 56 49	ORA	$4956			
	10E9-	52	???				
	10EA-	54	???				
	10EB-	44	???				
	10EC-	52	???				
	10ED-	49 56	EOR	#$56			
	10EF-	2E 54 45	ROL	$4554			
	10F2-	58	CLI				
	10F3-	54	???				
	10F4-	EA	NOP				
	10F5-	A4 00	LDY	$00			
	10F7-	02	???				
	10F8-	BB	???				
	10F9-	AE 9B 06	LDX	$069B			
	10FC-	A3	???				
	10FD-	06 03	ASL	$03			
	10FF-	00	BRK				
	1100-	0E 4F 54	ASL	$544F			
	1103-	48	PHA				
	1104-	45 52	EOR	$52			
	1106-	44	???				
	1107-	49 53	EOR	#$53			
	1109-	4B	???				
	110A-	2E 54 45	ROL	$4554			
	110D-	58	CLI				
	110E-	54	???				
	110F-	A4 00	LDY	$00			
	1111-	02	???				
	1112-	BB	???				
	1113-	AE D8 06	LDX	$06D8			
	1116-	FE 06 03	INC	$0306,X			
	1119-	00	BRK				
	111A-	0E 4F 4C	ASL	$4C4F			
	111D-	44	???				
	111E-	54	???				
	111F-	41 42	EOR	($42,X)			
	1121-	4C 45 53	JMP	$5345			
	1124-	2E 54 45	ROL	$4554			

```
1127-  58          CLI
1128-  54          ???
1129-  A5 00       LDA   $00
112B-  02          ???
112C-  BB          ???
112D-  AE FE 06    LDX   $06FE
1130-  02          ???
1131-  07          ???
1132-  03          ???
1133-  00          BRK
1134-  0D 4D 45    ORA   $454D
1137-  53          ???
1138-  53          ???
1139-  41 47       EOR   ($47,X)
113B-  45 53       EOR   $53
113D-  2E 54 45    ROL   $4554
1140-  58          CLI
1141-  54          ???
1142-  40          RTI
1143-  B2          ???
1144-  00          BRK
1145-  02          ???
1146-  BB          ???
1147-  AE 02 07    LDX   $0702
114A-  06 07       ASL   $07
114C-  02          ???
114D-  00          BRK
114E-  0D 4D 45    ORA   $454D
1151-  53          ???
1152-  53          ???
1153-  41 47       EOR   ($47,X)
1155-  45 53       EOR   $53
1157-  2E 43 4F    ROL   $4F43
115A-  44          ???
115B-  45 3A       EOR   $3A
115D-  B2          ???
115E-  00          BRK
115F-  02          ???
1160-  BB          ???
1161-  AE 06 07    LDX   $0706
1164-  1E 07 03    ASL   $0307,X
1167-  00          BRK
1168-  0A          ASL
1169-  52          ???
116A-  57          ???
116B-  54          ???
116C-  53          ???
116D-  32          ???
116E-  2E 54 45    ROL   $4554
1171-  58          CLI
1172-  54          ???
```

```
1173-  87          ???
1174-  00          BRK
1175-  00          BRK
1176-  EA          NOP
1177-  A4 00       LDY   $00
1179-  02          ???
117A-  BB          ???
117B-  AE 69 07    LDX   $0769
117E-  77          ???
117F-  07          ???
1180-  03          ???
1181-  00          BRK
1182-  0E 49 4E    ASL   $4E49
1185-  49 54       EOR   #$54
1187-  47          ???
1188-  52          ???
1189-  41 50       EOR   ($50,X)
118B-  48          PHA
118C-  2E 54 45    ROL   $4554
118F-  58          CLI
1190-  54          ???
1191-  A4 00       LDY   $00
1193-  02          ???
1194-  BB          ???
1195-  AE 9D 07    LDX   $079D
1198-  B3          ???
1199-  07          ???
119A-  03          ???
119B-  00          BRK
119C-  0C          ???
119D-  49 45       EOR   #$45
119F-  51 55       EOR   ($55),Y
11A1-  41 54       EOR   ($54,X)
11A3-  45 2E       EOR   $2E
11A5-  54          ???
11A6-  45 58       EOR   $58
11A8-  54          ???
11A9-  00          BRK
11AA-  EA          NOP
11AB-  A4 00       LDY   $00
11AD-  02          ???
11AE-  BB          ???
11AF-  AE 9D 07    LDX   $079D
11B2-  B3          ???
11B3-  07          ???
11B4-  03          ???
11B5-  00          BRK
11B6-  00          BRK
11B7-  49 45       EOR   #$45
11B9-  51 55       EOR   ($55),Y
11BB-  41 54       EOR   ($54,X)
```

	11BD-	45 2E	EOR	$2E			
	11BF-	54	???				
	11C0-	45 58	EOR	$58			
	11C2-	54	???				
	11C3-	00	BRK				
	11C4-	EA	NOP				
	11C5-	A4 00	LDY	$00			
	11C7-	02	???				
	11C8-	BB	???				
	11C9-	AE 9D 07	LDX	$079D			
	11CC-	B3	???				
	11CD-	07	???				
	11CE-	03	???				
	11CF-	00	BRK				
	11D0-	00	BRK				
	11D1-	49 45	EOR	#$45			
	11D3-	51 55	EOR	($55),Y			
	11D5-	41 54	EOR	($54,X)			
	11D7-	45 2E	EOR	$2E			
	11D9-	54	???				
	11DA-	45 58	EOR	$58			
	11DC-	54	???				
	11DD-	00	BRK				
	11DE-	EA	NOP				
	11DF-	A4 00	LDY	$00			
	11E1-	02	???				
	11E2-	BB	???				
	11E3-	AE 9D 07	LDX	$079D			
	11E6-	B3	???				
	11E7-	07	???				
	11E8-	03	???				
	11E9-	00	BRK				
	11EA-	00	BRK				
	11EB-	49 45	EOR	#$45			
	11ED-	51 55	EOR	($55),Y			
	11EF-	41 54	EOR	($54,X)			
	11F1-	45 2E	EOR	$2E			
	11F3-	54	???				
	11F4-	45 58	EOR	$58			
	11F6-	54	???				
	11F7-	00	BRK				
	11F8-	EA	NOP				
	11F9-	A4 00	LDY	$00			
	11FB-	02	???				
	11FC-	BB	???				
	11FD-	AE 9D 07	LDX	$079D			
					--- Block #5 is hijacked from Directory ---		
	1200-	FE 06 03	INC	$0306,X			
	1203-	00	BRK				
	1204-	0E 4F 4C	ASL	$4C4F			
	1207-	44	???				

	1208-	54	???				
	1209-	41 42	EOR	($42,X)			
	120B-	4C 45 53	JMP	$5345			
	120E-	2E 54 45	ROL	$4554			
	1211-	58	CLI				
	1212-	54	???				
	1213-	A5 00	LDA	$00			
	1215-	02	???				
	1216-	BB	???				
	1217-	AE FE 06	LDX	$06FE			
	121A-	02	???				
	121B-	07	???				
	121C-	03	???				
	121D-	00	BRK				
	121E-	0D 4D 45	ORA	$454D			
	1221-	53	???				
	1222-	53	???				
	1223-	41 47	EOR	($47,X)			
	1225-	45 53	EOR	$53			
	1227-	2E 54 45	ROL	$4554			
	122A-	58	CLI				
	122B-	54	???				
	122C-	40	RTI				
	122D-	B2	???				
	122E-	00	BRK				
	122F-	02	???				
	1230-	BB	???				
	1231-	AE 02 07	LDX	$0702			
	1234-	06 07	ASL	$07			
	1236-	02	???				
	1237-	00	BRK				
	1238-	0D 4D 45	ORA	$454D			
	123B-	53	???				
	123C-	53	???				
	123D-	41 47	EOR	($47,X)			
	123F-	45 53	EOR	$53			
	1241-	2E 43 4F	ROL	$4F43			
	1244-	44	???				
	1245-	45 3A	EOR	$3A			
	1247-	B2	???				
	1248-	00	BRK				
	1249-	02	???				
	124A-	BB	???				
	124B-	AE 06 07	LDX	$0706			
	124E-	1E 07 03	ASL	$0307,X			
	1251-	00	BRK				
	1252-	0A	ASL				
	1253-	52	???				
	1254-	57	???				
	1255-	54	???				
	1256-	53	???				

	1257-	32	???				
	1258-	2E 54 45	ROL	$4554			
	125B-	58	CLI				
	125C-	54	???				
	125D-	87	???				
	125E-	00	BRK				
	125F-	00	BRK				
	1260-	EA	NOP				
	1261-	A4 00	LDY	$00			
	1263-	02	???				
	1264-	BB	???				
	1265-	AE 39 07	LDX	$0739			
	1268-	53	???				
	1269-	07	???				
	126A-	02	???				
	126B-	00	BRK				
	126C-	06 54	ASL	$54			
	126E-	45 53	EOR	$53			
	1270-	54	???				
	1271-	31 34	AND	($34),Y			
	1273-	00	BRK				
	1274-	00	BRK				
	1275-	00	BRK				
	1276-	72	???				
	1277-	87	???				
	1278-	00	BRK				
	1279-	00	BRK				
	127A-	40	RTI				
	127B-	B2	???				
	127C-	00	BRK				
	127D-	02	???				
	127E-	BB	???				
	127F-	AE 69 07	LDX	$0769			
	1282-	77	???				
	1283-	07	???				
	1284-	03	???				
	1285-	00	BRK				
	1286-	0E 49 4E	ASL	$4E49			
	1289-	49 54	EOR	#$54			
	128B-	47	???				
	128C-	52	???				
	128D-	41 50	EOR	($50,X)			
	128F-	48	PHA				
	1290-	2E 54 45	ROL	$4554			
	1293-	58	CLI				
	1294-	54	???				
	1295-	A4 00	LDY	$00			
	1297-	02	???				
	1298-	BB	???				
	1299-	AE 77 07	LDX	$0777			
	129C-	9D 07 03	STA	$0307,X			

	129F-	00	BRK				
	12A0-	0C	???				
	12A1-	49 54	EOR	#$54			
	12A3-	41 42	EOR	($42,X)			
	12A5-	4C 45 53	JMP	$5345			
	12A8-	2E 54 45	ROL	$4554			
	12AB-	58	CLI				
	12AC-	54	???				
	12AD-	00	BRK				
	12AE-	EA	NOP				
	12AF-	A4 00	LDY	$00			
	12B1-	02	???				
	12B2-	BB	???				
	12B3-	AE 9D 07	LDX	$079D			
	12B6-	B3	???				
	12B7-	07	???				
	12B8-	03	???				
	12B9-	00	BRK				
	12BA-	0C	???				
	12BB-	49 45	EOR	#$45			
	12BD-	51 55	EOR	($55),Y			
	12BF-	41 54	EOR	($54,X)			
	12C1-	45 2E	EOR	$2E			
	12C3-	54	???				
	12C4-	45 58	EOR	$58			
	12C6-	54	???				
	12C7-	00	BRK				
	12C8-	EA	NOP				
	12C9-	A4 00	LDY	$00			
	12CB-	02	???				
	12CC-	BB	???				
	12CD-	AE 9D 07	LDX	$079D			
	12D0-	B3	???				
	12D1-	07	???				
	12D2-	03	???				
	12D3-	00	BRK				
	12D4-	00	BRK				
	12D5-	49 45	EOR	#$45			
	12D7-	51 55	EOR	($55),Y			
	12D9-	41 54	EOR	($54,X)			
	12DB-	45 2E	EOR	$2E			
	12DD-	54	???				
	12DE-	45 58	EOR	$58			
	12E0-	54	???				
	12E1-	00	BRK				
	12E2-	EA	NOP				
	12E3-	A4 00	LDY	$00			
	12E5-	02	???				
	12E6-	BB	???				
	12E7-	AE 9D 07	LDX	$079D			
	12EA-	B3	???				

	12EB-	07	???				
	12EC-	03	???				
	12ED-	00	BRK				
	12EE-	00	BRK				
	12EF-	49 45	EOR	#$45			
	12F1-	51 55	EOR	($55),Y			
	12F3-	41 54	EOR	($54,X)			
	12F5-	45 2E	EOR	$2E			
	12F7-	54	???				
	12F8-	45 58	EOR	$58			
	12FA-	54	???				
	12FB-	00	BRK				
	12FC-	EA	NOP				
	12FD-	A4 00	LDY	$00			
	12FF-	02	???				
	1300-	BB	???				
	1301-	AE 9D 07	LDX	$079D			
	1304-	B3	???				
	1305-	07	???				
	1306-	03	???				
	1307-	00	BRK				
	1308-	00	BRK				
	1309-	49 45	EOR	#$45			
	130B-	51 55	EOR	($55),Y			
	130D-	41 54	EOR	($54,X)			
	130F-	45 2E	EOR	$2E			
	1311-	54	???				
	1312-	45 58	EOR	$58			
	1314-	54	???				
	1315-	00	BRK				
	1316-	EA	NOP				
	1317-	A4 00	LDY	$00			
	1319-	02	???				
	131A-	BB	???				
	131B-	AE 9D 07	LDX	$079D			
	131E-	B3	???				
	131F-	07	???				
	1320-	03	???				
	1321-	00	BRK				
	1322-	00	BRK				
	1323-	49 45	EOR	#$45			
	1325-	51 55	EOR	($55),Y			
	1327-	41 54	EOR	($54,X)			
	1329-	45 2E	EOR	$2E			
	132B-	54	???				
	132C-	45 58	EOR	$58			
	132E-	54	???				
	132F-	00	BRK				
	1330-	EA	NOP				
	1331-	A4 00	LDY	$00			
	1333-	02	???				

	1334-	BB	???				
	1335-	AE 9D 07	LDX	$079D			
	1338-	B3	???				
	1339-	07	???				
	133A-	03	???				
	133B-	00	BRK				
	133C-	00	BRK				
	133D-	49 45	EOR	#$45			
	133F-	51 55	EOR	($55),Y			
	1341-	41 54	EOR	($54,X)			
	1343-	45 2E	EOR	$2E			
	1345-	54	???				
	1346-	45 58	EOR	$58			
	1348-	54	???				
	1349-	00	BRK				
	134A-	EA	NOP				
	134B-	A4 00	LDY	$00			
	134D-	02	???				
	134E-	BB	???				
	134F-	AE 9D 07	LDX	$079D			
	1352-	B3	???				
	1353-	07	???				
	1354-	03	???				
	1355-	00	BRK				
	1356-	00	BRK				
	1357-	49 45	EOR	#$45			
	1359-	51 55	EOR	($55),Y			
	135B-	41 54	EOR	($54,X)			
	135D-	45 2E	EOR	$2E			
	135F-	54	???				
	1360-	45 58	EOR	$58			
	1362-	54	???				
	1363-	00	BRK				
	1364-	EA	NOP				
	1365-	A4 00	LDY	$00			
	1367-	02	???				
	1368-	BB	???				
	1369-	AE 9D 07	LDX	$079D			
	136C-	B3	???				
	136D-	07	???				
	136E-	03	???				
	136F-	00	BRK				
	1370-	00	BRK				
	1371-	49 45	EOR	#$45			
	1373-	51 55	EOR	($55),Y			
	1375-	41 54	EOR	($54,X)			
	1377-	45 2E	EOR	$2E			
	1379-	54	???				
	137A-	45 58	EOR	$58			
	137C-	54	???				
	137D-	00	BRK				

	137E-	EA	NOP				
	137F-	A4 00	LDY	$00			
	1381-	02	???				
	1382-	BB	???				
	1383-	AE 9D 07	LDX	$079D			
	1386-	B3	???				
	1387-	07	???				
	1388-	03	???				
	1389-	00	BRK				
	138A-	00	BRK				
	138B-	49 45	EOR	#$45			
	138D-	51 55	EOR	($55),Y			
	138F-	41 54	EOR	($54,X)			
	1391-	45 2E	EOR	$2E			
	1393-	54	???				
	1394-	45 58	EOR	$58			
	1396-	54	???				
	1397-	00	BRK				
	1398-	EA	NOP				
	1399-	A4 00	LDY	$00			
	139B-	02	???				
	139C-	BB	???				
	139D-	AE 9D 07	LDX	$079D			
	13A0-	B3	???				
	13A1-	07	???				
	13A2-	03	???				
	13A3-	00	BRK				
	13A4-	00	BRK				
	13A5-	49 45	EOR	#$45			
	13A7-	51 55	EOR	($55),Y			
	13A9-	41 54	EOR	($54,X)			
	13AB-	45 2E	EOR	$2E			
	13AD-	54	???				
	13AE-	45 58	EOR	$58			
	13B0-	54	???				
	13B1-	00	BRK				
	13B2-	EA	NOP				
	13B3-	A4 00	LDY	$00			
	13B5-	02	???				
	13B6-	BB	???				
	13B7-	AE 9D 07	LDX	$079D			
	13BA-	B3	???				
	13BB-	07	???				
	13BC-	03	???				
	13BD-	00	BRK				
	13BE-	00	BRK				
	13BF-	49 45	EOR	#$45			
	13C1-	51 55	EOR	($55),Y			
	13C3-	41 54	EOR	($54,X)			
	13C5-	45 2E	EOR	$2E			
	13C7-	54	???				

```
13C8-   45 58       EOR   $58
13CA-   54          ???
13CB-   00          BRK
13CC-   EA          NOP
13CD-   A4 00       LDY   $00
13CF-   02          ???
13D0-   BB          ???
13D1-   AE 9D 07    LDX   $079D
13D4-   B3          ???
13D5-   07          ???
13D6-   03          ???
13D7-   00          BRK
13D8-   00          BRK
13D9-   49 45       EOR   #$45
13DB-   51 55       EOR   ($55),Y
13DD-   41 54       EOR   ($54,X)
13DF-   45 2E       EOR   $2E
13E1-   54          ???
13E2-   45 58       EOR   $58
13E4-   54          ???
13E5-   00          BRK
13E6-   EA          NOP
13E7-   A4 00       LDY   $00
13E9-   02          ???
13EA-   BB          ???
13EB-   AE 00 00    LDX   $0000
13EE-   06 00       ASL   $00
13F0-   00          BRK
13F1-   00          BRK
13F2-   07          ???
13F3-   52          ???
13F4-   41 4D       EOR   ($4D,X)
13F6-   53          ???
13F7-   52          ???
13F8-   43          ???
13F9-   45 00       EOR   $00
13FB-   08          PHP
13FC-   4D 00 00    EOR   $0000
13FF-   00          BRK

                    --- End of Pascal Directory

------------------------------------------------------
```

Updated June 8, 2015

```
SYSTEM.INTERP (Wizardry IV)

This code gets relocated to $D000..$FFFF.  Uses Bank 2.
  (Note, 200.CHARSET loaded to $D000..$DFFF uses Bank 1.)

*1400LLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLLL

[$D000]  INTERP                          $E7FE - D000 + 1400

  Before call:

    F3    = Boot slot ($C6)
    478   = Current Track
    47E   = S6D1, Current Track
    FA    = 0
    FC75  = 0 [4075]
    FC67  = Cx  (boot slot)
    F6.F7 = #$BCDE
    FB    = Slot# * 16 ($60)

                                Set up $2D..$35 to read last 4 sectors of
[$D000]                           SYSTEM.INTERP to $BC00.$BFFF.
1400-   A2 00       LDX   #$00
1402-   A9 04       LDA   #$04
1404-   85 33       STA   $33   Sector count := 4
1406-   86 32       STX   $32   $32 := 0

1408-   A5 F4       LDA   $F4   $34.35 := Block#
140A-   85 34       STA   $34             (last 4 sectors of SYSTEM.INTERP)
140C-   A5 F5       LDA   $F5
140E-   85 35       STA   $35

1410-   A9 BC       LDA   #$BC  $30.31 := BC00  I/O Buffer
1412-   85 31       STA   $31
1414-   86 30       STX   $30

1416-   A9 04       LDA   #$04  Device 4 is Slot 6, Drive 1 (usually)
1418-   85 2D       STA   $2D   $2D := 4
141A-   A9 FF       LDA   #$FF
141C-   85 F2       STA   $F2   $F2 "Virtual Disk" flag := #$FF

141E-   20 FE E7    JSR   $E7FE  [$2BFE]  Read last 4 sectors of INTERP to $BC00.

1421-   4C 00 BC    JMP   $BC00 Start last part of INTERP at $BC00
--------------------------------------------------------------------------------
                                    Block Transfer for Cache code (see $EAD8)

                                    $C8.C9  Source
                                    $CA.CB  Destination
                                    $CC.CD  Length

                                    Source and Destination usually overlap

[$D024]
1424-   A6 CD       LDX   $CD     X := $CD (# of full sectors)
1426-   F0 0B       BEQ   $1433
1428-   18          CLC           $C9 := $C9 + $CD
1429-   8A          TXA
142A-   65 C9       ADC   $C9
142C-   85 C9       STA   $C9
142E-   8A          TXA           $CB := $CB + $CD + Carry(?!)
142F-   65 CB       ADC   $CB
1431-   85 CB       STA   $CB
```

```
1433-   E8          INX           X := X + 1
1434-   A4 CC       LDY   $CC     Y := partial sector length
1436-   D0 0B       BNE   $1443   Partial? Yes, $D043
1438-   F0 10       BEQ   $144A             No,  $D04A


                                  Transfer bytes starting from the end

                                  OuterLoop:
143A-   C6 C9       DEC   $C9
143C-   C6 CB       DEC   $CB
143E-   88          DEY
                                    loop:
143F-   B1 C8       LDA   ($C8),Y
1441-   91 CA       STA   ($CA),Y

[$D043]
1443-   88          DEY
1444-   D0 F9       BNE   $143F      BNE loop
1446-   B1 C8       LDA   ($C8),Y
1448-   91 CA       STA   ($CA),Y

[$D04A]
144A-   CA          DEX
144B-   D0 ED       BNE   $143A   BNE OuterLoop
144D-   60          RTS
-------------------------------------------------------------------
144E-   A6 CD       LDX   $CD
1450-   E8          INX
1451-   A4 CC       LDY   $CC
1453-   F0 25       BEQ   $147A
1455-   98          TYA
1456-   18          CLC
1457-   65 C8       ADC   $C8
1459-   85 C8       STA   $C8
145B-   B0 02       BCS   $145F
145D-   C6 C9       DEC   $C9
145F-   18          CLC
1460-   98          TYA
1461-   65 CA       ADC   $CA
1463-   85 CA       STA   $CA
1465-   B0 03       BCS   $146A
1467-   C6 CB       DEC   $CB
1469-   38          SEC
146A-   A9 00       LDA   #$00
146C-   E5 CC       SBC   $CC
146E-   A8          TAY
146F-   B1 C8       LDA   ($C8),Y
1471-   91 CA       STA   ($CA),Y
1473-   C8          INY
1474-   D0 F9       BNE   $146F
1476-   E6 C9       INC   $C9
1478-   E6 CB       INC   $CB
147A-   CA          DEX
147B-   D0 F2       BNE   $146F
147D-   60          RTS
147E-   A0 01       LDY   #$01
1480-   B1 9E       LDA   ($9E),Y
1482-   85 CC       STA   $CC
1484-   E6 9E       INC   $9E
1486-   D0 02       BNE   $148A
1488-   E6 9F       INC   $9F
148A-   A5 BC       LDA   $BC
```

```
148C-   85 00       STA   $00
148E-   A5 BD       LDA   $BD
1490-   85 01       STA   $01
1492-   B1 00       LDA   ($00),Y
1494-   AA          TAX
1495-   88          DEY
1496-   B1 00       LDA   ($00),Y
1498-   85 00       STA   $00
149A-   86 01       STX   $01
149C-   C8          INY
149D-   C6 CC       DEC   $CC
149F-   D0 F1       BNE   $1492
14A1-   A8          TAY
14A2-   60          RTS
14A3-   A0 00       LDY   #$00
14A5-   E6 9E       INC   $9E
14A7-   D0 02       BNE   $14AB
14A9-   E6 9F       INC   $9F
14AB-   B1 9E       LDA   ($9E),Y
14AD-   0A          ASL
14AE-   B0 03       BCS   $14B3
14B0-   A2 00       LDX   #$00
14B2-   60          RTS
14B3-   AA          TAX
14B4-   E6 9E       INC   $9E
14B6-   D0 02       BNE   $14BA
14B8-   E6 9F       INC   $9F
14BA-   B1 9E       LDA   ($9E),Y
14BC-   0A          ASL
14BD-   90 01       BCC   $14C0
14BF-   E8          INX
14C0-   60          RTS
14C1-   86 C9       STX   $C9
14C3-   84 C8       STY   $C8
14C5-   A2 00       LDX   #$00
14C7-   A0 01       LDY   #$01
14C9-   B1 9E       LDA   ($9E),Y
14CB-   0A          ASL
14CC-   90 0E       BCC   $14DC
14CE-   AA          TAX
14CF-   E6 9E       INC   $9E
14D1-   D0 02       BNE   $14D5
14D3-   E6 9F       INC   $9F
14D5-   B1 9E       LDA   ($9E),Y
14D7-   0A          ASL
14D8-   90 02       BCC   $14DC
14DA-   E8          INX
14DB-   18          CLC
14DC-   65 C8       ADC   $C8
14DE-   85 C8       STA   $C8
14E0-   8A          TXA
14E1-   65 C9       ADC   $C9
14E3-   85 C9       STA   $C9
14E5-   60          RTS
        ------------------------------------------------
                                  pCode=$C7 LDCI
                                            (same)
[$D0E6]
14E6-   A0 02       LDY   #$02
14E8-   B1 9E       LDA   ($9E),Y  $9E.9F === IPCPTR
14EA-   48          PHA
14EB-   88          DEY
14EC-   B1 9E       LDA   ($9E),Y
14EE-   48          PHA
```

```
14EF-   38          SEC
14F0-   24 18       BIT   $18

[$D0F1]
   14F1    18       CLC

14F2-   A9 02       LDA   #$02
14F4-   65 9E       ADC   $9E     $9E.9F === IPCPTR
14F6-   85 9E       STA   $9E
14F8-   90 02       BCC   $14FC
14FA-   E6 9F       INC   $9F
14FC-   4C 9D 00    JMP   $009D
-------------------------------------------------

                                  pCode=$A1 FJP
                                           (same)
[$D0FF]
14FF-   68          PLA
1500-   4A          LSR
1501-   68          PLA
1502-   B0 ED       BCS   $14F1
                                  Note fall through
-------------------------------------------------
                                  pCode=$B9 UJP
                                           (same)
[$D104]
1504-   A0 01       LDY   #$01
1506-   18          CLC
1507-   B1 9E       LDA   ($9E),Y
1509-   30 04       BMI   $150F
150B-   69 02       ADC   #$02
150D-   90 E5       BCC   $14F4   positive SB for UJP

150F-   A8          TAY           negative SB for UJP
1510-   65 B8       ADC   $B8     $B8.B9 === JTBPTR
1512-   AA          TAX
1513-   A9 FF       LDA   #$FF
1515-   65 B9       ADC   $B9
1517-   48          PHA
1518-   38          SEC
1519-   8A          TXA
151A-   F1 BA       SBC   ($BA),Y
151C-   85 9E       STA   $9E
151E-   C8          INY
151F-   68          PLA
1520-   F1 BA       SBC   ($BA),Y
1522-   85 9F       STA   $9F
1524-   4C 9D 00    JMP   $009D    $9D === EXECPC
-------------------------------------------------
                                  pCode=$AC  XJP
[$D127]
1527-   A0 01       LDY   #$01
1529-   98          TYA
152A-   05 9E       ORA   $9E    Insure P-code ptr
152C-   85 9E       STA   $9E
152E-   38          SEC
152F-   68          PLA          Extract case value
1530-   85 C6       STA   $C6
1532-   F1 9E       SBC   ($9E),Y
1534-   AA          TAX
1535-   C8          INY
1536-   68          PLA
1537-   85 C7       STA   $C7
```

```
1539-  F1 9E     SBC  ($9E),Y
153B-  10 07     BPL  $1544
153D-  70 07     BVS  $1546
153F-  18        CLC
1540-  A9 05     LDA  #$05
1542-  90 B0     BCC  $14F4
1544-  70 F9     BVS  $153F
1546-  85 CD     STA  $CD
1548-  C8        INY
1549-  B1 9E     LDA  ($9E),Y
154B-  C5 C6     CMP  $C6
154D-  C8        INY
154E-  B1 9E     LDA  ($9E),Y
1550-  E5 C7     SBC  $C7
1552-  10 04     BPL  $1558
1554-  70 04     BVS  $155A
1556-  50 E7     BVC  $153F
1558-  70 E5     BVS  $153F
155A-  8A        TXA
155B-  0A        ASL
155C-  26 CD     ROL  $CD
155E-  69 07     ADC  #$07
1560-  90 03     BCC  $1565
1562-  E6 CD     INC  $CD
1564-  18        CLC
1565-  65 9E     ADC  $9E
1567-  85 C8     STA  $C8
1569-  A5 CD     LDA  $CD
156B-  65 9F     ADC  $9F
156D-  85 C9     STA  $C9
156F-  A0 00     LDY  #$00
1571-  38        SEC
1572-  A5 C8     LDA  $C8
1574-  F1 C8     SBC  ($C8),Y
1576-  85 9E     STA  $9E
1578-  A5 C9     LDA  $C9
157A-  C8        INY
157B-  F1 C8     SBC  ($C8),Y
157D-  85 9F     STA  $9F
157F-  4C 9D 00  JMP  $009D

                                pCode=$D8 SLDL 1
                                pCode=$D9 SLDL 2
                                pCode=$DA SLDL 3
                                pCode=$DB SLDL 4
                                pCode=$DC SLDL 5
                                pCode=$DD SLDL 6
                                pCode=$DE SLDL 7
                                pCode=$DF SLDL 8
                                pCode=$E0 SLDL 9
                                pCode=$E1 SLDL 10
                                pCode=$E2 SLDL 11
                                pCode=$E3 SLDL 12
                                pCode=$E4 SLDL 13
                                pCode=$E5 SLDL 14
                                pCode=$E6 SLDL 15
                                pCode=$E7 SLDL 16
                                          (same)
[$D182]
1582-  BC 10 FE  LDY  $FE10,X
1585-  B1 BC     LDA  ($BC),Y
1587-  48        PHA
1588-  88        DEY
1589-  B1 BC     LDA  ($BC),Y
```

```
158B-   4C 96 00    JMP   $0096

                                  pCode=$E8 SLDO 1
                                  pCode=$E9 SLDO 2
                                  pCode=$EA SLDO 3
                                  pCode=$EB SLDO 4
                                  pCode=$EC SLDO 5
                                  pCode=$ED SLDO 6
                                  pCode=$EE SLDO 7
                                  pCode=$EF SLDO 8
                                  pCode=$F0 SLDO 9
                                  pCode=$F1 SLDO 10
                                  pCode=$F2 SLDO 11
                                  pCode=$F3 SLDO 12
                                  pCode=$F4 SLDO 13
                                  pCode=$F5 SLDO 14
                                  pCode=$F6 SLDO 15
                                  pCode=$F7 SLDO 16
[$D18E]
158E-   BC 00 FE    LDY   $FE00,X
1591-   B1 B2       LDA   ($B2),Y  $B2 === MSPTR
1593-   48          PHA
1594-   88          DEY
1595-   B1 B2       LDA   ($B2),Y
1597-   4C 96 00    JMP   $0096

                                  pCode=$F8 SIND 0
                                  pCode=$F9 SIND 1
                                  pCode=$FA SIND 2
                                  pCode=$FB SIND 3
                                  pCode=$FC SIND 4
                                  pCode=$FD SIND 5
                                  pCode=$FE SIND 6
                                  pCode=$FF SIND 7
[$D19A]
159A-   BC 00 FE    LDY   $FE00,X
159D-   68          PLA
159E-   85 C8       STA   $C8
15A0-   68          PLA
15A1-   85 C9       STA   $C9
15A3-   B1 C8       LDA   ($C8),Y
15A5-   48          PHA
15A6-   88          DEY
15A7-   B1 C8       LDA   ($C8),Y
15A9-   4C 96 00    JMP   $0096

                                  pCode=$A9 LDO
[$D1AC]
15AC-   A4 B2       LDY   $B2
15AE-   A6 B3       LDX   $B3
15B0-   D0 0A       BNE   $15BC

                                  pCode=$B6 LOD
[$D1B2]
15B2-   20 7E D0    JSR   $D07E
15B5-   4C BC D1    JMP   $D1BC

                                  pCode=$CA LDL
[$D1B8]
15B8-   A6 BD       LDX   $BD
15BA-   A4 BC       LDY   $BC
15BC-   20 C1 D0    JSR   $D0C1
15BF-   A0 0B       LDY   #$0B
15C1-   B1 C8       LDA   ($C8),Y
```

```
15C3-   48          PHA
15C4-   88          DEY
15C5-   B1 C8       LDA   ($C8),Y
15C7-   48          PHA
15C8-   4C F1 D0    JMP   $D0F1

                                pCode=$B2 LDA
[$D1CB]
15CB-   20 7E D0    JSR   $D07E
15CE-   D0 0A       BNE   $15DA

                                pCode=$A5 LAO
[$D1D0]
15D0-   A4 B2       LDY   $B2
15D2-   A6 B3       LDX   $B3
15D4-   D0 04       BNE   $15DA

                                pCode=$C6 LLA
[$D1D6]
15D6-   A4 BC       LDY   $BC
15D8-   A6 BD       LDX   $BD
15DA-   98          TYA
15DB-   18          CLC
15DC-   69 0A       ADC   #$0A
15DE-   90 01       BCC   $15E1
15E0-   E8          INX
15E1-   A8          TAY
15E2-   20 C1 D0    JSR   $D0C1
15E5-   48          PHA
15E6-   A5 C8       LDA   $C8
15E8-   48          PHA
15E9-   4C F1 D0    JMP   $D0F1
                                pCode=$AB SRO
[$D1EC]
15EC-   A4 B2       LDY   $B2
15EE-   A6 B3       LDX   $B3
15F0-   D0 0A       BNE   $15FC

                                pCode=$B8 STR
[$D1F2]
15F2-   20 7E D0    JSR   $D07E
15F5-   4C FC D1    JMP   $D1FC

                                pCode=$CC STL
[$D1F8]
15F8-   A6 BD       LDX   $BD
15FA-   A4 BC       LDY   $BC
15FC-   20 C1 D0    JSR   $D0C1
15FF-   A0 0A       LDY   #$0A
1601-   68          PLA
1602-   91 C8       STA   ($C8),Y
1604-   C8          INY
1605-   68          PLA
1606-   91 C8       STA   ($C8),Y
1608-   4C F1 D0    JMP   $D0F1

                                pCode=$A3 IND
[$D20B]
160B-   68          PLA
160C-   A8          TAY
160D-   68          PLA
160E-   AA          TAX
160F-   20 C1 D0    JSR   $D0C1
1612-   A0 01       LDY   #$01
```

```
1614-   B1 C8       LDA   ($C8),Y
1616-   48          PHA
1617-   88          DEY
1618-   B1 C8       LDA   ($C8),Y
161A-   48          PHA
161B-   4C F1 D0    JMP   $D0F1

                                pCode=$9A STO
[$D21E]
161E-   68          PLA
161F-   A8          TAY
1620-   68          PLA
1621-   AA          TAX
1622-   68          PLA
1623-   85 C8       STA   $C8
1625-   68          PLA
1626-   85 C9       STA   $C9
1628-   98          TYA
1629-   A0 00       LDY   #$00
162B-   91 C8       STA   ($C8),Y
162D-   8A          TXA
162E-   C8          INY
162F-   91 C8       STA   ($C8),Y
1631-   4C 97 00    JMP   $0097

                                pCode=$BE LDB
[$D234]
1634-   BA          TSX
1635-   18          CLC
1636-   68          PLA
1637-   7D 03 01    ADC   $0103,X
163A-   85 C8       STA   $C8
163C-   68          PLA
163D-   7D 04 01    ADC   $0104,X
1640-   85 C9       STA   $C9
1642-   68          PLA
1643-   68          PLA
1644-   A0 00       LDY   #$00
1646-   98          TYA
1647-   48          PHA
1648-   B1 C8       LDA   ($C8),Y
164A-   4C 96 00    JMP   $0096

                                pCode=$BF STB
[$D24D]
164D-   BA          TSX
164E-   18          CLC
164F-   BD 03 01    LDA   $0103,X
1652-   7D 05 01    ADC   $0105,X
1655-   85 C8       STA   $C8
1657-   BD 04 01    LDA   $0104,X
165A-   7D 06 01    ADC   $0106,X
165D-   85 C9       STA   $C9
165F-   68          PLA
1660-   A0 00       LDY   #$00
1662-   91 C8       STA   ($C8),Y
1664-   8A          TXA
1665-   69 06       ADC   #$06
1667-   AA          TAX
1668-   9A          TXS
1669-   4C 97 00    JMP   $0097

                                pCode=$B3 LDC
[$D26C]
```

```
166C-  A0 01       LDY   #$01
166E-  B1 9E       LDA   ($9E),Y
1670-  F0 15       BEQ   $1687
1672-  AA          TAX
1673-  A5 9E       LDA   $9E
1675-  29 01       AND   #$01
1677-  F0 01       BEQ   $167A
1679-  C8          INY
167A-  C8          INY
167B-  C8          INY
167C-  B1 9E       LDA   ($9E),Y
167E-  48          PHA
167F-  88          DEY
1680-  B1 9E       LDA   ($9E),Y
1682-  48          PHA
1683-  C8          INY
1684-  CA          DEX
1685-  D0 F3       BNE   $167A
1687-  98          TYA
1688-  38          SEC
1689-  4C F4 D0    JMP   $D0F4

                                pCode=$BC LDM
[$D28C]
168C-  68          PLA
168D-  85 C8       STA   $C8
168F-  68          PLA
1690-  85 C9       STA   $C9
1692-  A0 01       LDY   #$01
1694-  B1 9E       LDA   ($9E),Y
1696-  F0 0C       BEQ   $16A4
1698-  0A          ASL
1699-  A8          TAY
169A-  88          DEY
169B-  B1 C8       LDA   ($C8),Y
169D-  48          PHA
169E-  88          DEY
169F-  D0 FA       BNE   $169B
16A1-  B1 C8       LDA   ($C8),Y
16A3-  48          PHA
16A4-  4C F1 D0    JMP   $D0F1

                                pCode=$BD STM
[$D2A7]
16A7-  A0 01       LDY   #$01
16A9-  B1 9E       LDA   ($9E),Y
16AB-  F0 1C       BEQ   $16C9
16AD-  0A          ASL
16AE-  85 CC       STA   $CC
16B0-  BA          TSX
16B1-  8A          TXA
16B2-  65 CC       ADC   $CC
16B4-  AA          TAX
16B5-  BD 01 01    LDA   $0101,X
16B8-  85 C8       STA   $C8
16BA-  BD 02 01    LDA   $0102,X
16BD-  85 C9       STA   $C9
16BF-  A6 CC       LDX   $CC
16C1-  88          DEY
16C2-  68          PLA
16C3-  91 C8       STA   ($C8),Y
16C5-  C8          INY
16C6-  CA          DEX
16C7-  D0 F9       BNE   $16C2
```

```
16C9-   68          PLA
16CA-   68          PLA
16CB-   4C F1 D0    JMP   $D0F1

                                pCode=$A6 LSA
[$D2CE]
16CE-   E6 9E       INC   $9E
16D0-   D0 02       BNE   $16D4
16D2-   E6 9F       INC   $9F
16D4-   A5 9F       LDA   $9F
16D6-   48          PHA
16D7-   A5 9E       LDA   $9E
16D9-   48          PHA
16DA-   A0 00       LDY   #$00
16DC-   B1 9E       LDA   ($9E),Y
16DE-   38          SEC
16DF-   4C F4 D0    JMP   $D0F4

                                pCode=$AA SAS
[$D2E2]
16E2-   68          PLA
16E3-   85 C8       STA   $C8
16E5-   68          PLA
16E6-   85 C9       STA   $C9
16E8-   68          PLA
16E9-   85 CA       STA   $CA
16EB-   68          PLA
16EC-   85 CB       STA   $CB
16EE-   A0 01       LDY   #$01
16F0-   B1 9E       LDA   ($9E),Y
16F2-   A6 C9       LDX   $C9
16F4-   D0 0E       BNE   $1704
16F6-   AA          TAX
16F7-   F0 21       BEQ   $171A
16F9-   A5 C8       LDA   $C8
16FB-   91 CA       STA   ($CA),Y
16FD-   98          TYA
16FE-   88          DEY
16FF-   91 CA       STA   ($CA),Y
1701-   4C F1 D0    JMP   $D0F1

1704-   88          DEY
1705-   D1 C8       CMP   ($C8),Y
1707-   90 11       BCC   $171A
1709-   B1 C8       LDA   ($C8),Y
170B-   91 CA       STA   ($CA),Y
170D-   F0 08       BEQ   $1717
170F-   A8          TAY
1710-   B1 C8       LDA   ($C8),Y
1712-   91 CA       STA   ($CA),Y
1714-   88          DEY
1715-   D0 F9       BNE   $1710
1717-   4C F1 D0    JMP   $D0F1
171A-   A0 00       LDY   #$00
171C-   B1 C8       LDA   ($C8),Y
171E-   48          PHA
171F-   C8          INY
1720-   B1 9E       LDA   ($9E),Y
1722-   A8          TAY
1723-   68          PLA
1724-   A2 0D       LDX   #$0D
1726-   4C 00 FE    JMP   $FE00

                                pCode=$9B IXS
```

```
[$D329]
1729-   BA          TSX
172A-   BD 02 01    LDA   $0102,X
172D-   D0 1A       BNE   $1749
172F-   BD 01 01    LDA   $0101,X
1732-   F0 15       BEQ   $1749
1734-   BC 03 01    LDY   $0103,X
1737-   84 C8       STY   $C8
1739-   BC 04 01    LDY   $0104,X
173C-   84 C9       STY   $C9
173E-   A0 00       LDY   #$00
1740-   D1 C8       CMP   ($C8),Y
1742-   90 02       BCC   $1746
1744-   D0 03       BNE   $1749
1746-   4C 97 00    JMP   $0097
1749-   A0 00       LDY   #$00
174B-   B1 C8       LDA   ($C8),Y
174D-   A8          TAY
174E-   BD 01 01    LDA   $0101,X
1751-   A2 01       LDX   #$01
1753-   4C 00 FE    JMP   $FE00

                                pCode=$A8 MOV
[$D356]
1756-   68          PLA
1757-   85 C8       STA   $C8
1759-   68          PLA
175A-   85 C9       STA   $C9
175C-   68          PLA
175D-   85 CA       STA   $CA
175F-   68          PLA
1760-   85 CB       STA   $CB
1762-   20 A3 D0    JSR   $D0A3
1765-   85 CC       STA   $CC
1767-   20 50 D0    JSR   $D050
176A-   4C 97 00    JMP   $0097

                                pCode=$A2 INC
[$D36D]
176D-   68          PLA
176E-   A8          TAY
176F-   68          PLA
1770-   AA          TAX
1771-   20 C1 D0    JSR   $D0C1
1774-   48          PHA
1775-   A5 C8       LDA   $C8
1777-   48          PHA
1778-   4C F1 D0    JMP   $D0F1

                                pCode=$A4 IXA
[$D37B]
177B-   20 A3 D0    JSR   $D0A3
177E-   C9 02       CMP   #$02
1780-   D0 0F       BNE   $1791
1782-   E0 00       CPX   #$00
1784-   D0 0B       BNE   $1791
1786-   68          PLA
1787-   0A          ASL
1788-   85 16       STA   $16
178A-   68          PLA
178B-   2A          ROL
178C-   85 17       STA   $17
178E-   4C 9E D3    JMP   $D39E
1791-   85 12       STA   $12
```

```
1793-  86 13     STX  $13
1795-  68        PLA
1796-  85 14     STA  $14
1798-  68        PLA
1799-  85 15     STA  $15
179B-  20 A9 DB  JSR  $DBA9
179E-  18        CLC
179F-  68        PLA
17A0-  65 16     ADC  $16
17A2-  AA        TAX
17A3-  68        PLA
17A4-  65 17     ADC  $17
17A6-  4C 94 00  JMP  $0094

                        pCode=$D0 LPA
[$D3A9]
17A9-  A0 01     LDY  #$01
17AB-  B1 9E     LDA  ($9E),Y
17AD-  A8        TAY
17AE-  A5 9E     LDA  $9E
17B0-  18        CLC
17B1-  69 02     ADC  #$02
17B3-  85 9E     STA  $9E
17B5-  A5 9F     LDA  $9F
17B7-  69 00     ADC  #$00
17B9-  85 9F     STA  $9F
17BB-  48        PHA
17BC-  A5 9E     LDA  $9E
17BE-  48        PHA
17BF-  98        TYA
17C0-  18        CLC
17C1-  4C F4 D0  JMP  $D0F4

                        pCode=$C0 IXP
[$D3C4]
17C4-  68        PLA
17C5-  85 0E     STA  $0E
17C7-  68        PLA
17C8-  85 0F     STA  $0F
17CA-  A0 01     LDY  #$01
17CC-  B1 9E     LDA  ($9E),Y
17CE-  85 0C     STA  $0C
17D0-  88        DEY
17D1-  84 0D     STY  $0D
17D3-  20 38 DC  JSR  $DC38
17D6-  06 0E     ASL  $0E
17D8-  26 0F     ROL  $0F
17DA-  68        PLA
17DB-  65 0E     ADC  $0E
17DD-  A8        TAY
17DE-  68        PLA
17DF-  65 0F     ADC  $0F
17E1-  48        PHA
17E2-  98        TYA
17E3-  48        PHA
17E4-  A9 00     LDA  #$00
17E6-  48        PHA
17E7-  A0 02     LDY  #$02
17E9-  B1 9E     LDA  ($9E),Y
17EB-  48        PHA
17EC-  AA        TAX
17ED-  A9 00     LDA  #$00
17EF-  48        PHA
17F0-  65 10     ADC  $10
```

```
17F2-  CA          DEX
17F3-  D0 FB       BNE   $17F0
17F5-  48          PHA
17F6-  4C EF D0    JMP   $D0EF

                                pCode=$BA LDP
[$D3F9]
17F9-  68          PLA
17FA-  C9 08       CMP   #$08
17FC-  90 02       BCC   $1800
17FE-  29 07       AND   #$07
1800-  85 22       STA   $22
1802-  68          PLA
1803-  68          PLA
1804-  AA          TAX
1805-  68          PLA
1806-  68          PLA
1807-  85 C8       STA   $C8
1809-  68          PLA
180A-  85 C9       STA   $C9
180C-  A0 01       LDY   #$01
180E-  B1 C8       LDA   ($C8),Y
1810-  B0 05       BCS   $1817
1812-  85 21       STA   $21
1814-  88          DEY
1815-  B1 C8       LDA   ($C8),Y
1817-  A4 22       LDY   $22
1819-  F0 06       BEQ   $1821
181B-  46 21       LSR   $21
181D-  6A          ROR
181E-  88          DEY
181F-  D0 FA       BNE   $181B
1821-  A8          TAY
1822-  A5 21       LDA   $21
1824-  3D F8 FA    AND   $FAF8,X
1827-  48          PHA
1828-  98          TYA
1829-  3D 00 FB    AND   $FB00,X
182C-  4C 96 00    JMP   $0096

                                pCode=$BB STP
[$D42F]
182F-  68          PLA
1830-  85 20       STA   $20
1832-  68          PLA
1833-  85 21       STA   $21
1835-  68          PLA
1836-  A8          TAY
1837-  85 23       STA   $23
1839-  68          PLA
183A-  68          PLA
183B-  18          CLC
183C-  65 23       ADC   $23
183E-  AA          TAX
183F-  68          PLA
1840-  68          PLA
1841-  85 C8       STA   $C8
1843-  68          PLA
1844-  85 C9       STA   $C9
1846-  B9 00 FB    LDA   $FB00,Y
1849-  1D 10 FB    ORA   $FB10,X
184C-  85 D2       STA   $D2
184E-  B9 F8 FA    LDA   $FAF8,Y
1851-  1D 08 FB    ORA   $FB08,X
```

```
1854-   85 D3       STA   $D3
1856-   A5 20       LDA   $20
1858-   C0 08       CPY   #$08
185A-   90 06       BCC   $1862
185C-   85 21       STA   $21
185E-   98          TYA
185F-   29 07       AND   #$07
1861-   A8          TAY
1862-   C0 00       CPY   #$00
1864-   F0 06       BEQ   $186C
1866-   0A          ASL
1867-   26 21       ROL   $21
1869-   88          DEY
186A-   D0 FA       BNE   $1866
186C-   85 20       STA   $20
186E-   51 C8       EOR   ($C8),Y
1870-   25 D2       AND   $D2
1872-   45 20       EOR   $20
1874-   91 C8       STA   ($C8),Y
1876-   C8          INY
1877-   B1 C8       LDA   ($C8),Y
1879-   45 21       EOR   $21
187B-   25 D3       AND   $D3
187D-   45 21       EOR   $21
187F-   91 C8       STA   ($C8),Y
1881-   4C 97 00    JMP   $0097

[$D484]
1884-   A6 B7       LDX   $B7
1886-   A5 B6       LDA   $B6

                              After reading in a segment
                              A.X --> Segment Dictionary in memory
[$D488]
1888-   38          SEC
1889-   A0 01       LDY   #$01        Get UB2 (Procedure #)
188B-   F1 9E       SBC   ($9E),Y        e.g., from $BFFF, first time= 1
188D-   B0 02       BCS   $1891       Use UB2 to calculate address
188F-   CA          DEX                 of relative procedure ptr
1890-   38          SEC
1891-   F1 9E       SBC   ($9E),Y
1893-   B0 02       BCS   $1897
1895-   CA          DEX
1896-   38          SEC
1897-   86 C9       STX   $C9         e.g., C8.C9 := #$BFEA
1899-   85 C8       STA   $C8         Address of relative procedure ptr
                                        for procedure UB1.
                              A.X is same (e.g., #$BFEA)

189B-   88          DEY               Y := 0
189C-   F1 C8       SBC   ($C8),Y     BA.BB := ptr to procedure code section
189E-   85 BA       STA   $BA              biased by #$100 (256)
18A0-   8A          TXA
18A1-   C8          INY               Procedure code section is at
18A2-   F1 C8       SBC   ($C8),Y        e.g., $BF3A
18A4-   E9 01       SBC   #$01
18A6-   85 BB       STA   $BB         e.g., BA.BB := $BE3A

18A8-   A0 F8       LDY   #$F8        #$F8 == $100 - $F8 = 8 === DataSize index
18AA-   A5 BE       LDA   $BE           e.g., BE = $88; BF = $A3
18AC-   F1 BA       SBC   ($BA),Y     Adjust KP;
18AE-   48          PHA               Expand for DATASIZE for this proc
18AF-   C8          INY
18B0-   A5 BF       LDA   $BF
```

```
18B2-  F1 BA     SBC   ($BA),Y
18B4-  AA        TAX
18B5-  C8        INY             Expand for PARAMETER SIZE
18B6-  68        PLA
18B7-  F1 BA     SBC   ($BA),Y
18B9-  B0 02     BCS   $18BD
18BB-  CA        DEX
18BC-  38        SEC
18BD-  E9 0E     SBC   #$0E      Expand another 14 bytes
18BF-  B0 01     BCS   $18C2
18C1-  CA        DEX
18C2-  86 05     STX   $05       e.g., 04.05 := #$9146
18C4-  85 04     STA   $04

18C6-  B1 BA     LDA   ($BA),Y
18C8-  F0 16     BEQ   $18E0

                                 ------------------------------
18CA-  AA        TAX             Pull JSR return off stack
18CB-  68        PLA             Save it in $00.$01
18CC-  85 00     STA   $00
18CE-  68        PLA
18CF-  85 01     STA   $01

18D1-  A0 0E     LDY   #$0E      Pull params off stack
18D3-  68        PLA
18D4-  91 04     STA   ($04),Y   Push them into new MARKSTACK
18D6-  C8        INY             (Pass them to Procedure)
18D7-  CA        DEX
18D8-  D0 F9     BNE   $18D3

18DA-  A5 01     LDA   $01       Restore JSR to stack
18DC-  48        PHA
18DD-  A5 00     LDA   $00
18DF-  48        PHA
                                 ------------------------------
18E0-  A0 00     LDY   #$00      Save old KP  (BE.BF)
18E2-  A5 BE     LDA   $BE
18E4-  91 04     STA   ($04),Y     after new Markstack
18E6-  C8        INY
18E7-  A5 BF     LDA   $BF
18E9-  91 04     STA   ($04),Y

18EB-  A0 04     LDY   #$04      Save BASE    (BC.BD)  (?)
18ED-  A5 BC     LDA   $BC         e.g., $BFF0
18EF-  91 04     STA   ($04),Y
18F1-  C8        INY
18F2-  A5 BD     LDA   $BD
18F4-  91 04     STA   ($04),Y

18F6-  C8        INY             Save JTAB     (B8.B9)
18F7-  A5 B8     LDA   $B8         e.g., $BC91
18F9-  91 04     STA   ($04),Y
18FB-  C8        INY
18FC-  A5 B9     LDA   $B9
18FE-  91 04     STA   ($04),Y

1900-  C8        INY             Save SEG ptr  (B6.B7)
1901-  A5 B6     LDA   $B6         e.g., $BC91
1903-  91 04     STA   ($04),Y
1905-  C8        INY
1906-  A5 B7     LDA   $B7
1908-  91 04     STA   ($04),Y
```

```
190A-  C8          INY             Save old IPC+2 (9E.9F)  (?)
190B-  18          CLC               Return point
190C-  A5 9E       LDA  $9E          e.g., $C000
190E-  69 02       ADC  #$02
1910-  91 04       STA  ($04),Y
1912-  C8          INY
1913-  A5 9F       LDA  $9F
1915-  69 00       ADC  #$00
1917-  91 04       STA  ($04),Y

1919-  C8          INY             Save SP
191A-  BA          TSX
191B-  E8          INX                (previous to $D488 call)
191C-  E8          INX
191D-  8A          TXA
191E-  91 04       STA  ($04),Y

1920-  A6 05       LDX  $05        KP updated (BE.BF)
1922-  A5 04       LDA  $04
1924-  86 BF       STX  $BF
1926-  85 BE       STA  $BE
1928-  69 02       ADC  #$02
192A-  90 01       BCC  $192D
192C-  E8          INX
192D-  86 BD       STX  $BD        BC.BD updated (Markstack ptr)
192F-  85 BC       STA  $BC

1931-  A6 BB       LDX  $BB
1933-  E8          INX
1934-  A5 BA       LDA  $BA
1936-  85 B8       STA  $B8        B8.B9 updated
1938-  86 B9       STX  $B9

193A-  38          SEC             IPC (9E.9F) updated
193B-  E9 02       SBC  #$02
193D-  B0 02       BCS  $1941
193F-  CA          DEX
1940-  38          SEC
1941-  A0 FE       LDY  #$FE
1943-  F1 BA       SBC  ($BA),Y
1945-  85 9E       STA  $9E
1947-  8A          TXA
1948-  C8          INY
1949-  F1 BA       SBC  ($BA),Y
194B-  85 9F       STA  $9F

194D-  A0 01       LDY  #$01
194F-  38          SEC
1950-  A5 B4       LDA  $B4
1952-  E5 BE       SBC  $BE
1954-  A8          TAY
1955-  A5 B5       LDA  $B5
1957-  E5 BF       SBC  $BF
1959-  B0 03       BCS  $195E
195B-  A0 01       LDY  #$01
195D-  60          RTS

195E-  A2 04       LDX  #$04
1960-  4C 00 FE    JMP  $FE00
-------------------------------------------------------------------------------
                              CGP ($CF) Call Global Procedure
[$D563]
1963-  20 84 D4    JSR  $D484
1966-  A5 B3       LDA  $B3
```

```
1968-   91 BC       STA   ($BC),Y
196A-   88          DEY
196B-   A5 B2       LDA   $B2
196D-   91 BC       STA   ($BC),Y
196F-   4C 9D 00    JMP   $009D
--------------------------------------------------------------------------------
                                CLP ($CE) Call Local Procedure
[$D572]
1972-   20 84 D4    JSR   $D484
1975-   C8          INY
1976-   B1 BC       LDA   ($BC),Y
1978-   91 BE       STA   ($BE),Y
197A-   C8          INY
197B-   B1 BC       LDA   ($BC),Y
197D-   91 BE       STA   ($BE),Y
197F-   4C 9D 00    JMP   $009D
--------------------------------------------------------------------------------
                                CBP ($C2) Call Base Procedure
[$D582]
1982-   20 84 D4    JSR   $D484

[$D585]
1985-   B1 B2       LDA   ($B2),Y
1987-   91 BC       STA   ($BC),Y
1989-   88          DEY
198A-   B1 B2       LDA   ($B2),Y
198C-   91 BC       STA   ($BC),Y
198E-   A5 B3       LDA   $B3
1990-   48          PHA
1991-   A5 B2       LDA   $B2
1993-   48          PHA
1994-   A5 BC       LDA   $BC
1996-   85 B2       STA   $B2
1998-   A5 BD       LDA   $BD
199A-   85 B3       STA   $B3
199C-   4C 9D 00    JMP   $009D
--------------------------------------------------------------------------------
                                CXP ($CD) Call External Procedure
[$D59F]
199F-   E6 9E       INC   $9E       Increment IPCPTR (to UB1)
19A1-   D0 02       BNE   $19A5
19A3-   E6 9F       INC   $9F
19A5-   A0 00       LDY   #$00
19A7-   B1 9E       LDA   ($9E),Y   Get UB1 = Segment#
19A9-   20 14 DE    JSR   $DE14     JSR $DE14 Read Segment into memory
                                              if not there already.

19AC-   BD 8A 0C    LDA   $0C8A,X   $0A.0B := Procedure Dictionary
19AF-   85 0B       STA   $0B                 (e.g., #$BFEC)
19B1-   BD 7A 0C    LDA   $0C7A,X
19B4-   85 0A       STA   $0A
19B6-   A6 0B       LDX   $0B       A.X := Procedure Dictionary

19B8-   20 88 D4    JSR   $D488     JSR $D488  Build new MARKSTACK
19BB-   A5 0A       LDA   $0A
19BD-   85 B6       STA   $B6
19BF-   A5 0B       LDA   $0B
19C1-   85 B7       STA   $B7
19C3-   B1 B8       LDA   ($B8),Y        A := Lex Level
19C5-   F0 BE       BEQ   $1985 [$D585]  Into CBP code
19C7-   30 BC       BMI   $1985 [$D585]  Into CBP code
19C9-   10 05       BPL   $19D0 [$D5D0]  Into CIP code (Lex Level > 0)
--------------------------------------------------------------------------------
                                CIP ($AE) Call Intermediate Procedure
```

```
[$D5CB]
19CB-   20 84 D4    JSR   $D484
19CE-   B1 B8       LDA   ($B8),Y  A := Lex Level

                                   A == Lex Level
                                   BC.BD == New Markstack
[$D5D0]
19D0-   AA          TAX
19D1-   CA          DEX
19D2-   86 24       STX   $24      $24 := Lex Level - 1

19D4-   A5 BC       LDA   $BC      $08.09 := Base? or MP?  Not sure
19D6-   85 08       STA   $08
19D8-   A5 BD       LDA   $BD
19DA-   85 09       STA   $09

19DC-   A0 04       LDY   #$04     $02.03 := Caller's JTAB (?)
19DE-   B1 08       LDA   ($08),Y
19E0-   85 02       STA   $02
19E2-   C8          INY
19E3-   B1 08       LDA   ($08),Y
19E5-   85 03       STA   $03

19E7-   A0 02       LDY   #$02
19E9-   B1 08       LDA   ($08),Y
19EB-   AA          TAX
19EC-   C8          INY
19ED-   B1 08       LDA   ($08),Y
19EF-   85 09       STA   $09
19F1-   86 08       STX   $08
19F3-   A0 01       LDY   #$01
19F5-   B1 02       LDA   ($02),Y  Get Lex Level for this JTAB
19F7-   C5 24       CMP   $24
19F9-   D0 E1       BNE   $19DC    Loop until we get to previous lex lev

19FB-   A5 09       LDA   $09      "Use that activation record's
19FD-   91 BC       STA   ($BC),Y   Static Link as the Static Link of the
19FF-   88          DEY             new Markstack."
1A00-   8A          TXA
1A01-   91 BC       STA   ($BC),Y
1A03-   4C 9D 00    JMP   $009D
-------------------------------------------------------------------------------
                                  pCode=$C1 RBP
[$D606]
1A06-   A0 0A       LDY   #$0A
1A08-   B1 BC       LDA   ($BC),Y
1A0A-   AA          TAX
1A0B-   CA          DEX
1A0C-   CA          DEX
1A0D-   9A          TXS
1A0E-   68          PLA
1A0F-   85 B2       STA   $B2
1A11-   68          PLA
1A12-   85 B3       STA   $B3
1A14-   D0 06       BNE   $1A1C

                                  pCode=$AD RNP
[$D616]
1A16-   A0 0A       LDY   #$0A
1A18-   B1 BC       LDA   ($BC),Y
1A1A-   AA          TAX
1A1B-   9A          TXS
1A1C-   A0 01       LDY   #$01
1A1E-   B1 9E       LDA   ($9E),Y
```

```
1A20-  F0 09       BEQ   $1A2B
1A22-  A0 0D       LDY   #$0D
1A24-  B1 BC       LDA   ($BC),Y
1A26-  48          PHA
1A27-  88          DEY
1A28-  B1 BC       LDA   ($BC),Y
1A2A-  48          PHA
1A2B-  A0 04       LDY   #$04
1A2D-  B1 BC       LDA   ($BC),Y
1A2F-  85 B8       STA   $B8
1A31-  85 BA       STA   $BA
1A33-  C8          INY
1A34-  B1 BC       LDA   ($BC),Y
1A36-  85 B9       STA   $B9
1A38-  85 BB       STA   $BB
1A3A-  C6 BB       DEC   $BB
1A3C-  C8          INY
1A3D-  B1 BC       LDA   ($BC),Y
1A3F-  85 C8       STA   $C8
1A41-  C8          INY
1A42-  B1 BC       LDA   ($BC),Y
1A44-  85 C9       STA   $C9
1A46-  C8          INY
1A47-  B1 BC       LDA   ($BC),Y
1A49-  85 9E       STA   $9E
1A4B-  C8          INY
1A4C-  B1 BC       LDA   ($BC),Y
1A4E-  85 9F       STA   $9F
1A50-  A0 02       LDY   #$02
1A52-  B1 BC       LDA   ($BC),Y
1A54-  AA          TAX
1A55-  C8          INY
1A56-  B1 BC       LDA   ($BC),Y
1A58-  85 BD       STA   $BD
1A5A-  86 BC       STX   $BC
1A5C-  A0 00       LDY   #$00
1A5E-  B1 BE       LDA   ($BE),Y
1A60-  AA          TAX
1A61-  C8          INY
1A62-  B1 BE       LDA   ($BE),Y
1A64-  85 BF       STA   $BF
1A66-  86 BE       STX   $BE
1A68-  88          DEY
1A69-  B1 B6       LDA   ($B6),Y
1A6B-  D1 C8       CMP   ($C8),Y
1A6D-  F0 0B       BEQ   $1A7A
1A6F-  20 F9 DD    JSR   $DDF9
1A72-  A5 C8       LDA   $C8
1A74-  85 B6       STA   $B6
1A76-  A5 C9       LDA   $C9
1A78-  85 B7       STA   $B7
1A7A-  4C 9D 00    JMP   $009D

1A7D-  68          PLA
1A7E-  85 26       STA   $26
1A80-  68          PLA
1A81-  68          PLA
1A82-  85 25       STA   $25
1A84-  68          PLA
1A85-  A0 01       LDY   #$01
1A87-  B1 B8       LDA   ($B8),Y
1A89-  10 03       BPL   $1A8E
1A8B-  4C 3D DF    JMP   $DF3D
1A8E-  A6 B9       LDX   $B9
```

```
1A90-  A5 B8     LDA  $B8
1A92-  38        SEC
1A93-  E9 04     SBC  #$04
1A95-  B0 02     BCS  $1A99
1A97-  CA        DEX
1A98-  38        SEC
1A99-  A0 FC     LDY  #$FC
1A9B-  F1 BA     SBC  ($BA),Y
1A9D-  85 9E     STA  $9E
1A9F-  8A        TXA
1AA0-  C8        INY
1AA1-  F1 BA     SBC  ($BA),Y
1AA3-  85 9F     STA  $9F
1AA5-  A0 00     LDY  #$00
1AA7-  B1 B8     LDA  ($B8),Y
1AA9-  C5 26     CMP  $26
1AAB-  D0 06     BNE  $1AB3
1AAD-  B1 B6     LDA  ($B6),Y
1AAF-  C5 25     CMP  $25
1AB1-  F0 56     BEQ  $1B09
1AB3-  A5 BC     LDA  $BC
1AB5-  A6 BD     LDX  $BD
1AB7-  D0 08     BNE  $1AC1
1AB9-  A0 03     LDY  #$03
1ABB-  B1 C8     LDA  ($C8),Y
1ABD-  AA        TAX
1ABE-  88        DEY
1ABF-  B1 C8     LDA  ($C8),Y
1AC1-  85 C8     STA  $C8
1AC3-  86 C9     STX  $C9
1AC5-  A0 05     LDY  #$05
1AC7-  B1 C8     LDA  ($C8),Y
1AC9-  AA        TAX
1ACA-  88        DEY
1ACB-  B1 C8     LDA  ($C8),Y
1ACD-  38        SEC
1ACE-  E9 04     SBC  #$04
1AD0-  B0 02     BCS  $1AD4
1AD2-  CA        DEX
1AD3-  38        SEC
1AD4-  85 CA     STA  $CA
1AD6-  86 CB     STX  $CB
1AD8-  A0 00     LDY  #$00
1ADA-  F1 CA     SBC  ($CA),Y
1ADC-  A0 08     LDY  #$08
1ADE-  91 C8     STA  ($C8),Y
1AE0-  8A        TXA
1AE1-  A0 01     LDY  #$01
1AE3-  F1 CA     SBC  ($CA),Y
1AE5-  A0 09     LDY  #$09
1AE7-  91 C8     STA  ($C8),Y
1AE9-  A0 05     LDY  #$05
1AEB-  B1 CA     LDA  ($CA),Y
1AED-  30 1D     BMI  $1B0C
1AEF-  88        DEY
1AF0-  B1 CA     LDA  ($CA),Y
1AF2-  C5 26     CMP  $26
1AF4-  D0 C3     BNE  $1AB9
1AF6-  A0 06     LDY  #$06
1AF8-  B1 C8     LDA  ($C8),Y
1AFA-  85 CA     STA  $CA
1AFC-  C8        INY
1AFD-  B1 C8     LDA  ($C8),Y
1AFF-  85 CB     STA  $CB
```

```
1B01-   A0 00       LDY   #$00
1B03-   B1 CA       LDA   ($CA),Y
1B05-   C5 25       CMP   $25
1B07-   D0 B0       BNE   $1AB9
1B09-   4C 9D 00    JMP   $009D

1B0C-   A5 25       LDA   $25
1B0E-   A4 26       LDY   $26
1B10-   A2 03       LDX   #$03
1B12-   4C 00 FE    JMP   $FE00

                            pCode=$C8 LEQI
[$D715]
1B15-   BA          TSX
1B16-   68          PLA
1B17-   DD 03 01    CMP   $0103,X
1B1A-   68          PLA
1B1B-   FD 04 01    SBC   $0104,X
1B1E-   30 06       BMI   $1B26
1B20-   50 53       BVC   $1B75
1B22-   70 6B       BVS   $1B8F
1B24-   68          PLA
1B25-   68          PLA
1B26-   70 4D       BVS   $1B75
1B28-   50 65       BVC   $1B8F

                            pCode=$C9 LESI
[$D72A]
1B2A-   BA          TSX
1B2B-   BD 03 01    LDA   $0103,X
1B2E-   DD 01 01    CMP   $0101,X
1B31-   BD 04 01    LDA   $0104,X
1B34-   FD 02 01    SBC   $0102,X
1B37-   30 70       BMI   $1BA9
1B39-   50 20       BVC   $1B5B
1B3B-   68          PLA
1B3C-   68          PLA
1B3D-   A9 01       LDA   #$01
1B3F-   9D 03 01    STA   $0103,X
1B42-   A9 00       LDA   #$00
1B44-   9D 04 01    STA   $0104,X
1B47-   4C 97 00    JMP   $0097


                            pCode=$C4 GEQI
[$D74A]
1B4A-   BA          TSX
1B4B-   BD 03 01    LDA   $0103,X
1B4E-   DD 01 01    CMP   $0101,X
1B51-   BD 04 01    LDA   $0104,X
1B54-   FD 02 01    SBC   $0102,X
1B57-   30 CB       BMI   $1B24
1B59-   50 E0       BVC   $1B3B
1B5B-   68          PLA
1B5C-   68          PLA
1B5D-   A9 00       LDA   #$00
1B5F-   9D 03 01    STA   $0103,X
1B62-   9D 04 01    STA   $0104,X
1B65-   4C 97 00    JMP   $0097

                            pCode=$C3 EQUI
[$D768]
1B68-   BA          TSX
1B69-   68          PLA
```

```
1B6A-   DD 03 01    CMP   $0103,X
1B6D-   D0 ED       BNE   $1B5C
1B6F-   68          PLA
1B70-   DD 04 01    CMP   $0104,X
1B73-   D0 1A       BNE   $1B8F
1B75-   A9 01       LDA   #$01
1B77-   9D 03 01    STA   $0103,X
1B7A-   A9 00       LDA   #$00
1B7C-   9D 04 01    STA   $0104,X
1B7F-   4C 97 00    JMP   $0097

                                  pCode=$CB NEQI
[$D782]
1B82-   BA          TSX
1B83-   68          PLA
1B84-   DD 03 01    CMP   $0103,X
1B87-   D0 B3       BNE   $1B3C
1B89-   68          PLA
1B8A-   DD 04 01    CMP   $0104,X
1B8D-   D0 E6       BNE   $1B75
1B8F-   A9 00       LDA   #$00
1B91-   9D 03 01    STA   $0103,X
1B94-   9D 04 01    STA   $0104,X
1B97-   4C 97 00    JMP   $0097


                                  pCode=$C5 GRTI
[D79A]
1B9A-   BA          TSX
1B9B-   68          PLA
1B9C-   DD 03 01    CMP   $0103,X
1B9F-   68          PLA
1BA0-   FD 04 01    SBC   $0104,X
1BA3-   30 06       BMI   $1BAB
1BA5-   50 E8       BVC   $1B8F
1BA7-   70 CC       BVS   $1B75
1BA9-   68          PLA
1BAA-   68          PLA
1BAB-   50 C8       BVC   $1B75
1BAD-   70 E0       BVS   $1B8F

                                  pCode=$AF EQU
                                  pCode=$AF GEQ
                                  pCode=$AF GRT
                                  pCode=$AF LEQ
                                  pCode=$AF LES
                                  pCode=$AF NEQ
[$D7AF]
1BAF-   BD 2F FE    LDA   $FE2F,X
1BB2-   85 1B       STA   $1B
1BB4-   A0 01       LDY   #$01
1BB6-   B1 9E       LDA   ($9E),Y
1BB8-   AA          TAX
1BB9-   BD D0 FE    LDA   $FED0,X
1BBC-   85 AD       STA   $AD
1BBE-   BD D1 FE    LDA   $FED1,X
1BC1-   85 AE       STA   $AE
1BC3-   6C AD 00    JMP   ($00AD)
1BC6-   38          SEC
1BC7-   24 18       BIT   $18
1BC9-   68          PLA
1BCA-   85 C0       STA   $C0
1BCC-   68          PLA
1BCD-   85 C1       STA   $C1
```

```
1BCF-  68          PLA
1BD0-  85 C2       STA   $C2
1BD2-  68          PLA
1BD3-  85 C3       STA   $C3
1BD5-  08          PHP
1BD6-  20 A5 D0    JSR   $D0A5
1BD9-  85 CC       STA   $CC
1BDB-  8A          TXA
1BDC-  28          PLP
1BDD-  90 03       BCC   $1BE2
1BDF-  4A          LSR
1BE0-  66 CC       ROR   $CC
1BE2-  88          DEY
1BE3-  AA          TAX
1BE4-  F0 0C       BEQ   $1BF2
1BE6-  B1 C2       LDA   ($C2),Y
1BE8-  D1 C0       CMP   ($C0),Y
1BEA-  D0 16       BNE   $1C02
1BEC-  C8          INY
1BED-  D0 F7       BNE   $1BE6
1BEF-  CA          DEX
1BF0-  D0 F4       BNE   $1BE6
1BF2-  A6 1A       LDX   $1A
1BF4-  F0 0A       BEQ   $1C00
1BF6-  B1 C2       LDA   ($C2),Y
1BF8-  D1 C0       CMP   ($C0),Y
1BFA-  D0 06       BNE   $1C02
1BFC-  C8          INY
1BFD-  CA          DEX
1BFE-  D0 F6       BNE   $1BF6
1C00-  F0 59       BEQ   $1C5B
1C02-  B0 54       BCS   $1C58
1C04-  90 4E       BCC   $1C54
1C06-  68          PLA
1C07-  AA          TAX
1C08-  68          PLA
1C09-  D0 06       BNE   $1C11
1C0B-  84 C4       STY   $C4
1C0D-  86 C5       STX   $C5
1C0F-  A2 C4       LDX   #$C4
1C11-  86 C0       STX   $C0
1C13-  85 C1       STA   $C1
1C15-  68          PLA
1C16-  AA          TAX
1C17-  68          PLA
1C18-  D0 06       BNE   $1C20
1C1A-  84 C6       STY   $C6
1C1C-  86 C7       STX   $C7
1C1E-  A2 C6       LDX   #$C6
1C20-  86 C2       STX   $C2
1C22-  85 C3       STA   $C3
1C24-  88          DEY
1C25-  B1 C0       LDA   ($C0),Y
1C27-  D1 C2       CMP   ($C2),Y
1C29-  90 02       BCC   $1C2D
1C2B-  B1 C2       LDA   ($C2),Y
1C2D-  AA          TAX
1C2E-  F0 0A       BEQ   $1C3A
1C30-  C8          INY
1C31-  B1 C0       LDA   ($C0),Y
1C33-  D1 C2       CMP   ($C2),Y
1C35-  D0 09       BNE   $1C40
1C37-  CA          DEX
1C38-  D0 F6       BNE   $1C30
```

	1C3A-	A1 C2	LDA	($C2,X)					
	1C3C-	C1 C0	CMP	($C0,X)					
	1C3E-	F0 1B	BEQ	$1C5B					
	1C40-	B0 16	BCS	$1C58					
	1C42-	90 10	BCC	$1C54					
	1C44-	68	PLA						
	1C45-	4A	LSR						
	1C46-	68	PLA						
	1C47-	68	PLA						
	1C48-	B0 06	BCS	$1C50					
	1C4A-	4A	LSR						
	1C4B-	68	PLA						
	1C4C-	B0 0D	BCS	$1C5B					
	1C4E-	90 08	BCC	$1C58					
	1C50-	4A	LSR						
	1C51-	68	PLA						
	1C52-	90 07	BCC	$1C5B					
	1C54-	A2 16	LDX	#$16					
	1C56-	D0 05	BNE	$1C5D					
	1C58-	A2 2A	LDX	#$2A					
	1C5A-	2C A2 31	BIT	$31A2					
	1C5D-	A9 00	LDA	#$00					
	1C5F-	48	PHA						
	1C60-	8A	TXA						
	1C61-	25 1B	AND	$1B					
	1C63-	F0 02	BEQ	$1C67					
	1C65-	A9 01	LDA	#$01					
	1C67-	48	PHA						
	1C68-	4C F1 D0	JMP	$D0F1					
	1C6B-	A9 00	LDA	#$00					
	1C6D-	85 18	STA	$18					
	1C6F-	85 19	STA	$19					
	1C71-	20 CC D9	JSR	$D9CC					
	1C74-	9A	TXS						
	1C75-	B0 13	BCS	$1C8A					
	1C77-	A4 C4	LDY	$C4					
	1C79-	B1 C0	LDA	($C0),Y					
	1C7B-	D0 07	BNE	$1C84					
	1C7D-	88	DEY						
	1C7E-	C4 C6	CPY	$C6					
	1C80-	D0 F7	BNE	$1C79					
	1C82-	F0 1B	BEQ	$1C9F					
	1C84-	C6 18	DEC	$18					
	1C86-	A4 C6	LDY	$C6					
	1C88-	D0 15	BNE	$1C9F					
	1C8A-	F0 36	BEQ	$1CC2					
	1C8C-	C4 C4	CPY	$C4					
	1C8E-	F0 0F	BEQ	$1C9F					
	1C90-	B1 C2	LDA	($C2),Y					
	1C92-	D0 07	BNE	$1C9B					
	1C94-	88	DEY						
	1C95-	C4 C4	CPY	$C4					
	1C97-	D0 F7	BNE	$1C90					
	1C99-	F0 04	BEQ	$1C9F					
	1C9B-	C6 19	DEC	$19					
	1C9D-	A4 C4	LDY	$C4					
	1C9F-	B1 C2	LDA	($C2),Y					
	1CA1-	51 C0	EOR	($C0),Y					
	1CA3-	F0 16	BEQ	$1CBB					
	1CA5-	AA	TAX						
	1CA6-	31 C0	AND	($C0),Y					
	1CA8-	F0 0B	BEQ	$1CB5					
	1CAA-	A5 18	LDA	$18					
	1CAC-	D0 1E	BNE	$1CCC					

```
1CAE-   C6 19       DEC   $19
1CB0-   8A          TXA
1CB1-   31 C2       AND   ($C2),Y
1CB3-   F0 06       BEQ   $1CBB
1CB5-   A5 19       LDA   $19
1CB7-   D0 13       BNE   $1CCC
1CB9-   C6 18       DEC   $18
1CBB-   88          DEY
1CBC-   D0 E1       BNE   $1C9F
1CBE-   A5 18       LDA   $18
1CC0-   D0 06       BNE   $1CC8
1CC2-   A5 19       LDA   $19
1CC4-   F0 95       BEQ   $1C5B
1CC6-   D0 8C       BNE   $1C54
1CC8-   A5 19       LDA   $19
1CCA-   F0 8C       BEQ   $1C58
1CCC-   A9 02       LDA   #$02
1CCE-   D0 8D       BNE   $1C5D

                                pCode=$A0 ADJ
[$D8D0]
1CD0-   68          PLA
1CD1-   85 C4       STA   $C4
1CD3-   68          PLA
1CD4-   38          SEC
1CD5-   A0 01       LDY   #$01
1CD7-   B1 9E       LDA   ($9E),Y
1CD9-   E5 C4       SBC   $C4
1CDB-   F0 1F       BEQ   $1CFC
1CDD-   0A          ASL
1CDE-   85 27       STA   $27
1CE0-   84 C1       STY   $C1
1CE2-   84 C3       STY   $C3
1CE4-   BA          TSX
1CE5-   8A          TXA
1CE6-   85 C0       STA   $C0
1CE8-   38          SEC
1CE9-   E5 27       SBC   $27
1CEB-   85 C2       STA   $C2
1CED-   AA          TAX
1CEE-   9A          TXS
1CEF-   B0 0E       BCS   $1CFF
1CF1-   B1 9E       LDA   ($9E),Y
1CF3-   0A          ASL
1CF4-   A8          TAY
1CF5-   B1 C0       LDA   ($C0),Y
1CF7-   91 C2       STA   ($C2),Y
1CF9-   88          DEY
1CFA-   D0 F9       BNE   $1CF5
1CFC-   4C F1 D0    JMP   $D0F1
1CFF-   A5 C4       LDA   $C4
1D01-   F0 0A       BEQ   $1D0D
1D03-   0A          ASL
1D04-   AA          TAX
1D05-   B1 C0       LDA   ($C0),Y
1D07-   91 C2       STA   ($C2),Y
1D09-   C8          INY
1D0A-   CA          DEX
1D0B-   D0 E8       BNE   $1CF5
1D0D-   A6 27       LDX   $27
1D0F-   A9 00       LDA   #$00
1D11-   91 C2       STA   ($C2),Y
1D13-   C8          INY
1D14-   CA          DEX
```

```
1D15-   D0 FA       BNE   $1D11
1D17-   4C F1 D0    JMP   $D0F1

                                pCode=$97 SGS
[$D91A]
1D1A-   BA          TSX
1D1B-   BD 02 01    LDA   $0102,X
1D1E-   48          PHA
1D1F-   BD 01 01    LDA   $0101,X
1D22-   48          PHA

                                pCode=$94 SRS
[$D923]
1D23-   68          PLA
1D24-   85 C0       STA   $C0
1D26-   29 0F       AND   #$0F
1D28-   A8          TAY
1D29-   68          PLA
1D2A-   85 C1       STA   $C1
1D2C-   4A          LSR
1D2D-   D0 63       BNE   $1D92
1D2F-   68          PLA
1D30-   85 C2       STA   $C2
1D32-   29 0F       AND   #$0F
1D34-   AA          TAX
1D35-   68          PLA
1D36-   85 C3       STA   $C3
1D38-   4A          LSR
1D39-   D0 57       BNE   $1D92
1D3B-   A5 C0       LDA   $C0
1D3D-   C5 C2       CMP   $C2
1D3F-   A5 C1       LDA   $C1
1D41-   E5 C3       SBC   $C3
1D43-   90 4A       BCC   $1D8F
1D45-   46 C3       LSR   $C3
1D47-   A5 C2       LDA   $C2
1D49-   6A          ROR
1D4A-   4A          LSR
1D4B-   4A          LSR
1D4C-   4A          LSR
1D4D-   85 C3       STA   $C3
1D4F-   46 C1       LSR   $C1
1D51-   A5 C0       LDA   $C0
1D53-   6A          ROR
1D54-   4A          LSR
1D55-   4A          LSR
1D56-   4A          LSR
1D57-   85 C1       STA   $C1
1D59-   B9 F9 FA    LDA   $FAF9,Y
1D5C-   48          PHA
1D5D-   B9 01 FB    LDA   $FB01,Y
1D60-   48          PHA
1D61-   38          SEC
1D62-   A5 C1       LDA   $C1
1D64-   E5 C3       SBC   $C3
1D66-   A8          TAY
1D67-   F0 07       BEQ   $1D70
1D69-   A9 FF       LDA   #$FF
1D6B-   48          PHA
1D6C-   48          PHA
1D6D-   88          DEY
1D6E-   D0 FB       BNE   $1D6B
1D70-   68          PLA
1D71-   A8          TAY
```

```
1D72-   68          PLA
1D73-   3D 08 FB    AND   $FB08,X
1D76-   48          PHA
1D77-   98          TYA
1D78-   3D 10 FB    AND   $FB10,X
1D7B-   48          PHA
1D7C-   A9 00       LDA   #$00
1D7E-   A4 C3       LDY   $C3
1D80-   F0 05       BEQ   $1D87
1D82-   48          PHA
1D83-   48          PHA
1D84-   CA          DEX
1D85-   D0 FB       BNE   $1D82
1D87-   48          PHA
1D88-   E6 C1       INC   $C1
1D8A-   A5 C1       LDA   $C1
1D8C-   4C 96 00    JMP   $0096
1D8F-   4C 90 00    JMP   $0090
1D92-   A2 01       LDX   #$01
1D94-   4C 00 FE    JMP   $FE00

                                pCode=$8B INN
[$D997]
1D97-   68          PLA
1D98-   0A          ASL
1D99-   85 C4       STA   $C4
1D9B-   68          PLA
1D9C-   BA          TSX
1D9D-   86 C0       STX   $C0
1D9F-   8A          TXA
1DA0-   65 C4       ADC   $C4
1DA2-   AA          TAX
1DA3-   9A          TXS
1DA4-   68          PLA
1DA5-   AA          TAX
1DA6-   68          PLA
1DA7-   4A          LSR
1DA8-   D0 1D       BNE   $1DC7
1DAA-   8A          TXA
1DAB-   6A          ROR
1DAC-   4A          LSR
1DAD-   4A          LSR
1DAE-   C5 C4       CMP   $C4
1DB0-   90 03       BCC   $1DB5
1DB2-   D0 13       BNE   $1DC7
1DB4-   18          CLC
1DB5-   65 C0       ADC   $C0
1DB7-   A8          TAY
1DB8-   8A          TXA
1DB9-   29 07       AND   #$07
1DBB-   AA          TAX
1DBC-   B9 01 01    LDA   $0101,Y
1DBF-   3D F0 FA    AND   $FAF0,X
1DC2-   F0 03       BEQ   $1DC7
1DC4-   A2 01       LDX   #$01
1DC6-   2C A2 00    BIT   $00A2
1DC9-   4C 92 00    JMP   $0092
1DCC-   BA          TSX
1DCD-   BD 03 01    LDA   $0103,X
1DD0-   0A          ASL
1DD1-   85 C4       STA   $C4
1DD3-   8A          TXA
1DD4-   69 04       ADC   #$04
1DD6-   85 C0       STA   $C0
```

```
1DD8-  65 C4     ADC  $C4
1DDA-  85 28     STA  $28
1DDC-  AA        TAX
1DDD-  BD 01 01  LDA  $0101,X
1DE0-  0A        ASL
1DE1-  85 C6     STA  $C6
1DE3-  8A        TXA
1DE4-  69 02     ADC  #$02
1DE6-  85 C2     STA  $C2
1DE8-  65 C6     ADC  $C6
1DEA-  AA        TAX
1DEB-  A9 01     LDA  #$01
1DED-  85 C1     STA  $C1
1DEF-  85 C3     STA  $C3
1DF1-  A5 C6     LDA  $C6
1DF3-  C5 C4     CMP  $C4
1DF5-  A8        TAY
1DF6-  60        RTS

                          pCode=$9C UNI
[$D9F7]
1DF7-  20 CC D9  JSR  $D9CC
1DFA-  B0 19     BCS  $1E15
1DFC-  F0 09     BEQ  $1E07
1DFE-  B1 C0     LDA  ($C0),Y

[$DA00]
1E00-  11 C2     ORA  ($C2),Y
1E02-  91 C0     STA  ($C0),Y
1E04-  88        DEY
1E05-  D0 F7     BNE  $1DFE
1E07-  68        PLA
1E08-  9A        TXS
1E09-  AA        TAX
1E0A-  A4 C4     LDY  $C4
1E0C-  B1 C0     LDA  ($C0),Y
1E0E-  48        PHA
1E0F-  88        DEY
1E10-  D0 FA     BNE  $1E0C
1E12-  4C 92 00  JMP  $0092
1E15-  A4 C4     LDY  $C4
1E17-  F0 09     BEQ  $1E22
1E19-  B1 C0     LDA  ($C0),Y
1E1B-  11 C2     ORA  ($C2),Y
1E1D-  91 C2     STA  ($C2),Y
1E1F-  88        DEY
1E20-  D0 F7     BNE  $1E19
1E22-  A6 28     LDX  $28
1E24-  9A        TXS
1E25-  4C 97 00  JMP  $0097

                          pCode=$8C INT
[$DA28]
1E28-  20 CC D9  JSR  $D9CC
1E2B-  F0 1A     BEQ  $1E47
1E2D-  9A        TXS
1E2E-  90 0F     BCC  $1E3F
1E30-  E5 C4     SBC  $C4
1E32-  F0 0B     BEQ  $1E3F
1E34-  A8        TAY
1E35-  A9 00     LDA  #$00
1E37-  48        PHA
1E38-  88        DEY
1E39-  D0 FC     BNE  $1E37
```

```
1E3B-   A4 C4       LDY   $C4
1E3D-   F0 08       BEQ   $1E47
1E3F-   B1 C0       LDA   ($C0),Y
1E41-   31 C2       AND   ($C2),Y
1E43-   48          PHA
1E44-   88          DEY
1E45-   D0 F8       BNE   $1E3F
1E47-   A6 28       LDX   $28
1E49-   9A          TXS
1E4A-   4C 97 00    JMP   $0097

                              pCode=$85 DIF
[$DA4D]
1E4D-   20 CC D9    JSR   $D9CC
1E50-   F0 11       BEQ   $1E63
1E52-   90 04       BCC   $1E58
1E54-   A4 C4       LDY   $C4
1E56-   F0 0B       BEQ   $1E63
1E58-   B1 C0       LDA   ($C0),Y
1E5A-   49 FF       EOR   #$FF
1E5C-   31 C2       AND   ($C2),Y
1E5E-   91 C2       STA   ($C2),Y
1E60-   88          DEY
1E61-   D0 F5       BNE   $1E58
1E63-   A6 28       LDX   $28
1E65-   9A          TXS
1E66-   4C 97 00    JMP   $0097

[$DA69]                       pCode=#$80  ABI
1E69-   BA          TSX
1E6A-   BD 02 01    LDA   $0102,X
1E6D-   10 12       BPL   $1E81

                              pCode=$91 NGI
[$DA6F]
1E6F-   BA          TSX
1E70-   38          SEC
1E71-   A9 00       LDA   #$00
1E73-   FD 01 01    SBC   $0101,X
1E76-   9D 01 01    STA   $0101,X
1E79-   A9 00       LDA   #$00
1E7B-   FD 02 01    SBC   $0102,X
1E7E-   9D 02 01    STA   $0102,X
1E81-   4C 97 00    JMP   $0097

[$DA84]                       pCode=#$82  ADI
1E84-   BA          TSX
1E85-   18          CLC
1E86-   68          PLA
1E87-   7D 03 01    ADC   $0103,X
1E8A-   9D 03 01    STA   $0103,X
1E8D-   68          PLA
1E8E-   7D 04 01    ADC   $0104,X
1E91-   9D 04 01    STA   $0104,X
1E94-   4C 97 00    JMP   $0097

                              pCode=$95 SBI
[$DA97]
1E97-   BA          TSX
1E98-   38          SEC
1E99-   BD 03 01    LDA   $0103,X
1E9C-   FD 01 01    SBC   $0101,X
1E9F-   9D 03 01    STA   $0103,X
1EA2-   E8          INX
```

```
1EA3-   BD 03 01    LDA   $0103,X
1EA6-   FD 01 01    SBC   $0101,X
1EA9-   9D 03 01    STA   $0103,X
1EAC-   E8          INX
1EAD-   9A          TXS
1EAE-   4C 97 00    JMP   $0097

                                pCode=$8F MPI
[$DAB1]
1EB1-   68          PLA
1EB2-   85 14       STA   $14
1EB4-   68          PLA
1EB5-   85 15       STA   $15
1EB7-   68          PLA
1EB8-   85 12       STA   $12
1EBA-   68          PLA
1EBB-   85 13       STA   $13
1EBD-   10 1D       BPL   $1EDC
1EBF-   A5 15       LDA   $15
1EC1-   10 19       BPL   $1EDC
1EC3-   38          SEC
1EC4-   A9 00       LDA   #$00
1EC6-   E5 12       SBC   $12
1EC8-   85 12       STA   $12
1ECA-   A9 00       LDA   #$00
1ECC-   E5 13       SBC   $13
1ECE-   85 13       STA   $13
1ED0-   A9 00       LDA   #$00
1ED2-   E5 14       SBC   $14
1ED4-   85 14       STA   $14
1ED6-   A9 00       LDA   #$00
1ED8-   E5 15       SBC   $15
1EDA-   85 15       STA   $15
1EDC-   20 A9 DB    JSR   $DBA9
1EDF-   A5 17       LDA   $17
1EE1-   48          PHA
1EE2-   A5 16       LDA   $16
1EE4-   4C 96 00    JMP   $0096

                                pCode=$98 SQI
[$DAE7]
1EE7-   68          PLA
1EE8-   85 14       STA   $14
1EEA-   85 12       STA   $12
1EEC-   68          PLA
1EED-   85 15       STA   $15
1EEF-   85 13       STA   $13
1EF1-   4C C1 DA    JMP   $DAC1

                                pCode=$86 DVI
[$DAF4]
1EF4-   68          PLA
1EF5-   85 0C       STA   $0C
1EF7-   68          PLA
1EF8-   85 0D       STA   $0D
1EFA-   68          PLA
1EFB-   85 0E       STA   $0E
1EFD-   68          PLA
1EFE-   85 0F       STA   $0F
1F00-   45 0D       EOR   $0D
1F02-   08          PHP
1F03-   20 0F DC    JSR   $DC0F
1F06-   28          PLP
1F07-   10 0D       BPL   $1F16
```

```
1F09-  38          SEC
1F0A-  A9 00       LDA   #$00
1F0C-  E5 0E       SBC   $0E
1F0E-  85 0E       STA   $0E
1F10-  A9 00       LDA   #$00
1F12-  E5 0F       SBC   $0F
1F14-  85 0F       STA   $0F
1F16-  A5 0F       LDA   $0F
1F18-  48          PHA
1F19-  A5 0E       LDA   $0E
1F1B-  4C 96 00    JMP   $0096

                                pCode=$8E MODI
[$DB1E]
1F1E-  68          PLA
1F1F-  85 0C       STA   $0C
1F21-  68          PLA
1F22-  85 0D       STA   $0D
1F24-  68          PLA
1F25-  85 0E       STA   $0E
1F27-  68          PLA
1F28-  85 0F       STA   $0F
1F2A-  08          PHP
1F2B-  20 0F DC    JSR   $DC0F
1F2E-  28          PLP
1F2F-  10 0D       BPL   $1F3E
1F31-  38          SEC
1F32-  A9 00       LDA   #$00
1F34-  E5 10       SBC   $10
1F36-  85 10       STA   $10
1F38-  A9 00       LDA   #$00
1F3A-  E5 11       SBC   $11
1F3C-  85 11       STA   $11
1F3E-  A5 11       LDA   $11
1F40-  48          PHA
1F41-  A5 10       LDA   $10
1F43-  4C 96 00    JMP   $0096

                                pCode=$88 CHK
[$DB46]
1F46-  BA          TSX
1F47-  68          PLA
1F48-  DD 05 01    CMP   $0105,X
1F4B-  68          PLA
1F4C-  FD 06 01    SBC   $0106,X
1F4F-  10 0E       BPL   $1F5F
1F51-  70 0E       BVS   $1F61
1F53-  BD 05 01    LDA   $0105,X
1F56-  A8          TAY
1F57-  BD 06 01    LDA   $0106,X
1F5A-  A2 01       LDX   #$01
1F5C-  4C 00 FE    JMP   $FE00
1F5F-  70 F2       BVS   $1F53
1F61-  68          PLA
1F62-  68          PLA
1F63-  BD 05 01    LDA   $0105,X
1F66-  DD 03 01    CMP   $0103,X
1F69-  BD 06 01    LDA   $0106,X
1F6C-  DD 04 01    CMP   $0104,X
1F6F-  10 05       BPL   $1F76
1F71-  50 E0       BVC   $1F53
1F73-  4C 97 00    JMP   $0097
1F76-  70 DB       BVS   $1F53
```

```
1F78-   4C 97 00    JMP   $0097

                                  pCode=$84 LAND
[$DB7B]
1F7B-   BA          TSX
1F7C-   68          PLA
1F7D-   3D 03 01    AND   $0103,X
1F80-   9D 03 01    STA   $0103,X
1F83-   68          PLA
1F84-   3D 04 01    AND   $0104,X
1F87-   9D 04 01    STA   $0104,X
1F8A-   4C 97 00    JMP   $0097

                                  pCode=$8D LOR
[$DB8D]
1F8D-   BA          TSX
1F8E-   68          PLA
1F8F-   1D 03 01    ORA   $0103,X
1F92-   9D 03 01    STA   $0103,X
1F95-   68          PLA
1F96-   1D 04 01    ORA   $0104,X
1F99-   9D 04 01    STA   $0104,X
1F9C-   4C 97 00    JMP   $0097


                                  pCode=$93 LNOT
[$DB9F]
1F9F-   68          PLA
1FA0-   49 FF       EOR   #$FF
1FA2-   AA          TAX
1FA3-   68          PLA
1FA4-   49 FF       EOR   #$FF
1FA6-   4C 94 00    JMP   $0094
1FA9-   A9 00       LDA   #$00
1FAB-   85 17       STA   $17
1FAD-   A6 15       LDX   $15
1FAF-   D0 1A       BNE   $1FCB
1FB1-   A4 13       LDY   $13
1FB3-   D0 24       BNE   $1FD9
1FB5-   A0 08       LDY   #$08
1FB7-   0A          ASL
1FB8-   26 17       ROL   $17
1FBA-   06 14       ASL   $14
1FBC-   90 07       BCC   $1FC5
1FBE-   18          CLC
1FBF-   65 12       ADC   $12
1FC1-   90 02       BCC   $1FC5
1FC3-   E6 17       INC   $17
1FC5-   88          DEY
1FC6-   D0 EF       BNE   $1FB7
1FC8-   85 16       STA   $16
1FCA-   60          RTS
1FCB-   A4 13       LDY   $13
1FCD-   D0 24       BNE   $1FF3
1FCF-   86 13       STX   $13
1FD1-   A6 14       LDX   $14
1FD3-   A4 12       LDY   $12
1FD5-   86 12       STX   $12
1FD7-   84 14       STY   $14
1FD9-   A0 08       LDY   #$08
1FDB-   0A          ASL
1FDC-   26 17       ROL   $17
1FDE-   06 14       ASL   $14
1FE0-   90 0B       BCC   $1FED
```

	1FE2-	18	CLC						
	1FE3-	65 12	ADC	$12					
	1FE5-	AA	TAX						
	1FE6-	A5 17	LDA	$17					
	1FE8-	65 13	ADC	$13					
	1FEA-	85 17	STA	$17					
	1FEC-	8A	TXA						
	1FED-	88	DEY						
	1FEE-	D0 EB	BNE	$1FDB					
	1FF0-	85 16	STA	$16					
	1FF2-	60	RTS						
	1FF3-	A0 10	LDY	#$10					
	1FF5-	0A	ASL						
	1FF6-	26 17	ROL	$17					
	1FF8-	06 14	ASL	$14					
	1FFA-	26 15	ROL	$15					
	1FFC-	90 0B	BCC	$2009					
	1FFE-	18	CLC						
	1FFF-	65 12	ADC	$12					
	2001-	AA	TAX						
	2002-	A5 17	LDA	$17					
	2004-	65 13	ADC	$13					
	2006-	85 17	STA	$17					
	2008-	8A	TXA						
	2009-	88	DEY						
	200A-	D0 E9	BNE	$1FF5					
	200C-	85 16	STA	$16					
	200E-	60	RTS						
	200F-	A4 0D	LDY	$0D					
	2011-	D0 05	BNE	$2018					
	2013-	A6 0C	LDX	$0C					
	2015-	F0 2A	BEQ	$2041					
	2017-	98	TYA						
	2018-	10 0D	BPL	$2027					
	201A-	38	SEC						
	201B-	A9 00	LDA	#$00					
	201D-	E5 0C	SBC	$0C					
	201F-	85 0C	STA	$0C					
	2021-	A9 00	LDA	#$00					
	2023-	E5 0D	SBC	$0D					
	2025-	85 0D	STA	$0D					
	2027-	A5 0F	LDA	$0F					
	2029-	10 0D	BPL	$2038					
	202B-	38	SEC						
	202C-	A9 00	LDA	#$00					
	202E-	E5 0E	SBC	$0E					
	2030-	85 0E	STA	$0E					
	2032-	A9 00	LDA	#$00					
	2034-	E5 0F	SBC	$0F					
	2036-	85 0F	STA	$0F					
	2038-	A8	TAY						
	2039-	A6 0E	LDX	$0E					
	203B-	E4 0C	CPX	$0C					
	203D-	E5 0D	SBC	$0D					
	203F-	B0 0B	BCS	$204C					
	2041-	A9 00	LDA	#$00					
	2043-	85 0E	STA	$0E					
	2045-	85 0F	STA	$0F					
	2047-	86 10	STX	$10					
	2049-	84 11	STY	$11					
	204B-	60	RTS						
	204C-	98	TYA						
	204D-	D0 07	BNE	$2056					
	204F-	84 0E	STY	$0E					

```
2051-   86 0F       STX   $0F
2053-   A0 08       LDY   #$08
2055-   2C A0 10    BIT   $10A0
2058-   A9 00       LDA   #$00
205A-   85 11       STA   $11
205C-   06 0E       ASL   $0E
205E-   26 0F       ROL   $0F
2060-   2A          ROL
2061-   26 11       ROL   $11
2063-   AA          TAX
2064-   C5 0C       CMP   $0C
2066-   A5 11       LDA   $11
2068-   E5 0D       SBC   $0D
206A-   90 08       BCC   $2074
206C-   85 11       STA   $11
206E-   8A          TXA
206F-   E5 0C       SBC   $0C
2071-   AA          TAX
2072-   E6 0E       INC   $0E
2074-   8A          TXA
2075-   88          DEY
2076-   D0 E4       BNE   $205C
2078-   85 10       STA   $10
207A-   60          RTS
        ------------------------------------------------------------------------------
                                  pCode=$9E CSP Call Standard Procedure
[$DC7B]
207B-   A0 01       LDY   #$01      Get <UB> (following $9E) as Pascal Procedure
207D-   B1 9E       LDA   ($9E),Y
207F-   AA          TAX
2080-   BD 80 FE    LDA   $FE80,X   For example, 9E 06 == UNITWRITE == $DEBO
2083-   85 AD       STA   $AD
2085-   BD A9 FE    LDA   $FEA9,X
2088-   85 AE       STA   $AE
208A-   6C AD 00    JMP   ($00AD)
        ------------------------------------------------------------------------------
                                  FILLCHAR   (from CSP 9E 0A)
[$DC8D]
208D-   68          PLA
208E-   A8          TAY
208F-   68          PLA
2090-   68          PLA
2091-   85 CC       STA   $CC
2093-   68          PLA
2094-   85 CD       STA   $CD
2096-   BA          TSX
2097-   18          CLC
2098-   68          PLA
2099-   7D 03 01    ADC   $0103,X
209C-   85 C8       STA   $C8
209E-   68          PLA
209F-   7D 04 01    ADC   $0104,X
20A2-   85 C9       STA   $C9
20A4-   68          PLA
20A5-   68          PLA
20A6-   98          TYA
20A7-   A0 00       LDY   #$00
20A9-   A6 CD       LDX   $CD
20AB-   F0 0A       BEQ   $20B7
20AD-   91 C8       STA   ($C8),Y
20AF-   C8          INY
20B0-   D0 FB       BNE   $20AD
20B2-   E6 C9       INC   $C9
20B4-   CA          DEX
```

	20B5-	D0 F6	BNE	$20AD					
	20B7-	A6 CC	LDX	$CC					
	20B9-	F0 06	BEQ	$20C1					
	20BB-	91 C8	STA	($C8),Y					
	20BD-	C8	INY						
	20BE-	CA	DEX						
	20BF-	D0 FA	BNE	$20BB					
	20C1-	4C F1 D0	JMP	$D0F1					
	20C4-	68	PLA						
	20C5-	68	PLA						
	20C6-	BA	TSX						
	20C7-	18	CLC						
	20C8-	68	PLA						
	20C9-	7D 03 01	ADC	$0103,X					
	20CC-	85 C8	STA	$C8					
	20CE-	85 CA	STA	$CA					
	20D0-	68	PLA						
	20D1-	7D 04 01	ADC	$0104,X					
	20D4-	85 C9	STA	$C9					
	20D6-	85 CB	STA	$CB					
	20D8-	68	PLA						
	20D9-	68	PLA						
	20DA-	68	PLA						
	20DB-	A8	TAY						
	20DC-	68	PLA						
	20DD-	68	PLA						
	20DE-	6A	ROR						
	20DF-	4A	LSR						
	20E0-	85 1A	STA	$1A					
	20E2-	24 1A	BIT	$1A					
	20E4-	68	PLA						
	20E5-	AA	TAX						
	20E6-	68	PLA						
	20E7-	85 CD	STA	$CD					
	20E9-	30 3B	BMI	$2126					
	20EB-	8A	TXA						
	20EC-	F0 02	BEQ	$20F0					
	20EE-	E6 CD	INC	$CD					
	20F0-	05 CD	ORA	$CD					
	20F2-	F0 2C	BEQ	$2120					
	20F4-	98	TYA						
	20F5-	A0 00	LDY	#$00					
	20F7-	D1 C8	CMP	($C8),Y					
	20F9-	F0 04	BEQ	$20FF					
	20FB-	50 04	BVC	$2101					
	20FD-	70 11	BVS	$2110					
	20FF-	50 0F	BVC	$2110					
	2101-	C8	INY						
	2102-	D0 02	BNE	$2106					
	2104-	E6 C9	INC	$C9					
	2106-	CA	DEX						
	2107-	D0 EE	BNE	$20F7					
	2109-	C6 CD	DEC	$CD					
	210B-	D0 EA	BNE	$20F7					
	210D-	18	CLC						
	210E-	F0 01	BEQ	$2111					
	2110-	38	SEC						
	2111-	98	TYA						
	2112-	65 C8	ADC	$C8					
	2114-	90 02	BCC	$2118					
	2116-	E6 C9	INC	$C9					
	2118-	38	SEC						
	2119-	E5 CA	SBC	$CA					
	211B-	AA	TAX						

```
211C-  A5 C9       LDA   $C9
211E-  E5 CB       SBC   $CB
2120-  48          PHA
2121-  8A          TXA
2122-  48          PHA
2123-  4C F1 D0    JMP   $D0F1
2126-  98          TYA
2127-  A0 00       LDY   #$00
2129-  D1 C8       CMP   ($C8),Y
212B-  F0 04       BEQ   $2131
212D-  50 04       BVC   $2133
212F-  70 12       BVS   $2143
2131-  50 10       BVC   $2143
2133-  C0 00       CPY   #$00
2135-  D0 02       BNE   $2139
2137-  C6 C9       DEC   $C9
2139-  88          DEY
213A-  E8          INX
213B-  D0 EC       BNE   $2129
213D-  E6 CD       INC   $CD
213F-  D0 E8       BNE   $2129
2141-  F0 07       BEQ   $214A
2143-  C0 00       CPY   #$00
2145-  D0 02       BNE   $2149
2147-  C6 C9       DEC   $C9
2149-  88          DEY
214A-  18          CLC
214B-  90 C4       BCC   $2111
214D-  68          PLA
214E-  85 CC       STA   $CC
2150-  68          PLA
2151-  85 CD       STA   $CD
2153-  BA          TSX
2154-  18          CLC
2155-  68          PLA
2156-  7D 03 01    ADC   $0103,X
2159-  85 CA       STA   $CA
215B-  68          PLA
215C-  7D 04 01    ADC   $0104,X
215F-  85 CB       STA   $CB
2161-  68          PLA
2162-  68          PLA
2163-  BA          TSX
2164-  18          CLC
2165-  68          PLA
2166-  7D 03 01    ADC   $0103,X
2169-  85 C8       STA   $C8
216B-  68          PLA
216C-  7D 04 01    ADC   $0104,X
216F-  85 C9       STA   $C9
2171-  68          PLA
2172-  68          PLA
2173-  B1 9E       LDA   ($9E),Y
2175-  C9 02       CMP   #$02
2177-  F0 06       BEQ   $217F

                             $C8.C9  Source
                             $CA.CB  Destination
                             $CC.CD  Length

2179-  20 24 D0    JSR   $D024     Block Transfer
217C-  4C F1 D0    JMP   $D0F1
217F-  20 4E D0    JSR   $D04E
2182-  4C F1 D0    JMP   $D0F1
2185-  68          PLA
2186-  0A          ASL
```

```
2187-   85 CC       STA   $CC
2189-   68          PLA
218A-   2A          ROL
218B-   85 CD       STA   $CD
218D-   68          PLA
218E-   85 C8       STA   $C8
2190-   68          PLA
2191-   85 C9       STA   $C9
2193-   88          DEY
2194-   18          CLC
2195-   A5 B4       LDA   $B4
2197-   91 C8       STA   ($C8),Y
2199-   65 CC       ADC   $CC
219B-   85 B4       STA   $B4
219D-   C8          INY
219E-   A5 B5       LDA   $B5
21A0-   91 C8       STA   ($C8),Y
21A2-   65 CD       ADC   $CD
21A4-   85 B5       STA   $B5
21A6-   20 4F D5    JSR   $D54F
21A9-   4C F1 D0    JMP   $D0F1
21AC-   68          PLA
21AD-   85 C8       STA   $C8
21AF-   68          PLA
21B0-   85 C9       STA   $C9
21B2-   A5 B5       LDA   $B5
21B4-   91 C8       STA   ($C8),Y
21B6-   88          DEY
21B7-   A5 B4       LDA   $B4
21B9-   91 C8       STA   ($C8),Y
21BB-   4C F1 D0    JMP   $D0F1
21BE-   68          PLA
21BF-   85 C8       STA   $C8
21C1-   68          PLA
21C2-   85 C9       STA   $C9
21C4-   B1 C8       LDA   ($C8),Y
21C6-   85 B5       STA   $B5
21C8-   88          DEY
21C9-   B1 C8       LDA   ($C8),Y
21CB-   85 B4       STA   $B4
21CD-   20 4F D5    JSR   $D54F
21D0-   4C F1 D0    JMP   $D0F1
        ------------------------------------------------------------------------------
                                  MEMAVAIL
21D3-   38          SEC
21D4-   A5 BE       LDA   $BE
21D6-   E5 B4       SBC   $B4
21D8-   AA          TAX
21D9-   A5 BD       LDA   $BD
21DB-   E5 B5       SBC   $B5
21DD-   4A          LSR
21DE-   48          PHA
21DF-   8A          TXA
21E0-   6A          ROR
21E1-   48          PHA
21E2-   4C F1 D0    JMP   $D0F1
        ------------------------------------------------------------------------------
21E5-   68          PLA
21E6-   AA          TAX
21E7-   68          PLA
21E8-   8A          TXA
21E9-   20 14 DE    JSR   $DE14
21EC-   4C F1 D0    JMP   $D0F1
```

```
21EF-   68          PLA
21F0-   AA          TAX
21F1-   68          PLA
21F2-   8A          TXA
21F3-   20 F9 DD    JSR   $DDF9
21F6-   4C F1 D0    JMP   $D0F1
21F9-   A8          TAY
21FA-   F0 17       BEQ   $2213
21FC-   BE 19 0C    LDX   $0C19,Y
21FF-   DE 9A 0C    DEC   $0C9A,X
2202-   D0 0F       BNE   $2213
2204-   18          CLC
2205-   A5 BE       LDA   $BE
2207-   7D 5A 0C    ADC   $0C5A,X
220A-   85 BE       STA   $BE
220C-   A5 BF       LDA   $BF
220E-   7D 6A 0C    ADC   $0C6A,X
2211-   85 BF       STA   $BF
2213-   60          RTS
-------------------------------------------------------------------------------
                                  To here processing CXP
                                    From $D59F
                                  A = Segment# (e.g. 1, 7, 8, 9, ...)
[$DE14]
2214-   A8          TAY           Segment#  (1 first time, #$C next time).
2215-   BE 19 0C    LDX   $0C19,Y Convert Seg# to $C3A index (0, 1, 2, 3...)
2218-   BD 9A 0C    LDA   $0C9A,X Use Count
221B-   D0 16       BNE   $2233   Already loaded?  $DE33

221D-   38          SEC           Need to load this segment
221E-   A5 BE       LDA   $BE     KP ($BE.BF) == Stack ptr
2220-   E9 02       SBC   #$02    Calc high end of Segment in memory
2222-   9D 7A 0C    STA   $0C7A,X Save Seg Dictionary in Seg Load Tables
2225-   A5 BF       LDA   $BF       $C7A,x $C8A,x   x == [$0..$F]
2227-   E9 00       SBC   #$00
2229-   9D 8A 0C    STA   $0C8A,X
222C-   8A          TXA
222D-   48          PHA           Save segment index (0..$F)
222E-   20 37 DE    JSR   $DE37   JSR $DE37
2231-   68          PLA           A := segment index (0..$F)
2232-   AA          TAX

[$DE33]
2233-   FE 9A 0C    INC   $0C9A,X Segment "use" count
2236-   60          RTS
-------------------------------------------------------------------------------
                                  X == Segment# index into $C3A...
[$DE37]
2237-   BD 3A 0C    LDA   $0C3A,X $0C3A,X is Disk ADDR
223A-   85 34       STA   $34
223C-   BD 4A 0C    LDA   $0C4A,X $0C4A,X is Disk ADDR
223F-   85 35       STA   $35
2241-   05 34       ORA   $34
2243-   D0 0B       BNE   $2250     Disk Addr <> 0, $DE50
2245-   8A          TXA
2246-   A2 00       LDX   #$00    Error, $FE00
2248-   4C 00 FE    JMP   $FE00

224B-   A2 02       LDX   #$02    Error, $FD00
224D-   4C 00 FD    JMP   $FD00

                                  X == Segment# index into $C3A...
[$DE50]
2250-   BD 5A 0C    LDA   $0C5A,X $0C5A,X is Disk Length
```

```
2253-   85 32       STA   $32
2255-   BD 6A 0C    LDA   $0C6A,X   $0C6A,X is Disk Length
2258-   85 33       STA   $33

225A-   38          SEC             Adjust KP
225B-   A5 BE       LDA   $BE          ($BE.BF)
225D-   E5 32       SBC   $32
225F-   85 BE       STA   $BE
2261-   85 30       STA   $30       $30.31 := Destination for I/O
2263-   A5 BF       LDA   $BF                (same as BE.BF now)
2265-   E5 33       SBC   $33
2267-   85 BF       STA   $BF
2269-   85 31       STA   $31

226B-   A9 04       LDA   #$04      2D := #$04 = Pascal Device S6,D1
226D-   85 2D       STA   $2D
226F-   A2 00       LDX   #$00      X := 0 (Read function)
2271-   86 2E       STX   $2E       2E := #$00
2273-   86 2F       STX   $2F       2F := #$00
2275-   20 5C E8    JSR   $E85C     Read Blocks

[$DE78]
2278-   8A          TXA
2279-   D0 D0       BNE   $224B   Error, $224B
227B-   60          RTS
-------------------------------------------------------------------------------
227C-   A6 F0       LDX   $F0
227E-   F0 09       BEQ   $2289
2280-   4C 00 FD    JMP   $FD00
2283-   A9 00       LDA   #$00
2285-   48          PHA
2286-   A5 F0       LDA   $F0
2288-   48          PHA
2289-   4C F1 D0    JMP   $D0F1
228C-   68          PLA
228D-   85 2E       STA   $2E
228F-   68          PLA
2290-   85 2F       STA   $2F
2292-   BA          TSX
2293-   18          CLC
2294-   68          PLA
2295-   7D 03 01    ADC   $0103,X
2298-   85 30       STA   $30
229A-   68          PLA
229B-   7D 04 01    ADC   $0104,X
229E-   85 31       STA   $31
22A0-   68          PLA
22A1-   68          PLA
22A2-   A2 04       LDX   #$04
22A4-   2C A2 05    BIT   $05A2
22A7-   2C A2 06    BIT   $06A2
22AA-   2C A2 02    BIT   $02A2
22AD-   4C D8 DE    JMP   $DED8

-------------------------------------------------------------------------------
                              WRITE TO VOL#  (CSP 9E 06) UNITWRITE()

                          UNITWRITE( UNIT, ARRAY, LENGTH, BLOCK#, MODE)
                                     $2D   $30.31 $32.33  $34.35  $2E.2F
[$DEB0]
22B0-   A2 01        LDX   #$01
22B2-    2C         BIT   $00A2
                              READ FROM VOL# (CSP 9E 05) UNITREAD()
[$DEB3]
```

```
         A2 00       LDX   #00    Pull parameters off in reverse order
22B5-    68          PLA
22B6-    85 2E       STA   $2E       Mode
22B8-    68          PLA
22B9-    85 2F       STA   $2F
22BB-    68          PLA
22BC-    85 34       STA   $34       Block Number to Read/Write
22BE-    68          PLA
22BF-    85 35       STA   $35
22C1-    68          PLA
22C2-    85 32       STA   $32       Length of I/O
22C4-    68          PLA             (for UNIT = #$D, this is the function)
22C5-    85 33       STA   $33
22C7-    68          PLA
22C8-    85 30       STA   $30       Dest Array subscript
22CA-    68          PLA
22CB-    85 31       STA   $31
22CD-    18          CLC
22CE-    68          PLA
22CF-    65 30       ADC   $30       Dest Array
22D1-    85 30       STA   $30
22D3-    68          PLA
22D4-    65 31       ADC   $31
22D6-    85 31       STA   $31
22D8-    68          PLA
22D9-    85 2D       STA   $2D       Unit (#$D special for Wiz IV)
22DB-    68          PLA
22DC-    A8          TAY

[$DEDD]
22DD-    20 E5 DE    JSR   $DEE5    JSR $DEE5
22E0-    86 F0       STX   $F0
22E2-    4C F1 D0    JMP   $D0F1    All done, advance IPC by 2, and continue

[$DEE5]
22E5-    20 7D E7    JSR   $E77D     Check keyboard for input
22E8-    A5 2D       LDA   $2D       UNIT = 0?
22EA-    F0 13       BEQ   $22FF       Yes, $DEFF
22EC-    C9 0F       CMP   #$0F      UNIT > #$0E?    (#$D is valid and #$E !?)
22EE-    B0 0F       BCS   $22FF       Yes, $DEFF

22F0-    0A          ASL             UNIT = 1..$D (or #$E!?)
22F1-    A8          TAY
22F2-    B9 06 DF    LDA   $DF06,Y
22F5-    85 AD       STA   $AD
22F7-    B9 07 DF    LDA   $DF07,Y
22FA-    85 AE       STA   $AE
22FC-    6C AD 00    JMP   ($00AD)   UNIT $D: ($ED10)

[$DEFF]
22FF-    8A          TXA
2300-    A8          TAY
2301-    A5 2D       LDA   $2D
2303-    A2 12       LDX   #$12
2305-    4C          JMP   $FE00

[$DF06, $DF07]
         00 FE
         BC E7
         BC E7
         FF DE           3             $DEFF
         5C E8     Device 4 (S6, D1)
         FE E7            5 (S6, D2)
         FF DE            6             $DEFF Printer
```

```
        FF DE            7              $DEFF REMIN
        FF DE            8              $DEFF REMOUT
        FE E7            9.
        FE E7           10.
        FE E7           11.
        FE E7           12.
        10 ED     #$D   13.             $ED10

--------------------------------------------------------------------------------
                                       Clear Primary Text Screen to blanks

                                       $FFF4 points here
                                       JSR from $BD4F after I/O error reading Dir
                                         during boot
[$DF22]
2322    A2 17       LDX   #$17     24 vertical rows

2324-   BD 61 E2    LDA   $E261,X  ptr lines for primary/secondary text pages
2327-   09 04       ORA   #$04     page 1
2329-   85 C9       STA   $C9
232B-   BD 79 E2    LDA   $E279,X
232E-   85 C8       STA   $C8
2330-   A9 A0       LDA   #$A0     " "
2332-   A0 27       LDY   #$27     40 horizontal characters

2334-   91 C8       STA   ($C8),Y
2336-   88          DEY
2337-   10 FB       BPL   $2334
2339-   CA          DEX
233A-   10 E8       BPL   $2324
233C-   60          RTS
--------------------------------------------------------------------------------
233D-   20 22 DF    JSR   $DF22
2340-   4C 00 D0    JMP   $D000
--------------------------------------------------------------------------------
2343-   00          BRK
2344-   00          BRK
2345-   00          BRK
2346-   00          BRK
2347-   00          BRK
2348-   00          BRK
2349-   00          BRK
234A-   00          BRK
234B-   00          BRK
234C-   00          BRK
234D-   00          BRK
234E-   00          BRK
234F-   00          BRK
2350-   00          BRK
2351-   00          BRK
2352-   00          BRK
2353-   00          BRK
2354-   00          BRK
2355-   00          BRK
2356-   00          BRK
2357-   00          BRK
2358-   00          BRK
2359-   00          BRK
235A-   00          BRK
235B-   00          BRK
235C-   00          BRK
235D-   00          BRK
235E-   00          BRK
235F-   00          BRK
```

	2360- 00	BRK						
	2361- 00	BRK						
	2362- 00	BRK						
	2363- 00	BRK						
	2364- 00	BRK						
	2365- 00	BRK						
	2366- 00	BRK						
	2367- 00	BRK						
	2368- 00	BRK						
	2369- 00	BRK						
	236A- 00	BRK						
	236B- 00	BRK						
	236C- 00	BRK						
	236D- 00	BRK						
	236E- 00	BRK						
	236F- 00	BRK						
	2370- 00	BRK						
	2371- 00	BRK						
	2372- 00	BRK						
	2373- 00	BRK						
	2374- 00	BRK						
	2375- 00	BRK						
	2376- 00	BRK						
	2377- 00	BRK						
	2378- 00	BRK						
	2379- 00	BRK						
	237A- 00	BRK						
	237B- 00	BRK						
	237C- 00	BRK						
	237D- 00	BRK						
	237E- 00	BRK						
	237F- 00	BRK						
	2380- 00	BRK						
	2381- 00	BRK						
	2382- 00	BRK						
	2383- 00	BRK						
	2384- 00	BRK						
	2385- 00	BRK						
	2386- 00	BRK						
	2387- 00	BRK						
	2388- 00	BRK						
	2389- 00	BRK						
	238A- 00	BRK						
	238B- 00	BRK						
	238C- 00	BRK						
	238D- 00	BRK						
	238E- 00	BRK						
	238F- 00	BRK						
	2390- 00	BRK						
	2391- 00	BRK						
	2392- 00	BRK						
	2393- 00	BRK						
	2394- 00	BRK						
	2395- 00	BRK						
	2396- 00	BRK						
	2397- 00	BRK						
	2398- 00	BRK						
	2399- 00	BRK						
	239A- 00	BRK						
	239B- 00	BRK						
	239C- 00	BRK						
	239D- 00	BRK						
	239E- 00	BRK						
	239F- 00	BRK						

	23A0- 00	BRK						
	23A1- 00	BRK						
	23A2- 00	BRK						
	23A3- 00	BRK						
	23A4- 00	BRK						
	23A5- 00	BRK						
	23A6- 00	BRK						
	23A7- 00	BRK						
	23A8- 00	BRK						
	23A9- 00	BRK						
	23AA- 00	BRK						
	23AB- 00	BRK						
	23AC- 00	BRK						
	23AD- 00	BRK						
	23AE- 00	BRK						
	23AF- 00	BRK						
	23B0- 00	BRK						
	23B1- 00	BRK						
	23B2- 00	BRK						
	23B3- 00	BRK						
	23B4- 00	BRK						
	23B5- 00	BRK						
	23B6- 00	BRK						
	23B7- 00	BRK						
	23B8- 00	BRK						
	23B9- 00	BRK						
	23BA- 00	BRK						
	23BB- 00	BRK						
	23BC- 00	BRK						
	23BD- 00	BRK						
	23BE- 00	BRK						
	23BF- 00	BRK						
	23C0- 00	BRK						
	23C1- 00	BRK						
	23C2- 00	BRK						
	23C3- 00	BRK						
	23C4- 00	BRK						
	23C5- 00	BRK						
	23C6- 00	BRK						
	23C7- 00	BRK						
	23C8- 00	BRK						
	23C9- 00	BRK						
	23CA- 00	BRK						
	23CB- 00	BRK						
	23CC- 00	BRK						
	23CD- 00	BRK						
	23CE- 00	BRK						
	23CF- 00	BRK						
	23D0- 00	BRK						
	23D1- 00	BRK						
	23D2- 00	BRK						
	23D3- 00	BRK						
	23D4- 00	BRK						
	23D5- 00	BRK						
	23D6- 00	BRK						
	23D7- 00	BRK						
	23D8- 00	BRK						
	23D9- 00	BRK						
	23DA- 00	BRK						
	23DB- 00	BRK						
	23DC- 00	BRK						
	23DD- 00	BRK						
	23DE- 00	BRK						
	23DF- 00	BRK						

```
23E0-   00          BRK
23E1-   00          BRK
23E2-   00          BRK
23E3-   00          BRK
23E4-   00          BRK
23E5-   00          BRK
23E6-   00          BRK
23E7-   00          BRK
23E8-   00          BRK
23E9-   00          BRK
23EA-   00          BRK
23EB-   00          BRK
23EC-   00          BRK
23ED-   00          BRK
23EE-   00          BRK
23EF-   00          BRK
23F0-   00          BRK
23F1-   00          BRK
23F2-   00          BRK
23F3-   00          BRK
23F4-   00          BRK
23F5-   00          BRK
23F6-   00          BRK
23F7-   00          BRK
23F8-   00          BRK
23F9-   00          BRK
23FA-   00          BRK
23FB-   00          BRK
23FC-   00          BRK
23FD-   00          BRK
23FE-   00          BRK
23FF-   00          BRK

--------------------------------------------------------------------------------
[$E000]
2400-   A2 00       LDX   #$00
2402-   A0 02       LDY   #$02
2404-   88          DEY
2405-   B1 30       LDA   ($30),Y
2407-   4A          LSR
2408-   3E AA 0C    ROL   $0CAA,X
240B-   4A          LSR
240C-   3E AA 0C    ROL   $0CAA,X
240F-   99 00 03    STA   $0300,Y
2412-   E8          INX
2413-   E0 56       CPX   #$56
2415-   90 ED       BCC   $2404
2417-   A2 00       LDX   #$00
2419-   98          TYA
241A-   D0 E8       BNE   $2404
241C-   A2 55       LDX   #$55
241E-   BD AA 0C    LDA   $0CAA,X
2421-   29 3F       AND   #$3F
2423-   9D AA 0C    STA   $0CAA,X
2426-   CA          DEX
2427-   10 F5       BPL   $241E
2429-   60          RTS
--------------------------------------------------------------------------------
                                Write to Disk
[$E02A]
242A-   38          SEC
242B-   A6 FB       LDX   $FB
242D-   BD 8D C0    LDA   $C08D,X  Load Data Latch
2430-   BD 8E C0    LDA   $C08E,X  Prepare Latch for input
```

```
2433-  30 7D       BMI   $24B2
2435-  AD AA 0C    LDA   $0CAA
2438-  85 4B       STA   $4B
243A-  A9 FF       LDA   #$FF
243C-  9D 8F C0    STA   $C08F,X  Prepare Latch for output
243F-  1D 8C C0    ORA   $C08C,X  Strobe Data Latch
2442-  48          PHA
2443-  68          PLA
2444-  EA          NOP
2445-  A0 04       LDY   #$04
2447-  48          PHA
2448-  68          PLA
2449-  20 B7 E0    JSR   $E0B7
244C-  88          DEY
244D-  D0 F8       BNE   $2447
                                 DATA FIELD

244F-  A9 D5       LDA   #$D5    #$D5
2451-  20 B6 E0    JSR   $E0B6
2454-  A9 AA       LDA   #$AA    #$AA
2456-  20 B6 E0    JSR   $E0B6
2459-  A9 AD       LDA   #$AD    #$AD
245B-  20 B6 E0    JSR   $E0B6
245E-  98          TYA
245F-  A0 56       LDY   #$56    86
2461-  D0 03       BNE   $2466

2463-  B9 AA 0C    LDA   $0CAA,Y  86 nibbles
2466-  59 A9 0C    EOR   $0CA9,Y
2469-  AA          TAX
246A-  BD 21 E2    LDA   $E221,X
246D-  AE F8 05    LDX   $05F8
2470-  9D 8D C0    STA   $C08D,X  Load Data Latch
2473-  BD 8C C0    LDA   $C08C,X  Strobe Data Latch
2476-  88          DEY
2477-  D0 EA       BNE   $2463

2479-  A5 4B       LDA   $4B
247B-  EA          NOP

247C-  59 00 03    EOR   $0300,Y  256 nibbles
247F-  AA          TAX
2480-  BD 21 E2    LDA   $E221,X
2483-  AE F8 05    LDX   $05F8
2486-  9D 8D C0    STA   $C08D,X  Load Data Latch
2489-  BD 8C C0    LDA   $C08C,X  Strobe Data Latch
248C-  B9 00 03    LDA   $0300,Y
248F-  C8          INY
2490-  D0 EA       BNE   $247C

2492-  AA          TAX
2493-  BD 21 E2    LDA   $E221,X
2496-  A6 FB       LDX   $FB
2498-  20 B9 E0    JSR   $E0B9    Epilogue
249B-  A9 DE       LDA   #$DE     #$DE
249D-  20 B6 E0    JSR   $E0B6
24A0-  A9 AA       LDA   #$AA     #$AA
24A2-  20 B6 E0    JSR   $E0B6
24A5-  A9 EB       LDA   #$EB     #$EB
24A7-  20 B6 E0    JSR   $E0B6
24AA-  A9 FF       LDA   #$FF     #$FF
24AC-  20 B6 E0    JSR   $E0B6
24AF-  BD 8E C0    LDA   $C08E,X  Prepare Latch for input
24B2-  BD 8C C0    LDA   $C08C,X  Strobe Data Latch
```

```
24B5-   60          RTS
--------------------------------
[$E0B6]
24B6-   18          CLC
24B7-   48          PHA
24B8-   68          PLA
24B9-   9D 8D C0    STA   $C08D,X  Load Data Latch
24BC-   1D 8C C0    ORA   $C08C,X  Strobe Data Latch
24BF-   60          RTS
--------------------------------
[$E0C0]                           De-Nibblize
24C0-   A0 00       LDY   #$00
24C2-   A2 56       LDX   #$56
24C4-   CA          DEX
24C5-   30 FB       BMI   $24C2
24C7-   B9 00 03    LDA   $0300,Y
24CA-   5E AA 0C    LSR   $0CAA,X
24CD-   2A          ROL
24CE-   5E AA 0C    LSR   $0CAA,X
24D1-   2A          ROL
24D2-   91 30       STA   ($30),Y    $30.31 buffer
24D4-   C8          INY
24D5-   C4 47       CPY   $47
24D7-   D0 EB       BNE   $24C4
24D9-   60          RTS
--------------------------------------------------------------------------------
[$E0DA]                           Read a Data Field
24DA-   A6 FB       LDX   $FB
24DC-   A0 20       LDY   #$20
24DE-   88          DEY
24DF-   F0 62       BEQ   $2543
24E1-   BD 8C C0    LDA   $C08C,X
24E4-   10 FB       BPL   $24E1
24E6-   49 D5       EOR   #$D5        #$D5
24E8-   D0 F4       BNE   $24DE
24EA-   EA          NOP
24EB-   BD 8C C0    LDA   $C08C,X
24EE-   10 FB       BPL   $24EB
24F0-   C9 AA       CMP   #$AA        #$AA
24F2-   D0 F2       BNE   $24E6
24F4-   EA          NOP
24F5-   BD 8C C0    LDA   $C08C,X
24F8-   10 FB       BPL   $24F5
24FA-   C9 AD       CMP   #$AD        #$AD
24FC-   D0 E8       BNE   $24E6

24FE-   A9 00       LDA   #$00
2500-   A0 56       LDY   #$56        256 + 86 nibbles per sector
2502-   88          DEY
2503-   84 4B       STY   $4B
2505-   BC 8C C0    LDY   $C08C,X
2508-   10 FB       BPL   $2505
250A-   59 00 E2    EOR   $E200,Y
250D-   A4 4B       LDY   $4B
250F-   99 AA 0C    STA   $0CAA,Y
2512-   D0 EE       BNE   $2502
2514-   84 4B       STY   $4B
2516-   BC 8C C0    LDY   $C08C,X
2519-   10 FB       BPL   $2516
251B-   59 00 E2    EOR   $E200,Y
251E-   A4 4B       LDY   $4B
2520-   99 00 03    STA   $0300,Y
2523-   C8          INY
2524-   D0 EE       BNE   $2514
```

```
2526-   BC 8C C0    LDY   $C08C,X
2529-   10 FB       BPL   $2526

252B-   D9 00 E2    CMP   $E200,Y
252E-   D0 13       BNE   $2543       bad checksum, $2543
2530-   BD 8C C0    LDA   $C08C,X     Look for Epilogue
2533-   10 FB       BPL   $2530
2535-   C9 DE       CMP   #$DE        #$DE
2537-   D0 0A       BNE   $2543
2539-   EA          NOP
253A-   BD 8C C0    LDA   $C08C,X
253D-   10 FB       BPL   $253A
253F-   C9 AA       CMP   #$AA        #$AA
2541-   F0 5E       BEQ   $25A1

2543-   38          SEC               Error return
2544-   60          RTS
--------------------------------------------------------------------------------
                                Read an Address Field

                                JSR from $E424
[$E145]
2545-   A6 FB       LDX   $FB
2547-   A0 FC       LDY   #$FC
2549-   84 4B       STY   $4B

254B-   C8          INY
254C-   D0 04       BNE   $2552
254E-   E6 4B       INC   $4B
2550-   F0 F1       BEQ   $2543
2552-   BD 8C C0    LDA   $C08C,X   Strobe Data
2555-   10 FB       BPL   $2552
2557-   C9 D5       CMP   #$D5      #$D5
2559-   D0 F0       BNE   $254B
255B-   EA          NOP
255C-   BD 8C C0    LDA   $C08C,X
255F-   10 FB       BPL   $255C
2561-   C9 AA       CMP   #$AA      #$AA
2563-   D0 F2       BNE   $2557
2565-   A0 03       LDY   #$03            Y := 3 for V/T/S
2567-   BD 8C C0    LDA   $C08C,X
256A-   10 FB       BPL   $2567
256C-   C9 96       CMP   #$96      #$96
256E-   D0 E7       BNE   $2557

                                  D5 AA 96 is Address field

[$E170]                           Now get Volume, Track, Sector, Checksum
2570-   A9 00       LDA   #$00
2572-   85 00       STA   $00
2574-   BD 8C C0    LDA   $C08C,X
2577-   10 FB       BPL   $2574
2579-   2A          ROL
257A-   85 4B       STA   $4B
257C-   BD 8C C0    LDA   $C08C,X
257F-   10 FB       BPL   $257C
2581-   25 4B       AND   $4B
2583-   99 01 00    STA   $0001,Y         $04 $03 $02 $01
2586-   45 00       EOR   $00               V   T   S   C
2588-   88          DEY
2589-   10 E7       BPL   $2572

[$E18B]
258B-   A8          TAY
```

```
258C-  D0 B5       BNE   $2543     Bad checksum, $2543

258E-  BD 8C C0    LDA   $C08C,X   Now check for EPILOGUE
2591-  10 FB       BPL   $258E       (but skip last byte $EB)
2593-  C9 DE       CMP   #$DE      #$DE
2595-  D0 AC       BNE   $2543
2597-  EA          NOP
2598-  BD 8C C0    LDA   $C08C,X
259B-  10 FB       BPL   $2598
259D-  C9 AA       CMP   #$AA      #$AA
259F-  D0 A2       BNE   $2543
25A1-  18          CLC
25A2-  60          RTS
--------------------------------------------------------------------------------
                                   Move Disk Arm to Requested Track

                                   Called from $E482 code (update prev track)

                                    0478 = Current track
                                     $FB = Requested track

                                   Uses:  $01, $4A, $4B

[$E1A3]
25A3-  A6 FB       LDX   $FB     Requested track
25A5-  CD 78 04    CMP   $0478   On the requested track?
25A8-  F0 53       BEQ   $25FD     Yes, return.

25AA-  A9 00       LDA   #$00    No
25AC-  85 4B       STA   $4B

25AE-  AD 78 04    LDA   $0478   Current Track
25B1-  85 01       STA   $01
25B3-  38          SEC
25B4-  E5 4A       SBC   $4A     Requested track (in 1/2 granularity)
25B6-  F0 33       BEQ   $25EB   Right track?  Yes, final stepper and RTS
25B8-  B0 07       BCS   $25C1
25BA-  49 FF       EOR   #$FF    Move inner 1 track
25BC-  EE 78 04    INC   $0478   Increment Current Track
25BF-  90 05       BCC   $25C6

25C1-  69 FE       ADC   #$FE    Move outer 1 track
25C3-  CE 78 04    DEC   $0478   Decrement Current Track

25C6-  C5 4B       CMP   $4B
25C8-  90 02       BCC   $25CC

25CA-  A5 4B       LDA   $4B
25CC-  C9 08       CMP   #$08
25CE-  B0 01       BCS   $25D1
25D0-  A8          TAY
25D1-  38          SEC
25D2-  20 EF E1    JSR   $E1EF    Stepper Motor
25D5-  B9 11 E2    LDA   $E211,Y
25D8-  20 00 E2    JSR   $E200    Wait (A = wait time)
25DB-  A5 01       LDA   $01
25DD-  18          CLC
25DE-  20 F2 E1    JSR   $E1F2    Stepper Motor
25E1-  B9 19 E2    LDA   $E219,Y
25E4-  20 00 E2    JSR   $E200    Wait (A = wait time)
25E7-  E6 4B       INC   $4B
25E9-  D0 C3       BNE   $25AE

[$E1EB]
```

```
25EB-   20 00 E2    JSR   $E200    Wait (A = wait time)
25EE-   18          CLC

[$E1EF]
25EF-   AD 78 04    LDA   $0478    Current Track
[$E1F2]
25F2-   29 03       AND   #$03
25F4-   2A          ROL
25F5-   05 FB       ORA   $FB
25F7-   AA          TAX
25F8-   BD 80 C0    LDA   $C080,X  Stepper motor
25FB-   A6 FB       LDX   $FB
25FD-   60          RTS
--------------------------------------------------------------------------------
25FE-   00          BRK
25FF-   00          BRK
--------------------------------------------------------------------------------
                                   Wait, A = wait time.
[$E200]
2600-   A2 11       LDX   #$11
2602-   CA          DEX
2603-   D0 FD       BNE   $2602
2605-   E6 49       INC   $49
2607-   D0 02       BNE   $260B
2609-   E6 48       INC   $48
260B-   38          SEC
260C-   E9 01       SBC   #$01
260E-   D0 F0       BNE   $2600
2610-   60          RTS
--------------------------------------------------------------------------------
2611-   01 30       ORA   ($30,X)
2613-   28          PLP
2614-   24 20       BIT   $20
2616-   1E 1D 1C    ASL   $1C1D,X
2619-   70 2C       BVS   $2647
261B-   26 22       ROL   $22
261D-   1F          ???
261E-   1E 1D 1C    ASL   $1C1D,X
2621-   96 97       STX   $97,Y
2623-   9A          TXS
2624-   9B          ???
2625-   9D 9E 9F    STA   $9F9E,X
2628-   A6 A7       LDX   $A7
262A-   AB          ???
262B-   AC AD AE    LDY   $AEAD
262E-   AF          ???
262F-   B2          ???
2630-   B3          ???
2631-   B4 B5       LDY   $B5,X
2633-   B6 B7       LDX   $B7,Y
2635-   B9 BA BB    LDA   $BBBA,Y
2638-   BC BD BE    LDY   $BEBD,X
263B-   BF          ???
263C-   CB          ???
263D-   CD CE CF    CMP   $CFCE
2640-   D3          ???
2641-   D6 D7       DEC   $D7,X
2643-   D9 DA DB    CMP   $DBDA,Y
2646-   DC          ???
2647-   DD DE DF    CMP   $DFDE,X
264A-   E5 E6       SBC   $E6
264C-   E7          ???
264D-   E9 EA       SBC   #$EA
264F-   EB          ???
```

```
2650-   EC ED EE    CPX   $EEED
2653-   EF          ???
2654-   F2          ???
2655-   F3          ???
2656-   F4          ???
2657-   F5 F6       SBC   $F6,X
2659-   F7          ???
265A-   F9 FA FB    SBC   $FBFA,Y
265D-   FC          ???
265E-   FD FE FF    SBC   $FFFE,X
--------------------------------------------------------------------------------
Table referenced from $F1C6          Text Screen PTRs

[$E261]
2661-   00          BRK
2662-   00          BRK
2663-   01
        01       ORA   ($01,X)
2665-   02          ???
2666-   02          ???
2667-   03          ???
2668-   03          ???
2669-   00          BRK
266A-   00          BRK
266B-   01
        01       ORA   ($01,X)
266D-   02          ???
266E-   02          ???
266F-   03          ???
2670-   03          ???
2671-   00          BRK
2672-   00          BRK
2673-   01
        01       ORA   ($01,X)
2675-   02          ???
2676-   02          ???
2677-   03          ???
2678-   03          ???
                                     Text Screen PTRs (?)
                                     Other 1/2 of $E261 table
[$E279]
2679-   00          BRK
267A-   80          ???
267B-   00          BRK
267C-   80          ???
267D-   00          BRK
267E-   80          ???
267F-   00          BRK
2680-   80          ???
2681-   28          PLP
2682-   A8          TAY
2683-   28          PLP
2684-   A8          TAY
2685-   28          PLP
2686-   A8          TAY
2687-   28          PLP
2688-   A8          TAY
2689-   50 D0       BVC   $265B
268B-   50 D0       BVC   $265D
268D-   50 D0       BVC   $265F
268F-   50 D0       BVC   $2661
--------------------------------------------------------------------------------
2691-   00          BRK
2692-   00          BRK
```

```
2693-  00          BRK
2694-  00          BRK
2695-  00          BRK
2696-  00          BRK
2697-  01 98       ORA   ($98,X)
2699-  99 02 03    STA   $0302,Y
269C-  9C          ???
269D-  04          ???
269E-  05 06       ORA   $06
26A0-  A0 A1       LDY   #$A1
26A2-  A2 A3       LDX   #$A3
26A4-  A4 A5       LDY   $A5
26A6-  07          ???
26A7-  08          PHP
26A8-  A8          TAY
26A9-  A9 AA       LDA   #$AA
26AB-  09 0A       ORA   #$0A
26AD-  0B          ???
26AE-  0C          ???
26AF-  0D B0 B1    ORA   $B1B0
26B2-  0E 0F 10    ASL   $100F
26B5-  11 12       ORA   ($12),Y
26B7-  13          ???
26B8-  B8          CLV
26B9-  14          ???
26BA-  15 16       ORA   $16,X
26BC-  17          ???
26BD-  18          CLC
26BE-  19 1A C0    ORA   $C01A,Y
26C1-  C1 C2       CMP   ($C2,X)
26C3-  C3          ???
26C4-  C4 C5       CPY   $C5
26C6-  C6 C7       DEC   $C7
26C8-  C8          INY
26C9-  C9 CA       CMP   #$CA
26CB-  1B          ???
26CC-  CC 1C 1D    CPY   $1D1C
26CF-  1E D0 D1    ASL   $D1D0,X
26D2-  D2          ???
26D3-  1F          ???
26D4-  D4          ???
26D5-  D5 20       CMP   $20,X
26D7-  21 D8       AND   ($D8,X)
26D9-  22          ???
26DA-  23          ???
26DB-  24 25       BIT   $25
26DD-  26 27       ROL   $27
26DF-  28          PLP
26E0-  E0 E1       CPX   #$E1
26E2-  E2          ???
26E3-  E3          ???
26E4-  E4 29       CPX   $29
26E6-  2A          ROL
26E7-  2B          ???
26E8-  E8          INX
26E9-  2C 2D 2E    BIT   $2E2D
26EC-  2F          ???
26ED-  30 31       BMI   $2720
26EF-  32          ???
26F0-  F0 F1       BEQ   $26E3
26F2-  33          ???
26F3-  34          ???
26F4-  35 36       AND   $36,X
26F6-  37          ???
```

	26F7-	38	SEC						
	26F8-	F8	SED						
	26F9-	39 3A 3B	AND	$3B3A,Y					
	26FC-	3C	???						
	26FD-	3D 3E 3F	AND	$3F3E,X					
	2700-	38	SEC						
	2701-	A6 FB	LDX	$FB					
	2703-	BD 8D C0	LDA	$C08D,X					
	2706-	BD 8E C0	LDA	$C08E,X					
	2709-	30 55	BMI	$2760					
	270B-	A5 2D	LDA	$2D					
	270D-	45 45	EOR	$45					
	270F-	45 46	EOR	$46					
	2711-	85 00	STA	$00					
	2713-	A9 FF	LDA	#$FF					
	2715-	9D 8F C0	STA	$C08F,X					
	2718-	DD 8C C0	CMP	$C08C,X					
	271B-	48	PHA						
	271C-	68	PLA						
	271D-	20 63 E3	JSR	$E363					
	2720-	20 63 E3	JSR	$E363					
	2723-	9D 8D C0	STA	$C08D,X					
	2726-	DD 8C C0	CMP	$C08C,X					
	2729-	EA	NOP						
	272A-	88	DEY						
	272B-	D0 F0	BNE	$271D					
	272D-	A9 D5	LDA	#$D5					
	272F-	20 75 E3	JSR	$E375					
	2732-	A9 AA	LDA	#$AA					
	2734-	20 75 E3	JSR	$E375					
	2737-	A9 96	LDA	#$96					
	2739-	20 75 E3	JSR	$E375					
	273C-	A5 2D	LDA	$2D					
	273E-	20 64 E3	JSR	$E364					
	2741-	A5 45	LDA	$45					
	2743-	20 64 E3	JSR	$E364					
	2746-	A5 46	LDA	$46					
	2748-	20 64 E3	JSR	$E364					
	274B-	A5 00	LDA	$00					
	274D-	20 64 E3	JSR	$E364					
	2750-	A9 DE	LDA	#$DE					
	2752-	20 75 E3	JSR	$E375					
	2755-	A9 AA	LDA	#$AA					
	2757-	20 75 E3	JSR	$E375					
	275A-	A9 EB	LDA	#$EB					
	275C-	20 75 E3	JSR	$E375					
	275F-	60	RTS						
	2760-	BD 8C C0	LDA	$C08C,X					
	2763-	60	RTS						
	2764-	48	PHA						
	2765-	4A	LSR						
	2766-	05 06	ORA	$06					
	2768-	9D 8D C0	STA	$C08D,X					
	276B-	DD 8C C0	CMP	$C08C,X					
	276E-	68	PLA						
	276F-	EA	NOP						
	2770-	EA	NOP						
	2771-	EA	NOP						
	2772-	09 AA	ORA	#$AA					
	2774-	EA	NOP						
	2775-	18	CLC						
	2776-	48	PHA						
	2777-	68	PLA						
	2778-	9D 8D C0	STA	$C08D,X					

```
277B-   1D 8C C0    ORA   $C08C,X
277E-   60          RTS
-------------------------------------------------------------------------------
                                  Similar to RWTS

                                  2D = Pascal Device (e.g., 4 = S6D1 usually)
                                  FC71 & FC63 Slot/Drive table initialized
                                  43   =
                                  FB   = Requested Slot# * 60  ($60)
                                  FA   = Requested Drive (1 or 2 (?))

[$E37F] RWTS
277F-   A0 02       LDY   #$02      $4F := 2
2781-   84 4F       STY   $4F
2783-   A0 04       LDY   #$04      $4E := 4
2785-   84 4E       STY   $4E

2787-   A6 2D       LDX   $2D       Device (4,5, 9,10, 11,12)
2789-   BD 71 FC    LDA   $FC71,X   Drive# (1 or 2) for Device
278C-   4A          LSR               4 = Drive1 = 00; 5 = Drive2 = 01;
278D-   66 43       ROR   $43       Set $43 (high bit) to Drive 1/2
278F-   BD 63 FC    LDA   $FC63,X   Slot# * 16 for Device 4,5,... (e.g., $60)
2792-   8D F8 05    STA   $05F8     BUG! #$FF if no previous successful I/O
                                    to 5 after boot.

                            *** This code is TRYING to compare the previous
                            *** disk I/O slot# to the requested slot#,
                            *** but it fails miserably!

2795-   C5 FB       CMP   $FB       Slot# requested
2797-   F0 14       BEQ   $27AD     Same as set up one?  Yes, $27AD

                                  This device not set up yet(?)
                                    in FC63,X and FC71,X

2799-   A6 FB       LDX   $FB       X := Requested Slot# * 16
279B-   85 FB       STA   $FB       Save previous set up(?) Slot# * 16
279D-   BD 8E C0    LDA   $C08E,X   Prepare latch for input

27A0-   A0 08       LDY   #$08      Wait for Requested drive to stop spinning
27A2-   BD 8C C0    LDA   $C08C,X   Strobe Data Latch
27A5-   DD 8C C0    CMP   $C08C,X   Strobe Data Latch
27A8-   D0 F6       BNE   $27A0
27AA-   88          DEY
27AB-   D0 F8       BNE   $27A5     Same value for 8 consecutive tries(?)
                                    Yes, drive has stopped spinning.

                           *** Here $FB is requested SLOT * 16,
                           *** but could be #$FF !!!!

27AD-   A6 FB       LDX   $FB       X := Requested Slot# * 16  ($60)
27AF-   BD 8E C0    LDA   $C08E,X   Prepare latch for input
27B2-   BD 8C C0    LDA   $C08C,X   Strobe Data Latch
27B5-   A0 08       LDY   #$08

                                  Wait for data to start changing
[$E3B7]
27B7-   BD 8C C0    LDA   $C08C,X     Strobe Data Latch
27BA-   48          PHA
27BB-   68          PLA
27BC-   48          PHA
27BD-   68          PLA
27BE-   DD 8C C0    CMP   $C08C,X     Strobe Data Latch
27C1-   D0 03       BNE   $27C6
```

```
27C3-   88          DEY
27C4-   D0 F1       BNE   $27B7     Wait for drive

27C6-   08          PHP
27C7-   BD 89 C0    LDA   $C089,X   ===== MotorOn =====
27CA-   A9 EF       LDA   #$EF      $48 := #$EF (used in timing)
27CC-   85 48       STA   $48
27CE-   A4 2D       LDY   $2D
27D0-   B9 71 FC    LDA   $FC71,Y
27D3-   C5 FA       CMP   $FA
27D5-   F0 06       BEQ   $27DD
27D7-   85 FA       STA   $FA
27D9-   28          PLP
27DA-   A0 00       LDY   #$00
27DC-   08          PHP

[$E3DD]
27DD-   6A          ROR
27DE-   90 05       BCC   $27E5
27E0-   BD 8B C0    LDA   $C08B,X  Engage Drive #2
27E3-   B0 03       BCS   $27E8
27E5-   BD 8A C0    LDA   $C08A,X  Engage Drive #1
27E8-   28          PLP
27E9-   08          PHP
27EA-   D0 08       BNE   $27F4    Drive up to speed yet? Yes, $27F4

27EC-   A0 07       LDY   #$07
27EE-   20 00 E2    JSR   $E200 [$2600] Wait a bit (A = wait time)
27F1-   88          DEY
27F2-   D0 FA       BNE   $27EE    Wait a bit longer

                                   --- Drive should be up to speed by now ---

[$E3F4]
27F4-   A5 45       LDA   $45     Requested Track
27F6-   A6 FB       LDX   $FB     Slot# * 16
27F8-   20 82 E4    JSR   $E482   Move Arm to Requested Track

27FB-   28          PLP
27FC-   D0 11       BNE   $280F
27FE-   A4 48       LDY   $48
2800-   10 0D       BPL   $280F

2802-   A0 12       LDY   #$12    Wait a bit
2804-   88          DEY
2805-   D0 FD       BNE   $2804
2807-   E6 49       INC   $49
2809-   D0 F7       BNE   $2802
280B-   E6 48       INC   $48
280D-   D0 F3       BNE   $2802   Y = $48 = $49 = 0 when done waiting
[$E40F]
280F-   A5 44       LDA   $44     Function:  0 = Read; 1 = Write; ....
2811-   C9 03       CMP   #$03               3 = INIT DISK
2813-   D0 03       BNE   $2818
2815-   4C BC E4    JMP   $E4BC [$28BC]

2818-   A8          TAY
2819-   C0 00       CPY   #$00
281B-   F0 03       BEQ   $2820

281D-   20 00 E0    JSR   $E000 [2400]

2820-   A0 60       LDY   #$60           Retry count
2822-   84 4D       STY   $4D            Retry count
```

```
2824-   20 45 E1    JSR   $E145 [2545]   Read an Address Field
2827-   90 1E       BCC   $2847          Error?
2829-   C6 4D       DEC   $4D              Yes
282B-   10 F7       BPL   $2824            Try again

282D-   A9 40       LDA   #$40
282F-   20 AA E4    JSR   $E4AA [28AA]
2832-   C6 4F       DEC   $4F
2834-   F0 21       BEQ   $2857
2836-   A9 04       LDA   #$04
2838-   85 4E       STA   $4E
283A-   A9 00       LDA   #$00          Track := 0
283C-   20 82 E4    JSR   $E482         Move Arm to Requested Track
283F-   A5 45       LDA   $45           Track
2841-   20 82 E4    JSR   $E482         Move Arm to Requested Track
2844-   4C 20 E4    JMP   $E420

[$E447]
2847-   A4 03       LDY   $03      Track just read
2849-   C4 45       CPY   $45      Track we want
284B-   F0 0F       BEQ   $285C
284D-   98          TYA
284E-   20 AA E4    JSR   $E4AA
2851-   C6 4E       DEC   $4E
2853-   D0 EA       BNE   $283F
2855-   F0 D6       BEQ   $282D

2857-   A2 40       LDX   #$40
2859-   4C 72 E4    JMP   $E472

                              On the proper track
[$E45C]
285C-   A5 46       LDA   $46      Sector we want
285E-   C5 02       CMP   $02      Sector we just read
2860-   D0 C7       BNE   $2829    Not the sector we want

                              SECTOR ADDRESS field we want!
2862-   A5 44       LDA   $44      Function:  0 = Read; 1 = Write; ....
2864-   D0 13       BNE   $2879    Write request, $2879

                              Read request

2866-   20 DA E0    JSR   $E0DA [24DA]   Read a DATA FIELD
2869-   B0 BE       BCS   $2829
286B-   20 C0 E0    JSR   $E0C0 [24C0]  De-Nibblize and start at $30.33
286E-   A2 00       LDX   #$00

2870-   18          CLC
2871-   24          BIT   $38
[$E472]
           38         SEC
2873-   A4 FB       LDY   $FB
2875-   B9 88 C0    LDA   $C088,Y  MotorOFF
2878-   60          RTS
                              Write request

2879-   20 2A E0    JSR   $E02A
287C-   90 F0       BCC   $286E
287E-   A2 10       LDX   #$10
2880-   B0 F0       BCS   $2872
--------------------------------------------------------------------------------
                              Move Arm to Requested Track

                                A = Requested Track
```

```
                                    X = Slot# * 16
                                   43 = Drive 0 or 1 flag.
                                   FB = requested Track (?)
                                    Drive is spinning

                                  Update Previous Track ($478) and
                                    Current Track for this disk drive.
                                  JMP to $E1A3
                                    (Move Arm to Requested Track)
[$E482]
2882-   0A          ASL
2883-   85 4A       STA   $4A      Track * 2 (1/2 track granularity)
2885-   20 A3 E4    JSR   $E4A3    Y := slot#  (e.g., 06)
2888-   A5 4A       LDA   $4A      Track
288A-   24 43       BIT   $43      Drive 0? or 1?
288C-   30 09       BMI   $2897
                                    Drive 0
288E-   BE 78 04    LDX   $0478,Y      Previous Track
2891-   99 78 04    STA   $0478,Y      Desired Track
2894-   4C 9D E4    JMP   $E49D
                                    Drive 1
2897-   BE F8 04    LDX   $04F8,Y      Previous Track
289A-   99 F8 04    STA   $04F8,Y      Desired Track
289D-   8E 78 04    STX   $0478    0478,0 === Previous Track (?)
28A0-   4C A3 E1    JMP   $E1A3 [25A3]
--------------------------------------------------------------------------------
                                  CONVERTSLOT16

                                  Convert slot# * 16 to slot#
                                    X = slot# * 16  (e.g., #$60)
                                  Returns:
                                    Y = slot#       (e.g., #$06)
[$E4A3]
28A3-   8A          TXA
28A4-   4A          LSR
28A5-   4A          LSR
28A6-   4A          LSR
28A7-   4A          LSR
28A8-   A8          TAY
28A9-   60          RTS
--------------------------------------------------------------------------------
[$E4AA]
28AA-   48          PHA
28AB-   20 A3 E4    JSR   $E4A3
28AE-   68          PLA
28AF-   0A          ASL
28B0-   24 43       BIT   $43
28B2-   30 04       BMI   $28B8
28B4-   99 78 04    STA   $0478,Y
28B7-   60          RTS
28B8-   99 F8 04    STA   $04F8,Y
28BB-   60          RTS
--------------------------------------------------------------------------------
                                  INIT DISK

[$E4BC]
28BC-   A9 AA       LDA   #$AA       #$AA
28BE-   85 06       STA   $06
28C0-   A9 FF       LDA   #$FF
28C2-   85 07       STA   $07
28C4-   A9 00       LDA   #$00
28C6-   85 45       STA   $45        Track
28C8-   A0 56       LDY   #$56
28CA-   99 A9 0C    STA   $0CA9,Y
```

```
28CD-   88          DEY
28CE-   D0 FA       BNE   $28CA
28D0-   99 00 03    STA   $0300,Y    $0300,Y    denibble?
28D3-   88          DEY
28D4-   D0 FA       BNE   $28D0
28D6-   A9 28       LDA   #$28
28D8-   20 AA E4    JSR   $E4AA
28DB-   A9 28       LDA   #$28
28DD-   85 05       STA   $05

28DF-   A5 45       LDA   $45      Track
28E1-   20 82 E4    JSR   $E482    Move Arm to Requested Track
28E4-   20 15 E5    JSR   $E515
28E7-   A2 08       LDX   #$08
28E9-   B0 23       BCS   $290E
28EB-   A9 30       LDA   #$30
28ED-   85 4F       STA   $4F
28EF-   38          SEC
28F0-   C6 4F       DEC   $4F
28F2-   F0 1A       BEQ   $290E
28F4-   20 45 E1    JSR   $E145
28F7-   B0 F6       BCS   $28EF
28F9-   A5 02       LDA   $02
28FB-   D0 F2       BNE   $28EF
28FD-   20 DA E0    JSR   $E0DA
2900-   B0 ED       BCS   $28EF
2902-   E6 45       INC   $45     Track := Track + 1
2904-   A5 45       LDA   $45
2906-   C9 23       CMP   #$23
2908-   90 D5       BCC   $28DF

290A-   A2 00       LDX   #$00

290C-   18          CLC
290D-   24          BIT   $38

[$E50E]
           38         SEC
290F-   A4 FB       LDY   $FB
2911-   B9 88 C0    LDA   $C088,Y ===== MotorOff =====
2914-   60          RTS
--------------------------------------------------------------------------------
[$E515]
2915-   A9 00       LDA   #$00
2917-   85 46       STA   $46
2919-   85 4B       STA   $4B
291B-   A0 80       LDY   #$80
291D-   D0 02       BNE   $2921
291F-   A4 05       LDY   $05
2921-   20 00 E3    JSR   $E300
2924-   B0 6F       BCS   $2995
2926-   A5 07       LDA   $07
2928-   20 B9 E0    JSR   $E0B9
292B-   EA          NOP
292C-   EA          NOP
292D-   A0 07       LDY   #$07
292F-   20 49 E0    JSR   $E049
2932-   E6 46       INC   $46
2934-   A5 46       LDA   $46
2936-   C9 10       CMP   #$10
2938-   90 E5       BCC   $291F
293A-   A0 0F       LDY   #$0F
293C-   84 46       STY   $46
293E-   A9 30       LDA   #$30
```

```
2940-  85 4D       STA   $4D
2942-  99 00 02    STA   $0200,Y
2945-  88          DEY
2946-  10 FA       BPL   $2942
2948-  A4 05       LDY   $05
294A-  20 95 E5    JSR   $E595
294D-  20 95 E5    JSR   $E595
2950-  20 95 E5    JSR   $E595
2953-  48          PHA
2954-  68          PLA
2955-  EA          NOP
2956-  88          DEY
2957-  D0 F1       BNE   $294A
2959-  20 45 E1    JSR   $E145
295C-  B0 22       BCS   $2980
295E-  A5 02       LDA   $02
2960-  F0 15       BEQ   $2977
2962-  A9 10       LDA   #$10
2964-  C5 05       CMP   $05
2966-  A5 05       LDA   $05
2968-  E9 01       SBC   #$01
296A-  85 05       STA   $05
296C-  C9 05       CMP   #$05
296E-  B0 10       BCS   $2980
2970-  38          SEC
2971-  60          RTS
2972-  20 45 E1    JSR   $E145
2975-  B0 05       BCS   $297C
2977-  20 DA E0    JSR   $E0DA
297A-  90 1A       BCC   $2996
297C-  C6 4D       DEC   $4D
297E-  D0 F2       BNE   $2972
2980-  20 45 E1    JSR   $E145
2983-  B0 0B       BCS   $2990
2985-  A5 02       LDA   $02
2987-  C9 0F       CMP   #$0F

2989-  D0 05       BNE   $2990
298B-  20 DA E0    JSR   $E0DA
298E-  90 85       BCC   $2915
2990-  C6 4D       DEC   $4D
2992-  D0 EC       BNE   $2980
2994-  38          SEC
2995-  60          RTS
2996-  A4 02       LDY   $02
2998-  B9 00 02    LDA   $0200,Y
299B-  30 DF       BMI   $297C
299D-  A9 FF       LDA   #$FF
299F-  99 00 02    STA   $0200,Y
29A2-  C6 46       DEC   $46

29A4-  10 CC       BPL   $2972
29A6-  A5 45       LDA   $45      Track(?)
29A8-  D0 0A       BNE   $29B4
29AA-  A5 05       LDA   $05
29AC-  C9 10       CMP   #$10
29AE-  90 E5       BCC   $2995
29B0-  C6 05       DEC   $05
29B2-  C6 05       DEC   $05
29B4-  18          CLC
29B5-  60          RTS
-------------------------------------------------------------------------------
                                 From $2C49
                                 Use heartbeat's $AD.AE
```

```
                                    For X = 0, JMP $E5C6
                                    For X = 3, JMP $E614
[$E5B6]
29B6-   8A          TXA
29B7-   0A          ASL
29B8-   A8          TAY
29B9-   B9 75 E6    LDA   $E675,Y
29BC-   85 AE       STA   $AE
29BE-   B9 74 E6    LDA   $E674,Y
29C1-   85 AD       STA   $AD
29C3-   6C AD 00    JMP   ($00AD)   e.g., JMP $E5C6
--------------------------------------------------------------------------------
                                    Read consecutive Sectors from disk

                                    See JMP at $E5C3
[$E5C6]
29C6-   86 44       STX   $44       $44 := Function: 0 = Read; 1 = Write;
29C8-   A9 00       LDA   #$00
29CA-   85 47       STA   $47
29CC-   A5 34       LDA   $34
29CE-   29 07       AND   #$07
29D0-   0A          ASL
29D1-   85 42       STA   $42       $42 := Sector
29D3-   A5 34       LDA   $34
29D5-   46 35       LSR   $35
29D7-   6A          ROR
29D8-   4A          LSR
29D9-   4A          LSR
29DA-   85 45       STA   $45       $45 := Track

[$E5DC]
29DC-   A5 33       LDA   $33       Sector Count
29DE-   D0 06       BNE   $29E6

29E0-   A5 32       LDA   $32
29E2-   F0 2C       BEQ   $2A10
29E4-   85 47       STA   $47       Amount of I/O needed to last sector

29E6-   20 7D E7    JSR   $E77D     Check Keyboard for Input
29E9-   A5 42       LDA   $42
29EB-   C9 08       CMP   #$08      Convert to Pascal Sector
29ED-   90 02       BCC   $29F1
29EF-   E9 08       SBC   #$08
29F1-   2A          ROL
29F2-   85 46       STA   $46
29F4-   20 7F E3    JSR   $E37F     RWTS
29F7-   8A          TXA
29F8-   D0 18       BNE   $2A12
29FA-   C6 33       DEC   $33       Sector count
29FC-   30 12       BMI   $2A10     Done, $E610

                                    Set up for next I/O
29FE-   E6 31       INC   $31       Buffer pointer...advance to next 256 bytes
2A00-   E6 42       INC   $42       Inc sector number
2A02-   A5 42       LDA   $42
2A04-   C9 10       CMP   #$10      End of Track?
2A06-   90 D4       BCC   $29DC       No, $E5DC (Do another Read)
2A08-   A9 00       LDA   #$00      Yes
2A0A-   85 42       STA   $42       Set Sector number to 0
2A0C-   E6 45       INC   $45       Inc Track number
2A0E-   D0 CC       BNE   $29DC     Do another Read

[$E610]
```

```
2A10-   A2 00       LDX   #$00      All done with this set of I/Os
2A12-   58          CLI
2A13-   60          RTS
--------------------------------------------------------------------------------
[$E614]
2A14-   86 44       STX   $44    Function:  0 = Read;
                                            1 = Write;
                                            3 = INIT Disk;
2A16-   20 7F E3    JSR   $E37F  RWTS
2A19-   B0 02       BCS   $2A1D
2A1B-   A9 00       LDA   #$00
2A1D-   58          CLI
2A1E-   AA          TAX
2A1F-   60          RTS
2A20-   68          PLA
2A21-   A8          TAY
2A22-   68          PLA
2A23-   AA          TAX
2A24-   A9 00       LDA   #$00
2A26-   48          PHA
2A27-   48          PHA
2A28-   8A          TXA
2A29-   48          PHA
2A2A-   98          TYA
2A2B-   48          PHA
---------------------------------
[$E62C]
2A2C-   A6 2D       LDX   $2D
2A2E-   BD 63 FC    LDA   $FC63,X   Slot * 16 for Device $2D  (e.g., $60)
2A31-   30 03       BMI   $2A36
2A33-   A2 00       LDX   #$00
2A35-   60          RTS

2A36-   A2 09       LDX   #$09
2A38-   60          RTS
--------------------------------------------------------------------------------
                                Return "# of Blocks" on a physical disk II

                                After INIT DISK call,
                                UNITREAD( BASE12, MP0E, 21, 0);
                                Set MP0E to #$0118 = 280 = # of BLOCKS

[$E639]
2A39-   20 2C E6    JSR   $E62C     Get slot * 16 using $2D
2A3C-   D0 0B       BNE   $2A49
2A3E-   A9 01       LDA   #$01
2A40-   A0 01       LDY   #$01
2A42-   91 30       STA   ($30),Y
2A44-   A9 18       LDA   #$18
2A46-   88          DEY
2A47-   91 30       STA   ($30),Y   $30.31 ==> $0118 = 280
2A49-   60          RTS                       # of blocks on a diskette
--------------------------------------------------------------------------------
2A4A-   20 2C E6    JSR   $E62C
2A4D-   D0 24       BNE   $2A73
2A4F-   A0 00       LDY   #$00
2A51-   98          TYA
2A52-   91 30       STA   ($30),Y
2A54-   C8          INY
2A55-   91 30       STA   ($30),Y
2A57-   C8          INY
2A58-   91 30       STA   ($30),Y
2A5A-   C8          INY
2A5B-   A9 01       LDA   #$01
```

```
2A5D-   91 30       STA   ($30),Y
2A5F-   C8          INY
2A60-   A9 10       LDA   #$10
2A62-   91 30       STA   ($30),Y
2A64-   C8          INY
2A65-   A9 00       LDA   #$00
2A67-   91 30       STA   ($30),Y
2A69-   C8          INY
2A6A-   A9 23       LDA   #$23
2A6C-   91 30       STA   ($30),Y
2A6E-   C8          INY
2A6F-   A9 00       LDA   #$00
2A71-   91 30       STA   ($30),Y
2A73-   60          RTS
--------------------------------------------------------------------------------
                                  Jmp table for Functions
[$E674]
2A74-
 C6 E5       Function 0 = Read
 C6 E5       Function 1 =            See $E5B6  JMP $E5C6
 2C E6       Function 2 =
 14 E6       Function 3 = INIT disk
 4A E6       Function 4 =
 2C E6       Function 5 =
 20 E6       Function 6 =
 2C E6       Function 7 =
 39 E6       Function 8 = Return "# of Blocks" on disk ($0118 = 280) $E639
--------------------------------------------------------------------------------

[$E686]
2A86    85 43
2A88-   B9 63 FC    LDA   $FC63,Y   Slot # for Pascal Device Y
2A8B-   85 B1       STA   $B1       $B1
2A8D-   A0 00       LDY   #$00
2A8F-   84 42       STY   $42
2A91-   84 B0       STY   $B0
2A93-   88          DEY
2A94-   B1 B0       LDA   ($B0),Y   $B0
2A96-   85 B0       STA   $B0
2A98-   8A          TXA
2A99-   0A          ASL
2A9A-   A8          TAY
2A9B-   B9 AE E6    LDA   $E6AE,Y
2A9E-   85 AD       STA   $AD
2AA0-   B9 AF E6    LDA   $E6AF,Y
2AA3-   85 AE       STA   $AE
2AA5-   20 AF 00    JSR   $00AF
2AA8-   8D D0 07    STA   $07D0
2AAB-   6C AD 00    JMP   ($00AD)

                              Some of the following looks like data

2AAE-   2C E7 BC    BIT   $BCE7
2AB1-   E6 EE       INC   $EE      not an instruction
2AB3-   E6 16       INC   $16
[$E6B5]
2AB5-   E7          ???           ???? nop ????
2AB6-   FE E6 FE    INC   $FEE6,X
2AB9-   E6 08       INC   $08
2ABB-   E7          ???           ???? nop ????
2ABC-   B0 32       BCS   $2AF0

2ABE-   A2 02       LDX   #$02
2AC0-   86 42       STX   $42
```

```
2AC2-  A5 34     LDA  $34
2AC4-  85 46     STA  $46
2AC6-  A5 35     LDA  $35
2AC8-  85 47     STA  $47
2ACA-  A5 30     LDA  $30
2ACC-  85 44     STA  $44
2ACE-  A5 31     LDA  $31
2AD0-  85 45     STA  $45
2AD2-  20 AF 00  JSR  $00AF   JSR $00AF  (AF: JMP xx xx, B0 & B1 target)
2AD5-  B0 19     BCS  $2AF0
2AD7-  E6 34     INC  $34
2AD9-  D0 02     BNE  $2ADD
2ADB-  E6 35     INC  $35
2ADD-  E6 31     INC  $31
2ADF-  E6 31     INC  $31
2AE1-  38        SEC
2AE2-  A5 33     LDA  $33
2AE4-  E9 02     SBC  #$02
2AE6-  85 33     STA  $33
2AE8-  90 04     BCC  $2AEE
2AEA-  05 32     ORA  $32
2AEC-  D0 D0     BNE  $2ABE
2AEE-  A9 00     LDA  #$00
2AF0-  AA        TAX
2AF1-  60        RTS
```

```
2AF2-  68        PLA
2AF3-  A8        TAY
2AF4-  68        PLA
2AF5-  AA        TAX
2AF6-  A9 00     LDA  #$00
2AF8-  48        PHA
2AF9-  48        PHA
2AFA-  8A        TXA
2AFB-  48        PHA
2AFC-  98        TYA
2AFD-  48        PHA
2AFE-  A9 01     LDA  #$01
2B00-  E9 00     SBC  #$00
2B02-  D0 EA     BNE  $2AEE
2B04-  A9 09     LDA  #$09
2B06-  D0 E8     BNE  $2AF0
2B08-  B0 E6     BCS  $2AF0
2B0A-  98        TYA
2B0B-  A0 01     LDY  #$01
2B0D-  91 30     STA  ($30),Y
2B0F-  88        DEY
2B10-  8A        TXA
2B11-  91 30     STA  ($30),Y
2B13-  A2 00     LDX  #$00
2B15-  60        RTS
2B16-  B0 D8     BCS  $2AF0
2B18-  98        TYA
2B19-  A0 01     LDY  #$01
2B1B-  91 30     STA  ($30),Y
2B1D-  88        DEY
2B1E-  8A        TXA
2B1F-  91 30     STA  ($30),Y
2B21-  A2 03     LDX  #$03
2B23-  86 42     STX  $42
2B25-  20 AF 00  JSR  $00AF
2B28-  B0 C6     BCS  $2AF0
2B2A-  90 C2     BCC  $2AEE
```

```
2B2C-   B0 C2       BCS   $2AF0
2B2E-   A5 34       LDA   $34
2B30-   85 46       STA   $46
2B32-   A5 35       LDA   $35
2B34-   85 47       STA   $47
2B36-   A2 02       LDX   #$02
2B38-   86 45       STX   $45
2B3A-   86 C9       STX   $C9
2B3C-   CA          DEX
2B3D-   86 42       STX   $42
2B3F-   CA          DEX
2B40-   86 44       STX   $44
2B42-   86 C8       STX   $C8
2B44-   20 AF 00    JSR   $00AF
2B47-   B0 A7       BCS   $2AF0
2B49-   A5 30       LDA   $30
2B4B-   85 CA       STA   $CA
2B4D-   A5 31       LDA   $31
2B4F-   85 CB       STA   $CB
2B51-   A0 00       LDY   #$00
2B53-   A2 02       LDX   #$02
2B55-   A5 33       LDA   $33
2B57-   C9 02       CMP   #$02
2B59-   B0 03       BCS   $2B5E
2B5B-   AA          TAX
2B5C-   A4 32       LDY   $32
2B5E-   84 CC       STY   $CC
2B60-   86 CD       STX   $CD
2B62-   20 4E D0    JSR   $D04E
2B65-   E6 34       INC   $34
2B67-   D0 02       BNE   $2B6B
2B69-   E6 35       INC   $35
2B6B-   E6 31       INC   $31
2B6D-   E6 31       INC   $31
2B6F-   38          SEC
2B70-   A5 33       LDA   $33
2B72-   E9 02       SBC   #$02
2B74-   85 33       STA   $33
2B76-   90 04       BCC   $2B7C
2B78-   05 32       ORA   $32
2B7A-   D0 B2       BNE   $2B2E
2B7C-   60          RTS
        ------------------------------------Check Keyboard for Input -----------
[$E77D]
2B7D-   08          PHP          Save Processor Status
2B7E-   48          PHA          Save A
2B7F-   98          TYA          Save Y
2B80-   48          PHA
2B81-   AD 00 C0    LDA   $C000  Keyboard Data Input
2B84-   85 E5       STA   $E5
2B86-   10 2F       BPL   $2BB7  Key pressed?  No, RTS
2B88-   8D 10 C0    STA   $C010    Yes, Clear KEYBOARD STROBE
2B8B-   29 7F       AND   #$7F   Strip high bit (Make it ASCII)
2B8D-   C9 05       CMP   #$05   CTRL-E pressed?
2B8F-   F0 26       BEQ   $2BB7    Yes, RTS  (Ignore CTRL-E)
2B91-   C9 04       CMP   #$04   CTRL-D pressed?
2B93-   D0 04       BNE   $2B99
2B95-   A9 80       LDA   #$80     Yes, A := #$80
2B97-   D0 06       BNE   $2B9F

[$E799]
2B99-   C9 13       CMP   #$13     No.
2B9B-   D0 08       BNE   $2BA5  CTRL-S pressed?
2B9D-   A9 40       LDA   #$40     Yes, A := #$40
```

```
2B9F-   45 E7       EOR   $E7    Set/Clear FLAG in $E7  (10000000 = CTRL-D)
2BA1-   85 E7       STA   $E7                           (01000000 = CTRL-S)
2BA3-   B0 12       BCS   $2BB7  CTRL-D bit set?  Yes, RTS
2BA5-   A4 E1       LDY   $E1    Input buffer ptr (?)
2BA7-   C8          INY
2BA8-   C0 1A       CPY   #$1A   At end of buffer? (?)
2BAA-   90 02       BCC   $2BAE
2BAC-   A0 00       LDY   #$00     Yes, so reset to beginning (?)
2BAE-   C4 E0       CPY   $E0    Buffer full?  (?)
2BB0-   F0 05       BEQ   $2BB7    Yes RTS (?)
2BB2-   99 00 0C    STA   $0C00,Y No, save character in Input buffer.
2BB5-   84 E1       STY   $E1       and update buffer ptr.

[$E7B7]
2BB7-   68          PLA             Restore Y
2BB8-   98          TYA
2BB9-   68          PLA             Restore A
2BBA-   28          PLP             Restore Processor Status
2BBB-   60          RTS           Return
-------------------------------------------------------------------------
[$E7BC
2BBC-   E0 04       CPX   #$04
2BBE-   B0 36       BCS   $2BF6
2BC0-   F0 31       BEQ   $2BF3
2BC2-   E0 01       CPX   #$01
2BC4-   B0 29       BCS   $2BEF
2BC6-   F0 2B       BEQ   $2BF3
2BC8-   A6 32       LDX   $32
2BCA-   F0 1E       BEQ   $2BEA
2BCC-   20 7D E7    JSR   $E77D   Check keyboard for input
2BCF-   A4 E0       LDY   $E0
2BD1-   C4 E1       CPY   $E1
2BD3-   F0 F7       BEQ   $2BCC
2BD5-   C8          INY
2BD6-   C0 1A       CPY   #$1A
2BD8-   90 02       BCC   $2BDC
2BDA-   A0 00       LDY   #$00
2BDC-   84 E0       STY   $E0
2BDE-   B9 00 0C    LDA   $0C00,Y
2BE1-   A0 00       LDY   #$00
2BE3-   91 30       STA   ($30),Y
2BE5-   CA          DEX
2BE6-   D0 E4       BNE   $2BCC
2BE8-   E6 31       INC   $31
2BEA-   C6 33       DEC   $33
2BEC-   10 DE       BPL   $2BCC
2BEE-   60          RTS
2BEF-   A5 E0       LDA   $E0
2BF1-   85 E1       STA   $E1
2BF3-   A2 00       LDX   #$00
2BF5-   60          RTS
2BF6-   38          SEC
2BF7-   A5 E0       LDA   $E0
2BF9-   E5 E1       SBC   $E1
2BFB-   A2 00       LDX   #$00
2BFD-   60          RTS
-------------------------------------------------------------------------------
                              From $141E, at start of SYSTEM.INTERP
                              and from $F904 JMP $E7FE while loading a segment

Before JSR:

 Set up $2D..$35 for I/O to read last 4 sectors of SYSTEM.INTERP to $BC00..BFFF.
```

```
  X   := 0
  $2D    := 4  Device (4 == Slot 6, Drive 1)
  $30.31 := Destination (e.g., $BC00)
  $32.33 := Length (#$400 = 2 blocks)
  $34.35 := Block # to start with
  $F2 "Virtual Disk" flag := #$FF

From $F904
  X := 00
  $2D := 4
  $30.31 := #$A388
  $32.33 := #$1C66
  $34.35 := #$0047
  $F2 "Virtual Disk" flag := #$00

From $xxxx
  X   := 3   (Format a Diskette)
  $2D := 4 or 5

From $EE9C
  X   := 8   ( (?) )
  $2D := 4  S6D1

[$E7FE]
2BFE-   86 3F       STX   $3F    Check Function value
2C00-   E0 09       CPX   #$09
2C02-   B0 17       BCS   $2C1B    Function > 8? Yes, $E81B

2C04-   A5 2D       LDA   $2D    Check Device value
2C06-   C9 04       CMP   #$04
2C08-   90 0C       BCC   $2C16
2C0A-   C9 06       CMP   #$06
2C0C-   90 16       BCC   $2C24
2C0E-   C9 09       CMP   #$09
2C10-   90 04       BCC   $2C16
2C12-   C9 0D       CMP   #$0D
2C14-   90 0E       BCC   $2C24

[$E816]
2C16-   85 EF       STA   $EF    Device = [0, 1, 2, 3]  [6, 7, 8]  [D, E,...]
2C18-   A2 18     LDX   #$18
2C1A-   2C A2 19    BIT   $19A2

[$E81B]           LDX #$19
2C1D-   A4 3F       LDY   $3F
2C1F-   A5 2D       LDA   $2D
2C21-   4C 00 FD    JMP   $FD00
--------------------------------
                                Device = DISK [4, 5]  [9, A, B, C]

                                Move 9 bytes from $2D..$35 to $36..$3E
                                  $2D $2E $2F $30 $31 $32 $33 $34 $35
                                  $36 $37 $38 $39 $3A $3B $3C $3D $3E

                                  $3F seems to also be input to $E847
[$E824]
2C24-   A2 08       LDX   #$08
2C26-   B5 2D       LDA   $2D,X
2C28-   95 36       STA   $36,X
2C2A-   CA          DEX             04 60 A6 00 BC 00 04 22 00
2C2B-   10 F9       BPL   $2C26      Input to "READ BLOCK OF SECTORS" $2D..$35
2C2D-   30 18       BMI   $2C47
--------------------------------
```

```
[$E82F]
2C2F-   E6 F6       INC   $F6
2C31-   C6 F7       DEC   $F7
2C33-   24 E6       BIT   $E6        Special flag, set by initial ClearScreen
2C35-   10 18       BPL   $2C4F        0 or Positive?  Yes, RTS
                                     $E6 is Negative

2C37-   24 38       BIT   $38        $38 ($2F)
2C39-   30 14       BMI   $2C4F          Negative, RTS
                                     $38 = 0 or Positive
2C3B-   20 00 FD    JSR   $FD00      Error routine (?)
2C3E-   A2 08       LDX   #$08
2C40-   B5 36       LDA   $36,X
2C42-   95 2D       STA   $2D,X
2C44-   CA          DEX
2C45-   10 F9       BPL   $2C40
--------------------------------
                                    $3F = function
[$E847]
2C47-   A6 3F       LDX   $3F            e.g., 0 = Read operation
2C49-   20 B6 E5    JSR   $E5B6
2C4C-   8A          TXA
2C4D-   D0 E0       BNE   $2C2F [$E82F]
2C4F-   60          RTS
--------------------------------------------------------------------------------
                                  Aux Section Sizes

                                  During Bootstrap:
                                    $FBE4:  03 04 01 05
                                            -- -- --
                                  03, 04, and 01 are indices into $E850 table

                                  03 04 01 identify 3 Sections of 64K Aux Bank
[$E850]      Index
2C50-   0F     0:
        5F     1: $0200..$BFFF    $5F is # of BLOCKS
        40     2:
        08     3: $D000..$CFFF    $08 is # of BLOCKS (Bank 1)
        18     4: $D000..$FFFF    $18 is # of BLOCKS (Bank 2)
2C55-   00     5:

------------------------------
                                  Aux Section Start Addresses

                                  This table also indexed by
                                     03 04 01 for Aux memory
                                     -- -- --
[$E856]
2C56-   02
        02     1: $0200..$BFFF
        40     2:
        D0     3: $D000..$CFFF
        D0     4: $D000..$FFFF
2C5B-   00
--------------------------------------------------------------------------------
                                  X == 0  Read
                                       1  Write
                                       4  Aux Cache

                                  Y == 1 for $F084 call to here.
[$E85C]
2C5C-   A5 2D       LDA   $2D      Device = 4?  (S6D1)
2C5E-   C9 04       CMP   #$04
2C60-   D0 0D       BNE   $2C6F          No,  $F8EE
```

```
2C62-   E0 01       CPX   #$01     Read?
2C64-   90 0C       BCC   $2C72            Yes, $E872
2C66-   F0 07       BEQ   $2C6F    Write?  Yes, $F8EE

                                   No.  (could be X = 4)
2C68-   24 2F       BIT   $2F          *** $2F set to #$60 in $F084 code ***
2C6A-   50 03       BVC   $2C6F    Normal I/O; S6D1 and Function > 1 and $2F(bit 6)=0
2C6C-   4C F3 E9    JMP   $E9F3    CACHE  I/O; S6D1 and Function > 1 and $2F(bit 6)=1
2C6F-   4C EE F8    JMP   $F8EE    Normal I/O; S5, S6D2, S6D1 Write, S6D1 Read and $EE=0


                                   Read function, Device 4 (S6D1)

                                   UNITWRITE( 13, -2, 25, 0) sets $EE to -1
[$E872]
2C72-   24 EE       BIT   $EE      $EE negative means Aux Cache is available
2C74-   10 F9       BPL   $2C6F    No Cache, $F8EE

2C76-   A5 E9       LDA   $E9      # of Cache records
2C78-   F0 F5       BEQ   $2C6F    No Cache, $F8EE

                                 ************************
                                 *  Look at Cache first! *
                                 ************************

                                 X = 0
[$E87A]
2C7A-   85 84       STA   $84    $84 := $E9 (# of Cache records)
2C7C-   A5 32       LDA   $32    $88.89 := $32.33 (I/O Length in bytes)
2C7E-   85 88       STA   $88
2C80-   A5 33       LDA   $33
2C82-   85 89       STA   $89    Length = 0?
2C84-   05 88       ORA   $88
2C86-   F0 4E       BEQ   $2CD6    Yes, RTS

2C88-   A5 34       LDA   $34    $8A.8B := $34.35 (I/O Block Request)
2C8A-   85 8A       STA   $8A
2C8C-   A5 35       LDA   $35
2C8E-   85 8B       STA   $8B
2C90-   A5 30       LDA   $30    $86.87 := $30.31 (I/O Dest)
2C92-   85 86       STA   $86
2C94-   A5 31       LDA   $31
2C96-   85 87       STA   $87
2C98-   86 8C       STX   $8C    $8C := X  (0 because this is Read function)
                                           (but $8C is Cache record block #
                                            for $E988)

                                 Begin Loop to process each cache record.
                                 Each cache block has up to 64 cache records
                                 in Aux Bank #0:  $D000..DFFF (bank 1)
[$E89A]
2C9A-   A9 00       LDA   #$00   $82.83 := #$0200 (Not input to $E988)
2C9C-   85 82       STA   $82
2C9E-   A9 02       LDA   #$02   Start at cache record 0 in a cache record block
2CA0-   85 83       STA   $83     (Cache record block is read into $0200 main ram)
2CA2-   A2 40       LDX   #$40   $7E := #$40      (Not input to $E988)
2CA4-   86 7E       STX   $7E    Max of 64 cache records in a cache record block

                               *** Re-Entry from Bottom of this routine
                                   Sometimes the I/O request spans multiple
                                   cache record blocks.

                                   $8C is the cache record block to read
[$E8A6]                            ------------------------------------------------
```

```
2CA6-   20 88 E9    JSR   $E988  Read  Aux $D000..DFFF; Bank 1; 1 block to $0200
                                  ------------------------------------------------
[$E8A9]
2CA9-   A0 06       LDY   #$06   (82.83)-->Cache record,6 is LastBlock(+1)
2CAB-   38          SEC
2CAC-   B1 82       LDA   ($82),Y      LastBlock(+1) (low)

2CAE-   E5 8A       SBC   $8A    $8E.8F := Last Block - $8A.8B (Requested Block)
2CB0-   85 8E       STA   $8E
2CB2-   C8          INY
2CB3-   B1 82       LDA   ($82),Y      LastBlock(+1) (high)
2CB5-   E5 8B       SBC   $8B
2CB7-   85 8F       STA   $8F
2CB9-   05 8E       ORA   $8E
2CBB-   F0 02       BEQ   $2CBF   Requested Block = LastBlock(+1)?  Yes, $E8BF

2CBD-   B0 34       BCS   $2CF3  [$E8F3] If Requested Block < Last Block + 1
                                         this might be entry for Requested
                                         Block.  Branch to $E8F3

                                 Haven't found Requested Block yet.
                                 Advance to next cache record.
[$E8BF]
2CBF-   18          CLC          $82.83 := $82.83 + 8 (size of a cache record)
2CC0-   A5 82       LDA   $82
2CC2-   69 08       ADC   #$08
2CC4-   85 82       STA   $82
2CC6-   90 02       BCC   $2CCA
2CC8-   E6 83       INC   $83

2CCA-   C6 84       DEC   $84    One less cache record to look at

[$E8CC]
2CCC-   F0 09       BEQ   $2CD7  [$E8D7]  No more to examine --> $E8D7

2CCE-   C6 7E       DEC   $7E    One less cache record in this cache record block
2CD0-   D0 D7       BNE   $2CA9  [$E8A9]  Check next record, $E8A9
2CD2-   E6 8C       INC   $8C    Advance to next cache record block in $D000..DFFF
2CD4-   D0 C4       BNE   $2C9A  [$E89A]  Get next cache record block, $E89A
2CD6-   60          RTS          Never fall through to this RTS!!!
        ----------------------------------------
                                 No more cache records to examine :(
[$E8D7]
2CD7-   A9 04       LDA   #$04   A := 4 so that $E922 always branches to $E92A.
2CD9-   D0 0B       BNE   $2CE6  [$E8E6] skip some code           Tricky stuff!
        --------------------------------
                               Requested Block is in a "gap".
                               It is >= LastBlock(+1) in the previous record,
                               and is < FirstBlock in the current record.
[$E8DB]
2CDB-   38          SEC          $8E.8F := 0 - $20.21
2CDC-   A9 00       LDA   #$00            (Positive difference between
2CDE-   E5 20       SBC   $20              Requested Block and FirstBlock)
2CE0-   85 8E       STA   $8E      This helps determine how much I/O to Disk
2CE2-   A9 00       LDA   #$00     is required.  It might be all of the gap or
2CE4-   E5 21       SBC   $21      just part of the gap.

                                   $8E.8F is the maximum I/O we could ever do
                                   for this "gap" using Requested Block
                                   as the first block.
        --------------------------------
[$E8E6]
2CE6-   85 8F       STA   $8F   $8F := (could be #$4 from $E8D7)  Tricks!
2CE8-   A5 8A       LDA   $8A   $34.35 := $8A.8B (Requested Block #)
```

```
2CEA-   85 34       STA   $34
2CEC-   A5 8B       LDA   $8B
2CEE-   85 35       STA   $35
2CF0-   38          SEC         C := 1 (no find, so need to read from Disk)
2CF1-   B0 27       BCS   $2D1A [$E91A] skip some code
--------------------------------
                                Almost Found entry for Requested Block!
[$E8F3]
2CF3-   A0 00       LDY   #$00    $20.21 := $8A.8B (Requested) -
2CF5-   38          SEC                    FirstBlock in current cache record
2CF6-   A5 8A       LDA   $8A
2CF8-   F1 82       SBC   ($82),Y
2CFA-   85 20       STA   $20
2CFC-   C8          INY
2CFD-   A5 8B       LDA   $8B
2CFF-   F1 82       SBC   ($82),Y

[$E901]
2D01-   85 21       STA   $21
2D03-   90 D6       BCC   $2CDB   Requested < FirstBlock so there must
                                  not be a cache record for this Requested Block
                                  Continue at $$E8DB.

                                  ************************************
                                  * Now we truly have found the entry! *
                                  ************************************

                                  The current cache record includes at
                                  least the first Requested Block.
[$E905]
2D05-   C8          INY           Get Cache record SectionOffset (Word 1)
2D06-   18          CLC           $34.35 := SectionOffset +
2D07-   B1 82       LDA   ($82),Y           (Requested - First Block)
2D09-   65 20       ADC   $20
2D0B-   85 34       STA   $34       In other words, calculate the address
2D0D-   C8          INY             relative to the start of the Aux Section
2D0E-   B1 82       LDA   ($82),Y   where the real data is found.
2D10-   65 21       ADC   $21
2D12-   85 35       STA   $35

2D14-   C8          INY           $85 := Aux Section Index (0, 1, 2, 3, 4,...)
2D15-   B1 82       LDA   ($82),Y        for Aux Section:  (03 04 01)
2D17-   85 85       STA   $85
2D19-   18          CLC
--------------------------------
[$E91A]
2D1A-   08          PHP           C=0 means we found entry; C=1 not found
2D1B-   A2 00       LDX   #$00    Calculate $8E.8F in byte granularity
2D1D-   A5 8E       LDA   $8E     $8E.8F := 2 * 8E.8F (but 8E is not updated)
2D1F-   0A          ASL           A := 2 * $8E and C set/cleared
2D20-   26 8F       ROL   $8F     For the "find" case, 2 * $8F = 0 since #$5F
2D22-   D0 06       BNE   $2D2A     is the max I/O (in blocks) for 1 cache record

                                  $8F <> 0 for the tricks case at $E8D7,
                                  and for extremely large "gaps".

                                  A = max blocks I/O based on this cache record.

2D24-   C5 89       CMP   $89     Compare with I/O Length Request
2D26-   F0 06       BEQ   $2D2E   i.e., make sure we don't do too much I/O
2D28-   90 04       BCC   $2D2E   This could be just a subset for this cache record
                                  If $88.89 > this cache record we scale it back and will
                                  do another I/O after this one.
```

```
2D2A-   A6 88       LDX   $88    X.A := $88.89     I/O Length
2D2C-   A5 89       LDA   $89

2D2E-   86 8E       STX   $8E    $8E.8F := X.A     Length
2D30-   85 8F       STA   $8F
2D32-   86 32       STX   $32    $32.33 := X.A     Length
2D34-   85 33       STA   $33
2D36-   A5 86       LDA   $86    $30.31 := $86.87  Dest
2D38-   85 30       STA   $30
2D3A-   A5 87       LDA   $87
2D3C-   85 31       STA   $31
2D3E-   28          PLP
2D3F-   B0 0C       BCS   $2D4D [$E94D] Get block the "normal" way

                                Get block from Aux Memory
2D41-   A9 00       LDA   #$00   $8D := Read Aux
2D43-   85 8D       STA   $8D
2D45-   A6 85       LDX   $85    Aux Section Index (0, 1, 2, 3, 4, 5, ...)
                                  for Aux Section: (03 04 01)
                                ---------------------
2D47-   20 A0 E9    JSR   $E9A0  Read from Aux Memory
                                ---------------------
2D4A-   4C 52 E9    JMP   $E952  skip some code --> $E952

[$E94D]
2D4D-   A2 00       LDX   #$00   Read function
                                -----------------------
2D4F-   20 EE F8    JSR   $F8EE  Read from Disk
                                -----------------------
[$E952]
2D52-   20 7D E7    JSR   $E77D  check keyboard for input

2D55-   38          SEC          $88.89 := $88.89 - $8E.8F
2D56-   A5 88       LDA   $88              I/O Length request -
2D58-   E5 8E       SBC   $8E              # of blocks from just performed I/O
2D5A-   85 88       STA   $88
2D5C-   A5 89       LDA   $89
2D5E-   E5 8F       SBC   $8F    If $88.89 is 0, then we have completed the I/O.
2D60-   85 89       STA   $89
2D62-   05 88       ORA   $88    If not 0, then more I/O is needed.
2D64-   F0 1E       BEQ   $2D84    0? Yes, RTS

[$E966]
2D66-   18          CLC          No.
2D67-   A5 87       LDA   $87    $87:= $87 + $8F
2D69-   65 8F       ADC   $8F           (Update I/O Dest)
2D6B-   85 87       STA   $87
2D6D-   A5 8F       LDA   $8F    $8A.8B := $8A.8B + ($8F / 2)
2D6F-   4A          LSR                 (Update Request Block)
2D70-   18          CLC
2D71-   65 8A       ADC   $8A
2D73-   85 8A       STA   $8A
2D75-   90 02       BCC   $2D79
2D77-   E6 8B       INC   $8B
[$E979]
2D79-   4C A6 E8    JMP   $E8A6 [E8A6] Start again near the top
--------------------------------------------------------------------------------
                                Return Bank Section Size
                                Input:
                                  X = index into $FBE4 table.

                                Output:
                                  A.X = size (in blocks)
                                  X is always 0
```

```
                                $FBE4:   03 04 01 05 by BOOTSTRAP code.
                                         3 values = 1 bank in Aux.
                                         05 is terminator

                                Value from $FBE4 is index into $E850 table:

                                X   $FBE4  $E850
                                0     03     08   $D000..$CFFF  (Bank 1)
                                1     04     18   $D000..$FFFF  (Bank 2)
                                2     01     5F   $0200..$BFFF

[$E97C]
2D7C-   BC E4 FB    LDY   $FBE4,X   Memory Banks table
2D7F-   B9 50 E8    LDA   $E850,Y   A := Section size (in blocks)
2D82-   A2 00       LDX   #$00      X := 0 (High part (?))
2D84-   60          RTS
-------------------------------------------------------------
                            Read/Write Aux memory

                          $E985 Write Aux $D000; Bank 1; 1 block
                                Update cache record block
                                  $8C = Cache block record index
                                  e.g., $8C = $03, means $D600.
                                There are up to 64 x 4 word cache records
                                in each cache record block.
                                A cache record:
                                  1. FirstBlock
                                  2. Offset into Section (Blocks)
                                  3. Aux Section index (0, 1, 2, 3, 4, 5, ...)
                                                       03 04 01 03 04 01 ...
                                  4. LastBlock(+1)

                          $E988 Read  Aux $D000; Bank 1; 1 block
                                Get cache record block
                                  $8C (same as Write $E985)

                          $E994 is Alternate Entry Point Read (or Write)
                                1. Read  #$0200 bytes
                                         From: Aux Section X + Offset ($34)
                                         To:   #$0200

                                2. Write #$0200 bytes
                                         From: #$0200
                                         To:   Aux Section X + Offset ($34)

                                8D     = 0(Read) or 1(Write)
                                $34    = Offset
                                X      = Aux section (0, 1, 2)

                          $E9A0 is Alternate Entry Point Read (or Write)
                                1. Read $32.33 bytes
                                        From: Aux Section X + Offset ($34)
                                        To:   $30.31

                                2. Write $32.33 bytes
                                        From: $30.31
                                        To:   Aux Section X + Offset ($34)

                                8D     = 0(Read) or 1(Write)
                                $34    = Offset
                                X      = Aux section (0, 1, 2)
                                $32.33 = Length
                                $30.31 = Dest / Src
```

```
[$E985]
2D85-   A9 01     LDA   #$01    *** WRITE Aux $D000 Bank 1
2D87-   2C         BIT   $00A9       #$0200 bytes, from $0200

[$E988]
      A9 00       LDA   #$00    *** READ Aux $D000 Bank 1
                                      #$0200 bytes, to $0200
2D8A-   85 8D       STA   $8D   $8D := 00 = READ; 01=WRITE
2D8C-   A2 00       LDX   #$00  X := 0 == Aux Section 03 ($D000..DFFF, Bank 1)
2D8E-   86 35       STX   $35   $35 := 0 never used(?)
2D90-   A5 8C       LDA   $8C   $34 := $8C (Offset)
2D92-   85 34       STA   $34

[$E994]                         *** ALTERNATE ENTRY TO R/W AUX MEMORY
2D94-   A9 02       LDA   #$02  $32.33 := #$0200 (Length in bytes)
2D96-   A0 00       LDY   #$00
2D98-   85 33       STA   $33
2D9A-   84 32       STY   $32
2D9C-   85 31       STA   $31   $30.31 := #$0200 (Dest (Read) / Src (Write))
2D9E-   84 30       STY   $30

[$E9A0]                         *** ALTERNATE ENTRY TO R/W AUX MEMORY
                                    Memory Banks table   X :=  0, 1, 2
2DA0-   BC E4 FB    LDY   $FBE4,X   Memory Banks table   Y := 03 04 01 [05]

2DA3-   A5 32       LDA   $32   $0CAC.0CAD := $32.33
2DA5-   8D AC 0C    STA   $0CAC   Length (in bytes) for xfer
2DA8-   A5 33       LDA   $33
2DAA-   8D AD 0C    STA   $0CAD
2DAD-   A5 30       LDA   $30   $0CAA.0CAB := $30.31
2DAF-   8D AA 0C    STA   $0CAA   (DEST / SRC) for Aux Memory xfer
2DB2-   A5 31       LDA   $31
2DB4-   8D AB 0C    STA   $0CAB

2DB7-   A5 34       LDA   $34   $34 := $34 * 2 (Calc offset into Aux section)
2DB9-   0A          ASL                 Y = 00 01 02 03 04
2DBA-   79 56 E8    ADC   $E856,Y  $E856,Y = 02 02 40 D0 D0 (Start of Aux section)
2DBD-   8D AF 0C    STA   $0CAF  $0CAE.0CAF := $xx00 + 256 * Offset  e.g., $D600
2DC0-   A9 00       LDA   #$00     (SRC / DEST) for Aux Memory xfer
2DC2-   8D AE 0C    STA   $0CAE

2DC5-   A0 17       LDY   #$17      Move code to $0CB0
2DC7-   B9 DB E9    LDA   $E9DB,Y     (So it can execute while
2DCA-   99 B0 0C    STA   $0CB0,Y      Aux Bank switched ram is active
2DCD-   88          DEY                in $D000..$FFFF. See $C009 switch.)
2DCE-   10 F7       BPL   $2DC7

2DD0-   BC E4 FB    LDY   $FBE4,X   Y := Memory Banks Section   Y := 03 04 01 [05]
2DD3-   BD 24 FC    LDA   $FC24,X   A := Bank to access in Aux. AppleWin: A := 0;
2DD6-   A6 8D       LDX   $8D       X := 00 = READ; 01 = WRITE
[$E9D8]
2DD8-   4C B0 0C    JMP   $0CB0     Execute $0CB0 code
-----------------------------------------------------
                                Code is moved to $0CB0 and called from above.

                                Transfer from (to) Aux memory.

                                A = Aux Bank (64K size) = 0 AppleWin
                                X = 0 (READ) or 1 (WRITE)
                                Y = Aux Bank section (03, 04, or 01)

                                $0CAA.0CAB = Dest (Read) / Src  (Write)
```

```
                                  $0CAE.0CAF = Src  (Read) / Dest (Write)

                                  $0CAC.0CAD = Length (in bytes)
[$E9DB]
2DDB-   8D 09 C0    STA   $C009   Use Aux page 0 1 & Aux $D000..FFFF
                                    ($0000..$01FF & $D000..$FFFF in Aux memory)
2DDE-   8E 50 07    STX   $0750   Save Read / Write (why?)
2DE1-   8D 73 C0    STA   $C073   Select bank using ACCUMULATOR (0 for APPLEWIN)
2DE4-   8C 51 07    STY   $0751   Save Aux Bank Section (03 04 01) (why?)
2DE7-   20 40 00    JSR   $0040   Transfer From / To Aux Memory
                                    See BOOT0.TXT
2DEA-   A9 00       LDA   #$00    Select Aux bank #0
2DEC-   8D 73 C0    STA   $C073   Select Aux bank #0
2DEF-   8D 08 C0    STA   $C008   Use Main $00.$1FF (zpage + stack + RAM)
2DF2-   60          RTS
--------------------------------------------------------------------------------

                                  UNITWRITE( 13,  0, 25, 0);
                                  UNITWRITE( 13, ++, 25, 0);
                                                $30.31 (2 words of input)
                                                  Disk Block Request,
                                                  I/O Block Count

                                  CACHE stuff

                                  $7C.7D = Disk Block#
                                  $7B    = (???)

                                  $F084 -> $E85C -> $E9F3

                                  $FC = 3 * number of 64K banks in Aux mem
                                        See bootstrap at $0B54
                                        For AppleWin, $FC = 3,
                                          so 1 Aux Bank with 3 sections

                                  Y = 1 = index to high part of Param2 word 0.

                                  $EA.EB = 00 00 see UNITWRITE( 13, -3, 25, 0)
                                           # of blocks written to this Aux Section already
                                  $E9    = # cache records in Aux $D000..DFFF

                                  $E8    = 01    see UNITWRITE( 13, -3, 25, 0)
                                     With $E8 = 01, skip first Aux Section
                                       in first Aux Bank, which is
                                       03 ($D000.DFFF bank 1)
[$E9F3]
2DF3-   C8          INY           Y := 2;  Param2 word 1 byte 0
2DF4-   A5 30       LDA   $30     $80.81 := $30.31 Param2 Word 0
2DF6-   85 80       STA   $80               Disk Block Request
2DF8-   A5 31       LDA   $31
2DFA-   85 81       STA   $81

2DFC-   B1 30       LDA   ($30),Y  $8E.8F := PARAM2 word 1
2DFE-   85 8E       STA   $8E         (I/O Block Count;
2E00-   C8          INY                Total I/O requested for this Aux bank.
2E01-   B1 30       LDA   ($30),Y      Should be <= Max size of this Aux bank.)
2E03-   85 8F       STA   $8F

2E05-   05 8E       ORA   $8E
2E07-   F0 51       BEQ   $2E5A    $8E.8F = 0?  Yes, $EA5A (Return Aux bank(s) total size)


[$EA09]                         BEGIN LOOP for one Section of Aux I/O
```

```
                                  Calculate $22.$23 (Size of I/O) for JSR $EAD8

2E09-   A6 E8       LDX   $E8       Aux section (0, 1, 2) (for 03 04 01)
2E0B-   E4 FC       CPX   $FC       3 * number of 64K Aux banks
2E0D-   B0 4B       BCS   $2E5A     Any more Aux sections to process?
                                      No, $EA5A
                                    Do 1 of the 3 Aux Bank Sections
                                    ($D000..DFFF, $D000..FFFF, $0200..BFFF)
2E0F-   20 7C E9    JSR   $E97C     Get # of blocks in this Aux section
2E12-   38         SEC               A := $08 or $18 or $5F
2E13-   E5 EA      SBC   $EA         Subtract $EA.EB from A
2E15-   A8         TAY                 ($EA.EB is # of blocks already read)
2E16-   8A         TXA               Save in Y.X  (X=high)
2E17-   E5 EB      SBC   $EB         A := (remaining) block count
2E19-   AA         TAX                     (e.g., #$18, D000..FFFF)

2E1A-   C4 8E      CPY   $8E     If (remaining) blocks in this Aux section >
2E1C-   E5 8F      SBC   $8F       total I/O requested
2E1E-   90 04      BCC   $2E24   then
2E20-   A4 8E      LDY   $8E       Y.X := total I/O requested
2E22-   A6 8F      LDX   $8F         (i.e., Y.X could be partial Aux section size)

2E24-   84 22      STY   $22     $22.23 := Y.X
2E26-   86 23      STX   $23       Amount of I/O for this Aux section

                                  $E8    = Aux Section (0, 1, 2) (for 03 04 01)
                                  $E9    = # of cache records
                                  $EA.EB = # of blocks written to this Aux Section
                                  $7C.7D = Disk Block #
                                  $22.23 = I/O Block Count
                                  ----------------------------
2E28-   20 D8 EA    JSR   $EAD8   JSR $EAD8  LOTS OF CODE HERE
                                  ----------------------------
                                  ...But no disk i/o yet

2E2B-   38         SEC            $8E.8F := $8E.8F - $22.23
2E2C-   A5 8E      LDA   $8E       Remaining I/O after this
2E2E-   E5 22      SBC   $22       Aux section I/O completed
2E30-   85 8E      STA   $8E
2E32-   A5 8F      LDA   $8F
2E34-   E5 23      SBC   $23
2E36-   85 8F      STA   $8F
                                  -------------------------------------
2E38-   20 8F EA    JSR   $EA8F   Read Blocks from DISK to Aux Section
                                  -------------------------------------

2E3B-   A6 E8       LDX   $E8     Aux Bank section index (0, 1, 2) (03, 04, 01)
2E3D-   20 7C E9    JSR   $E97C   Get SIZE of this Aux section
                                  X := 0 (always)
                                  A := size (in blocks)
2E40-   E4 EB       CPX   $EB     $EB > 0?
2E42-   90 08       BCC   $2E4C     $EB > 0 --> $EA4C Hopefully never taken!?
2E44-   D0 0E       BNE   $2E54   BNE can't be taken with X always 0.

2E46-   C5 EA       CMP   $EA     SIZE of Aux section = $EA?
2E48-   F0 02       BEQ   $2E4C     Yes, $EA4C (we filled this Aux section)
2E4A-   B0 08       BCS   $2E54   SIZE of Aux section > $EA?
                                    Yes, $EA54 (did not fill Aux section)
                                               (should be done (?))
                                  No. (Can't ever just fall through!?)
                                      (Would mean we "overfilled" this section.)
[$EA4C]
2E4C-   E6 E8       INC   $E8     E8 := E8 + 1; Increment to next Aux section
2E4E-   A9 00       LDA   #$00      (Possibly to next 64K bank section)
```

```
2E50-   85 EA       STA   $EA     Reset $EA.EB := 0;
2E52-   85 EB       STA   $EB           # of blocks written to this Aux section

2E54-   A5 8F       LDA   $8F
2E56-   05 8E       ORA   $8E
2E58-   D0 AF       BNE   $2E09 JMP LOOP (if more to read)
--------------------------------
                                Done with all 64K banks.
                                Note fallthrough from above too,
                                and branch from EA07.
[$EA5A]
                                Return Aux Bank(s) size (less first section)

2E5A-   A9 00       LDA   #$00   $8E.8F := 0;
2E5C-   85 8E       STA   $8E
2E5E-   85 8F       STA   $8F
2E60-   A6 FC       LDX   $FC    X := Aux Bank Sections (3* # of 64K banks)
2E62-   F0 18       BEQ   $2E7C  $FC = 0?  Yes, $EA7C (No Aux memory)

[$EA64]                         AUX ADD LOOP:
2E64-   CA          DEX           Dec to next Aux section in $FBE4: 03 04 01 05
2E65-   E4 E8       CPX   $E8     Process all (except first, 03)
2E67-   90 13       BCC   $2E7C   $EA7C when complete
2E69-   86 85       STX   $85     X is index into $FBE4 table:  03 04 01 05
                                  ************************************
2E6B-   20 7C E9    JSR   $E97C   A.X := # of blocks in this Aux section
                                  ************************************
                                    X   $FBE4  $E850
                                    0     03     08  $C000..$CFFF
                                    1     04     18  $D000..$FFFF
                                    2     01     5F  $0200..$BFFF

                                  A.X = Size in Blocks for this Aux Section
                                        (X is ALWAYS 0)

2E6E-   18          CLC           $8E.8F := $8E.8F + Size of this Aux Section
2E6F-   65 8E       ADC   $8E                       ($5F, $18, etc.)
2E71-   85 8E       STA   $8E
2E73-   8A          TXA
2E74-   65 8F       ADC   $8F
2E76-   85 8F       STA   $8F
2E78-   A6 85       LDX   $85     X := Aux Bank Section we just processed
2E7A-   90 E8       BCC   $2E64  JMP Aux ADD LOOP (Always)

2E7C-   A0 00       LDY   #$00   To here after summing $5F + $18
2E7E-   38          SEC          Now subtract $EA.EB
2E7F-   A5 8E       LDA   $8E
2E81-   E5 EA       SBC   $EA
2E83-   91 80       STA   ($80),Y   and save results in 2nd param first word
2E85-   C8          INY
2E86-   A5 8F       LDA   $8F
2E88-   E5 EB       SBC   $EB
2E8A-   91 80       STA   ($80),Y
2E8C-   A2 00       LDX   #$00
2E8E-   60          RTS
-----------------------------------------------------------------------
                               Read Blocks from DISK to CACHE

                               $22.23 = # of blocks I/O for this Aux section
                               $7C.7D = first Disk Block#
                               $EA.EB = # of blocks written to this Aux Section
                               $E8 = Aux Section index (0, 1, 2) for (03, 04, 01)

[$EA8F]
```

```
2E8F-   A5 22       LDA   $22
2E91-   F0 02       BEQ   $2E95
2E93-   E6 23       INC   $23
                              LOOP:
2E95-   A5 7C       LDA   $7C     $34.35 := Disk Block#
2E97-   85 34       STA   $34
2E99-   A5 7D       LDA   $7D
2E9B-   85 35       STA   $35

2E9D-   A9 04       LDA   #$04    $2D := #$4  (S6D1)
2E9F-   85 2D       STA   $2D
2EA1-   A9 02       LDA   #$02    $32.33 :=  #$0200 Length (bytes)
2EA3-   85 33       STA   $33     $30.31 :=  #$0200 Destination
2EA5-   85 31       STA   $31
2EA7-   A2 00       LDX   #$00    X := 0 = Read
2EA9-   86 32       STX   $32
2EAB-   86 30       STX   $30
2EAD-   86 2F       STX   $2F     $2F := 0

[$EAAF]                           ----------------------
2EAF-   20 EE F8    JSR   $F8EE   Normal Disk Read I/O
                                  (not Virtual (2F = 0))
                                  ----------------------

2EB2-   A5 EA       LDA   $EA     $34.35 := $EA.EB
2EB4-   85 34       STA   $34
2EB6-   A5 EB       LDA   $EB
2EB8-   85 35       STA   $35
2EBA-   A2 01       LDX   #$01    $8D := #$01 = Write to Aux
2EBC-   86 8D       STX   $8D
2EBE-   A6 E8       LDX   $E8     X := $E8 = Memory Bank Section
                                             Index (3, 4, 1)
                                  ---------------------------------------
2EC0-   20 94 E9    JSR   $E994   ALTERNATE entry to R/W AUX
                                    Write #$0200 bytes
                                      From: #$0200
                                      To:   Aux Section X + Offset ($34)
                                  ---------------------------------------
2EC3-   E6 EA       INC   $EA     $EA.EB := $EA.EB + 1
2EC5-   D0 02       BNE   $2EC9       INC # of blocks written to this section
2EC7-   E6 EB       INC   $EB
2EC9-   E6 7C       INC   $7C     $7C.7D := 7C.7D + 1
2ECB-   D0 02       BNE   $2ECF       Advance to next disk block
2ECD-   E6 7D       INC   $7D

2ECF-   C6 22       DEC   $22     $22.23 := $22.23 - 1
2ED1-   D0 C2       BNE   $2E95   JMP LOOP
2ED3-   C6 23       DEC   $23
2ED5-   D0 BE       BNE   $2E95   JMP LOOP
2ED7-   60          RTS
-------------------------------------------------------------------------------
                                Insert Cache Record into Aux Bank #0,
                                  Aux Section 3:  $D000..DFFF (bank 1)

                                $E8     Aux Section Index (0, 1, 2, 3, 4, 5, ...)
                                                          (03 04 01)

                                $E9     # of cache records in Aux Bank #0,
                                        Aux Section 03: $D000..$D1FF (bank 1)
                                        Reset to 0 by UNITWRITE( 13, -3, 25, 0)

                                $22.23  Amount of Disk I/O (in blocks) needed
                                        for this Aux Section (could be a partial
                                        amount).
```

```
                                $7C.7D  Disk Block Request

                                $EA.EB  # of blocks written to Aux Section

                                $7B     Lucky to be 0 first time!

                                $18..$1F become one cache record
[$EAD8]
2ED8-   A5 E9       LDA   $E9   $84 := $E9 = # of cache records in Aux
2EDA-   85 84       STA   $84       ($84 is decremented during search)
2EDC-   A5 7C       LDA   $7C   $18.19 := $7C.7D (Requested Disk Block)
2EDE-   85 18       STA   $18
2EE0-   A5 7D       LDA   $7D
2EE2-   85 19       STA   $19
2EE4-   A5 EA       LDA   $EA   $1A.1B := $EA.EB (# of blocks already written)
2EE6-   85 1A       STA   $1A
2EE8-   A5 EB       LDA   $EB
2EEA-   85 1B       STA   $1B
2EEC-   A5 E8       LDA   $E8   $1C := $E8 (Aux Section index: 0, 1, 2)
2EEE-   85 1C       STA   $1C                                  (03 04 01)

2EF0-   18          CLC         $1E.1F := $7C.7D + $22.23
2EF1-   A5 7C       LDA   $7C             Last Disk Block(+1) for this I/O
2EF3-   65 22       ADC   $22
2EF5-   85 1E       STA   $1E
2EF7-   A5 7D       LDA   $7D
2EF9-   65 23       ADC   $23
2EFB-   85 1F       STA   $1F

2EFD-   A9 00       LDA   #$00  $1D := 0;
2EFF-   85 1D       STA   $1D   $8C := 0; cache record block index
2F01-   85 8C       STA   $8C             0 = $D000; 1 = $D200; ...

                                  Scan through cache records in Bank #0,
                                  Section 03, $D000..DFFF (bank 1) until the
                                  Requested Disk block ($7C.7D) is less than
                                  the FirstBlock in a cache record.

                                  If 64 cache records are searched, then $8C
                                  increments to the next block in $D000..DFFF.
                                  $82.83 is the pointer to the current cache
                                    record in the buffer at $0200.

                                Search until Requested Disk Block < FirstBlock,
                                then exit OUTER LOOP.

[$EB03]                         OUTER LOOP:
2F03-   A9 00       LDA   #$00    $82.83 := $0200
2F05-   85 82       STA   $82
2F07-   A9 02       LDA   #$02
2F09-   85 83       STA   $83     ------------------------------------
2F0B-   20 88 E9    JSR   $E988   Read Aux ($D000 + $8C * 512) Bank 1,
                                     #$0200 bytes to RAM at $0200
                                  ------------------------------------
2F0E-   A9 40       LDA   #$40    $7E := #$40
2F10-   85 7E       STA   $7E       Max of 64 cache records in a block

[$EB12]                         Loop:
2F12-   A5 84       LDA   $84       If no (more) cache records to search
2F14-   F0 42       BEQ   $2F58        JMP MiddleNextLoop

2F16-   A0 01       LDY   #$01      If Requested Disk Block <
2F18-   A5 7D       LDA   $7D          FirstBlock in current Aux record
```

```
2F1A-   D1 82       CMP   ($82),Y
2F1C-   90 3A       BCC   $2F58
2F1E-   D0 07       BNE   $2F27
2F20-   88          DEY
2F21-   A5 7C       LDA   $7C
2F23-   D1 82       CMP   ($82),Y
2F25-   90 31       BCC   $2F58          JMP Middle Next Loop

                                  The "=" case is interesting.
                                  Seems we are allowing multiple FirstBlocks
                                  with the same values!?

2F27-   C6 84       DEC   $84        $84 := $84 - 1  Try next cache record
2F29-   A9 08       LDA   #$08       $82.83 := $82.83 + 8
2F2B-   18          CLC                        Advance to next cache record
2F2C-   65 82       ADC   $82
2F2E-   85 82       STA   $82
2F30-   90 02       BCC   $2F34
2F32-   E6 83       INC   $83
2F34-   C6 7E       DEC   $7E        $7E := $7E - 1 (do max of 64 in block)
2F36-   D0 DA       BNE   $2F12    Jmp Loop (Process next cache record)

2F38-   E6 8C       INC   $8C      $8C := $8C + 1 (More than 64 cache records,
                                     so advance to next block in $D000..DFFF)
2F3A-   D0 C7       BNE   $2F03  Jmp OUTER LOOP

                                (Falling through should never happen (?))

[$EB3C]                         BEGIN ANOTHER LOOP
2F3C-   A2 07       LDX   #$07     Retrieve saved entry from end of previous block
2F3E-   B5 10       LDA   $10,X      From: $10..$17
2F40-   95 18       STA   $18,X      To:   $18..$1F
2F42-   CA          DEX
2F43-   10 F9       BPL   $2F3E

2F45-   C6 84       DEC   $84
2F47-   E6 8C       INC   $8C      $8C := $8C + 1 (cache block index)
2F49-   A9 00       LDA   #$00     $82.83 := #$0200
2F4B-   85 82       STA   $82
2F4D-   A9 02       LDA   #$02
2F4F-   85 83       STA   $83      ------------------------
2F51-   20 88 E9    JSR   $E988    Read Aux $D000 Bank 1,
                                    #$0200 bytes, to $0200
                                   ------------------------
2F54-   A9 40       LDA   #$40     $7E := #$40
2F56-   85 7E       STA   $7E

----------------------------------------
[$EB58] From middle of first loop to here
----------------------------------------

                                One of the following is true:

                                  1. There are 0 cache records in $D000..DFFF
                                  2. Requested Block < FirstBlock
                                  3. Requested Block >= all FirstBlocks
                                       in $D000..DFFF

2F58-   A6 84       LDX   $84          If 0 cache records to search
2F5A-   F0 33       BEQ   $2F8F [$EB8F]     JMP $EB8F

2F5C-   E4 7E       CPX   $7E          If this block is not completely full,
2F5E-   90 0D       BCC   $2F6D [$EB6D]    JMP $EB6D
```

```
                                The block we want to insert into already has
                                64 cache records.  Save the last one,
                                move all the ones downward from "insertion
                                point", insert the Requested Disk record,
                                and then loop back to insert the saved
                                entry to the beginning of next block.
                                Repeat as many times as necessary.

2F60-   A2 07       LDX   #$07
2F62-   BD F8 02    LDA   $02F8,X   From:  $02F8..$02FF
2F65-   95 10       STA   $10,X     To:      $10..  $17
2F67-   CA          DEX
2F68-   10 F8       BPL   $2F62

2F6A-   A6 7E       LDX   $7E       $7B := $7E - 1
2F6C-   CA          DEX                    $7E is # of cache record "slots" from
                                              "insertion point" to end of block
[$EB6D]
2F6D-   86 7B       STX   $7B       Note: only place where $7B is "set".

2F6F-   8A          TXA             CC.CD := 8 * $7B      (Size)
2F70-   A2 00       LDX   #$00                             bytes for BT at $D024
2F72-   86 CD       STX   $CD
2F74-   0A          ASL
2F75-   0A          ASL
2F76-   0A          ASL
2F77-   26 CD       ROL   $CD
2F79-   85 CC       STA   $CC
2F7B-   A5 82       LDA   $82       $C8.C9 := $82.83      (SRC)
2F7D-   A4 83       LDY   $83
2F7F-   85 C8       STA   $C8
2F81-   84 C9       STY   $C9
2F83-   69 08       ADC   #$08      $CA.CB := $C8.C9 + 8  (DEST)
2F85-   90 01       BCC   $2F88
2F87-   C8          INY
2F88-   85 CA       STA   $CA
2F8A-   84 CB       STY   $CB
                              Shift existing entries down to open up a slot

                                $C8.C9  SRC
                                $CA.CB  DEST
                                $CC.CD  Size
                                --------------
2F8C-   20 24 D0    JSR   $D024     Block Transfer
                                --------------
[$EB8F]
2F8F-   A0 07       LDY   #$07      Xfer $18..$1F to Aux cache buffer (at $0200)
2F91-   B9 18 00    LDA   $0018,Y     $18.19 = FirstBlock (Requested Disk Block)
2F94-   91 82       STA   ($82),Y     $1A.1B = SectionOffset
2F96-   88          DEY               $1C    = Aux Section (0, 1, 2)
2F97-   10 F8       BPL   $2F91       $1D    = 0
                                      $1E.1F = LastBlock(+1)

                                $8C = Cache block index into $D000..DFFF
                                --------------------------------
2F99-   20 85 E9    JSR   $E985     Write AUX $D000; Bank 1; 1 block
                                --------------------------------
2F9C-   38          SEC             $84 := $84 - $7B
2F9D-   A5 84       LDA   $84                  $7B was lucky to be 00 first time
2F9F-   E5 7B       SBC   $7B
2FA1-   85 84       STA   $84       If $84 <> 0 then
2FA3-   D0 97       BNE   $2F3C   JMP BEGIN ANOTHER LOOP

2FA5-   E6 E9       INC   $E9     Increment # of cache records in Aux
```

```
2FA7-   18          CLC
2FA8-   60          RTS
-------------------------------------------------------------------------------
                                  Some sort of error routine
                                    -- OR --
                                  Please Insert Scenario Diskette...

                                Input:
                                  CBA.CBB := #0 #0
                                  CD2.CD3 := #$FF1C = "Please Insert ..." msg
                                  C4      := "B" == insertion for disk#
                                  CE9.CEA := #$FF3F = "Press ~ To Continue"

[$EBA9]
2FA9-   A0 00       LDY   #$00     Clear $200 (#$20 #$00 values)
2FAB-   A9 20       LDA   #$20
2FAD-   99 00 02    STA   $0200,Y
2FB0-   88          DEY
2FB1-   A9 00       LDA   #$00
2FB3-   99 00 02    STA   $0200,Y
2FB6-   88          DEY
2FB7-   D0 F2       BNE   $2FAB

2FB9-   A9 20       LDA   #$20     Clear $300 (#$20 #$00 values)
2FBB-   99 00 03    STA   $0300,Y
2FBE-   88          DEY
2FBF-   A9 00       LDA   #$00
2FC1-   99 00 03    STA   $0300,Y
2FC4-   88          DEY
2FC5-   D0 F2       BNE   $2FB9

2FC7-   A9 02       LDA   #$02     $0A.0B := #$020A
2FC9-   85 0B       STA   $0B
2FCB-   A9 0A       LDA   #$0A
2FCD-   85 0A       STA   $0A

2FCF-   A9 4C       LDA   #$4C     $0E := #$4C
2FD1-   85 0E       STA   $0E      $10 := #$4C
2FD3-   85 10       STA   $10
2FD5-   A9 BA       LDA   #$BA     A.X := #$0CBA  PTR to ???? no message ????
2FD7-   A2 0C       LDX   #$0C
2FD9-   20 2B EC    JSR   $EC2B    JSR $EC2B

[$EBDC]
2FDC-   A9 D2       LDA   #$D2     A.X := #$0CD2  PTR to $FF1C message
2FDE-   A2 0C       LDX   #$0C         $FF1C "Please Insert SCENARIO Diskette %c"
2FE0-   20 2B EC    JSR   $EC2B    JSR $EC2B

2FE3-   A9 E9       LDA   #$E9     A.X := #$0CE9  PTR to $FF3F message
2FE5-   A2 0C       LDX   #$0C         $FF3F "Press ~ To Continue"
2FE7-   20 2B EC    JSR   $EC2B    JSR $EC2B

2FEA-   A2 00       LDX   #$00
2FEC-   8E 02 02    STX   $0202
2FEF-   A9 09       LDA   #$09
2FF1-   8D 03 02    STA   $0203
2FF4-   A9 28       LDA   #$28
2FF6-   8D 04 02    STA   $0204
2FF9-   A9 05       LDA   #$05
2FFB-   8D 05 02    STA   $0205
2FFE-   A9 3F       LDA   #$3F
3000-   8D 08 02    STA   $0208
3003-   A9 02       LDA   #$02
3005-   85 31       STA   $31
```

```
3007-   A9 00       LDA   #$00
3009-   85 30       STA   $30
300B-   A9 03       LDA   #$03    Function := #03 = Display a Window (including Frame)
300D-   20 26 EC    JSR   $EC26   JSR $EC26  ($F15B)

3010-   AD 00 C0    LDA   $C000   Get a key
3013-   10 FB       BPL   $3010
3015-   8D 10 C0    STA   $C010
3018-   C9 8D       CMP   #$8D    <CR>?
301A-   D0 F4       BNE   $3010   Go back and Get a key

301C-   A5 EC       LDA   $EC
301E-   85 30       STA   $30
3020-   A5 ED       LDA   $ED
3022-   85 31       STA   $31
3024-   A9 02       LDA   #$02   Function #2 = Paint All Windows

[$EC26]
3026-   85 32       STA   $32    Function
3028-   4C 5B F1    JMP   $F15B
--------------------------------

[$EC2B]
302B-   86 01       STX   $01      $00.01 := X.A   ($0CBA, $0CD2, $0CE9)
302D-   85 00       STA   $00
302F-   A0 00       LDY   #$00
3031-   84 06       STY   $06
3033-   84 07       STY   $07

3035-   B1 00       LDA   ($00),Y  $02.03 := ($00.01),0 ($00.01),1
3037-   85 02       STA   $02
3039-   C8          INY
303A-   B1 00       LDA   ($00),Y
303C-   85 03       STA   $03
303E-   05 02       ORA   $02
3040-   F0 64       BEQ   $30A6    $02.03 = 0000?  Yes, $ECA6
                                   No.  e.g., $FF1C
3042-   C8          INY
3043-   84 09       STY   $09
3045-   24 F1       BIT   $F1  (??)
3047-   A0 00       LDY   #$00

3049-   B1 02       LDA   ($02),Y  Loop:      Transfer Message to ($0A),Y
304B-   F0 18       BEQ   $3065
304D-   E6 07       INC   $07
304F-   C9 25       CMP   #$25       "%" is insertion character in raw message
3051-   F0 5F       BEQ   $30B2 [$ECB2]
                                     "return" to $EC53 or to $EC5F
[$EC53]
3053-   A4 06       LDY   $06
3055-   24 F1       BIT   $F1  (??)
3057-   C4 0E       CPY   $0E
3059-   B0 02       BCS   $305D
305B-   91 0A       STA   ($0A),Y
305D-   C8          INY
305E-   C8          INY

[$EC5F]
305F-   84 06       STY   $06
3061-   A4 07       LDY   $07
3063-   D0 E4       BNE   $3049    JMP Loop

3065-   24 F1       BIT   $F1      $F1 flag
3067-   70 0D       BVS   $3076      bit6 set?  Yes, $EC76
```

```
3069-   A9 A0       LDA   #$A0
306B-   A4 06       LDY   $06

306D-   C4 0E       CPY   $0E
306F-   B0 35       BCS   $30A6
3071-   91 0A       STA   ($0A),Y
3073-   C8          INY
3074-   D0 F7       BNE   $306D

[$EC76]
3076-   38          SEC
3077-   A5 0E       LDA   $0E
3079-   E5 06       SBC   $06
307B-   90 29       BCC   $30A6
307D-   4A          LSR
307E-   29 7E       AND   #$7E
3080-   F0 24       BEQ   $30A6
3082-   48          PHA
3083-   18          CLC
3084-   65 0A       ADC   $0A
3086-   85 04       STA   $04
3088-   A9 00       LDA   #$00
308A-   65 0B       ADC   $0B
308C-   85 05       STA   $05
308E-   A4 06       LDY   $06
3090-   88          DEY
3091-   88          DEY

[$EC92]
3092-   B1 0A       LDA   ($0A),Y   Transfer Message
3094-   91 04       STA   ($04),Y   "Please Insert Scenario Diskette...."
3096-   88          DEY
3097-   88          DEY
3098-   10 F8       BPL   $3092

309A-   68          PLA
309B-   A8          TAY
309C-   A9 20       LDA   #$20    ' '
309E-   D0 02       BNE   $30A2

30A0-   91 0A       STA   ($0A),Y
30A2-   88          DEY
30A3-   88          DEY
30A4-   10 FA       BPL   $30A0

30A6-   A5 10       LDA   $10
30A8-   18          CLC
30A9-   65 0A       ADC   $0A
30AB-   85 0A       STA   $0A
30AD-   90 02       BCC   $30B1
30AF-   E6 0B       INC   $0B
30B1-   60          RTS
-----------------------------------

                              Found "%" in message
[$ECB2]
30B2-   C8          INY
30B3-   B1 02       LDA   ($02),Y
30B5-   E6 07       INC   $07
30B7-   AA          TAX
30B8-   A4 09       LDY   $09
30BA-   B1 00       LDA   ($00),Y
30BC-   E6 09       INC   $09
30BE-   E0 63       CPX   #$63
```

```
30C0-   F0 91       BEQ   $3053    "return"

30C2-   E0 68       CPX   #$68
30C4-   D0 05       BNE   $30CB
30C6-   AA          TAX
30C7-   A9 00       LDA   #$00
30C9-   F0 17       BEQ   $30E2
30CB-   E0 62       CPX   #$62
30CD-   D0 09       BNE   $30D8
30CF-   A0 01       LDY   #$01
30D1-   20 43 FE    JSR   $FE43    JSR $FE43
30D4-   A2 01       LDX   #$01
30D6-   D0 19       BNE   $30F1
30D8-   E6 09       INC   $09
30DA-   C8          INY
30DB-   E0 78       CPX   #$78
30DD-   F0 27       BEQ   $3106
30DF-   AA          TAX
30E0-   B1 00       LDA   ($00),Y
30E2-   20 C8 FA    JSR   $FAC8    JSR $FAC8
30E5-   A2 05       LDX   #$05

30E7-   CA          DEX
30E8-   F0 07       BEQ   $30F1
30EA-   BD AA 0C    LDA   $0CAA,X        $0CAA
30ED-   C9 30       CMP   #$30
30EF-   F0 F6       BEQ   $30E7

30F1-   A4 06       LDY   $06
30F3-   24 F1       BIT   $F1     (??)

30F5-   BD AA 0C    LDA   $0CAA,X        $0CAA
30F8-   C4 0E       CPY   $0E
30FA-   B0 02       BCS   $30FE
30FC-   91 0A       STA   ($0A),Y
30FE-   C8          INY
30FF-   C8          INY
3100-   CA          DEX
3101-   10 F2       BPL   $30F5

3103-   4C 5F EC    JMP   $EC5F   "return"

3106-   AA          TAX
3107-   B1 00       LDA   ($00),Y
3109-   20 3D FE    JSR   $FE3D
310C-   A2 03       LDX   #$03
310E-   D0 E1       BNE   $30F1
-----------------------------------------------------------------------
                          Processing UNITNUMBER #$D UNITREAD/WRITE
                          UNITWRITE( UNIT, ARRAY, LENGTH, BLOCK#, MODE)
                                     $2D   $30.31 $32.33  $34.35  $2E.2F
                                                  FUNCTION
[$ED10]
3110-   A9 00       LDA   #$00
3112-   85 F0       STA   $F0
3114-   A5 32       LDA   $32       Function  e.g., #$1F=Read SYSTEM.RELOC to $1000
3116-   C9 23       CMP   #$23        > 34?
3118-   B0 05       BCS   $311F       Yes, $ED1F ERROR
311A-   0A          ASL
311B-   A8          TAY             Y := index into $ED29 table
311C-   20 22 ED    JSR   $ED22     JSR $ED22
311F-   A6 F0       LDX   $F0
3121-   60          RTS
```

```
---------------------------------
[$ED22]
3122-   B9 2A ED    LDA   $ED2A,Y
3125-   48          PHA
3126-   B9 29 ED    LDA   $ED29,Y
3129-   48          PHA             $ED29 Table starts here (!)
312A-   60          RTS
---------------------------------
[$ED2B]
        5A F1    [$01] (01) $F15B Initial CLEARSCREEN, return KANMP04[] config, sets $F1
        6E ED    [$02] (02) $ED6F DRAWSCR() UNITWRITE(BASE12, WINCHAIN^, 2, 0)
        5A F1    [$03] (03) $F15B In P01060E after "S)tart Game" displayed
        5A F1    [$04] (04)
        5A F1    [$05] (05)
        1E ED    [$06] (06) xxxxx
        1E ED    [$07] (07) xxxxx
        1E ED    [$08] (08) xxxxx
        1E ED    [$09] (09) xxxxx
        52 EE    [$0A] (10) $EE53 RANDOM() function call
        88 EE    [$0B] (11)       in KANJIREA, after title screen displayed. and S)tart M)ake displ
        97 EE    [$0C] (12) $EE98
        5A F1    [$0D] (13) $F15B after random chars chosen to display
        5A F1    [$0E] (14) $F15B processing '200.CHARSET'
        5A F1    [$0F] (15)
        DC EE    [$10] (16) $EEDD SearchBF()
        5A F1    [$11] (17)
        5A F1    [$12] (18) $F15B processing '200.MONSTERS'
        A4 EE    [$13] (19) $EEA5 after displaying 4 MONSTER pics waiting for S) or M)
        7A F0    [$14] (20) $F07B "Formatting Diskette"
        9B EE    [$15] (21) $EE9C After "Formatting Diskette" and before COPYING disk
        1E EE    [$16] (22) $EE1F
        5A F1    [$17] (23) $F15B
        31 EE    [$18] (24) $EE32
        83 F0    [$19] (25) $F084 Cache stuff
        E8 EF    [$1A] (26) $EFE9 Concatonate strings
        B9 F0    [$1B] (27) $F0BA from P010055() with string messages
        1E ED    [$1C] (28) xxxxx
        1E F0    [$1D] (29) $F01F
        5A F1    [$1E] (30) $F15B From CENTSTR() and P010022()
        79 ED    [$1F] (31) $ED7A
        76 EF    [$20] (32) $EF77 ASCII.HUFF to $0D00..$0EFF
        91 EF    [$21] (33) $EF92 Message string processing
        5A F1    [$22] (34) $F15B DrawMaze code (?)  See P010B04().
-------------------------------------------------------------------------------
[$ED6F]                         From above for DRAWSCR() and DRAWSCR2()
        A5 30       LDA   $30
        85 EC       STA   $EC
        A5 31       LDA   $31
        85 ED       STA   $ED
        4C 5B F1    JMP   $F15B
-------------------------------------------------------------------------------
                                UNITREAD( BASE12, MP04, 31, MP03)
                                          $2D    $30.31 $32.33  $34.35  $2E.2F

                                  MP04 = 0 Read SYSTEM.RELOC
                                  MP04 = 1
                                  MP04 = 2 in SHOPS P01080D()
                                  MP04 = 4 Convert from "Virtual"
[$ED7A]
317A-   A0 00       LDY   #$00      Check input buffer (BASE.9) 0, 1, etc.
317C-   B1 30       LDA   ($30),Y   #BASE.9 in WIZARDRY Proc 1
317E-   F0 2C       BEQ   $31AC [$EDAC] 0?  Yes, $EDAC (...first time) Read SYSTEM.RELOC
3180-   C9 02       CMP   #$02
3182-   F0 0F       BEQ   $3193         2?  Yes, $3193
```

```
3184-   B0 12       BCS   $3198         3+? Yes, $3198
                                        1?  Yes, $ED88
[$ED88]
3186-   A9 80       LDA   #$80      "Virtual Disk" flag := #$80
3188-   85 F2       STA   $F2       $F2 := #$80
318A-   85 2F       STA   $2F       $2F := #$80
318C-   A2 04       LDX   #$04      Function := 4 = Convert from "Virtual"
318E-   86 2D       STX   $2D       $2D := #$4
3190-   4C EE F8    JMP   $F8EE

[$ED93]
3193-   A9 00       LDA   #$00  Reset "Virtual Disk" flag for SYSTEM.RELOC
3195-   85 F2       STA   $F2   This happens when "M)aking" Scenario
3197-   60          RTS           disks.  And M)ove Save Games.

3198-   C9 03       CMP   #$03
319A-   F0 0B       BEQ   $31A7
319C-   A2 04       LDX   #$04    Function := 4 = Convert from "Virtual" (?)
319E-   86 2D       STX   $2D
31A0-   A9 C0       LDA   #$C0    $2F := #$C0
31A2-   85 2F       STA   $2F
31A4-   4C EE F8    JMP   $F8EE

31A7-   A5 34       LDA   $34
31A9-   4C 36 FD    JMP   $FD36
-------------------------------------------------------
              Read SYSTEM.RELOC to $1000 and adjust NP.
                UNITWRITE (13, 0, 31, 0)   (WRITE VOL#)


[$EDAC]
31AC-   A9 00       LDA   #$00        $32.33 := 6 == Length of I/O #$600
31AE-   85 32       STA   $32                        6 Sectors (3 Blocks)
31B0-   A9 06       LDA   #$06
31B2-   85 33       STA   $33

31B4-   A9 00       LDA   #$00        $30.31 := #$1000  Buffer address
31B6-   85 30       STA   $30
31B8-   A9 10       LDA   #$10
31BA-   85 31       STA   $31
31BC-   A9 04       LDA   #$04        $2D := 4  (Slot 6, Drive 1)
31BE-   85 2D       STA   $2D
31C0-   A2 00       LDX   #$00        X := 0 === Read request

[$EDC2]
31C2-   20 5C E8    JSR   $E85C       Read a bunch of sectors
31C5-   A9 10       LDA   #$10        $C8.C9 := #$1002 == Disk Section #1
31C7-   85 C9       STA   $C9
31C9-   A9 02       LDA   #$02           5 Disk Sections for WIZARDRY IV
31CB-   85 C8       STA   $C8
[$EDCD]
31CD-   A0 00       LDY   #$00      LOOP outer
31CF-   B1 C8       LDA   ($C8),Y     Get DISK NUMBER (1, 2, 3, 4, or 5)
31D1-   F0 3E       BEQ   $3211
31D3-   A0 02       LDY   #$02        Get # of Segment Sections
31D5-   B1 C8       LDA   ($C8),Y       for this DISK NUMBER
31D7-   AA         TAX
31D8-   C8         INY
31D9-   B1 C8       LDA   ($C8),Y
31DB-   85 1A       STA   $1A

31DD-   C8         INY            LOOP inner
31DE-   B1 C8       LDA   ($C8),Y
31E0-   85 34       STA   $34         Groups of 3 words:
```

```
31E2-   C8          INY                 Add LowBlock(Word 0) to Length(Word 2).
31E3-   B1 C8       LDA   ($C8),Y       Word 2 is length of Segment Section
31E5-   85 35       STA   $35           This makes Word 0 and Word 2
31E7-   18          CLC                   a "range" of blocks.
31E8-   98          TYA
31E9-   69 03       ADC   #$03
31EB-   A8          TAY
31EC-   90 03       BCC   $31F1
31EE-   E6 C9       INC   $C9

31F0-   18          CLC
31F1-   B1 C8       LDA   ($C8),Y
31F3-   65 34       ADC   $34
31F5-   91 C8       STA   ($C8),Y
31F7-   C8          INY
31F8-   B1 C8       LDA   ($C8),Y
31FA-   65 35       ADC   $35
31FC-   91 C8       STA   ($C8),Y

31FE-   CA          DEX               Do it for all groups of 3 words.
31FF-   D0 DC       BNE   $31DD     LOOP inner

[$EE01]
3201-   C6 1A       DEC   $1A
3203-   10 D8       BPL   $31DD     LOOP inner

3205-   98          TYA             Reset $C8.C9 to include Y value.
3206-   38          SEC             So that we are at the next DISK section.
3207-   65 C8       ADC   $C8
3209-   85 C8       STA   $C8
320B-   90 C0       BCC   $31CD     LOOP outer
320D-   E6 C9       INC   $C9
320F-   B0 BC       BCS   $31CD     LOOP outer

3211-   18          CLC
3212-   A5 C8       LDA   $C8       Adjust NP ($B4.B5)
3214-   69 02       ADC   #$02        +2
3216-   85 B4       STA   $B4
3218-   A5 C9       LDA   $C9
321A-   69 00       ADC   #$00
321C-   85 B5       STA   $B5
321E-   60          RTS
--------------------------------------------------------------------------------

                              UNITWRITE(BASE12, 1, 22, 1)

                         UNITWRITE( UNIT, ARRAY, LENGTH, BLOCK#, MODE)
                                     $2D   $30.31 $32.33  $34.35  $2E.2F

                              From P010C06() while aborting COPY DISK
                              From P010612() after "S)tart Game" chosen

[$EE1F]
321F-   A5 34       LDA   $34
3221-   F0 02       BEQ   $3225
3223-   A9 80       LDA   #$80
3225-   85 E6       STA   $E6    Special flag
3227-   60          RTS
--------------------------------------------------------------------------------
3228-   88          DEY
3229-   D0 FD       BNE   $3228
322B-   60          RTS
--------------------------------------------------------------------
                                Tables used in $EE32 to
```

```
                                        BEEP! the Speaker
[$EE2C]
       2C      $2C F0
       16      $16 60
       24      $24 D0
[$EE2F]
       F0
         60
       D0
--------------------------------
                                    BEEP! the Speaker

                                    UNITWRITE( xxx, 0, 24, xxx)
        $34
                                    See P010038() and P010002()

[$EE32]
3232   24 E7       BIT   $E7      $E7 10000000 = CTRL-D pressed
                                      01000000 = CTRL-S pressed
3234-  70 1C       BVS   $3252    CTRL-S?  Yes, RTS (Ignore it)
3236-  A6 34       LDX   $34        No.
3238-  E0 03       CPX   #$03     $34 > 2?
323A-  B0 16       BCS   $3252      Yes, $EE52 RTS

   1/2/2000
                                           -- -- --
323C-  BD 2C EE    LDA   $EE2C,X  $02 := F0 60 D0
323F-  85 02       STA   $02      $01 := 2C 16 24
3241-  BD 2F EE    LDA   $EE2F,X
3244-  85 01       STA   $01
                                    Timing Loop
3246-  A4 02       LDY   $02
3248-  20 28 EE    JSR   $EE28      Wait a bit
324B-  AD 30 C0    LDA   $C030      Click the SPEAKER (Sound)
324E-  C6 01       DEC   $01
3250-  D0 F4       BNE   $3246

3252-  60          RTS
------------------------------------------------------------
                                    RANDOM() function call
[$EE53]
3253-  A0 07       LDY   #$07
3255-  06 F6       ASL   $F6
3257-  08          PHP
3258-  26 F7       ROL   $F7
325A-  26 F8       ROL   $F8
325C-  26 F9       ROL   $F9
325E-  30 05       BMI   $3265
3260-  28          PLP
3261-  10 07       BPL   $326A
3263-  30 03       BMI   $3268
3265-  28          PLP
3266-  30 02       BMI   $326A
3268-  E6 F6       INC   $F6
326A-  88          DEY
326B-  D0 E8       BNE   $3255
326D-  A5 F6       LDA   $F6
326F-  4A          LSR
3270-  AA          TAX
3271-  A5 F8       LDA   $F8
3273-  4C 8F EE    JMP   $EE8F   Return the RANDOM() number
------------------------------------------------------------
[$EE76]
3276-  20 7D E7    JSR   $E77D  check keyboard for input
```

```
3279-   E6 F6       INC   $F6    Random #
327B-   D0 02       BNE   $327F
327D-   E6 F7       INC   $F7    Random #
327F-   A2 00       LDX   #$00   x := 0   (FALSE) = no char avail
3281-   A5 E0       LDA   $E0    Input buffer ptr (first char avail)
3283-   C5 E1       CMP   $E1    Input buffer ptr (last char avail)
3285-   F0 01       BEQ   $3288  Was buffer full?  Yes, RTS
3287-   E8          INX          x := 1   (TRUE) = character avail
3288-   60          RTS
-------------------------------------------------------------
                                 UNITREAD( $D, mp03, 11, 0)
                                 KeyPress
[$EE89]
3289-   20 76 EE    JSR   $EE76
328C-   8A          TXA              A := 1 if character gotten
328D-   A2 00       LDX   #$00

[$EE8F]
328F-   A0 00       LDY   #$00
3291-   91 30       STA   ($30),Y    Return status
3293-   8A          TXA
3294-   C8          INY
3295-   91 30       STA   ($30),Y    Whole word for boolean(!?)
3297-   60          RTS              Small bug if A = 1
                                       (00000001 00000001) == TRUE (!)
-------------------------------------------------------------
3298-   A9 01       LDA   #$01
329A-   D0 F1       BNE   $328D
-------------------------------
                                   Just after Formatting Diskette
                                     and before COPYING
[$EE9C]
329C-   A2 08       LDX   #$08   Function := 8
329E-   A9 04       LDA   #$04   Device := 4
32A0-   85 2D       STA   $2D
[$EEA2]
32A2-   4C FE E7    JMP   $E7FE
-------------------------------------------------------------------
                                    MP01 := 2;  Timing value for MAJOR LOOP
                                    UNITWRITE( BASE12, MP01, 19, MP01)
[$EEA5]
32A5-   24 E7       BIT   $E7   Check $E7 for CTRL-D toggle (???)
32A7-   30 33       BMI   $32DC   Set, so just RTS

32A9-   24 FE       BIT   $FE   Apple Configuration (?)
32AB-   50 04       BVC   $32B1   Bit 6 is clear?  $EEB1
                              Bit 6 is set.  This is APPLE II GS (?)
32AD-   06 34       ASL   $34   $34.35 := 2 * $34.35
32AF-   26 35       ROL   $35

32B1-   A4 34       LDY   $34
                              MAJOR LOOP
32B3-   A2 60       LDX   #$60    Prepare for MINOR LOOP
32B5-   86 00       STX   $00
                                 MINOR LOOP
32B7-   A2 C5       LDX   #$C5      Timing value

32B9-   CA          DEX             Timing loop
32BA-   D0 FD       BNE   $32B9
                                 **********************************
32BC-   20 76 EE    JSR   $EE76     JSR EE76  Check keyboard for input
                                           ...and update RANDOM #
                                 **********************************
32BF-   A5 E5       LDA   $E5       Key pressed?   ($E5 is the char)
```

```
32C1-   30 19       BMI   $32DC       Yes, RTS     (in INPUT buffer)

32C3-   24 FE       BIT   $FE       Keyboard have Open and Closed Apple Keys?
32C5-   30 0A       BMI   $32D1       No, $EED1
                                     Yes, check them
32C7-   AD 61 C0    LDA   $C061       "Open Apple" key pressed?
32CA-   30 10       BMI   $32DC         Yes, RTS

32CC-   AD 62 C0    LDA   $C062       "Closed Apple" key pressed?
32CF-   30 0B       BMI   $32DC         Yes, RTS

32D1-   C6 00       DEC   $00
32D3-   D0 E2       BNE   $32B7   MINOR LOOP, up to #$60 times

32D5-   88          DEY
32D6-   D0 DB       BNE   $32B3 MAJOR LOOP

32D8-   C6 35       DEC   $35
32DA-   10 D7       BPL   $32B3 MAJOR LOOP

32DC-   60          RTS
--------------------------------------------------------------------------------
 UNITREAD( #$D, array[0..1], 16, # like 2040) from P01003F().   SEARCHBF()

  Input:  $30.31 --> array with 2(?) pointers to TCACHEP (?) structures.
          $34    Number like 2040 to 2046, etc.

  Structures are in a doubly linked list. "MSG" or cache buffers (?)

  This code takes the "FOUND" packet and moves it to the front of
  the chain.

  If the packet is not found then it still moves the last
  packet in the chain to the front to be reused.


[$EEDD]
32DD-   A0 00       LDY   #$00      A.B -> First "MSG" ptr buffer
32DF-   B1 30       LDA   ($30),Y   8.9 -> First "MSG" ptr buffer
32E1-   85 0A       STA   $0A
32E3-   85 08       STA   $08
32E5-   C8          INY
32E6-   B1 30       LDA   ($30),Y
32E8-   85 0B       STA   $0B
32EA-   85 09       STA   $09
                                 TRYANOTHER:
32EC-   A0 00       LDY   #$00      A.B point to magic number?
32EE-   B1 0A       LDA   ($0A),Y
32F0-   C5 34       CMP   $34
32F2-   D0 07       BNE   $32FB        No, $EEFB
32F4-   C8          INY
32F5-   B1 0A       LDA   ($0A),Y
32F7-   C5 35       CMP   $35
32F9-   F0 1B       BEQ   $3316        Yes, $EF16

32FB-   A0 02       LDY   #$02         No.  Try next buffer in chain
32FD-   B1 0A       LDA   ($0A),Y
32FF-   AA          TAX
[$EF00]
3300-   C8          INY
3301-   B1 0A       LDA   ($0A),Y
3303-   85 0B       STA   $0B
3305-   86 0A       STX   $0A
3307-   05 0A       ORA   $0A
```

```
3309-   D0 E1       BNE   $32EC     JMP TRYANOTHER if not NIL

330B-   A0 02       LDY   #$02      Not found in first chain,
330D-   B1 30       LDA   ($30),Y     so use 2nd ptr
330F-   85 0A       STA   $0A
3311-   C8          INY
3312-   B1 30       LDA   ($30),Y
3314-   85 0B       STA   $0B

                                    Found "Magic" number
                                 -or- Not found so use 2nd ptr
[$EF16]
3316-   A0 05       LDY   #$05     C.D := back ptr in buffer
3318-   B1 0A       LDA   ($0A),Y
331A-   85 0D       STA   $0D
331C-   88          DEY
331D-   B1 0A       LDA   ($0A),Y
331F-   85 0C       STA   $0C
3321-   05 0D       ORA   $0D
3323-   F0 51       BEQ   $3376    If NIL, then RTS

3325-   A5 0A       LDA   $0A
3327-   91 08       STA   ($08),Y
3329-   C8          INY
332A-   A5 0B       LDA   $0B
332C-   91 08       STA   ($08),Y
332E-   A0 02       LDY   #$02
3330-   B1 0A       LDA   ($0A),Y
3332-   85 0E       STA   $0E
3334-   91 0C       STA   ($0C),Y
3336-   C8          INY
3337-   B1 0A       LDA   ($0A),Y
3339-   85 0F       STA   $0F
333B-   91 0C       STA   ($0C),Y
333D-   05 0E       ORA   $0E
333F-   F0 0D       BEQ   $334E
                                    Do this...
3341-   C8          INY
3342-   A5 0C       LDA   $0C
3344-   91 0E       STA   ($0E),Y
3346-   C8          INY
3347-   A5 0D       LDA   $0D
3349-   91 0E       STA   ($0E),Y
334B-   4C 5B EF    JMP   $EF5B
                                    or this...
334E-   A0 02       LDY   #$02
3350-   A5 0C       LDA   $0C
3352-   91 30       STA   ($30),Y
3354-   C8          INY
3355-   A5 0D       LDA   $0D
3357-   91 30       STA   ($30),Y
3359-   A0 05       LDY   #$05
                                    ...and continue here
[$EF5B]
335B-   A9 00       LDA   #$00
335D-   91 0A       STA   ($0A),Y
335F-   88          DEY
3360-   91 0A       STA   ($0A),Y
3362-   88          DEY
3363-   A5 09       LDA   $09
3365-   91 0A       STA   ($0A),Y
3367-   88          DEY
3368-   A5 08       LDA   $08
336A-   91 0A       STA   ($0A),Y
```

```
336C-   88          DEY
336D-   A5 0B       LDA   $0B
336F-   91 30       STA   ($30),Y
3371-   88          DEY
3372-   A5 0A       LDA   $0A
3374-   91 30       STA   ($30),Y
3376-   60          RTS
--------------------------------------------------------------------------------
                                    ASCII.HUFF
                                    (T$1B S$00; T$1B S$0E; T$1B S$0D)
[$EF77]
3377-   A9 00       LDA   #$00     $34.35 := #$0D00
3379-   85 34       STA   $34
337B-   A9 0D       LDA   #$0D
337D-   85 35       STA   $35
337F-   A2 03       LDX   #$03     Xfer 3 sectors (less than 2 Blocks)
3381-   A0 00       LDY   #$00

3383-   B1 30       LDA   ($30),Y  From BUFFER[]...
3385-   91 34       STA   ($34),Y  ...to $0D00..$0FFF
3387-   88          DEY
3388-   D0 F9       BNE   $3383

338A-   E6 31       INC   $31
338C-   E6 35       INC   $35
338E-   CA          DEX
338F-   D0 F2       BNE   $3383

3391-   60          RTS
--------------------------------------------------------------------------------
                                Huffman decoding for Messages

                                Message string processing
                                  $30.31 == Disk Buffer with MESSAGES
                                        512 bytes  (No Header)
                                        ($1C62 first time)
                                  $2E.$2F is #0000
                                  $34.35 ("TO" buffer)
                                         ...first time $7FB4
[$EF92]
3392-   A0 00       LDY   #$00
3394-   84 14       STY   $14
3396-   B1 30       LDA   ($30),Y
3398-   85 12       STA   $12
339A-   A5 12       LDA   $12
339C-   C8          INY
339D-   84 15       STY   $15
339F-   A2 08       LDX   #$08
33A1-   20 B1 EF    JSR   $EFB1   JSR $EFB1  Unscramble the first char
33A4-   A8          TAY                      (message length)
33A5-   F0 09       BEQ   $33B0   RTS
33A7-   85 11       STA   $11
                                Loop:
33A9-   20 B1 EF    JSR   $EFB1     JSR $EFB1  Unscramble a char
33AC-   C6 11       DEC   $11
33AE-   D0 F9       BNE   $33A9   JMP Loop
33B0-   60          RTS
--------------------------------
  (from above)                  Process "from" bits, and return
                                  an ASCII character in the buffer.

                                X := 8 ( 8 bits per byte)
                                $12 := a "from" byte
[$EFB1]
```

```
33B1-   A9 00       LDA   #$00
                                    LOOP:  $EFB3
33B3-   A8          TAY
33B4-   CA          DEX
33B5-   30 22       BMI   $33D9     8 bits processed. Get next "from" byte

[$EFB7]
33B7-   46 12       LSR   $12       TAG1:
33B9-   B9 00 0D    LDA   $0D00,Y
33BC-   B0 0E       BCS   $33CC
33BE-   4A          LSR
33BF-   4A          LSR
33C0-   B9 00 0F    LDA   $0F00,Y
33C3-   90 EE       BCC   $33B3     JMP LOOP: $EFB3
33C5-   A4 14       LDY   $14
33C7-   91 34       STA   ($34),Y   Store "TO" char
33C9-   E6 14       INC   $14
33CB-   60          RTS           RETURN

33CC-   4A          LSR
33CD-   B9 00 0E    LDA   $0E00,Y
33D0-   90 E1       BCC   $33B3     JMP LOOP: $EFB3
33D2-   A4 14       LDY   $14
33D4-   91 34       STA   ($34),Y   Store "TO" char
33D6-   E6 14       INC   $14
33D8-   60          RTS           RETURN
--------------------------------
(from above)
                                      Get next "from" byte

33D9-   84 1A       STY   $1A        Save Y
33DB-   A4 15       LDY   $15
33DD-   B1 30       LDA   ($30),Y    Get next byte
33DF-   85 12       STA   $12
33E1-   E6 15       INC   $15        Advance to next "from" byte
33E3-   A4 1A       LDY   $1A        Restore Y
33E5-   A2 07       LDX   #$07       X := 7 process 8 bits in byte
33E7-   D0 CE       BNE   $33B7    JMP TAG1: $EFB7
--------------------------------------------------------------------------------
                                    Concatonate Strings
[$EFE9]
33E9-   A5 2E       LDA   $2E       Mode
33EB-   05 2F       ORA   $2F
33ED-   F0 0B       BEQ   $33FA

33EF-   20 FA EF    JSR   $EFFA
33F2-   A5 2E       LDA   $2E
33F4-   85 34       STA   $34
33F6-   A5 2F       LDA   $2F
33F8-   85 35       STA   $35

[$EFFA]
33FA-   18          CLC
33FB-   A0 00       LDY   #$00
33FD-   B1 30       LDA   ($30),Y
33FF-   AA          TAX
3400-   71 34       ADC   ($34),Y
3402-   91 30       STA   ($30),Y
3404-   18          CLC
3405-   8A          TXA
3406-   65 30       ADC   $30
3408-   85 0A       STA   $0A
340A-   A9 00       LDA   #$00
340C-   65 31       ADC   $31
```

```
340E-   85 0B       STA   $0B
3410-   B1 34       LDA   ($34),Y
3412-   F0 08       BEQ   $341C

3414-   A8          TAY
3415-   B1 34       LDA   ($34),Y
3417-   91 0A       STA   ($0A),Y
3419-   88          DEY
341A-   D0 F9       BNE   $3415

341C-   A2 00       LDX   #$00
341E-   60          RTS
-------------------------------------------------------------------------------
            POS() STRING Function === Find SUBSTRING in another STRING

            UNITWRITE( #$0D, MP106, 29, GETADDR( MP04), GETADDR( MP03));
                              $30.31     $34.35          $2E.2F
                                         These are STRINGS.  e.g.:
                                         '^'             Insert Master Diskette into Drive ^
                                         First byte = LENGTH
            After answering "N" in M)ake Scenario,
              where the disk already has data on it.
              (and this is SKIPPED if ESC is pressed to earlier solicit)

[$F01F]
341F-   A9 00       LDA   #$00      $30.31 := 0 (Word 0)
3421-   A0 01       LDY   #$01
3423-   91 30       STA   ($30),Y
3425-   20 5E F0    JSR   $F05E     JSR $F05E
                                    Y := 0; STA ($30),Y; LDX #0

3428-   B1 34       LDA   ($34),Y   LENGTH( MP04) = 0?
342A-   F0 36       BEQ   $3462       Yes, F062  RTS

342C-   85 13       STA   $13       $13 := LENGTH( MP04)
342E-   B1 2E       LDA   ($2E),Y   LENGTH( MP03) = 0?
3430-   F0 30       BEQ   $3462       Yes, F062  RTS

3432-   38          SEC             LENGTH( MP04) > LENGTH( MP03)?
3433-   E5 13       SBC   $13
3435-   90 2B       BCC   $3462       Yes, F062  RTS

3437-   69 00       ADC   #$00      $0C := difference in lengths + 1 (?)
3439-   85 0C       STA   $0C
343B-   84 0D       STY   $0D       $0D := 0;  Length in $30.31 string

343D-   A5 2E       LDA   $2E       $0A.0B := $2E.2F
343F-   85 0A       STA   $0A
3441-   A5 2F       LDA   $2F
3443-   85 0B       STA   $0B
                                  $3445:
3445-   A4 0D       LDY   $0D
3447-   C4 0C       CPY   $0C
3449-   B0 17       BCS   $3462     Done?  Yes, F062 RTS

344B-   C8          INY             Increment $30.31 string length
344C-   84 0D       STY   $0D
344E-   A6 13       LDX   $13
3450-   A0 01       LDY   #$01     First char in $34.35
                                   LOOP:
3452-   B1 34       LDA   ($34),Y  Get char from MP04 string
3454-   D1 0A       CMP   ($0A),Y  Match char in MP03?
3456-   D0 0D       BNE   $3465      No, $F065
```

```
3458-   C8          INY             Yes, that char matches
3459-   CA          DEX
345A-   D0 F6       BNE   $3452     JMP LOOP:

345C-   A5 0D       LDA   $0D

[$F05E]
345E-   A0 00       LDY   #$00
3460-   91 30       STA   ($30),Y

3462-   A2 00       LDX   #$00
3464-   60          RTS
--------------------------------
3465-   E6 0A       INC   $0A       INC $A.$B,D to next
3467-   D0 02       BNE   $346B
3469-   E6 0B       INC   $0B
346B-   A4 0D       LDY   $0D
346D-   B1 2E       LDA   ($2E),Y
346F-   10 D4       BPL   $3445       JMP to $3445 with this ASCII char

3471-   E6 0A       INC   $0A         Char is non-ASCII
3473-   D0 02       BNE   $3477       So skip over it (?)
3475-   E6 0B       INC   $0B
3477-   E6 0D       INC   $0D
3479-   D0 CA       BNE   $3445       JMP (always) to $3445
-------------------------------------------------------------------------------
                                UNITWRITE( BASE12, MP11^, 20, DCMP03)
                                "Formatting Diskette"
                                See P010C01() after P010C0A() call.
[$F07B]
347B-   A5 34       LDA   $34     Disk unit (4 or 5)
347D-   85 2D       STA   $2D
347F-   A2 03       LDX   #$03    INIT DISK
[$F081]
3481-   4C FE E7    JMP   $E7FE       JMP $E7FE
-------------------------------------------------------------------------------
                                UNITWRITE( 13,  0, 25, 0)  Cache Size
                                UNITWRITE( 13, -1, 25, 0)  CachOff    $EE := 0
                                UNITWRITE( 13, -2, 25, 0)  CachOn     $EE := -1
                                UNITWRITE( 13, -3, 25, 0)
                                UNITWRITE( 13, ++, 25, 0) Do I/O to Aux Cache
                                                ---
                                                2 words, both positive

                                In DOCOPY()
                                CACHE

[$F084]
3484-   A0 00       LDY   #$00     $7C.7D      := Param2 Word 0
3486-   B1 30       LDA   ($30),Y
3488-   85 7C       STA   $7C
348A-   8D BE 0C    STA   $0CBE    Clobbers in $CB0..CC7
348D-   C8          INY              (not needed, no need to (?))
348E-   B1 30       LDA   ($30),Y
3490-   85 7D       STA   $7D
3492-   8D BF 0C    STA   $0CBF    Clobbers in $CB0..CC7
3495-   10 18       BPL   $34AF   Param2 Word 0 >= 0? Yes, $F0AF


3497-   A6 7C       LDX   $7C     Negative
3499-   E0 FD       CPX   #$FD    Compare to -3
349B-   90 12       BCC   $34AF   < -3? (-4, -5, etc.) Yes, $F0AF
349D-   D0 0C       BNE   $34AB   -1 or -2?  Yes, F0AB
```

```
349F-   A2 00       LDX   #$00    -3
34A1-   86 E9       STX   $E9     $E9 := $EA := $EB := 0;
34A3-   86 EA       STX   $EA     $EA.EB == # of blocks written to Aux CACHE
34A5-   86 EB       STX   $EB
34A7-   E8          INX
34A8-   86 E8       STX   $E8     $E8 := 1;
34AA-   60          RTS

                                  -1 or -2
[$F0AB]
34AB-   E8          INX          X := X + 1.
34AC-   86 EE       STX   $EE    $EE := X (0 or -1)
34AE-   60          RTS

                                  Param2 word 0 >= 0  ( or < -3 really?)
[$F0AF]
34AF-   A9 60       LDA   #$60
34B1-   85 2F       STA   $2F    $2F := #$60
34B3-   A2 04       LDX   #$04   X := 4 == Function = Aux cache I/O
34B5-   86 2D       STX   $2D    Device := 4;
34B7-   4C 5C E8    JMP   $E85C  Y = 1 (Param2, word 0 byte 1)
--------------------------------------------------------------------------------
                                  ASCII.HUFF processing

                                  UNITWRITE( BASE12, buffer, 27, 0, #)

                                  Message String Processing
                                    Part 1 - Determine relative offset in
                                             block to HUFF bytes for message.

                                  $30.31 == Disk Buffer with MESSAGES
                                            6 byte header + 512 bytes

                                  $2E.$2F is msg # (relative to first in block)
                                     e.g., if first msg in block is 2019, and we
                                        want msg 2040, this is 2040-2019=21
                                  $34.35 is 0 if not looking at BLOCK 0,
                                         is 4 if looking at BLOCK 0
                                             To account for 2 ptrs at beginning!
[$F0BA]
34BA-   18          CLC           C8.C9 := 30.31 + #$0106
34BB-   A5 30       LDA   $30             2nd half of BLOCK
34BD-   69 06       ADC   #$06
34BF-   85 C8       STA   $C8
34C1-   A5 31       LDA   $31
34C3-   69 01       ADC   #$01
34C5-   85 C9       STA   $C9

34C7-   A0 FF       LDY   #$FF    0C := last byte of 2nd BLOCK
34C9-   B1 C8       LDA   ($C8),Y       (# of msg GROUPS in this block)
34CB-   85 0C       STA   $0C
34CD-   88          DEY
                                  BEGIN LOOP
[$F0CE]
34CE-   A5 0C       LDA   $0C
34D0-   F0 57       BEQ   $3529 [$F129]  Error return (didn't find msg)

34D2-   C6 0C       DEC   $0C
34D4-   A5 2F       LDA   $2F      Negative message number? (>32767)
34D6-   30 51       BMI   $3529 [$F129]  Yes,  Error return

34D8-   B1 C8       LDA   ($C8),Y  # of messages in this GROUP.
34DA-   85 0D       STA   $0D
34DC-   20 53 F1    JSR   $F153     DEC C8,Y ptr
```

```
34DF-   A5 2F       LDA   $2F      Upper half of $2E.2F > 0?
34E1-   D0 06       BNE   $34E9      Yes, $F0E9

34E3-   A5 2E       LDA   $2E      Compare relative MSG# to
34E5-   C5 0D       CMP   $0D        # of messages in this group
34E7-   90 4E       BCC   $3537    Branch when we found the message group
                                     that contains our message.
                                   34.35 is now positioned at
                                     HUFF byte GROUP for the message.

[$F0E9]
34E9-   A6 0D       LDX   $0D      # of messages in this GROUP
34EB-   A5 34       LDA   $34
                                    loop2  (count # of HUFF bytes
34ED-   18          CLC                    for messages in this group)
34EE-   71 C8       ADC   ($C8),Y
34F0-   90 02       BCC   $34F4
34F2-   E6 35       INC   $35
34F4-   20 53 F1    JSR   $F153       DEC C8,Y ptr
34F7-   CA          DEX
34F8-   D0 F3       BNE   $34ED     jmp loop2

34FA-   85 34       STA   $34      34.35 := 34.35 + SUM of HUFF bytes
34FC-   C6 0D       DEC   $0D      2E.2F := 2E.2F - 0D + 1
34FE-   A5 2E       LDA   $2E                        (0D is #$0E first time)
[$F100]
3500-   38          SEC                    (2E.2F is #$0015 first time)
3501-   E5 0D       SBC   $0D
3503-   85 2E       STA   $2E
3505-   B0 02       BCS   $3509
3507-   C6 2F       DEC   $2F        2E.2F becomes #$0008 first time

3509-   86 0D       STX   $0D      0D := #$0

350B-   B1 C8       LDA   ($C8),Y  Get message GAP
350D-   10 07       BPL   $3516       This is # of "skipped" message numbers

350F-   29 7F       AND   #$7F     If "gap" > 127, then use 2 bytes for "gap"
3511-   85 0D       STA   $0D
3513-   20 53 F1    JSR   $F153      DEC C8,Y ptr

3516-   38          SEC              2E.2F := 2E.2F - GAP ("skipped" message numbers)
3517-   A5 2E       LDA   $2E                         (#08 first time)
3519-   F1 C8       SBC   ($C8),Y
351B-   85 2E       STA   $2E
351D-   A5 2F       LDA   $2F
351F-   E5 0D       SBC   $0D
3521-   85 2F       STA   $2F           2E.2F becomes 0 First Time
3523-   20 53 F1    JSR   $F153      DEC C8,Y ptr
3526-   4C CE F0    JMP   $F0CE  JMP BEGIN LOOP ($F0CE)
---------------------------------
[$F129]
3529-   A9 FF       LDA   #$FF   Error RETURN
352B-   85 35       STA   $35

[$F12D]
352D-   A0 00       LDY   #$00   Normal RETURN
352F-   91 30       STA   ($30),Y
3531-   A5 35       LDA   $35
3533-   C8          INY
3534-   91 30       STA   ($30),Y
3536-   60          RTS
---------------------------------
[$F137]
```

```
3537-   E6 2F       INC   $2F     Almost found the message
3539-   A5 34       LDA   $34     $34.35 is now positioned at the proper "group".
353B-   A6 2E       LDX   $2E
353D-   F0 0D       BEQ   $354C   First message in GROUP the one we want?  Yes, 354C
                                   LOOP
353F-   18          CLC
3540-   71 C8       ADC   ($C8),Y
3542-   90 02       BCC   $3546
3544-   E6 35       INC   $35
3546-   20 53 F1    JSR   $F153       DEC C8,Y ptr
3549-   CA          DEX
354A-   D0 F3       BNE   $353F    JMP LOOP

354C-   C6 2F       DEC   $2F
354E-   D0 EF       BNE   $353F    JMP LOOP
3550-   4C 2D F1    JMP   $F12D   Return
-------------------------------------
[$F153]
3553-   C0 00       CPY   #$00     DEC  (C8),Y pointer
3555-   D0 02       BNE   $3559
3557-   C6 C9       DEC   $C9
3559-   88          DEY
355A-   60          RTS
-------------------------------------------------------------------------------
                          To here from $ED10 for a lot of UNITWRITE() with #$D

                          UNITWRITE( UNIT, ARRAY, LENGTH, BLOCK#, MODE)
                                     $2D   $30.31 $32.33  $34.35  $2E.2F
                                                          func

[$F15B]
355B-   20 7D E7    JSR   $E77D     Check Keyboard for Input
355E-   A9 00       LDA   #$00
3560-   85 F0       STA   $F0
3562-   A5 32       LDA   $32       A := func
3564-   C9 23       CMP   #$23
3566-   B0 0E       BCS   $3576
3568-   0A          ASL
3569-   A8          TAY            *********************************
356A-   AD 8B C0    LDA   $C08B     Aux Ram, Bank 1 ($D000.$DFFF)
356D-   AD 8B C0    LDA   $C08B     Write Enabled
                                   *********************************
3570-   20 79 F1    JSR   $F179        Process the UNITWRITE( $D, ..., Y, ...)
                                   ************************
3573-   AD 80 C0    LDA   $C080     Back to Bank 2, no write.
                                   ************************
3576-   A6 F0       LDX   $F0
3578-   60          RTS
---------------------------------
[$F179]
3579-   B9 81 F1    LDA   $F181,Y
357C-   48          PHA
357D-   B9 80 F1    LDA   $F180,Y
3580-   48          PHA           $F180 table starts here
3581-   60          RTS
---------------------------------
3582-   04 F7           [01. = #$01] $F705 Initial ClearScreen, return KANMP04[] config
3584-   E0 F3           [02. = #$02] $F3E1 DRAWSCR() and DRAWSCR2()
3586-   64 F3           [03. = #$03] $F365 In KANJIREA, in loop for "S)tart game; M)ake Sc"
3588-   27 F5           [04. = #$04]
3589-   C7 F5           [05. = #$05] $F5C8  RTS (?)
358B-   75 F1            06.         xxxxx
358D-   75 F1            07.         xxxxx
358F-   75 F1            08.         xxxxx
```

```
3591-   75 F1            09.         xxxxx
3593-   75 F1            10.         xxxxx
3595-   75 F1            11.         xxxxx
3597-   75 F1            12.         xxxxx
3599-   CE F7           [13. = #$D]  $F7CF after random pics chosen (S)tart Game)
359C-   D8 F6           [14. = #$E]  processing '200.CHARSET' $F6D9
359D-   2A F7            15.         xxxxx
35A0-   75 F1            16.         xxxxx
35A2-   B5 F7           [17. = $11] in KANJIREA, in loop for "S)tart game; M)ake Scenario"
35A4-   F2 F6           [18. = #$12] processing '200.MONSTERS' $F6F3
35A5-   75 F1            19.         xxxxx
35A7-   75 F1            20.         xxxxx
35A9-   75 F1            21.         xxxxx
35AB-   75 F1            22.         xxxxx
35AD-   3A F7           [23. = #$17] $F73B
35B0-   75 F1            24.         xxxxx
35B2-   75 F1            25.         xxxxx
35B4-   75 F1            26.         xxxxx
35B6-   75 F1            27.         xxxxx
35B8-   75 F1            28.         xxxxx
35BA-   75 F1            29.         xxxxx
35BC-   37 F5           [30. = #$1E] $F538 Move STRING to Window and to SCREEN  CENTSTR() and P010022()
35BE-   75 F1            31.         xxxxx
35C0-   75 F1            32.         xxxxx
35C2-   75 F1            33.         xxxxx
35C4-   C8 F5            34. = #$22  $F5C9 DrawMaze code (?)  See P010B04().

---------------------------------
From $F1D6
                                     $19 is VPOS on Window

                                     Sets 16.17 to Secondary TEXT screen ptr
                                     Sets AX to relative TEXT screen ptr
[$F1C6]
        A4 19       LDY   $19          $16.17 --> Secondary TEXT screen ptr
        B9 61 E2    LDA   $E261,Y      Relative Text Screen PTRs    (00, 01, 02, 03)
35CB-   AA          TAX
35CC-   09 08       ORA   #$08         $08, 09, 0A, 0B
35CE-   85 17       STA   $17
35D0-   B9 79 E2    LDA   $E279,Y      Text Screen PTRs    (00, 80, 28, A8, 50, D0)
35D3-   85 16       STA   $16
35D5-   60          RTS
---------------------------------
                                  $19 = VPOS

                                  Modifies $F294 to $F306 for Primary HiRes screen

                                  Returns:
                                    $12.13 --> Primary TEXT screen ptr
                                    $16.17 --> Secondary TEXT screen ptr

[$F1D6]
35D6-   20 C6 F1    JSR   $F1C6    Set $16.17 to Secondary TEXT screen ptr $800, $880, etc.
                                      and AX to *relative* TEXT screen ptr, $0000, $0080...
35D9-   A8          TAY                AX (Low, High)
35DA-   8A          TXA
35DB-   09 04       ORA   #$04     Primary TEXT screen:  $0400, $0480, etc.
35DD-   85 13       STA   $13      $12.13 := Primary TEXT screen ptr ($400, $480, etc.)
35DF-   84 12       STY   $12
35E1-   8A          TXA
35E2-   09 20       ORA   #$20     High Res Screen 1:  A := $20, $21, $22, etc.
35E4-   18          CLC            A := $20, 21 etc.;
                                   Y := 00, 80, etc.
```

```
35E5-   8C DB F2    STY   $F2DB    Modify a TON of code for screen manipulation
35E8-   8C 95 F2    STY   $F295      $F294..$F306
35EB-   8D DC F2    STA   $F2DC
35EE-   8D 96 F2    STA   $F296
35F1-   69 04       ADC   #$04
35F3-   8C E1 F2    STY   $F2E1
35F6-   8C 98 F2    STY   $F298
35F9-   8D E2 F2    STA   $F2E2
35FC-   8D 99 F2    STA   $F299
35FF-   69 04       ADC   #$04
3601-   8C E7 F2    STY   $F2E7
3604-   8C 9B F2    STY   $F29B
3607-   8D E8 F2    STA   $F2E8
360A-   8D 9C F2    STA   $F29C
360D-   69 04       ADC   #$04
360F-   8C ED F2    STY   $F2ED
3612-   8C 9E F2    STY   $F29E
3615-   8D EE F2    STA   $F2EE
3618-   8D 9F F2    STA   $F29F
361B-   69 04       ADC   #$04
361D-   8C F3 F2    STY   $F2F3
3620-   8C A1 F2    STY   $F2A1
3623-   8D F4 F2    STA   $F2F4
3626-   8D A2 F2    STA   $F2A2
3629-   69 04       ADC   #$04
362B-   8C F9 F2    STY   $F2F9
362E-   8C A4 F2    STY   $F2A4
3631-   8D FA F2    STA   $F2FA
3634-   8D A5 F2    STA   $F2A5
3637-   69 04       ADC   #$04
3639-   8C FF F2    STY   $F2FF
363C-   8C A7 F2    STY   $F2A7
363F-   8D 00 F3    STA   $F300
3642-   8D A8 F2    STA   $F2A8
3645-   69 04       ADC   #$04
3647-   8C 05 F3    STY   $F305
364A-   8C AA F2    STY   $F2AA
364D-   8D 06 F3    STA   $F306
3650-   8D AB F2    STA   $F2AB
3653-   60          RTS
-------------------------------------------------------------------------------
[$F254]
                  See $F266 below
3654-   01 Upper Left Corner Window Frame
        02 Upper Edge Window Frame
        03 Upper Right Corner Window Frame
        04 Left Edge Window Frame
        05 Right Edge Window Frame
        06 Bottom Left Corner Window Frame
        07 Bottom Edge Window Frame
        08 Bottom Right Corner Window Frame

365C-   0D
        0A
        0E
        0B
        0C
        0E
           9
        0D
-------------------------------------------------------------------------------
                                Store a "Char" to the HiRes Screen

                                $06 = HPOS
```

```
                                        $0A = Low Byte of char
                                        $0B = High Byte of char
                                              <0, 0, 01, $20, $32, or >0
                                        $1F = index into $F254,Y table.  Special chars

                                        $12.13 and $16.17 correspond to Hi-Res display coordinates
                                          $12.13 = Primary TEXT screeen (characters)
                                          $16.17 = Secondary TEXT screen (priorities)

[$F264]
3664-   A4 1F       LDY   $1F       0 === special upper-left-corner char for WindowFrame (?)
3666-   B9 54 F2    LDA   $F254,Y   A := Special char (Window Frame)

                                    [ALTERNATE ENTRY to $F264]
[$F269]
3669-   85 0A       STA   $0A       $0A := Low Byte of char to display

                                    [ALTERNATE ENTRY to $F264]
[$F26B]
366B-   A5 1E       LDA   $1E       Priority
366D-   A4 06       LDY   $06       Horizontal position
366F-   D1 16       CMP   ($16),Y   This window have higher priority than current location on screen?
3671-   90 0E       BCC   $3681      No, RETURN w/o displaying

3673-   91 16       STA   ($16),Y   Save priority (bigger is higher) in Secondary TEXT memory
3675-   A5 0B       LDA   $0B       A := $0B
3677-   F0 09       BEQ   $3682     $0B = 0, Yes, $F282
3679-   C9 32       CMP   #$32      $0B = 50, Yes, $F282
367B-   F0 05       BEQ   $3682
367D-   A9 FF       LDA   #$FF      No, A := #$FF
367F-   D0 07       BNE   $3688     JMP $F288

3681-   60          RTS

[$F282]
3682-   A5 0A       LDA   $0A       Char to display; ...first time, value from $F254 table
3684-   D1 12       CMP   ($12),Y   Compare with current TEXT screen value
3686-   F0 F9       BEQ   $3681     Same value, so no update needed.

[$F288]
3688-   91 12       STA   ($12),Y   Char to display (or #$FF) (?) Update Primary TEXT Screen
368A-   A5 0B       LDA   $0B
368C-   C9 20       CMP   #$20      $0B = " "
368E-   D0 1D       BNE   $36AD [$F2AD]  No, $F2AD
3690-   A9 00       LDA   #$00        Yes, Display " " char

3692-   A4 06       LDY   $06       Horizontal position

                                    1 Hi-Res character "block"
[$F294]
3694-   99 00 20    STA   $2000,Y    modified
3697-   99 00 24    STA   $2400,Y    modified
369A-   99 00 28    STA   $2800,Y    modified
369D-   99 00 2C    STA   $2C00,Y    modified
36A0-   99 00 30    STA   $3000,Y    modified
36A3-   99 00 34    STA   $3400,Y    modified
36A6-   99 00 38    STA   $3800,Y    modified
36A9-   99 00 3C    STA   $3C00,Y    modified
36AC-   60          RTS
--------------------------------
                                    From above, A == $0B value
[$F2AD]
36AD-   A0 00       LDY   #$00      Y := 0
36AF-   AA          TAX             $0B value
```

```
36B0-   F0 09       BEQ   $36BB       0?  Yes, $F2BB
36B2-   30 54       BMI   $3708       <0? Yes, $F308
36B4-   09 80       ORA   #$80        +?  Yes, set high bit (this is a GRAPHIC pic)
36B6-   E0 01       CPX   #$01      $0B = 1?
36B8-   D0 07       BNE   $36C1       No, $F2C1
36BA-   C8          INY             Yes, Y := Y + 1

[$F2BB]
36BB-   A5 0A       LDA   $0A       Character to store = " "?
36BD-   C9 20       CMP   #$20
36BF-   F0 CF       BEQ   $3690       Yes, $F290 to clear hi-res screen for char

[$F2C1]
36C1-   84 02       STY   $02       $2E.2F := PTR to "char" in 200.charset at $D000
36C3-   0A          ASL                       #$D0mm == 200.charset at bank1
36C4-   26 02       ROL   $02
36C6-   0A          ASL                       Multiply Char index by 8
36C7-   26 02       ROL   $02                 Since 8 bytes per Char at $D000.
36C9-   0A          ASL
36CA-   26 02       ROL   $02                 #$D410 Area for MONSTER pic (#$F0 bytes)
36CC-   85 2E       STA   $2E                 #$D510 Area for MONSTER pic
36CE-   A5 02       LDA   $02                 #$D610 Area for MONSTER pic
36D0-   69 D0       ADC   #$D0                #$D710 Area for MONSTER pic
36D2-   85 2F       STA   $2F
                              *** $F2D4 Alternate Entry to this Code
36D4-   A0 00       LDY   #$00
36D6-   A6 06       LDX   $06        X := HPOS
36D8-   B1 2E       LDA   ($2E),Y
36DA-   9D 00 20    STA   $2000,X    modified   $F2DB   $2000 is Primary Hires Screen
36DD-   C8          INY
36DE-   B1 2E       LDA   ($2E),Y
36E0-   9D 00 24    STA   $2400,X    modified
36E3-   C8          INY
36E4-   B1 2E       LDA   ($2E),Y
36E6-   9D 00 28    STA   $2800,X    modified
36E9-   C8          INY
36EA-   B1 2E       LDA   ($2E),Y
36EC-   9D 00 2C    STA   $2C00,X    modified
36EF-   C8          INY
36F0-   B1 2E       LDA   ($2E),Y
36F2-   9D 00 30    STA   $3000,X    modified
36F5-   C8          INY
36F6-   B1 2E       LDA   ($2E),Y
36F8-   9D 00 34    STA   $3400,X    modified
36FB-   C8          INY
36FC-   B1 2E       LDA   ($2E),Y
36FE-   9D 00 38    STA   $3800,X    modified
[$F301]
3701-   C8          INY
3702-   B1 2E       LDA   ($2E),Y
3704-   9D 00 3C    STA   $3C00,X    modified
3707-   60          RTS
-----------------------------------------
3708-   A9 7F       LDA   #$7F
370A-   D0 86       BNE   $3692
-------------------------------------------------------------------------------
                              From below at $F365
                              While processing UNITWRITE( x, x, 3, x) (and (x, x, 3, x) (?)
                              $30.31 == "Dest Array"
                                     == MainWindow (including HEADER)
[$F30C]
370C-   A0 08       LDY   #$08    $18 := HPOS     Move: HPOS, VPOS, HSIZE, VSIZE,
370E-   A2 06       LDX   #$06    $19 := VPOS           HCURSOR, VCURSOR, PRIORITY
3710-   B1 30       LDA   ($30),Y $1A := HSIZE    to: $18..$1E
```

```
3712-   95 18       STA   $18,X   $1B := VSIZE
3714-   88          DEY           $1C := HCURSOR
3715-   CA          DEX           $1D := VCURSOR
3716-   10 F8       BPL   $3710   $1E := PRIORITY

3718-   38          SEC
3719-   A5 1A       LDA   $1A     $0D := HSIZE - 2
371B-   E9 02       SBC   #$02
371D-   85 0D       STA   $0D

371F-   A5 1B       LDA   $1B     $0C := VSIZE - 2
3721-   E9 02       SBC   #$02
3723-   85 0C       STA   $0C
3725-   60          RTS
-----------------------------------
                                  Clear Secondary TEXT Screen values to 0.
                                  Priority values.
[$F326]
3726-   A0 17       LDY   #$17    Y := 23 (24 lines)
3728-   84 19       STY   $19     VPOS
372A-   20 C6 F1    JSR   $F1C6   Get Secondary TEXT screen ptr using VPOS
372D-   A0 27       LDY   #$27    Y := 39 (40 columns)
372F-   A9 00       LDA   #$00
3731-   91 16       STA   ($16),Y Secondary TEXT Screen
3733-   88          DEY
3734-   10 FB       BPL   $3731
3736-   C6 19       DEC   $19
3738-   10 F0       BPL   $372A
373A-   60          RTS
-----------------------------------
                                Display TOP (or BOTTOM) edge of WINDOW FRAME.

                                  From below, with
                                  $0B := 0
                                  $1F := 0 === Upper Left Corner
                                         6 === Bottom Left Corner
                                  $19 == VPOS

[$F33B]
373B-   20 D6 F1    JSR   $F1D6   Get SCREEN ptrs, and Self-Modify Code!
                                  $16.17 --> Secondary Text Screen
373E-   A4 18       LDY   $18     $06 := HPOS
3740-   84 06       STY   $06
3742-   20 64 F2    JSR   $F264   Store "LeftUpperCorner" to the screen
3745-   E6 1F       INC   $1F     $1F := 1 === "TopEdge"
3747-   A5 0D       LDA   $0D     HSIZE - 2 value
3749-   F0 12       BEQ   $375D
374B-   85 00       STA   $00
374D-   A4 1F       LDY   $1F
374F-   B9 54 F2    LDA   $F254,Y Get "Edge" character  ("-")
3752-   85 0A       STA   $0A
                                  LOOP:
3754-   E6 06       INC   $06       HPOS := HPOS + 1
3756-   20 6B F2    JSR   $F26B     Store "-" to the HiRes screen
3759-   C6 00       DEC   $00
375B-   D0 F7       BNE   $3754   JMP LOOP:

375D-   E6 1F       INC   $1F     $1F := 2 === Right Upper Corner
375F-   E6 06       INC   $06
3761-   20 64 F2    JSR   $F264   Store "RightUpperCorner" to the screen
3764-   60          RTS
-------------------------------------------------------------------------------
                                  Display a Window (including the Frame)
```

```
                                    UNITWRITE( xxx, Mainwin.PMEM^, 3, 0);
                                         and (x, x, 2, x) for DRAWSCR and DRAWSCR2

                                    $30.31 == MainWindow (with HEADER)
[$F365]
3765-   20 0C F3    JSR   $F30C    Set up $18..$1E with Header info
3768-   A5 1E       LDA   $1E      Set $0C to VSIZE - 2; $0D to HSIZE - 2
376A-   30 F8       BMI   $3764    Priority < 0?  Yes, RETURN!
376C-   A9 00       LDA   #$00
376E-   85 0B       STA   $0B      $0B := 0
3770-   85 1F       STA   $1F      $1F := 0 "LeftUpperCorner" Window Frame
3772-   20 3B F3    JSR   $F33B    Display Top Edge of Window Frame

3775-   A4 1F       LDY   $1F      $04 := next special char (Left Edge)
3777-   C8          INY
3778-   B9 54 F2    LDA   $F254,Y
377B-   85 04       STA   $04
377D-   C8          INY            $05 := next special char (Right Edge)
377E-   B9 54 F2    LDA   $F254,Y
3781-   85 05       STA   $05
3783-   C8          INY            $1F := point to next special char
3784-   84 1F       STY   $1F

3786-   A5 0C       LDA   $0C    $0C (VSIZE - 2)
3788-   F0 52       BEQ   $37DC  Done?  Yes, do bottom edge of Frame ($F3DC)
378A-   A0 0A       LDY   #$0A   $10 := #$0A  Skip over HEADER bytes to get to SCREEN DATA.
378C-   84 10       STY   $10
378E-   A5 30       LDA   $30    $0E.0F := $30.31  (Window w/header)
3790-   85 0E       STA   $0E
3792-   A5 31       LDA   $31
3794-   85 0F       STA   $0F
                                  Do all rows of Window
3796-   E6 19       INC   $19        Verical Row Position
3798-   20 D6 F1    JSR   $F1D6      Get Screen Ptrs, etc.
379B-   A4 18       LDY   $18        $06 := Horizontal Postion
379D-   84 06       STY   $06
379F-   A9 00       LDA   #$00
37A1-   85 0B       STA   $0B
37A3-   A5 04       LDA   $04        Left Window Edge char
37A5-   20 69 F2    JSR   $F269      Store a "|" to the screen
37A8-   A5 0D       LDA   $0D        $0D (HSIZE - 2)
37AA-   F0 21       BEQ   $37CD      Done with this row?  Yes, $F3CD
37AC-   85 00       STA   $00

37AE-   A4 10       LDY   $10        Do a single char
37B0-   B1 0E       LDA   ($0E),Y      $0A := low byte of char from Window Data
37B2-   85 0A       STA   $0A
37B4-   C8          INY
37B5-   D0 02       BNE   $37B9
37B7-   E6 0F       INC   $0F
37B9-   B1 0E       LDA   ($0E),Y      $0B := high byte of char from Window Data
37BB-   85 0B       STA   $0B           **** 0B might be non-zero here! ****
37BD-   C8          INY                      And is for "graphic" chars
37BE-   D0 02       BNE   $37C2
37C0-   E6 0F       INC   $0F
37C2-   84 10       STY   $10
37C4-   E6 06       INC   $06
37C6-   20 6B F2    JSR   $F26B        Store a "char" to the screen
37C9-   C6 00       DEC   $00
37CB-   D0 E1       BNE   $37AE      Do a single char

[$F3CD]
37CD-   A9 00       LDA   #$00
37CF-   85 0B       STA   $0B
```

```
37D1-   A5 05       LDA   $05        Right Window Edge char
37D3-   E6 06       INC   $06
37D5-   20 69 F2    JSR   $F269      JSR $F269  Store a "|" to the screen
37D8-   C6 0C       DEC   $0C
37DA-   D0 BA       BNE   $3796  Do all rows

[$F3DC]
37DC-   E6 19       INC   $19    Increase to last line (Window Frame)
37DE-   4C 3B F3    JMP   $F33B  Code + RTS  Process last line of Window
-------------------------------------------------------------------------------
                                 PAINT all windows on the screen.

                                 UNITWRITE( x, x, 2, x)
                                 DRAWSCR() and DRAWSCR2()
[$F3E1]
37E1-   20 26 F3    JSR   $F326   Clear Secondary TEXT screen values to 0.
37E4-   A5 31       LDA   $31           (Priority Values)
37E6-   05 30       ORA   $30
37E8-   D0 03       BNE   $37ED   Null Window Handle?
37EA-   4C B0 F6    JMP   $F6B0     Yes, $F6B0, Clear Screens

                                 REPEAT
37ED-   20 65 F3    JSR   $F365     PAINT a Window (including the Frame)

37F0-   A0 00       LDY   #$00      $30.31 := Get Next Window in Chain
37F2-   B1 30       LDA   ($30),Y
37F4-   AA          TAX
37F5-   C8          INY
37F6-   B1 30       LDA   ($30),Y
37F8-   85 31       STA   $31
37FA-   86 30       STX   $30
37FC-   05 30       ORA   $30
37FE-   D0 ED       BNE   $37ED    UNTIL Window Handle = NULL
[$F400]
3800-   A9 00       LDA   #$00    $19 := 0 (Vertical Position)
3802-   85 19       STA   $19
3804-   A9 18       LDA   #$18    $1B := 24 (do 24 rows)
3806-   85 1B       STA   $1B
                                 Go through entire screen (24 x 40),
                                 look for any space not covered by a Window and
                                 put a blank character there.

3808-   20 D6 F1    JSR   $F1D6    Get Screen Ptrs, Self modify code, etc.
                                   $16.17 = Secondary TEXT screen == Priority values
                                   $12.13 = Primary TEXT screen = Characters

380B-   A0 27       LDY   #$27     Do 40 columns
                                   Do entire row
380D-   B1 16       LDA   ($16),Y    Priority = 0?
380F-   D0 05       BNE   $3816      No, Don't store
3811-   91 12       STA   ($12),Y    Yes, store 0 on TEXT screen
3813-   20 94 F2    JSR   $F294           and put " " on the screen
3816-   88          DEY
3817-   10 F4       BPL   $380D    Do entire row

3819-   E6 19       INC   $19
381B-   C6 1B       DEC   $1B
381D-   10 E9       BPL   $3808   Do entire screen
381F-   60          RTS
-------------------------------------------------------------------------------
                                    From below, processing CENTSTR(), P010022(),
                                    UNITWRITE( BASE12, Window^, 30, GETADDR( MP01);

                                    $30.31 = Window with Header (?)
```

```
                                        A = VCursor
                                        X = HCursor
                                        $0C = Vsize - 2
                                        $0D = Hsize - 2

                                        Output:  $0E.0F (?)
[$F420]
3820-   0A          ASL
3821-   85 03       STA   $03    $03 := 2 * VCursor
3823-   8A          TXA         A := HCursor
3824-   30 3A       BMI   $3860  HCursor < 0?  Yes, $F460
3826-   F0 18       BEQ   $3840  HCursor = 0?  Yes, $F440
3828-   A6 0D       LDX   $0D    Hsize - 2
382A-   E0 26       CPX   #$26   Hsize = 38?   Full width of screen?
382C-   F0 10       BEQ   $383E    Yes, $F43E
382E-   E0 24       CPX   #$24   Hsize = 36? (Nearly full width of screen?)
3830-   F0 0E       BEQ   $3840    Yes, $F440
3832-   EC 26 F5    CPX   $F526  Hsize = $F526  (?????)
3835-   D0 2A       BNE   $3861    No, $F461   Yes.
3837-   CD 27 F5    CMP   $F527  HCursor >= $F527  (?????)
383A-   B0 2C       BCS   $3868    Yes, $F468  No.

[$F43C]
383C-   69 19       ADC   #$19   A := A + 25
383E-   69 17       ADC   #$17   A := A + 23        (?????)

[$F440]
3840-   AA          TAX           $0E.0F := $30.31 + F494,X . $F4DD,X ????
3841-   18          CLC
3842-   BD 94 F4    LDA   $F494,X
3845-   65 30       ADC   $30
3847-   85 0E       STA   $0E
3849-   BD DD F4    LDA   $F4DD,X
384C-   65 31       ADC   $31
384E-   85 0F       STA   $0F

3850-   A4 03       LDY   $03
3852-   10 0C       BPL   $3860
3854-   A0 00       LDY   #$00
3856-   A5 0E       LDA   $0E
3858-   65 03       ADC   $03
385A-   85 0E       STA   $0E
385C-   B0 02       BCS   $3860
385E-   C6 0F       DEC   $0F
3860-   60          RTS
--------------------------------------------------------------------------------

[$F461]
3861-   8E 26 F5    STX   $F526    Save Hsize - 2
3864-   A0 01       LDY   #$01
3866-   D0 03       BNE   $386B

[$F468]
3868-   AC 27 F5    LDY   $F527

386B-   85 00       STA   $00
386D-   8A          TXA
386E-   0A          ASL
386F-   85 02       STA   $02
3871-   BE 0C F5    LDX   $F50C,Y
3874-   B9 C3 F4    LDA   $F4C3,Y
3877-   18          CLC
3878-   65 02       ADC   $02
387A-   99 C4 F4    STA   $F4C4,Y
```

```
387D-   90 02       BCC   $3881
387F-   E8          INX
3880-   18          CLC
3881-   8A          TXA
3882-   99 0D F5    STA   $F50D,Y
3885-   C8          INY
3886-   C4 00       CPY   $00
3888-   90 EA       BCC   $3874
388A-   F0 E8       BEQ   $3874
388C-   8C 27 F5    STY   $F527
388F-   88          DEY
3890-   98          TYA
3891-   18          CLC
3892-   90 A8       BCC   $383C [$F43C]    Always branch
--------------------------------------------------------------------------------

[$F494]                                Low Part, paired with $F4DD   (73 entries(?))
3894-   0A          ASL
3895-   52          ???
3896-   9A          TXS
3897-   E2          ???
3898-   2A          ROL
3899-   72          ???
389A-   BA          TSX
389B-   02          ???
389C-   4A          LSR
389D-   92          ???
389E-   DA          ???
389F-   22          ???
38A0-   6A          ROR
38A1-   B2          ???
38A2-   FA          ???
38A3-   42          ???
38A4-   8A          TXA
38A5-   D2          ???
38A6-   1A          ???
38A7-   62          ???
38A8-   AA          TAX
38A9-   F2          ???
38AA-   3A          ???
38AB-   82          ???
38AC-   0A          ASL
38AD-   56 A2       LSR   $A2,X
38AF-   EE 3A 86    INC   $863A
38B2-   D2          ???
38B3-   1E 6A B6    ASL   $B66A,X
38B6-   02          ???
38B7-   4E 9A E6    LSR   $E69A
38BA-   32          ???
38BB-   7E CA 16    ROR   $16CA,X
38BE-   62          ???
38BF-   AE FA 46    LDX   $46FA
38C2-   92          ???

[$F4C3]
38C3-   DE 0A 0A    DEC   $0A0A,X
38C6-   0A          ASL
38C7-   0A          ASL
38C8-   0A          ASL
38C9-   0A          ASL
38CA-   0A          ASL
38CB-   0A          ASL
38CC-   0A          ASL
38CD-   0A          ASL
```

```
38CE-   0A          ASL
38CF-   0A          ASL
38D0-   0A          ASL
38D1-   0A          ASL
38D2-   0A          ASL
38D3-   0A          ASL
38D4-   0A          ASL
38D5-   0A          ASL
38D6-   0A          ASL
38D7-   0A          ASL
38D8-   0A          ASL
38D9-   0A          ASL
38DA-   0A          ASL
38DB-   0A          ASL
38DC-   0A          ASL

[$F4DD]                          High Part, paired with $F494  (73 entries(?))
38DD-   00          BRK
38DE-   00          BRK
38DF-   00          BRK
38E0-   00          BRK
38E1-   01 01       ORA   ($01,X)
38E3-   01 02       ORA   ($02,X)
38E5-   02          ???
38E6-   02          ???
38E7-   02          ???
38E8-   03          ???
38E9-   03          ???
38EA-   03          ???
38EB-   03          ???
38EC-   04          ???
38ED-   04          ???
38EE-   04          ???
38EF-   05 05       ORA   $05
38F1-   05 05       ORA   $05
38F3-   06 06       ASL   $06
38F5-   00          BRK
38F6-   00          BRK
38F7-   00          BRK
38F8-   00          BRK
38F9-   01 01       ORA   ($01,X)
38FB-   01 02       ORA   ($02,X)
38FD-   02          ???
38FE-   02          ???
38FF-   03          ???
3900-   03          ???
3901-   03          ???
3902-   03          ???
3903-   04          ???
3904-   04          ???
3905-   04          ???
3906-   05 05       ORA   $05
3908-   05 05       ORA   $05
390A-   06 06       ASL   $06

[$F50C]
390C-   06 00       ASL   $00
390E-   00          BRK
390F-   00          BRK
3910-   00          BRK
3911-   00          BRK
3912-   00          BRK
3913-   00          BRK
3914-   00          BRK
```

```
3915-   00          BRK
3916-   00          BRK
3917-   00          BRK
3918-   00          BRK
3919-   00          BRK
391A-   00          BRK
391B-   00          BRK
391C-   00          BRK
391D-   00          BRK
391E-   00          BRK
391F-   00          BRK
3920-   00          BRK
3921-   00          BRK
3922-   00          BRK
3923-   00          BRK
3924-   00          BRK
3925-   00          BRK

[$F526]
3926-   00          BRK
[F527]
3927-   01
-------------------------------------------------------
                                Move CHAR to Window and to SCREEN
                                UNITWRITE( xxx, window^, 4, Char)
                                         $2D   $30.31  $32.33 $34.35

[$F528]
        A9 33       LDA   #33     $20 := #$33   $20.21 = "String" addr
392A-   85 20       STA   $20             fake out!  #$33 is "length of string"
                                                    #$34 is first char!
392C-   A0 00       LDY   #$00    $21 := 0
392E-   84 21       STY   $21
3930-   84 0B       STY   $0B     $0B := 0
3932-   A9 01       LDA   #$01
3934-   85 11       STA   $11     $11 := #$01  "String" length
3936-   D0 10       BNE   $3948   JMP $F548
-----------------------------------------------
                                  Move STRING to Window and to SCREEN

                                  UNITWRITE( BASE12, Window^, 30, GETADDR( MP01));
                                  From CENTSTR() and P010022()
                                  $34.35 --> STRING (with first byte = LENGTH)

[$F538]
3938-   A5 34       LDA   $34       $20.21 := STRING addr
393A-   85 20       STA   $20
393C-   A5 35       LDA   $35
393E-   85 21       STA   $21
3940-   A0 00       LDY   #$00      $0B (High Order of 2 byte character)
3942-   84 0B       STY   $0B           is ALWAYS 0 when storing from STRING (?)
3944-   B1 20       LDA   ($20),Y   $11 := STRING length
3946-   85 11       STA   $11

                                ***Alternate Entry Point (for Char) ***
[$F548]
3948-   84 1F       STY   $1F       $1F := Index into STRING
394A-   20 0C F3    JSR   $F30C     Set up $18..$1E from Window header info
                                  Set $0C to VSIZE - 2; $0D to HSIZE - 2

394D-   A5 1C       LDA   $1C       VCursor
394F-   A6 1D       LDX   $1D       HCursor
3951-   20 20 F4    JSR   $F420     JSR $F420   Calculate $0E.0F Window Data area(?)
```

```
3954-   84 10       STY   $10       $10 := Window Data ptr
3956-   38          SEC
3957-   A5 1C       LDA   $1C       HCursor
3959-   65 18       ADC   $18       H
395B-   85 06       STA   $06       $06 := Horizontal Position on SCREEN
395D-   A5 19       LDA   $19       V
395F-   85 22       STA   $22       $22 := Save V
3961-   4C A8 F5    JMP   $F5A8

                                    Begin Loop, Do all characters in string
[$F564]
3964-   A4 10       LDY   $10         Store 2 bytes (1 char) of STRING into
3966-   A5 0A       LDA   $0A           Window Data area.
3968-   91 0E       STA   ($0E),Y
396A-   C8          INY
396B-   A5 0B       LDA   $0B
396D-   91 0E       STA   ($0E),Y
396F-   C8          INY
3970-   D0 02       BNE   $3974
3972-   E6 0F       INC   $0F

3974-   84 10       STY   $10         $10 := Window Data ptr for next char
3976-   A5 1E       LDA   $1E
3978-   30 05       BMI   $397F       If Priority >= 0
397A-   20 6B F2    JSR   $F26B         Store a "char" to the screen
397D-   E6 06       INC   $06           Inc Horizontal Position on Screen ptr

397F-   A9 00       LDA   #$00
3981-   E6 1C       INC   $1C         HCursor
3983-   A6 1C       LDX   $1C
3985-   E4 0D       CPX   $0D         Beyond right side of Window?
3987-   90 27       BCC   $39B0         No

3989-   85 1C       STA   $1C         Yes, try next line
398B-   A4 18       LDY   $18         H
398D-   C8          INY
398E-   84 06       STY   $06         New line, first position
3990-   A4 1D       LDY   $1D         VCursor
3992-   C8          INY
3993-   C4 0C       CPY   $0C         Beyond bottom of Window?
3995-   90 0D       BCC   $39A4         No, $F5A4
3997-   A8          TAY               Yes, Start Over at TOP of WINDOW Data
3998-   A5 30       LDA   $30         $0E.0F := Window Buffer
399A-   85 0E       STA   $0E
399C-   A5 31       LDA   $31
399E-   85 0F       STA   $0F
39A0-   A9 0A       LDA   #$0A        Skip past HEADER bytes
39A2-   85 10       STA   $10         $10 := Window Data ptr for next char

[$F5A4]
39A4-   84 1D       STY   $1D     VCursor
39A6-   A5 22       LDA   $22     V

                                     From above
[$F5A8]
39A8-   38          SEC
39A9-   65 1D       ADC   $1D     Vcursor
39AB-   85 19       STA   $19     Vertical Position
39AD-   20 D6 F1    JSR   $F1D6   Get Screen Ptrs, Self modify code, etc.
                                    $16.17 = Secondary TEXT screen == Priority values
                                    $12.13 = Primary TEXT screen = Characters
[$F5B0]
39B0-   E6 1F       INC   $1F       $0A := Get next char from STRING
39B2-   A4 1F       LDY   $1F
```

```
39B4-   B1 20       LDA   ($20),Y
39B6-   85 0A       STA   $0A
39B8-   C6 11       DEC   $11       Do all characters in STRING
39BA-   10 A8       BPL   $3964   JMP LOOP [$F564]

39BC-   A5 1C       LDA   $1C     Update HCursor in Window Header
39BE-   A0 06       LDY   #$06
39C0-   91 30       STA   ($30),Y
39C2-   A5 1D       LDA   $1D     ...and VCursor in Window HEader
39C4-   C8          INY
39C5-   91 30       STA   ($30),Y
39C7-   60          RTS
--------------------------------------------------------------------------------
[$F5C8]
39C8-   60          RTS
--------------------------------------------------------------------------------
                                During DRAWMAZE  P010B04()
                                UNITWRITE( BASE12, P02MP08, 34, ORD( MAINWIN))
                                           $2D     $30.31  $32.33  $34.35  $2E.2F
                                                   14 byte structure
[$F5C9]
39C9-   A0 0C       LDY   #$0C      $16.17 := $30.31-->xxx.C, .D
39CB-   B1 30       LDA   ($30),Y
39CD-   85 16       STA   $16
39CF-   C8          INY
39D0-   B1 30       LDA   ($30),Y
39D2-   85 17       STA   $17

39D4-   A0 0A       LDY   #$0A      $14 := $30.31-->xxx.A
39D6-   B1 30       LDA   ($30),Y         + $30.31-->xxx.6
39D8-   A0 06       LDY   #$06
39DA-   18          CLC
39DB-   71 30       ADC   ($30),Y
39DD-   85 14       STA   $14

39DF-   A0 08       LDY   #$08      $13 := $30.31-->xxx.8
39E1-   B1 30       LDA   ($30),Y         + $30.31-->xxx.4
39E3-   A0 04       LDY   #$04
39E5-   71 30       ADC   ($30),Y
39E7-   85 13       STA   $13
39E9-   85 06       STA   $06       $06 := $13

39EB-   A0 02       LDY   #$02      $12 := $30.31-->xxx.2
39ED-   B1 30       LDA   ($30),Y
39EF-   85 12       STA   $12

39F1-   A0 00       LDY   #$00      $11 := $30.31-->xxx.0 (if +, else #$0)
39F3-   B1 30       LDA   ($30),Y
39F5-   10 01       BPL   $39F8
39F7-   98          TYA
39F8-   85 11       STA   $11

39FA-   84 15       STY   $15       $15 := 0

39FC-   A9 01       LDA   #$01      $B := 0
39FE-   85 0B       STA   $0B
[$F600]
3A00-   A5 34       LDA   $34       $30.31 := $34.35 (MAINWIN)
3A02-   85 30       STA   $30
3A04-   A5 35       LDA   $35
3A06-   85 31       STA   $31
3A08-   20 0C F3    JSR   $F30C   JSR $F30C
                                  Set up $18..$1E from Window header info
                                  Set $0C to VSIZE - 2; $0D to HSIZE - 2
```

```
3A0B-   E6 12       INC   $12     Set $12 to larger of $12 or $0D
3A0D-   A6 0D       LDX   $0D
3A0F-   E4 12       CPX   $12
3A11-   B0 02       BCS   $3A15
3A13-   86 12       STX   $12

3A15-   2C                        NOP or
                                  BEFORE LOOPA
<$3A16> E6 14       BIT   $14E6     INC $14

3A18-   A6 0C       LDX   $0C     MAINWIN VSIZE - 2
3A1A-   E4 14       CPX   $14     Vertical
3A1C-   30 60       BMI   $3A7E
3A1E-   A5 13       LDA   $13
3A20-   85 06       STA   $06
3A22-   A6 14       LDX   $14
3A24-   20 20 F4    JSR   $F420   JSR $F420
                                    Calculate $0E.0F Window Data area(?)
3A27-   84 10       STY   $10
                                  LOOPA:
[$FA29]
3A29-   A4 15       LDY   $15
3A2B-   E6 15       INC   $15
3A2D-   B1 16       LDA   ($16),Y
3A2F-   F0 E5       BEQ   $3A16     JMP $3A16 (before LOOPA)
3A31-   10 1C       BPL   $3A4F     $FA4F
3A33-   C9 FF       CMP   #$FF
3A35-   F0 47       BEQ   $3A7E     RTS
3A37-   C9 C0       CMP   #$C0
3A39-   90 44       BCC   $3A7F     $F67F
3A3B-   E9 C0       SBC   #$C0
3A3D-   AA          TAX
3A3E-   18          CLC
3A3F-   65 06       ADC   $06
3A41-   85 06       STA   $06
3A43-   8A          TXA
3A44-   0A          ASL
3A45-   65 0E       ADC   $0E
3A47-   85 0E       STA   $0E
3A49-   90 DE       BCC   $3A29   JMP LOOPA

3A4B-   E6 0F       INC   $0F
3A4D-   B0 DA       BCS   $3A29   JMP LOOPA

[$FA4F]
3A4F-   AA          TAX
3A50-   C8          INY
3A51-   B1 16       LDA   ($16),Y
3A53-   85 0A       STA   $0A
3A55-   E6 15       INC   $15
3A57-   A4 10       LDY   $10
                                  LOOPB:
3A59-   A5 06       LDA   $06
3A5B-   C5 11       CMP   $11
3A5D-   90 0E       BCC   $3A6D
3A5F-   C5 12       CMP   $12
3A61-   B0 0A       BCS   $3A6D

3A63-   A5 0A       LDA   $0A
3A65-   91 0E       STA   ($0E),Y
3A67-   C8          INY
3A68-   A5 0B       LDA   $0B
3A6A-   91 0E       STA   ($0E),Y
```

```
3A6C-   24                          NOP or INY
<3A6D>     C8       BIT   $C8       C8==INY
3A6E-   C8          INY
3A6F-   D0 02       BNE   $3A73
3A71-   E6 0F       INC   $0F
3A73-   E6 06       INC   $06
3A75-   CA          DEX
3A76-   D0 E1       BNE   $3A59   JMP LOOPB:

3A78-   84 10       STY   $10
3A7A-   A4 15       LDY   $15
3A7C-   D0 AB       BNE   $3A29   JMP LOOPB:
3A7E-   60          RTS

[$F67F]
3A7F-   38          SEC
3A80-   E9 80       SBC   #$80
3A82-   85 18       STA   $18
3A84-   C8          INY
                                  LOOPC:
3A85-   E6 15       INC   $15
3A87-   B1 16       LDA   ($16),Y
3A89-   A4 10       LDY   $10
3A8B-   A6 06       LDX   $06
3A8D-   E4 11       CPX   $11
3A8F-   90 0C       BCC   $3A9D
3A91-   E4 12       CPX   $12
3A93-   B0 08       BCS   $3A9D

3A95-   91 0E       STA   ($0E),Y
3A97-   C8          INY
3A98-   A9 01       LDA   #$01
3A9A-   91 0E       STA   ($0E),Y
3A9C-   24                          NOP or
<3A9D>  C8          BIT   $C8       INY
3A9E-   C8          INY
3A9F-   D0 02       BNE   $3AA3
3AA1-   E6 0F       INC   $0F
3AA3-   E6 06       INC   $06
3AA5-   84 10       STY   $10
3AA7-   A4 15       LDY   $15
3AA9-   C6 18       DEC   $18
3AAB-   D0 D8       BNE   $3A85    JMP LOOPC:

3AAD-   4C 29 F6    JMP   $F629  JMP LOOPA
--------------------------------------------------------------------------------
                                  Clear Screens
                                    Primary TEXT
                                    Secondary TEXT
                                    Primary HiRes
                                  Executed ONCE at start of WizIV
                                  -- or -- when Window handle is NULL
[$F6B0]
3AB0-   A9 00       LDA   #$00
3AB2-   85 06       STA   $06
3AB4-   85 19       STA   $19    $19 === VPOS
3AB6-   A9 18       LDA   #$18   24 lines
3AB8-   85 1B       STA   $1B
3ABA-   A9 28       LDA   #$28   40 columns
3ABC-   85 1A       STA   $1A
                               *** $F6BE = Alternate Entry Point
                                  Do all lines on screen
[$F6BE]
```

```
3ABE-   20 D6 F1    JSR   $F1D6      Self-modify Hi-Res code ($F294 code)
                                     Return 12.13 Primary TEXT ptr
                                     Return 16.17 Secondary TEXT ptr
3AC1-   A4 06       LDY   $06      Y := Horizontal Position
3AC3-   A6 1A       LDX   $1A
3AC5-   A9 00       LDA   #$00     A := #$00
                                   Do next character on this line
3AC7-   91 16       STA   ($16),Y    " " to Secondary TEXT screen
3AC9-   91 12       STA   ($12),Y    " " to Primary TEXT screen
3ACB-   20 94 F2    JSR   $F294      " " to Hi-Res (1 "block")

3ACE-   C8          INY
3ACF-   CA          DEX
3AD0-   D0 F5       BNE   $3AC7    Do next character on this line

3AD2-   E6 19       INC   $19      VPOS := VPOS + 1
3AD4-   C6 1B       DEC   $1B
3AD6-   D0 E6       BNE   $3ABE   Do all lines on screen
3AD8-   60          RTS
--------------------------------------------------------------------------------
                                     Processing '200.CHARSET'
                                     from pascal code P01060C()
3AD9-   A9 04       LDA   #$04   $2D := S6D1 disk drive
3ADB-   85 2D       STA   $2D
3ADD-   A9 D0       LDA   #$D0   $30.31 := destination  #$D000 Bank 1
3ADF-   85 31       STA   $31
3AE1-   A9 10       LDA   #$10   $32.33 := length (#$10 = 16 Sectors? or Blocks?)
3AE3-   85 33       STA   $33
3AE5-   A2 00       LDX   #$00   X := 0 == Read function
3AE7-   86 30       STX   $30
3AE9-   86 32       STX   $32

3AEB-   20 5C E8    JSR   $E85C  Read Blocks

3AEE-   A9 FF       LDA   #$FF   $F1 := #$FF  Special $F1 flag
3AF0-   85 F1       STA   $F1
3AF2-   60          RTS
--------------------------------------------------------------------------------
[$F6F3]
                                 Processing "200.MONSTERS"
3AF3-   A5 34       LDA   $34       $66.67 := 200.MONSTERS FIRSTBLOCK
3AF5-   85 66       STA   $66
3AF7-   A5 35       LDA   $35
3AF9-   85 67       STA   $67
3AFB-   A9 00       LDA   #$00      Clear $74 MONSTERS table (4 entries)
3AFD-   A2 03       LDX   #$03
3AFF-   95 74       STA   $74,X
3B01-   CA          DEX
3B02-   10 FB       BPL   $3AFF
3B04-   60          RTS
--------------------------------------------------------------------------------

                          UNITWRITE( UNIT, ARRAY, LENGTH, BLOCK#, MODE)
                                     $2D   $30.31 $32.33  $34.35  $2E.2F

                                 UNITWRITE( $D, 0, 1, 0)
                                 Clears screens and RETURN KANMP04[] config bits
                                 $F1 is set non-zero here.
                                 $E6 set to #$FF (?)

[$F705]
3B05-   A5 F1       LDA   $F1      $F1 = 0 only on first call to $F705; See $F715.
3B07-   D0 21       BNE   $3B2A     (After 200.CHARSET is loaded, $F1:=#$FF)
```

```
3B09-   A9 31       LDA   #$31     This code executed only ONCE for WizIV
3B0B-   A0 00       LDY   #$00      KANMP04[0] = TRUE  ==ASCII.KRN
3B0D-   91 30       STA   ($30),Y   KANMP04[1] = FALSE ==KANA
3B0F-   98          TYA             KANMP04[2] = FALSE ==KANJI
3B10-   C8          INY             KANMP04[3] = FALSE ==?
3B11-   91 30       STA   ($30),Y   KANMP04[4] = TRUE  ==200.CHARSET
3B13-   A9 80       LDA   #$80      KANMP04[5] = TRUE  ==200.MONSTERS
3B15-   85 F1       STA   $F1      $F1 := #$80; Don't execute this code again!

3B17-   20 B0 F6    JSR   $F6B0    Clear Screens (Primary & Seondary Text; Primary HiRes)

3B1A-   8D 54 C0    STA   $C054    Primary
3B1D-   8D 57 C0    STA   $C057    HiRes
3B20-   8D 52 C0    STA   $C052    No Mixed
3B23-   8D 50 C0    STA   $C050    Graphics
3B26-   A9 FF       LDA   #$FF
3B28-   85 E6       STA   $E6      Special flag
3B2A-   60          RTS            Return
--------------------------------
3B2B-   A5 F1       LDA   $F1      $F1 flag
3B2D-   F0 0B       BEQ   $3B3A
3B2F-   AD 51 C0    LDA   $C051
3B32-   A9 00       LDA   #$00
3B34-   85 F1       STA   $F1
3B36-   85 34       STA   $34
3B38-   85 E6       STA   $E6      special flag
3B3A-   60          RTS
-------------------------------------------------------------------------------
                          MP02 := 0;
                          UNITWRITE( BASE12, MP02, 23, MP02)

                          UNITWRITE( UNIT, ARRAY, LENGTH, BLOCK#, MODE)
                                     $2D   $30.31 $32.33  $34.35  $2E.2F
                                                  Func=23 MP02=0
                          At end of SOLICSM()

[$F73B]
3B3B-   A5 34       LDA   $34   MP02 = 0?
3B3D-   05 35       ORA   $35
3B3F-   F0 4A       BEQ   $3B8B   Yes, F78B Clear top 6 lines of screen
                                        (e.g., the Credits at boot up)

3B41-   A9 60       LDA   #$60
3B43-   85 32       STA   $32
3B45-   A9 06       LDA   #$06
3B47-   85 33       STA   $33
3B49-   A5 B4       LDA   $B4
3B4B-   85 30       STA   $30
3B4D-   85 2E       STA   $2E
3B4F-   A5 B5       LDA   $B5
3B51-   85 31       STA   $31
3B53-   85 2F       STA   $2F

3B55-   20 EE F8    JSR   $F8EE  Disk Read I/O

3B58-   A0 00       LDY   #$00
3B5A-   84 3C       STY   $3C
3B5C-   A9 06       LDA   #$06
3B5E-   85 1B       STA   $1B
3B60-   A9 02       LDA   #$02
3B62-   85 19       STA   $19
                                Outerloop:
3B64-   20 D6 F1    JSR   $F1D6    Get SCREEN ptrs, and Self-Modify Code!
                                   $16.17 --> Secondary Text Screen
```

```
3B67-   18          CLC
3B68-   A9 03       LDA   #$03
3B6A-   85 06       STA   $06
3B6C-   A2 22       LDX   #$22
3B6E-   86 00       STX   $00
                                  Loop:
3B70-   20 D4 F2    JSR   $F2D4

3B73-   E6 06       INC   $06
3B75-   A9 08       LDA   #$08
3B77-   65 2E       ADC   $2E
3B79-   85 2E       STA   $2E
3B7B-   90 03       BCC   $3B80
3B7D-   E6 2F       INC   $2F
3B7F-   18          CLC
3B80-   C6 00       DEC   $00
3B82-   D0 EC       BNE   $3B70    Jmp Loop:

3B84-   E6 19       INC   $19
3B86-   C6 1B       DEC   $1B
3B88-   D0 DA       BNE   $3B64  Jmp OuterLoop:

3B8A-   60          RTS         RTS
--------------------------------
3B8B-   A2 02       LDX   #$02   $19 := #$02  VPOS
3B8D-   86 19       STX   $19
3B8F-   E8          INX         $06 := #$03  HPOS
3B90-   86 06       STX   $06
3B92-   A9 06       LDA   #$06   $1B := #$06  6 lines
3B94-   85 1B       STA   $1B
3B96-   A9 22       LDA   #$22   $1A := #$22  34 columns
3B98-   85 1A       STA   $1A
3B9A-   4C BE F6    JMP   $F6BE  Clear Screen
-------------------------------------------------------------------------------
                                  Table for MONSTER PICs

  Horizontal displacement in MAINWIN to display 4
  ENEMY PICS.

  If 4 displayed then:
    00 = 00.  (since 2 bytes per displayed "char"
    14 = 10.   divide by 2)
    28 = 20.
    3C = 30.
  If 3 displayed then:
    08 = 04.
    1C = 14.
    30 = 24.
  If 2 displayed then:
    14 = 10.
    28 = 20.
  If 1 displayed then:
    1C = 14.

  Displacements are from left edge of MAINWIN.

[$F79D]
3B9D-   3C 30
        28 1C
        14 08
           0
[$F7A4]
        02      This leads to #$D410 where a MONSTER PIC has been read
```

```
        22      This leads to #$D510 ditto
        42      This leads to #$D610 ditto
        62      This leads to #$D710 ditto
--------------------------------------------------------------------------------
[$F7A8]
3BA8-   18          CLC         First time, $78.79 === 00 00,
3BA9-   A9 93       LDA   #$93  $0E.0F := $78.79 + #$293
3BAB-   65 78       ADC   $78
3BAD-   85 0E       STA   $0E
3BAF-   A9 02       LDA   #$02
3BB1-   65 79       ADC   $79
3BB3-   85 0F       STA   $0F
3BB5-   60          RTS
--------------------------------------------------------------------------------
                                  UNITWRITE( $D, xx, 17, 0) From KANJIREA
[$F7B6]
3BB6-   20 A8 F7    JSR   $F7A8     Set every other byte from #$293 to $3F9 to #1.
3BB9-   E6 0F       INC   $0F         ??????? Gross Bug !!!!
3BBB-   A9 01       LDA   #$01        $78.79 should be WINDOW BUFFER
3BBD-   A2 01       LDX   #$01                with HEADER.
3BBF-   A0 66       LDY   #$66        See for example the code at $F7CF

3BC1-   91 0E       STA   ($0E),Y
3BC3-   88          DEY
3BC4-   88          DEY
3BC5-   D0 FA       BNE   $3BC1

3BC7-   91 0E       STA   ($0E),Y
3BC9-   C6 0F       DEC   $0F
3BCB-   CA          DEX
3BCC-   10 F3       BPL   $3BC1

3BCE-   60          RTS
--------------------------------------------------------------------------------
                                  UNITWRITE( BASE12, KANMP02, 13, MAINWIN.I);
                                             13      $30.31   13, $34.35

                                  Display 1 to 4 MONSTER PICS on MAINWIN

                                    -or-

                                  Display 1 dead WERDNA ( 41, 42, 43, 44)

                                    -or-

                                  Display 1 Ghost of ROBERT ( 45, 46, 47, 48)

                                    -or-

                                  Display anything else from 200.MONSTERS (?)

                                  From P01060E()
                                  After solicit "S)tart Game"
                                  and after random ENEMY PIC #s chosen.
                                  KANMP02[0], [1], [2], [3] contains random numbers
                                    from 1 to 24 (or -1). Could be only 1 with non neg.
[$F7CF]
3BCF-   A5 34       LDA   $34    $78.79 := MAINWIN
3BD1-   85 78       STA   $78
3BD3-   A5 35       LDA   $35
3BD5-   85 79       STA   $79
3BD7-   20 B6 F7    JSR   $F7B6  JSR $F7B6  set high order bytes in MAINWIN
                                     to #$1 where ENEMY PICS appear. 5 rows
```

```
                                ---------------------------------
3BDA-   A2 03       LDX   #$03    Init $68, $6C, $70 tables (4 entries per)
3BDC-   A9 FF       LDA   #$FF    Init to #$FF
3BDE-   95 6C       STA   $6C,X
3BE0-   95 70       STA   $70,X
3BE2-   95 68       STA   $68,X
3BE4-   CA          DEX
3BE5-   10 F7       BPL   $3BDE
                                ---------------------------------
                                  Set up DISPLACEMENTS (to pic 0, 1, 2, 3) on screen.
                                  Use $F79D table to set up $70,X (0..3) table.
                                  Displacements from Left Edge of MAINWIN
                                  to display ENEMY PIC.
                                  There are 1 to 4 pics CENTER displayed.

3BE7-   A2 00       LDX   #$00    Process up to 4 ENEMY PICs
3BE9-   A0 03       LDY   #$03
3BEB-   B1 30       LDA   ($30),Y Get KANMP02[ Y] value (ENEMY index, 1..24)
3BED-   30 07       BMI   $3BF6   -1 means no pic
3BEF-   BD 9D F7    LDA   $F79D,X
3BF2-   99 70 00    STA   $0070,Y
3BF5-   E8          INX
3BF6-   E8          INX
3BF7-   88          DEY
3BF8-   10 F1       BPL   $3BEB   Process next KANMP02[ Y] value
                                ---------------------------------
3BFA-   A0 03       LDY   #$03
                                  $F7FC:  To Here From Next Section too.
                                          To see if the one we just inserted
                                          already exists in the list.

                                  See if any in new list of ENEMY pics are
                                  same as previously displayed.
                                  If YES, then we can skip some I/O.
                                  If NO, this is a giant NOP.
[$F7FC]
3BFC-   B1 30       LDA   ($30),Y Get KANMP02[ Y] (ENEMY index, 1..24)
3BFE-   30 1B       BMI   $3C1B

[$F800]
3C00-   A2 03       LDX   #$03
                                  MONSTER INDEX TABLE
                                  ($74,0; $74,1; $74,2) all set to 0 in
                                  PROCESSING MONSTERS.

3C02-   D5 74       CMP   $74,X   KANMP02[ Y] ENEMY INDEX already in MONSTER INDEX TABLE?
3C04-   D0 12       BNE   $3C18     No. Different enemy # in KANMP02[ Y] as $74,X
3C06-   B5 68       LDA   $68,X     Yes. Same enemy # in KANMP02[ Y] as $74,X
3C08-   10 04       BPL   $3C0E

3C0A-   A9 01       LDA   #$01    Indicate enemy index is there,
3C0C-   95 68       STA   $68,X     and no I/O needed
                                  Set up $68,X table
3C0E-   8A          TXA
3C0F-   99 6C 00    STA   $006C,Y ...and $6C,Y table
3C12-   A9 FF       LDA   #$FF
3C14-   91 30       STA   ($30),Y ...and clear the input from P01060E()
3C16-   D0 03       BNE   $3C1B

3C18-   CA          DEX
3C19-   10 E7       BPL   $3C02   Advance to next $74,X to compare with KANMP02[]

3C1B-   88          DEY
3C1C-   10 DE       BPL   $3BFC   Process up to 4 entries
```

```
--------------------------------
                                  ...first time, above here, we do not set anything
                                  in $68,x table or $6C,Y table.

3C1E-   A0 03       LDY   #$03     Process up to 4 entries
3C20-   B1 30       LDA   ($30),Y
3C22-   30 20       BMI   $3C44    -1 means no enemy pic

3C24-   85 11       STA   $11      Found one in list.  Save it at $11
3C26-   A2 03       LDX   #$03

3C28-   B5 68       LDA   $68,X   Look for "empty" entry in $68,X table.
3C2A-   10 15       BPL   $3C41

3C2C-   A9 00       LDA   #$00      Found an available entry.
3C2E-   95 68       STA   $68,X     $68,X := 0 (indicate this enemy # there.)
3C30-   A9 FF       LDA   #$FF                (and need to do I/O)
3C32-   91 30       STA   ($30),Y   ...and clear input from P01060E()
3C34-   A5 11       LDA   $11
3C36-   95 74       STA   $74,X     ...and save the enemy # in $74 table.
3C38-   8A          TXA
3C39-   99 6C 00    STA   $006C,Y   Save index into $74 table in the $6C table
3C3C-   88          DEY
3C3D-   10 BD       BPL   $3BFC   [$F7FC]   Back to previous loop (? ok ?)
3C3F-   30 06       BMI   $3C47             (...first time, not back)

3C41-   CA          DEX
3C42-   10 E4       BPL   $3C28

3C44-   88          DEY
3C45-   10 D9       BPL   $3C20   Process up to 4 entries
--------------------------------
3C47-   A2 03       LDX   #$03    Process up to 4 PICs
3C49-   86 7A       STX   $7A

[$F84B]
3C4B-   A6 7A       LDX   $7A
3C4D-   B4 6C       LDY   $6C,X   Entry have enemy #?
3C4F-   30 05       BMI   $3C56     No, $F856 (JMP $F8B8 and to $F8E6 to DEC $7A)

3C51-   B9 68 00    LDA   $0068,Y   $68,Y = 0?
3C54-   F0 03       BEQ   $3C59       Yes, $F859
                                      No, (-1 or +1) JMP $F8B8
3C56-   4C B8 F8    JMP   $F8B8   SKIP DOING THE I/O.  It is not needed

---------------------------------
[$F859]
3C59-   B9 74 00    LDA   $0074,Y  A := enemy index (1..24)  (1..120!!!)
3C5C-   38          SEC           $34.35 := $66.67 + (index - 1) DIV 2
3C5D-   E9 01       SBC   #$01    A := Make index relative to 0
3C5F-   4A          LSR           A := A / 2
3C60-   08          PHP           Save least bit of index (0 = 0, 1 = 1, 2 = 0, 3 = 1..)
3C61-   18          CLC
3C62-   65 66       ADC   $66     $66.67 is FIRSTBLK of 200.MONSTERS
3C64-   85 34       STA   $34
3C66-   A9 00       LDA   #$00
3C68-   65 67       ADC   $67
3C6A-   85 35       STA   $35     $34.35 is now BLOCK address in 200.MONSTERS for this monster

3C6C-   A9 02       LDA   #$02    $30.31 := $0200 === I/O buffer
3C6E-   85 31       STA   $31
3C70-   A9 00       LDA   #$00
3C72-   85 30       STA   $30
```

```
3C74-   28          PLP
3C75-   08          PHP           Resave least bit of index (Sector 0 or Sector 1)

3C76-   A9 00       LDA   #$00
3C78-   2A          ROL
3C79-   85 33       STA   $33     $33 := 0 or 1 (Sector 0 or Sector 1)

3C7B-   A9 F0       LDA   #$F0    $32.33 := Length of I/O
3C7D-   85 32       STA   $32
3C7F-   A9 04       LDA   #$04    Device := 4 (S6, D1)
3C81-   85 2D       STA   $2D
3C83-   A2 00       LDX   #$00    X := 0 == Read function
3C85-   AD 80 C0    LDA   $C080   Bank 2, No Write
                                  ********************************************
3C88-   20 5C E8    JSR   $E85C   JSR $E85C         Read 200.MONSTER [ index]
                                  ********************************************
3C8B-   AD 8B C0    LDA   $C08B   RamCard Ram, Bank 1 ($D000..$DFFF)
3C8E-   AD 8B C0    LDA   $C08B   Write Enabled
3C91-   A6 7A       LDX   $7A     $CA.CB := #$D010 + (#000001xx 00000000)
3C93-   B5 6C       LDA   $6C,X                       xx is 0 1 2 3
3C95-   09 04       ORA   #$04                           (index into $68 and $74)
3C97-   18          CLC                                 #$D410
3C98-   69 D0       ADC   #$D0                          #$D510
3C9A-   85 CB       STA   $CB                           #$D610
3C9C-   A9 10       LDA   #$10                          #$D710   ...first time
3C9E-   85 CA       STA   $CA

3CA0-   A9 02       LDA   #$02    $C8.C9 := Address in I/O buffer ($200 or $300) to
3CA2-   28          PLP                   Sector 0 or Sector 1 for MONSTER
3CA3-   69 00       ADC   #$00
3CA5-   85 C9       STA   $C9
3CA7-   A9 00       LDA   #$00
3CA9-   85 C8       STA   $C8

3CAB-   A0 EF       LDY   #$EF    XFER from $C8 to $CA buffer (#$F0 bytes for MONSTER)
3CAD-   B1 C8       LDA   ($C8),Y
3CAF-   91 CA       STA   ($CA),Y
3CB1-   88          DEY
3CB2-   D0 F9       BNE   $3CAD
3CB4-   B1 C8       LDA   ($C8),Y
3CB6-   91 CA       STA   ($CA),Y
--------------------------------

[$F8B8]
3CB8-   B4 6C       LDY   $6C,X   Y := index into $68 and $74 tables
3CBA-   30 2A       BMI   $3CE6     (-1, 0, 1, 2, 3)

3CBC-   20 A8 F7    JSR   $F7A8   JSR $F7A8   $0E.0F := $78.79 + #$293
3CBF-   B9 A4 F7    LDA   $F7A4,Y $F7A4 table ($02, $22, $42, $62)
3CC2-   B4 70       LDY   $70,X   Offset to area 0, 1, 2, 3 on screen for PIC

3CC4-   A2 05       LDX   #$05    5 x 6 "chars" to screen for MONSTER picture (?)
3CC6-   86 10       STX   $10

3CC8-   A2 06       LDX   #$06

3CCA-   91 0E       STA   ($0E),Y   Do 1 horizontal row
3CCC-   C8          INY
3CCD-   C8          INY
3CCE-   D0 02       BNE   $3CD2
3CD0-   E6 0F       INC   $0F
3CD2-   69 01       ADC   #$01
3CD4-   CA          DEX
3CD5-   D0 F3       BNE   $3CCA     Do 1 horizontal row (6 "char"s)
```

```
3CD7-   48          PHA             Advance to next row in MAINWIN
3CD8-   98          TYA
3CD9-   69 3C       ADC   #$3C      add #$3C == 60. (60 + 2*6 stores already = 72 =
3CDB-   A8          TAY                           # of bytes for 1 row in this window)
3CDC-   90 03       BCC   $3CE1
3CDE-   E6 0F       INC   $0F
3CE0-   18          CLC
3CE1-   68          PLA
3CE2-   C6 10       DEC   $10
3CE4-   D0 E2       BNE   $3CC8    5 x 6 "char"s to screen

3CE6-   C6 7A       DEC   $7A       Process up to 4 PICs
3CE8-   30 03       BMI   $3CED     All done, $F8ED
3CEA-   4C 4B F8    JMP   $F84B
[$F8ED]
3CED-   60          RTS
-------------------------------------------------------------------------------
                                  $2D = Device

                                  X = Function
                                        0 = Read
                                        1 = Write

                                  $F2 = "Virtual Disk" flag
                                  $2F = ? flag with upper 2 bits (BPL and BVC)

                                  $50 = ?

                                  Determine which code to execute:

                                    $E7FE Normal Disk I/O (?)
                                    $F90B Virtual Disk I/O (?)
                                    $FAAE No I/O but convert disk addr (?)
                                    $F9DE Scan all Disks and init Disk IDs table (?)
[$F8EE]
3CEE-   A5 2D       LDA   $2D      Device = 4?  (S6D1)
3CF0-   C9 04       CMP   #$04
3CF2-   D0 10       BNE   $3D04       No, $E7FE
3CF4-   E0 02       CPX   #$02     Yes.
3CF6-   90 0F       BCC   $3D07    Read? or Write?  Yes, $F907

3CF8-   24 2F       BIT   $2F
3CFA-   10 08       BPL   $3D04    $2F Positive?  Yes, $E7FE (Read Blocks)
3CFC-   50 03       BVC   $3D01    $2F Carry set?  No, $F9DE       2nd UNITWRITE, #$D and BASE.09=1
3CFE-   4C AE FA    JMP   $FAAE      Yes, $FAAE
3D01-   4C DE F9    JMP   $F9DE    2nd UNITWRITE, #$D and BASE.09=1
3D04-   4C FE E7    JMP   $E7FE    $E7FE  Read Blocks

[$F907]
3D07-   24 F2       BIT   $F2      "Virtual Disk" flag
3D09-   10 F9       BPL   $3D04    Not virtual, $F904 -> $E7FE

-------------------------------------------------------------------------------
                                  Virtual Disk I/O using SYSTEM.RELOC
[$F90B]
3D0B-   86 51       STX   $51      Here processing CXP C,1
3D0D-   A5 34       LDA   $34      Loading Segment KANJIREA
3D0F-   85 56       STA   $56
3D11-   A5 35       LDA   $35       $51 := X     (0 coming from $DE37 code)
3D13-   85 57       STA   $57       $52 := $30   Destination for I/O
3D15-   A5 32       LDA   $32       $53 := $31
3D17-   85 54       STA   $54       $54 := $32   Length (in Bytes)
3D19-   A5 33       LDA   $33       $55 := $33
```

```
3D1B-   85 55       STA   $55       $56 := $34   Disk Block Addr (#$85 for KANJIREA)
3D1D-   A5 30       LDA   $30       $57 := $35
3D1F-   85 52       STA   $52
3D21-   A5 31       LDA   $31
3D23-   85 53       STA   $53

[$F925]
3D25-   A9 01       LDA   #$01      $50 := #$01  = 0 means "Please Insert Scenario Disk..."
3D27-   85 50       STA   $50
                                    Convert from Virtual Disk Addr
[$F929]                                     to Physical Disk Addr
3D29-   20 14 FA    JSR   $FA14     JSR $FA14 with $50..$57 set.
                                    $5A.5B get Physical Disk Addr.
3D2C-   90 09       BCC   $3D37
                                    C=1 if Block is not "on-line"
3D2E-   C6 50       DEC   $50       $50 = 0 == Get Virtual Address even if we
                                              will need to read physical
                                              and return with $EF = DISK# (1, 2, 3, 4, 5)
3D30-   10 F7       BPL   $3D29     Now try reading from Physical Disk
3D32-   A2 1E       LDX   #$1E
3D34-   4C 00 FD    JMP   $FD00     Error

[$F937]
3D37-   A2 00       LDX   #$00  Calc difference between beginning of
                                the requested segment and the
                                end of the Segment Section.  Calc Sectors.

3D39-   A5 58       LDA   $58   ...#$1A   length in Blocks to end of
3D3B-   0A          ASL                   whole Segment Section (?)
3D3C-   26 59       ROL   $59   ...#$00   $58.59 := # of sectors (?)
3D3E-   D0 06       BNE   $3D46                   in whole Seg Section(?)
3D40-   C5 55       CMP   $55    weird check (?)
3D42-   F0 06       BEQ   $3D4A    if diff to end of whole Seg Sect is
3D44-   90 04       BCC   $3D4A      less than this segments length us diff (?)
3D46-   A6 54       LDX   $54      else
3D48-   A5 55       LDA   $55        Length (in bytes) for this 1 Segment

3D4A-   85 59       STA   $59   $58.59 := Length needed for this Segment
3D4C-   86 58       STX   $58

3D4E-   A5 50       LDA   $50
3D50-   D0 12       BNE   $3D64  =0 if we need to read disk(?)...#$01 first time

3D52-   A5 EF       LDA   $EF    Disk# (1, 2, 3, 4, 5)
3D54-   20 36 FD    JSR   $FD36  "Please Insert SCENARIO Diskette %n"; Get <CR>
3D57-   20 DE F9    JSR   $F9DE  Set up disk IDs table for diskettes (1, 2, 3, 4, 5)
3D5A-   A5 EF       LDA   $EF
3D5C-   20 03 FA    JSR   $FA03  See if $EF matches one
3D5F-   A8          TAY
3D60-   F0 F0       BEQ   $3D52  Repeat Message
3D62-   86 2D       STX   $2D

[$F964]
3D64-   A6 51       LDX   $51     X := $51
3D66-   A9 00       LDA   #$00    A := #$00
3D68-   85 2E       STA   $2E    2E := 0
3D6A-   85 2F       STA   $2F    2F := 0
3D6C-   A5 58       LDA   $58    30.31 := $52.53 Destination for I/O
3D6E-   85 32       STA   $32    32.33 := $58.59 Length (in bytes)
3D70-   A5 59       LDA   $59    34.35 := $5A.5B Physical Disk Block
3D72-   85 33       STA   $33
3D74-   A5 52       LDA   $52
3D76-   85 30       STA   $30
3D78-   A5 53       LDA   $53
```

```
3D7A-   85 31       STA   $31
3D7C-   A5 5A       LDA   $5A
3D7E-   85 34       STA   $34
3D80-   A5 5B       LDA   $5B
3D82-   85 35       STA   $35
3D84-   20 FE E7    JSR   $E7FE  READ BLOCKS

3D87-   38          SEC
3D88-   A5 54       LDA   $54
3D8A-   E5 58       SBC   $58
3D8C-   85 54       STA   $54
3D8E-   A5 55       LDA   $55
3D90-   E5 59       SBC   $59
3D92-   85 55       STA   $55
3D94-   05 54       ORA   $54
3D96-   F0 15       BEQ   $3DAD

3D98-   18          CLC
3D99-   A5 53       LDA   $53
3D9B-   65 59       ADC   $59
3D9D-   85 53       STA   $53
3D9F-   A5 59       LDA   $59
3DA1-   4A          LSR
3DA2-   65 56       ADC   $56
3DA4-   85 56       STA   $56
3DA6-   90 02       BCC   $3DAA
3DA8-   E6 57       INC   $57
3DAA-   4C 25 F9    JMP   $F925    JMP $F925

3DAD-   60          RTS
--------------------------------------------------------------------------------
[$F9AE]
3DAE-   A9 80       LDA   #$80     2F := #$80
3DB0-   85 2F       STA   $2F
3DB2-   A9 02       LDA   #$02     30.31 := #$200  Destination ($200..$3FF)
3DB4-   85 31       STA   $31
3DB6-   A2 02       LDX   #$02
3DB8-   86 33       STX   $33      32.33 := #$200  Length (1 Block = $200 bytes)
3DBA-   CA          DEX
3DBB-   86 34       STX   $34      34.35 := #$0001 Block # to read
3DBD-   CA          DEX                X : = #$0   (Read (?))
3DBE-   86 32       STX   $32
3DC0-   86 35       STX   $35
3DC2-   86 30       STX   $30
3DC4-   A5 5C       LDA   $5C      Drive (e.g., 4 = S6D1)
3DC6-   85 2D       STA   $2D      2D := Drive
3DC8-   20 FE E7    JSR   $E7FE    Read Blocks
3DCB-   8A          TXA
3DCC-   D0 0D       BNE   $3DDB

3DCE-   A0 FE       LDY   #$FE      A := #$FE
3DD0-   A9 03       LDA   #$03      00.01 := #$300
3DD2-   85 01       STA   $01
3DD4-   A9 00       LDA   #$00
3DD6-   85 00       STA   $00
3DD8-   B1 00       LDA   ($00),Y   A := $3FE value   Disk# e.g., 1, 2, 3, 4, 5
3DDA-   60          RTS             Return

3DDB-   A9 00       LDA   #$00      A := 0
3DDD-   60          RTS             Return
--------------------------------------------------------------------------------
                                from $F901, processing Param2 := #$01
                                          at WRITE VOL#    UNITWRITE(#$D...)
```

```
                                  Set up $5D..$65 table with Disk IDs
                                    (1, 2, 3, 4, 5) from end of BLOCK #1 on disks.
                                    $5D is for Device 4

[$F9DE]
3DDE-   A2 09       LDX   #$09      Init zpage locs $5D..$65 to #$FF
3DE0-   A9 FF       LDA   #$FF
3DE2-   95 5C       STA   $5C,X
3DE4-   CA          DEX
3DE5-   D0 FB       BNE   $3DE2

3DE7-   A2 0C       LDX   #$0C      Check 12, 11, 10, 9
                                      ...then 5, 4 for valid drives
3DE9-   86 5C       STX   $5C
3DEB-   BD 63 FC    LDA   $FC63,X   Slot# * 16 for this Pascal Device (4=S6D1)
3DEE-   30 07       BMI   $3DF7
[$F9F0]
3DF0-   20 AE F9    JSR   $F9AE     Found one!  $F9AE
3DF3-   A6 5C       LDX   $5C
3DF5-   95 59       STA   $59,X

3DF7-   CA          DEX
3DF8-   E0 08       CPX   #$08
3DFA-   D0 02       BNE   $3DFE
3DFC-   A2 05       LDX   #$05
3DFE-   E0 04       CPX   #$04
3E00-   B0 E7       BCS   $3DE9
3E02-   60          RTS
--------------------------------------------------------------------------------
[$FA03]
3E03-   A2 0C       LDX   #$0C
3E05-   D5 59       CMP   $59,X
3E07-   F0 08       BEQ   $3E11
3E09-   CA          DEX
3E0A-   E0 04       CPX   #$04
3E0C-   B0 F7       BCS   $3E05
3E0E-   A9 00       LDA   #$00
3E10-   60          RTS
3E11-   A9 01       LDA   #$01
3E13-   60          RTS
--------------------------------------------------------------------------------
                                Convert "virtual" disk# to physical disk#
                                SYSTEM.RELOC has this format:
                                  00.01:  0002
                                  02.xx:
                                     Each disk represented by 2 words,
                                       followed by list of 3 word entries
                                     02.03: Disk# (1,2,3,4,5)
                                     04.05: # of Segments this Disk#
                                     06.yy:
                                        Each segment represented by 3 words
                                        06.07: Virtual Block#
                                        08.09: Physical Block#
                                        0A.0B: Length( in Blocks)

                                $50 := #$01  (?) or #$00 (?)
                                $51 := X    (Sometimes 0, other times 4?)
                                $52 := $30  Destination for I/O to load Segment into
                                $53 := $31
                                $54 := $32  Length of I/O in Bytes
                                $55 := $33
                                $56 := $34  Disk Block Addr (#$85 for KANJIREA)
                                $57 := $35    (From Directory +
                                              SYSTEM.PASCAL FirstBlock)
```

```
[$FA14]
3E14-   A9 10       LDA   #$10       to here processing KANJIREA
3E16-   85 01       STA   $01
3E18-   A9 02       LDA   #$02
3E1A-   85 00       STA   $00        $00.$01 := #$1002  (SYSTEM.RELOC loaded to $1000 and altered)

[$FA1C]
3E1C-   A0 00       LDY   #$00       To here to process up to 5 disks
3E1E-   B1 00       LDA   ($00),Y
3E20-   85 EF       STA   $EF        $EF := DISK NUMBER (1, 2, 3, 4, 5)
3E22-   F0 66       BEQ   $3E8A

3E24-   A0 02       LDY   #$02
3E26-   B1 00       LDA   ($00),Y    $02.03 := # of Segment Sections
3E28-   85 02       STA   $02                  on this Disk
3E2A-   C8          INY
3E2B-   B1 00       LDA   ($00),Y
3E2D-   85 03       STA   $03

3E2F-   A9 04       LDA   #$04       Advance $00 to point to first "triplet"
3E31-   18          CLC
3E32-   65 00       ADC   $00
3E34-   85 00       STA   $00
3E36-   90 02       BCC   $3E3A
3E38-   E6 01       INC   $01

3E3A-   24 2F       BIT   $2F        [$2F is 0 first time (?)]
3E3C-   70 0B       BVS   $3E49
3E3E-   A5 EF       LDA   $EF        A := disk #  (1, 2, 3, 4, 5)
3E40-   20 03 FA    JSR   $FA03      X := Device(A); If found,
3E43-   86 2D       STX   $2D             Convert to UNIT (4,5, 9,10, 11,12)
3E45-   45 50       EOR   $50             A=0 == Not found;  A=1 == Found
3E47-   D0 2B       BNE   $3E74      Disk not "on-line",
                                       Branch to $FA74 to try next disk.

                                     Found disk # (1, 2, 3..) in drive X

                                        ==OR== Its not on-line but we really ===
                                        ====== want to check it, so we may   ===
                                        ====== need to "LOAD DISK".          ===
[$FA49]
3E49-   A0 04       LDY   #$04          LOOP
3E4B-   38          SEC                    $58.59 := Word 3 - $56.57
3E4C-   B1 00       LDA   ($00),Y          Look up $56.57 in RELOC table.
3E4E-   E5 56       SBC   $56              (3 words per segment)
3E50-   85 58       STA   $58              (Look for match with Word 3)
3E52-   C8          INY
3E53-   B1 00       LDA   ($00),Y
3E55-   E5 57       SBC   $57
3E57-   85 59       STA   $59

3E59-   05 58       ORA   $58
3E5B-   F0 02       BEQ   $3E5F            Equal, then $FA5F
3E5D-   B0 2D       BCS   $3E8C [$FA8C]    Carry=1 if Word 3 > $56.57 (MATCH!?)
                                           Carry=0 if Word 3 < $56.57
[$FA5F]
3E5F-   A9 06       LDA   #$06             Advance $00.01 to next
3E61-   18          CLC                      Segment entry in RELOC table
3E62-   65 00       ADC   $00                (or to next disk section)
3E64-   85 00       STA   $00
3E66-   90 02       BCC   $3E6A
3E68-   E6 01       INC   $01
3E6A-   C6 02       DEC   $02              Decrement Segment Counter
```

```
3E6C-   D0 DB       BNE   $3E49 [$FA49] LOOP
3E6E-   C6 03       DEC   $03             Decrement Segment Counter (high)
3E70-   10 D7       BPL   $3E49 [$FA49] LOOP
3E72-   30 A8       BMI   $3E1C [$FA1C] Not found on that disk, BRANCH always

                                        LOOPB advances $00.01 to
                                          next disk entry in RELOC table
[$FA74]
3E74-   A9 06       LDA   #$06            Advance $00.01 to next
3E76-   18          CLC                     Segment entry in RELOC table
3E77-   65 00       ADC   $00               (or to next disk section)
3E79-   85 00       STA   $00
3E7B-   90 02       BCC   $3E7F
3E7D-   E6 01       INC   $01
3E7F-   C6 02       DEC   $02             Decrement Segment Counter
3E81-   D0 F1       BNE   $3E74
3E83-   C6 03       DEC   $03             Decrement Segment Counter
3E85-   10 ED       BPL   $3E74          LOOPB
3E87-   4C 1C FA    JMP   $FA1C          $00.01 point to next disk entry
                                        Try to find it on next disk

3E8A-   38          SEC
3E8B-   60          RTS

                                    By definition, word 1 is always
                                      greater than word 3 above it,
                                      and always less than word 3 for
                                      this entry.

                                    Word 3 > $56.57 (Match!?)
[$FA8C]
3E8C-   A0 00       LDY   #$00    Let's see if it really matches
3E8E-   38          SEC
3E8F-   A5 56       LDA   $56     $5A.5B := $56.57 - word 1
3E91-   F1 00       SBC   ($00),Y
3E93-   85 5A       STA   $5A
3E95-   C8          INY
3E96-   A5 57       LDA   $57
3E98-   F1 00       SBC   ($00),Y
3E9A-   85 5B       STA   $5B
3E9C-   90 D6       BCC   $3E74   Not found on this disk, try next one

                                    Found Segment entry on this disk!!!
3E9E-   C8          INY           Point to word 2 in RELOC table
3E9F-   18          CLC
3EA0-   B1 00       LDA   ($00),Y   If the Disk ADDR we are looking for
3EA2-   65 5A       ADC   $5A       is >= word 1 entry and less than
3EA4-   85 5A       STA   $5A       word 3 entry, then $5A.5B holds
3EA6-   C8          INY             the difference.  Add that difference
3EA7-   B1 00       LDA   ($00),Y   to word 2 to get the REAL DISK ADDR.
3EA9-   65 5B       ADC   $5B       Note, if diff = 0, then this is
3EAB-   85 5B       STA   $5B       simply word 2.
3EAD-   60          RTS
--------------------------------------------------------------------------------

[$FAAE]
3EAE-   A5 34       LDA   $34
3EB0-   85 56       STA   $56
3EB2-   A5 35       LDA   $35
3EB4-   85 57       STA   $57
3EB6-   20 14 FA    JSR   $FA14  Convert from "Virtual" Disk Addr to Physical
3EB9-   A0 00       LDY   #$00
3EBB-   90 02       BCC   $3EBF
3EBD-   84 EF       STY   $EF
```

```
3EBF-   A5 EF       LDA   $EF      Return disk#  (1, 2, 3, 4, 5)
3EC1-   91 30       STA   ($30),Y
3EC3-   98          TYA
3EC4-   C8          INY
3EC5-   91 30       STA   ($30),Y
3EC7-   60          RTS
--------------------------------------------------------------------------------

[$FAC8]
3EC8-   A0 04       LDY   #$04
3ECA-   86 04       STX   $04
3ECC-   85 05       STA   $05

3ECE-   A2 30       LDX   #$30
3ED0-   A5 04       LDA   $04

3ED2-   D9 3A FE    CMP   $FE3A,Y   $FE3A
3ED5-   A5 05       LDA   $05
3ED7-   F9 35 FE    SBC   $FE35,Y   $FE35
3EDA-   90 0C       BCC   $3EE8
3EDC-   85 05       STA   $05
3EDE-   A5 04       LDA   $04
3EE0-   F9 3A FE    SBC   $FE3A,Y   $FE3A
3EE3-   85 04       STA   $04
3EE5-   E8          INX
3EE6-   D0 EA       BNE   $3ED2

3EE8-   8A          TXA
3EE9-   99 AA 0C    STA   $0CAA,Y
3EEC-   88          DEY
3EED-   10 DF       BPL   $3ECE
3EEF-   60          RTS
------------------------------------------------
3EF0-   01 02       ORA   ($02,X)
3EF2-   04          ???
3EF3-   08          PHP
3EF4-   10 20       BPL   $3F16
3EF6-   40          RTI
3EF7-   80          ???
3EF8-   00          BRK
3EF9-   00          BRK
3EFA-   00          BRK
3EFB-   00          BRK
3EFC-   00          BRK
3EFD-   00          BRK
3EFE-   00          BRK
3EFF-   00          BRK

[$FB00]
3F00-   00          BRK
3F01-   01 03       ORA   ($03,X)
3F03-   07          ???
3F04-   0F          ???
3F05-   1F          ???
3F06-   3F          ???
3F07-   7F          ???
3F08-   FF          ???
3F09-   FF          ???
3F0A-   FF          ???
3F0B-   FF          ???
3F0C-   FF          ???
3F0D-   FF          ???
3F0E-   FF          ???
3F0F-   FF          ???
```

	3F10-	FF	???						
	3F11-	FE FC F8	INC	$F8FC,X					
	3F14-	F0 E0	BEQ	$3EF6					
	3F16-	C0 80	CPY	#$80					
	3F18-	00	BRK						
	3F19-	00	BRK						
	3F1A-	00	BRK						
	3F1B-	00	BRK						
	3F1C-	00	BRK						
	3F1D-	00	BRK						
	3F1E-	00	BRK						
	3F1F-	00	BRK						
	3F20-	00	BRK						
	3F21-	01 02	ORA	($02,X)					
	3F23-	0A	ASL						
	3F24-	0B	???						
	3F25-	0C	???						
	3F26-	03	???						
	3F27-	04	???						
	3F28-	05 06	ORA	$06					
	3F2A-	00	BRK						
	3F2B-	00	BRK						
	3F2C-	00	BRK						
	3F2D-	00	BRK						
	3F2E-	00	BRK						
	3F2F-	00	BRK						
	3F30-	00	BRK						
	3F31-	00	BRK						
	3F32-	00	BRK						
	3F33-	00	BRK						
	3F34-	00	BRK						
	3F35-	00	BRK						
	3F36-	00	BRK						
	3F37-	00	BRK						
	3F38-	00	BRK						
	3F39-	00	BRK						
	3F3A-	00	BRK						
	3F3B-	00	BRK						
	3F3C-	00	BRK						
	3F3D-	00	BRK						
	3F3E-	00	BRK						
	3F3F-	00	BRK						
	3F40-	00	BRK						
	3F41-	00	BRK						
	3F42-	00	BRK						
	3F43-	00	BRK						
	3F44-	00	BRK						
	3F45-	00	BRK						
	3F46-	00	BRK						
	3F47-	00	BRK						
	3F48-	00	BRK						
	3F49-	00	BRK						
	3F4A-	00	BRK						
	3F4B-	00	BRK						
	3F4C-	00	BRK						
	3F4D-	00	BRK						
	3F4E-	00	BRK						
	3F4F-	00	BRK						
	3F50-	00	BRK						
	3F51-	00	BRK						
	3F52-	00	BRK						
	3F53-	00	BRK						
	3F54-	00	BRK						
	3F55-	00	BRK						

	3F56- 00	BRK						
	3F57- 00	BRK						
	3F58- 00	BRK						
	3F59- 00	BRK						
	3F5A- 00	BRK						
	3F5B- 00	BRK						
	3F5C- 00	BRK						
	3F5D- 00	BRK						
	3F5E- 00	BRK						
	3F5F- 00	BRK						
	3F60- 00	BRK						
	3F61- 00	BRK						
	3F62- 00	BRK						
	3F63- 00	BRK						
	3F64- 00	BRK						
	3F65- 00	BRK						
	3F66- 00	BRK						
	3F67- 00	BRK						
	3F68- 00	BRK						
	3F69- 00	BRK						
	3F6A- 00	BRK						
	3F6B- 00	BRK						
	3F6C- 00	BRK						
	3F6D- 00	BRK						
	3F6E- 00	BRK						
	3F6F- 00	BRK						
	3F70- 00	BRK						
	3F71- 00	BRK						
	3F72- 00	BRK						
	3F73- 00	BRK						
	3F74- 00	BRK						
	3F75- 00	BRK						
	3F76- 00	BRK						
	3F77- 00	BRK						
	3F78- 00	BRK						
	3F79- 00	BRK						
	3F7A- 00	BRK						
	3F7B- 00	BRK						
	3F7C- 00	BRK						
	3F7D- 00	BRK						
	3F7E- 00	BRK						
	3F7F- 00	BRK						
	3F80- 00	BRK						
	3F81- 00	BRK						
	3F82- 00	BRK						
	3F83- 00	BRK						
	3F84- 00	BRK						
	3F85- 00	BRK						
	3F86- 00	BRK						
	3F87- 00	BRK						
	3F88- 00	BRK						
	3F89- 00	BRK						
	3F8A- 00	BRK						
	3F8B- 00	BRK						
	3F8C- 00	BRK						
	3F8D- 00	BRK						
	3F8E- 00	BRK						
	3F8F- 00	BRK						
	3F90- 00	BRK						
	3F91- 00	BRK						
	3F92- 00	BRK						
	3F93- 00	BRK						
	3F94- 00	BRK						
	3F95- 00	BRK						

	3F96- 00	BRK						
	3F97- 00	BRK						
	3F98- 00	BRK						
	3F99- 00	BRK						
	3F9A- 00	BRK						
	3F9B- 00	BRK						
	3F9C- 00	BRK						
	3F9D- 00	BRK						
	3F9E- 00	BRK						
	3F9F- 00	BRK						
	3FA0- 00	BRK						
	3FA1- 00	BRK						
	3FA2- 00	BRK						
	3FA3- 00	BRK						
	3FA4- 00	BRK						
	3FA5- 00	BRK						
	3FA6- 00	BRK						
	3FA7- 00	BRK						
	3FA8- 00	BRK						
	3FA9- 00	BRK						
	3FAA- 00	BRK						
	3FAB- 00	BRK						
	3FAC- 00	BRK						
	3FAD- 00	BRK						
	3FAE- 00	BRK						
	3FAF- 00	BRK						
	3FB0- 00	BRK						
	3FB1- 00	BRK						
	3FB2- 00	BRK						
	3FB3- 00	BRK						
	3FB4- 00	BRK						
	3FB5- 00	BRK						
	3FB6- 00	BRK						
	3FB7- 00	BRK						
	3FB8- 00	BRK						
	3FB9- 00	BRK						
	3FBA- 00	BRK						
	3FBB- 00	BRK						
	3FBC- 00	BRK						
	3FBD- 00	BRK						
	3FBE- 00	BRK						
	3FBF- 00	BRK						
	3FC0- 00	BRK						
	3FC1- 00	BRK						
	3FC2- 00	BRK						
	3FC3- 00	BRK						
	3FC4- 00	BRK						
	3FC5- 00	BRK						
	3FC6- 00	BRK						
	3FC7- 00	BRK						
	3FC8- 00	BRK						
	3FC9- 00	BRK						
	3FCA- 00	BRK						
	3FCB- 00	BRK						
	3FCC- 00	BRK						
	3FCD- 00	BRK						
	3FCE- 00	BRK						
	3FCF- 00	BRK						
	3FD0- 00	BRK						
	3FD1- 00	BRK						
	3FD2- 00	BRK						
	3FD3- 00	BRK						
	3FD4- 00	BRK						
	3FD5- 00	BRK						

```
3FD6-   00          BRK
3FD7-   00          BRK
3FD8-   00          BRK
3FD9-   00          BRK
3FDA-   00          BRK
3FDB-   00          BRK
3FDC-   00          BRK
3FDD-   00          BRK
3FDE-   00          BRK
3FDF-   00          BRK
3FE0-   00          BRK
3FE1-   00          BRK
3FE2-   00          BRK
3FE3-   00          BRK
--------------------------------------------------------------------------------
                        $FBE4 Banks table (see bootstrap)
[$FBE4]
                        $FBE4 table changed to:
                           03 04 01 05
3FE4-   00          BRK
3FE5-   00          BRK    03 04 01 05 indicates 1 bank in Aux Mem (?)
3FE6-   00          BRK              05 is a terminator to list.
3FE7-   00          BRK              If more had been found, then
3FE8-   00          BRK              repeat 03 04 01 in the list.
3FE9-   00          BRK
3FEA-   00          BRK
3FEB-   00          BRK
3FEC-   00          BRK
3FED-   00          BRK
3FEE-   00          BRK
3FEF-   00          BRK
3FF0-   00          BRK
3FF1-   00          BRK
3FF2-   00          BRK
3FF3-   00          BRK
3FF4-   00          BRK
3FF5-   00          BRK
3FF6-   00          BRK
3FF7-   00          BRK
3FF8-   00          BRK
3FF9-   00          BRK
3FFA-   00          BRK
3FFB-   00          BRK
3FFC-   00          BRK
3FFD-   00          BRK
3FFE-   00          BRK
3FFF-   00          BRK

                    Table of p-code JMPs to instructions
                      (or possibly pseudo instructions?)
                      $FD00 is high half. indexed by X.
                      X=CD is first time.
[$FC00]
4000-   00          BRK
4001-   00          BRK
4002-   00          BRK
4003-   00          BRK
4004-   00          BRK
4005-   00          BRK
4006-   00          BRK
4007-   00          BRK
4008-   00          BRK
4009-   00          BRK
400A-   00          BRK
```

```
400B-   00          BRK
400C-   00          BRK
400D-   00          BRK
400E-   00          BRK
400F-   00          BRK
4010-   00          BRK
4011-   00          BRK
4012-   00          BRK
4013-   00          BRK
4014-   00          BRK
4015-   00          BRK
4016-   00          BRK
4017-   00          BRK
4018-   00          BRK
4019-   00          BRK
401A-   00          BRK
401B-   00          BRK
401C-   00          BRK
401D-   00          BRK
401E-   00          BRK
401F-   00          BRK
4020-   00          BRK
4021-   00          BRK
4022-   00          BRK
4023-   00          BRK
-------------------------------------------------------------------------------
                    Aux Memory table used with $FBE4 table
                    Set up during BOOTSTRAP.
[$FC24]
4024-   00          BRK   FC24: 00 00 00 00     <= After BOOTSTRAP
4025-   00          BRK                 00 Terminator
4026-   00          BRK         First 3 values are the BANK#.
4027-   00          BRK         For AppleWin, Aux memory has only 1 bank.
4028-   00          BRK         RAMWorks can have multiple 64K banks.
4029-   00          BRK         There are 3 entries for each 64K bank.
402A-   00          BRK
402B-   00          BRK
402C-   00          BRK
402D-   00          BRK
402E-   00          BRK
402F-   00          BRK
4030-   00          BRK
4031-   00          BRK
4032-   00          BRK
4033-   00          BRK
4034-   00          BRK
4035-   00          BRK
4036-   00          BRK
4037-   00          BRK
4038-   00          BRK
4039-   00          BRK
403A-   00          BRK
403B-   00          BRK
403C-   00          BRK
403D-   00          BRK
403E-   00          BRK
403F-   00          BRK
4040-   00          BRK
4041-   00          BRK
4042-   00          BRK
4043-   00          BRK
4044-   00          BRK
4045-   00          BRK
4046-   00          BRK
```

```
4047-   00          BRK
4048-   00          BRK
4049-   00          BRK
404A-   00          BRK
404B-   00          BRK
404C-   00          BRK
404D-   00          BRK
404E-   00          BRK
404F-   00          BRK
4050-   00          BRK
4051-   00          BRK
4052-   00          BRK
4053-   00          BRK
4054-   00          BRK
4055-   00          BRK
4056-   00          BRK
4057-   00          BRK
4058-   00          BRK
4059-   00          BRK
405A-   00          BRK
405B-   00          BRK
405C-   00          BRK
405D-   00          BRK
405E-   00          BRK
405F-   00          BRK
4060-   00          BRK
4061-   00          BRK
4062-   00          BRK

[$FC63]
4063-   00          BRK     Device table (?)
4064-   FF          ???
4065-   FF          ???
4066-   FF          ???
4067-   FF          ???  4  Set to $60 (Boot slot)  4 = slot 6 drive 1 maybe
4068-   FF          ???  5
4069-   FF          ???
406A-   FF          ???
406B-   FF          ???
406C-   FF          ???  9
406D-   FF          ??? 10
406E-   FF          ??? 11
406F-   FF          ??? 12
4070-   FF          ???

[$FC71]
4071-   FF          ???     Device table (?)   LSB = Drive 1 or 2
4072-   FF          ???
4073-   FF          ???
4074-   FF          ???
4075-   FF          ???  4   Set to 0 before INTERP starts (4 = slot 6 drive 1 maybe)
4076-   FF          ???  5
4077-   FF          ???
4078-   FF          ???
4079-   FF          ???
407A-   FF          ???  9
407B-   FF          ??? 10
407C-   FF          ??? 11
407D-   FF          ??? 12
407E-   FF          ???
407F-   FF          ???

                                $FC00,80 table for Pcodes with $FD00,80
4080-   69 03       ADC   #$03
```

```
4082-  84 03       STY   $03
4084-  7B          ???
4085-  4D F4 03    EOR   $03F4
4088-  46 03       LSR   $03
408A-  03          ???
408B-  97          ???
408C-  28          PLP
408D-  8D 1E B1    STA   $B11E
4090-  03          ???
4091-  6F          ???
4092-  03          ???
4093-  9F          ???
4094-  23          ???
4095-  97          ???
4096-  03          ???
4097-  1A          ???
4098-  E7          ???
4099-  03          ???
409A-  1E 29 F7    ASL   $F729,X
409D-  03          ???
409E-  7B          ???            $FC00,9E == CSP $DC7B
409F-  90 D0       BCC   $4071
40A1-  FF          ???
40A2-  6D 0B 7B    ADC   $7B0B
40A5-  D0 CE       BNE   $4075
40A7-  03          ???
40A8-  56 AC       LSR   $AC,X
40AA-  E2          ???
40AB-  EC 27 16    CPX   $1627
40AE-  CB          ???
40AF-  AF          ???
40B0-  AF          ???
40B1-  AF          ???
40B2-  CB          ???
40B3-  6C AF AF    JMP   ($AFAF)
40B6-  B2          ???
40B7-  AF          ???
40B8-  F2          ???
40B9-  04          ???
40BA-  F9 2F 8C    SBC   $8C2F,Y
40BD-  A7          ???
40BE-  34          ???
40BF-  4D C4 06    EOR   $06C4
40C2-  82          ???
40C3-  68          PLA
40C4-  4A          LSR
40C5-  9A          TXS
40C6-  D6 E6       DEC   $E6,X
40C8-  15 2A       ORA   $2A,X
40CA-  B8          CLV
40CB-  82          ???
40CC-  F8          SED

[$FCCD]
40CD-  9F          ???            CXP  $D59F  (See $FxCD)
40CE-  72          ???
40CF-  63          ???
40D0-  A9 03       LDA   #$03
40D2-  03          ???
40D3-  03          ???
40D4-  03          ???
40D5-  03          ???
40D6-  03          ???
40D7-  97          ???
```

```
40D8-   82          ???
40D9-   82          ???
40DA-   82          ???
40DB-   82          ???
40DC-   82          ???
40DD-   82          ???
40DE-   82          ???
40DF-   82          ???
40E0-   82          ???
40E1-   82          ???
40E2-   82          ???
40E3-   82          ???
40E4-   82          ???
40E5-   82          ???
40E6-   82          ???
40E7-   82          ???
40E8-   8E 8E 8E    STX   $8E8E
40EB-   8E 8E 8E    STX   $8E8E
40EE-   8E 8E 8E    STX   $8E8E
40F1-   8E 8E 8E    STX   $8E8E
40F4-   8E 8E 8E    STX   $8E8E
40F7-   8E 9A 9A    STX   $9A9A
40FA-   9A          TXS
40FB-   9A          TXS
40FC-   9A          TXS
40FD-   9A          TXS
40FE-   9A          TXS
40FF-   9A          TXS

[$FD00]                         $FD00,x = Pcode Table
4100-   E0 10       CPX   #$10
4102-   F0 24       BEQ   $4128
4104-   A9 A8       LDA   #$A8
4106-   8D BA 0C    STA   $0CBA
4109-   A2 FF       LDX   #$FF
410B-   8E BB 0C    STX   $0CBB
410E-   A0 CD       LDY   #$CD
4110-   A5 EF       LDA   $EF
4112-   09 40       ORA   #$40
4114-   24 F2       BIT   $F2     "Virtual Disk" flag
4116-   30 07       BMI   $411F
4118-   A4 2D       LDY   $2D
411A-   B9 1D FB    LDA   $FB1D,Y
411D-   A0 DB       LDY   #$DB
411F-   8C D2 0C    STY   $0CD2
4122-   8D D4 0C    STA   $0CD4
4125-   4C 48 FD    JMP   $FD48

4128-   A9 00       LDA   #$00
412A-   8D BA 0C    STA   $0CBA
412D-   8D EA 0C    STA   $0CEA
4130-   A9 FF       LDA   #$FF
4132-   A5 EF       LDA   $EF
4134-   10 08       BPL   $413E
-------------------------------------------------------
[$FD36]
                                CBA.CBB := #0.0
                                CD2.CD3 := #$FF1C = "Please Insert ..." msg
                                CD4     := "B" == insertion for disk#
                                CE9.CEA := #$FF3F = "Press ~ To Continue"

4136-   A2 00       LDX   #$00
4138-   8E BB 0C    STX   $0CBB
413B-   8E BA 0C    STX   $0CBA
```

```
413E-   09 40       ORA   #$40
4140-   8D D4 0C    STA   $0CD4
4143-   A9 1C       LDA   #$1C
4145-   8D D2 0C    STA   $0CD2

4148-   A2 FF       LDX   #$FF
414A-   8E D3 0C    STX   $0CD3
414D-   A9 3F       LDA   #$3F
414F-   8D E9 0C    STA   $0CE9
4152-   8E EA 0C    STX   $0CEA
4155-   4C A9 EB    JMP   $EBA9
-------------------------------------------------------
4158-   00          BRK
4159-   00          BRK
415A-   00          BRK
415B-   00          BRK
415C-   00          BRK
415D-   00          BRK
415E-   00          BRK
415F-   00          BRK
4160-   00          BRK
4161-   00          BRK
4162-   00          BRK
4163-   00          BRK
4164-   00          BRK
4165-   00          BRK
4166-   00          BRK
4167-   00          BRK
4168-   00          BRK
4169-   00          BRK
416A-   00          BRK
416B-   00          BRK
416C-   00          BRK
416D-   00          BRK
416E-   00          BRK
416F-   00          BRK
4170-   00          BRK
4171-   00          BRK
4172-   00          BRK
4173-   00          BRK
4174-   00          BRK
4175-   00          BRK
4176-   00          BRK
4177-   00          BRK
4178-   00          BRK
4179-   00          BRK
417A-   00          BRK
417B-   00          BRK
417C-   00          BRK
417D-   00          BRK
417E-   00          BRK
417F-   00          BRK

[$FD80]                          $FD80,x = Pcode Table
4180-   DA          ???
4181-   FE DA FE    INC   $FEDA,X
4184-   DB          ???
4185-   DA          ???
4186-   DA          ???
4187-   FE DB FE    INC   $FEDB,X
418A-   FE D9 DA    INC   $DAD9,X
418D-   DB          ???
418E-   DB          ???
418F-   DA          ???
```

```
4190-   FE DA FE    INC   $FEDA,X
4193-   DB          ???
4194-   D9 DA FE    CMP   $FEDA,Y
4197-   D9 DA FE    CMP   $FEDA,Y
419A-   D2          ???
419B-   D3          ???
419C-   D9 FE DC    CMP   $DCFE,Y    $FC00,9E == CSP $DC7B
419F-   00          BRK
41A0-   D8          CLD
41A1-   D0 D3       BNE   $4176
41A3-   D2          ???
41A4-   D3          ???
41A5-   D1 D2       CMP   ($D2),Y
41A7-   FE D3 D1    INC   $D1D3,X
41AA-   D2          ???
41AB-   D1 D1       CMP   ($D1),Y
41AD-   D6 D5       DEC   $D5,X
41AF-   D7          ???
41B0-   D7          ???
41B1-   D7          ???
41B2-   D1 D2       CMP   ($D2),Y
41B4-   D7          ???
41B5-   D7          ???
41B6-   D1 D7       CMP   ($D7),Y
41B8-   D1 D1       CMP   ($D1),Y
41BA-   D3          ???
41BB-   D4          ???
41BC-   D2          ???
41BD-   D2          ???
41BE-   D2          ???
41BF-   D2          ???
41C0-   D3          ???
41C1-   D6 D5       DEC   $D5,X
41C3-   D7          ???
41C4-   D7          ???
41C5-   D7          ???
41C6-   D1 D0       CMP   ($D0),Y
41C8-   D7          ???
41C9-   D7          ???
41CA-   D1 D7       CMP   ($D7),Y
41CC-   D1

[$FDCD]
        D5                       CXP  $D59F  (See $FCCD)
41CE-   D5 D5       CMP   $D5,X
41D0-   D3          ???
41D1-   FE 00 FE    INC   $FE00,X
41D4-   FE FE FE    INC   $FEFE,X
41D7-   00          BRK
41D8-   D1 D1       CMP   ($D1),Y
41DA-   D1 D1       CMP   ($D1),Y
41DC-   D1 D1       CMP   ($D1),Y
41DE-   D1 D1       CMP   ($D1),Y
41E0-   D1 D1       CMP   ($D1),Y
41E2-   D1 D1       CMP   ($D1),Y
41E4-   D1 D1       CMP   ($D1),Y
41E6-   D1 D1       CMP   ($D1),Y
41E8-   D1 D1       CMP   ($D1),Y
41EA-   D1 D1       CMP   ($D1),Y
41EC-   D1 D1       CMP   ($D1),Y
41EE-   D1 D1       CMP   ($D1),Y
41F0-   D1 D1       CMP   ($D1),Y
41F2-   D1 D1       CMP   ($D1),Y
41F4-   D1 D1       CMP   ($D1),Y
```

```
41F6-   D1 D1       CMP   ($D1),Y
41F8-   D1 D1       CMP   ($D1),Y
41FA-   D1 D1       CMP   ($D1),Y
41FC-   D1 D1       CMP   ($D1),Y
41FE-   D1 D1       CMP   ($D1),Y

4200-   8E D9 0C    STX   $0CD9
--------------------------------------------------------------------------------
[$FE03] --- Not implemented pCodes (?)
                          pCode=#$81  ABR Absolute of Real
                          pCode=#$83  ADR Add Reals
                          pCode=#$87  DVR Divide Reals
                          pCode=#$89  FLO Float TOS-1
                          pCode=#$8A  FLT Float TOS
                          pCode=#$90  MPR Multiply Reals
                          pCode=#$92  NGR Negate Real
                          pCode=#$96  SBR Subtract Reals
                          pCode=#$99  SQR Square Real

                          pCode=#$9D  LDE Load Extended Word
                          pCode=#$A7  LAE Load Extended Addr
                          pCode=#$D1  STE Store Extended Word

                          pCode=#$D3  EFJ Equal False Jump
                          pCode=#$D4  NFJ Not Equal False Jump
                          pCode=#$D5  BPT Breakpoint
                          pCode=#$D6  XIT Exit OS

4203-   8D D8 0C    STA   $0CD8
4206-   8C D7 0C    STY   $0CD7
4209-   A0 00       LDY   #$00
420B-   B1 B6       LDA   ($B6),Y
420D-   8D D6 0C    STA   $0CD6
4210-   B1 B8       LDA   ($B8),Y
4212-   8D D5 0C    STA   $0CD5
4215-   A9 53       LDA   #$53
4217-   8D BA 0C    STA   $0CBA
421A-   A9 72       LDA   #$72
421C-   8D D2 0C    STA   $0CD2
421F-   A9 87       LDA   #$87
4221-   8D E9 0C    STA   $0CE9
4224-   A2 FF       LDX   #$FF
4226-   8E BB 0C    STX   $0CBB
4229-   8E D3 0C    STX   $0CD3
422C-   8E EA 0C    STX   $0CEA
422F-   20 A9 EB    JSR   $EBA9
[$FE32]
4232-   4C 32 FE    JMP   $FE32   Infinite Loop here!
--------------------------------------------------------------------------------

[$FE35]
4235-   00          BRK
4236-   00          BRK
4237-   00          BRK
4238-   03          ???
4239-   27          ???

[$FE3A]
423A-   01 0A       ORA   ($0A,X)
423C-   64          ???
423D-   A0 03       LDY   #$03
423F-   20 43 FE    JSR   $FE43
4242-   8A          TXA
```

```
[$FE43]
4243-   48          PHA
4244-   4A          LSR
4245-   4A          LSR
4246-   4A          LSR
4247-   4A          LSR
4248-   20 4E FE    JSR   $FE4E
424B-   68          PLA
424C-   29 0F       AND   #$0F

[$FE4E]
424E-   C9 0A       CMP   #$0A
4250-   90 02       BCC   $4254
4252-   69 06       ADC   #$06
4254-   69 30       ADC   #$30
4256-   99 AA 0C    STA   $0CAA,Y
4259-   88          DEY
425A-   60          RTS
-------------------------------------------------------
425B-   00          BRK
425C-   00          BRK
425D-   00          BRK
425E-   00          BRK
425F-   00          BRK
4260-   00          BRK
4261-   00          BRK
4262-   00          BRK
4263-   00          BRK
4264-   00          BRK
4265-   00          BRK
4266-   00          BRK
4267-   00          BRK
4268-   00          BRK
4269-   00          BRK
426A-   00          BRK
426B-   00          BRK
426C-   00          BRK
426D-   00          BRK
426E-   00          BRK
426F-   00          BRK
4270-   00          BRK
4271-   00          BRK
4272-   00          BRK
4273-   00          BRK
4274-   00          BRK
4275-   00          BRK
4276-   00          BRK
4277-   00          BRK
4278-   00          BRK
4279-   00          BRK
427A-   00          BRK
427B-   00          BRK
427C-   00          BRK
427D-   00          BRK
427E-   00          BRK
427F-   00          BRK
-------------------------------------------------------------------------------
                              FE80,UB and FEA9,UB === CSP <UB> subroutines
                                9E 06 == UNITWRITE ==
[$FE80]
4280-   84 85       STY   $85
4282-   4D 4D 7D    EOR   $7D4D
4285-   B3          ???
4286-   B0 03       BCS   $428B   FE80,6 == $DEB0 == UNITWRITE
```

```
4288-  03          ???
4289-  03          ???
428A-  8D C4 8C    STA   $8CC4
428D-  03          ???
428E-  03          ???
428F-  03          ???
4290-  03          ???
4291-  03          ???
4292-  03          ???
4293-  03          ???
4294-  03          ???
4295-  E5 EF       SBC   $EF
4297-  03          ???
4298-  03          ???
4299-  03          ???
429A-  03          ???
429B-  03          ???
429C-  03          ???
429D-  03          ???
429E-  03          ???
429F-  03          ???
42A0-  AC BE 83    LDY   $83BE
42A3-  A8          TAY
42A4-  03          ???
42A5-  A5 AB       LDA   $AB
42A7-  03          ???
42A8-  D3          ???

                        FE80,UB and FEA9,UB === CSP <UB> subroutines
[$FEA9]
42A9-  FF          ???
42AA-  DD DD DD    CMP   $DDDD,X
42AD-  D6 DE       DEC   $DE,X
42AF-  DE 00 00    DEC   $0000,X   FEA9,6 == $DEBO == UNITWRITE
42B2-  FE DC DC    INC   $DCDC,X
42B5-  DE FE FE    DEC   $FEFE,X
42B8-  FE FE FE    INC   $FEFE,X
42BB-  FE FE FE    INC   $FEFE,X
42BE-  DD DD FE    CMP   $FEDD,X
42C1-  FE FE FE    INC   $FEFE,X
42C4-  FE FE FE    INC   $FEFE,X
42C7-  FE FE DD    INC   $DDFE,X
42CA-  DD DE DE    CMP   $DEDE,X
42CD-  FE DE DE    INC   $DEDE,X
42D0-  FE DD 03    INC   $03DD,X
42D3-  FE 06 D8    INC   $D806,X
42D6-  44          ???
42D7-  D8          CLD
42D8-  6B          ???
42D9-  D8          CLD
42DA-  C6 D7       DEC   $D7
42DC-  C8          INY
42DD-  D7          ???
42DE-  01 20       ORA   ($20,X)
42E0-  08          PHP
42E1-  00          BRK
42E2-  00          BRK
42E3-  10 04       BPL   $42E9
42E5-  00          BRK
42E6-  02          ???
42E7-  00          BRK
42E8-  0D 0F 11    ORA   $110F
42EB-  13          ???
42EC-  15 17       ORA   $17,X
```

```
42EE-   19 1B 1D    ORA   $1D1B,Y
42F1-   1F          ???
42F2-   21 23       AND   ($23,X)
42F4-   25 27       AND   $27
42F6-   29 2B       AND   #$2B
42F8-   01 03       ORA   ($03,X)
42FA-   05 07       ORA   $07
42FC-   09 0B       ORA   #$0B
42FE-   0D 0F 44    ORA   $440F
4301-   69 73       ADC   #$73
4303-   6B          ???
4304-   65 74       ADC   $74
4306-   74          ???
4307-   65 20       ADC   $20
4309-   49 73       EOR   #$73
430B-   20 57 72    JSR   $7257
430E-   69 74       ADC   #$74
????????????????????????????????????????????????
                                  Table:  FE10,X
                                  X = 1..16
                                  SLDL X
[$FE10]??????????????????????????????????????????
4310-   65 2D       ADC   $2D
4312-   50 72       BVC   $4386
4314-   6F          ???
4315-   74          ???
4316-   65 63       ADC   $63
4318-   74          ???
4319-   65 64       ADC   $64
431B-   00          BRK

[$FF1C]
431C-   50 6C       BVC   $438A   Please Insert SCENARIO Diskett %c
431E-   65 61       ADC   $61
4320-   73          ???
4321-   65 20       ADC   $20
4323-   49 6E       EOR   #$6E
4325-   73          ???
4326-   65 72       ADC   $72
4328-   74          ???
4329-   20 53 43    JSR   $4353
432C-   45 4E       EOR   $4E
432E-   41 52       EOR   ($52,X)
4330-   49 4F       EOR   #$4F
4332-   20 44 69    JSR   $6944
4335-   73          ???
4336-   6B          ???
4337-   65 74       ADC   $74
4339-   74          ???
433A-   65 20       ADC   $20
433C-   25 63       AND   $63
433E-   00          BRK

[$FF3F]
433F-   50 72       BVC   $43B3   Press ~ To Continue
4341-   65 73       ADC   $73
4343-   73          ???
4344-   20 7E 20    JSR   $207E
4347-   54          ???
4348-   6F          ???
4349-   20 43 6F    JSR   $6F43
434C-   6E 74 69    ROR   $6974
434F-   6E 75 65    ROR   $6575
4352-   00          BRK
```

```
[$FF53]
4353-   41 20       EOR   ($20,X)  A Serious Problem Has Occurred
4355-   53          ???
4356-   65 72       ADC   $72
4358-   69 6F       ADC   #$6F
435A-   75 73       ADC   $73,X
435C-   20 50 72    JSR   $7250
435F-   6F          ???
4360-   62          ???
4361-   6C 65 6D    JMP   ($6D65)
4364-   20 48 61    JSR   $6148
4367-   73          ???
4368-   20 4F 63    JSR   $634F
436B-   63          ???
436C-   75 72       ADC   $72,X
436E-   72          ???
436F-   65 64       ADC   $64
4371-   00          BRK

[$FF71]
4372-   4E 6F 74    LSR   $746F   Note: %h %h %h %h %h
4375-   65 3A       ADC   $3A
4377-   20 25 68    JSR   $6825
437A-   20 25 68    JSR   $6825
437D-   20 25 68    JSR   $6825
4380-   20 25 68    JSR   $6825
4383-   20 25 68    JSR   $6825
4386-   00          BRK

[$FF87]
4387-   50 6C       BVC   $43F5  Please Restart From Diskette "A"
4389-   65 61       ADC   $61
438B-   73          ???
438C-   65 20       ADC   $20
438E-   52          ???
438F-   65 73       ADC   $73
4391-   74          ???
4392-   61 72       ADC   ($72,X)
4394-   74          ???
4395-   20 46 72    JSR   $7246
4398-   6F          ???
4399-   6D 20 44    ADC   $4420
439C-   69 73       ADC   #$73
439E-   6B          ???
439F-   65 74       ADC   $74
43A1-   74          ???
43A2-   65 20       ADC   $20
43A4-   22          ???
43A5-   41 22       EOR   ($22,X)
43A7-   00          BRK

[$FFA8]
43A8-   41 20       EOR   ($20,X)  A Problem Has Occurred While Reading
43AA-   50 72       BVC   $441E
43AC-   6F          ???
43AD-   62          ???
43AE-   6C 65 6D    JMP   ($6D65)
43B1-   20 48 61    JSR   $6148
43B4-   73          ???
43B5-   20 4F 63    JSR   $634F
43B8-   63          ???
43B9-   75 72       ADC   $72,X
43BB-   72          ???
```

```
43BC-   65 64       ADC   $64
43BE-   20 57 68    JSR   $6857
43C1-   69 6C       ADC   #$6C
43C3-   65 20       ADC   $20
43C5-   52          ???
43C6-   65 61       ADC   $61
43C8-   64          ???
43C9-   69 6E       ADC   #$6E
43CB-   67          ???
43CC-   00          BRK

[$FFCD]
43CD-   44          ???         The Diskette In Drive %h
43CE-   69 73       ADC   #$73
43D0-   6B          ???
43D1-   65 74       ADC   $74
43D3-   74          ???
43D4-   65 20       ADC   $20
43D6-   22          ???
43D7-   25 63       AND   $63
43D9-   22          ???
43DA-   00          BRK

[$FFDB]
43DB-   54          ???
43DC-   68          PLA
43DD-   65 20       ADC   $20
43DF-   44          ???
43E0-   69 73       ADC   #$73
43E2-   6B          ???
43E3-   65 74       ADC   $74
43E5-   74          ???
43E6-   65 20       ADC   $20
43E8-   49 6E       EOR   #$6E
43EA-   20 44 72    JSR   $7244
43ED-   69 76       ADC   #$76
43EF-   65 20       ADC   $20
43F1-   25 68       AND   $68
43F3-   00          BRK

[$FFF4]
43F4-   4C 22 DF    JMP   $DF22   Jump after I/O error to Dir during boot

[$FFF7]
43F7-   4C FE E7    JMP   $E7FE   Jump to read block of sectors (Dir(?))
43FA-   BB          ???
43FB-   E7          ???
43FC-   BB          ???
43FD-   E7          ???
43FE-   BB          ???
43FF-   E7          ???
```

Updated June 8, 2015

```
INTERP last 4 sectors.  Loaded to $BC00..BFFF

 $D000..FFFF contains first part of INTERP
 $BC00.$BFFF contains last 2 blocks of INTERP

 JMP from $D021 during boot.

 $FB = Slot# * 16 (e.g., $60)
 $F3 = $Cx (x = Boot Slot, e.g., $C6)

[$BC00]
0900-   A2 FF       LDX   #$FF
0902-   9A          TXS
0903-   58          CLI
0904-   D8          CLD
0905-   20 A4 BC    JSR   $BCA4  Init lots of memory
                                 Transfer Pascal "heart beat" code to zpage at $90
                                 Display "WIZARDRY FOR THE APPLE II"
0908-   20 F9 BC    JSR   $BCF9  Read SYSTEM.PASCAL Segment Table into Active Seg table(s)
090B-   20 EF BD    JSR   $BDEF  Check for Disk Devices
                                 (update tables at $FC63 $FC71, 4=boot Slot 6 Drive 1)
[$BC0E]
090E-   4C 9D 00    JMP   $009D   (Code at $BC74 transferred to $90.)

                                 [ $BC3A points to $BC08.
                                   Use this to get to "SYSTEM.PASCAL" string below.]
-------------------------------------------------------------
0911-   0D 53 59    ORA   $5953    $0D "SYSTEM.PASCAL"
0914-   53          ???
0915-   54          ???
0916-   45 4D       EOR   $4D
0918-   2E 50 41    ROL   $4150
091B-   53          ???
091C-   43          ???
091D-   41 4C       EOR   ($4C,X)
091F-   00          BRK
-------------------------------------------------------------
0920-   57          ???            "WIZARDRY FOR THE APPLE II"
0921-   49 5A       EOR   #$5A
0923-   41 52       EOR   ($52,X)
0925-   44          ???
0926-   52          ???
0927-   59 20 46    EOR   $4620,Y
092A-   4F          ???
092B-   52          ???
092C-   20 54 48    JSR   $4854
092F-   45 20       EOR   $20
0931-   41 50       EOR   ($50,X)
0933-   50 4C       BVC   $0981
0935-   45 20       EOR   $20
0937-   49 49       EOR   #$49
0939-   00          BRK
-------------------------------------------------------------
093A-   0B          ???            Pointer $BC0B to get to "SYSTEM.PASCAL"
093B-   BC                           string while checking DIRECTORY
-------------------------------------------------------------
```

```
       45 52                    "ERROR: SYSTEM.PASCAL NOT FOUND"
093E-  52          ???
093F-  4F          ???
0940-  52          ???
0941-  3A          ???
0942-  20 53 59    JSR   $5953
0945-  53          ???
0946-  54          ???
0947-  45 4D       EOR   $4D
0949-  2E 50 41    ROL   $4150
094C-  53          ???
094D-  43          ???
094E-  41 4C       EOR   ($4C,X)
0950-  20 4E 4F    JSR   $4F4E
0953-  54          ???
0954-  20 46 4F    JSR   $4F46
0957-  55 4E       EOR   $4E,X
0959-  44          ???
095A-  00          BRK
-------------------------------------------------------------
095B-  54          ???           "TYPE <RETURN> TO RESTART"
095C-  59 50 45    EOR   $4550,Y
095F-  20 3C 52    JSR   $523C
0962-  45 54       EOR   $54
0964-  55 52       EOR   $52,X
0966-  4E 3E 20    LSR   $203E
0969-  54          ???
096A-  4F          ???
096B-  20 52 45    JSR   $4552
096E-  53          ???
096F-  54          ???
0970-  41 52       EOR   ($52,X)
0972-  54          ???
0973-  00          BRK
-------------------------------------------------------------
  --------------------------------
  --- Heart of the Beast       ---
  --- pCode interpretted here! ---
  --------------------------------
                                   See $BCEA, 48 bytes transfered to z-page
                                   To:  $90..BF
[$BC74]
[90]
0974-  A2 00       LDX   #$00

0976-  A9 00       LDA   #$00
0978-  48          PHA
0979-  8A          TXA
097A-  48          PHA
097B-  E6 9E       INC   $9E        Self Modify some code!
097D-  D0 02       BNE   $0981
097F-  E6 9F       INC   $9F        Self Modify some code!

[9D]                               To $9D at $BC0E
0981-  AE FB BF    LDX   $BFFB     This code self modified!
0984-  10 F0       BPL   $0976
```

```
0986-   BD 00 FC    LDA   $FC00,X
0989-   85 AD       STA   $AD
098B-   BD 00 FD    LDA   $FD00,X
098E-   85 AE       STA   $AE

[AC]
0990-   4C 00 FE    JMP   $xxxx    Self Modified code!
0993-   4C 00 00    JMP   $0000

0996-   F0 BF
[B4]
0998-   00 10                      B4.B5 is buffer to Directory (?)
        91 BC
        91 BC
099E-   91 BB
09A0-   F0 BF

[BE]
09A2-   EE BF                    $BE.BF === KP stack ptr
-------------------------------------------------------------

[$BCA4]                          To here from:  $BC05

                    LDA   #$04      $C8.C9 --> $0400
09A6-   85 C9       STA   $C9
09A8-   A0 00       LDY   #$00
09AA-   84 C8       STY   $C8

09AC-   A2 08       LDX   #$08      Clear $0400..0BFF to " "
09AE-   A9 A0       LDA   #$A0        Primary and Secondary Text Screens
09B0-   91 C8       STA   ($C8),Y
09B2-   C8          INY
09B3-   D0 FB       BNE   $09B0
09B5-   E6 C9       INC   $C9
09B7-   CA          DEX
09B8-   D0 F6       BNE   $09B0

09BA-   BD 20 BC    LDA   $BC20,X   Move "WIZARDRY FOR THE APPLE II"
09BD-   F0 08       BEQ   $09C7
09BF-   09 80       ORA   #$80
09C1-   9D 87 04    STA   $0487,X   To:  text screen at line 1 (0 based)
09C4-   E8          INX
09C5-   D0 F3       BNE   $09BA

09C7-   A9 00       LDA   #$00      Clear rest of RAM to 0
09C9-   A2 AF       LDX   #$AF      $0C00..$BBFF
09CB-   91 C8       STA   ($C8),Y
09CD-   C8          INY
09CE-   D0 FB       BNE   $09CB
09D0-   E6 C9       INC   $C9
09D2-   CA          DEX
09D3-   D0 F6       BNE   $09CB

09D5-   A2 F2       LDX   #$F2      Clear Z-page $01..$F2
09D7-   95 00       STA   $00,X
09D9-   CA          DEX
09DA-   D0 FB       BNE   $09D7
```

```
09DC-   A0 07       LDY   #$07      Clear some "holes" in screen pages to zeros (0).
09DE-   99 78 04    STA   $0478,Y     7 of these
09E1-   99 F8 04    STA   $04F8,Y     7 of these
09E4-   88          DEY
09E5-   D0 F7       BNE   $09DE

09E7-   8D F8 04    STA   $04F8       + 1 more
                                      (Leave $0478 alone!   (Current Disk Track(?))

[$BCEA]
09EA-   A2 2F       LDX   #$2F      Transfer $BC74..BCA3
09EC-   BD 74 BC    LDA   $BC74,X        To:   $90..BF
09EF-   95 90       STA   $90,X
09F1-   CA          DEX
09F2-   10 F8       BPL   $09EC

09F4-   A9 01       LDA   #$01      $E8 := 1
09F6-   85 E8       STA   $E8              UNITWRITE( 13, -3, 25, 0) also sets to 1
09F8-   60          RTS
-------------------------------------------------------------------------------------

  Read Directory & Find "SYSTEM.PASCAL"

[$BCF9]                         From $BC00 after initializing memory, etc.
09F9-   A5 B4       LDA   $B4    $30 := ($B4)
09FB-   85 30       STA   $30
09FD-   A5 B5       LDA   $B5    $31 := ($B5)
09FF-   85 31       STA   $31


                                Read PASCAL Directory to $B4.B5-->  ($1000)
                                  2D := 4  Device
                                  2E := 0
                                  2F := 0
                                  30.31 :=   I/O Buffer  (Initially $1000)
                                  32.33 := Length (#$800 = 8 sectors)
                                  34 := 2  34.35 = block addr for I/O
                                  35 := 0
                                  X  := 0
[$BD01]
0901-   A2 00       LDX   #00
0903-   86 2E       STX   $2E
0905-   86 2F       STX   $2F
0907-   A9 08       LDA   #$08
0909-   86 32       STX   $32
090B-   85 33       STA   $33
090D-   A9 02       LDA   #$02
090F-   85 34       STA   $34
0911-   86 35       STX   $35
0913-   A9 04       LDA   #$04
0915-   85 2D       STA   $2D
0917-   20 F7 FF    JSR   $FFF7  JMP $E7FE (Read block of sectors;  The DIR)

091A-   8A          TXA
091B-   D0 32       BNE   $094F  Error, Give bad news
```

```
091D-   A5 B4       LDA   $B4    $C8.C9 --> Directory
091F-   85 C8       STA   $C8
0921-   A5 B5       LDA   $B5
0923-   85 C9       STA   $C9
0925-   AD 3A BC    LDA   $BC3A  $CA.CB --> 5 bytes before "SYSTEM.PASCAL"
0928-   85 CA       STA   $CA
092A-   AD 3B BC    LDA   $BC3B
092D-   85 CB       STA   $CB
092F-   A0 10       LDY   #$10
0931-   B1 C8       LDA   ($C8),Y  # of files in Directory
0933-   AA          TAX
0934-   10 0B       BPL   $0941

0936-   A0 05       LDY   #$05
0938-   C8          INY
0939-   B1 CA       LDA   ($CA),Y
093B-   F0 45       BEQ   $0982    Found "SYSTEM.PASCAL" in DIR, $BD82
093D-   D1 C8       CMP   ($C8),Y
093F-   F0 F7       BEQ   $0938

0941-   18          CLC           Advance $C8.C9 to next file in DIR
0942-   A5 C8       LDA   $C8
0944-   69 1A       ADC   #$1A
0946-   85 C8       STA   $C8
0948-   90 02       BCC   $094C
094A-   E6 C9       INC   $C9
094C-   CA          DEX
094D-   10 E7       BPL   $0936    Note error fallthrough to $BD4F

-----------------------------------------------------------------------
[$BD4F]
094F-   20 F4 FF    JSR   $FFF4   Clear Primary Text Screen to Blanks (see $DF22)
0952-   AD 51 C0    LDA   $C051   All Text
0955-   A2 00       LDX   #$00

0957-   BD 3C BC    LDA   $BC3C,X   "ERROR: SYSTEM.PASCAL NOT FOUND"
095A-   F0 08       BEQ   $0964
095C-   09 80       ORA   #$80
095E-   9D 05 06    STA   $0605,X   line 4 (0 based) of screen
0961-   E8          INX
0962-   D0 F3       BNE   $0957

0964-   A0 00       LDY   #$00
0966-   B9 5B BC    LDA   $BC5B,Y   "TYPE <RETURN> TO RESTART"
0969-   F0 08       BEQ   $0973
096B-   09 80       ORA   #$80
096D-   99 88 06    STA   $0688,Y   line 5 (0 base) of screen
0970-   C8          INY
0971-   D0 F3       BNE   $0966

0973-   AD 00 C0    LDA   $C000     Keyboard strobe
0976-   10 FB       BPL   $0973     Wait until a key is pressed
0978-   8D 10 C0    STA   $C010
097B-   C9 8D       CMP   #$8D      <CR>?
097D-   D0 F4       BNE   $0973     No, try again

097F-   4C 00 D0    JMP   $D000     Yes, try restarting INTERP
```

```
--------------------------------------------------------------------

    Read SYSTEM.PASCAL Segment Dictionary into memory
    Update "Active Segment" table(s) at $0C19 and $0C3A.

[$BD82]                          After finding "SYSTEM.PASCAL" Dir entry

                                 $C8.C9 -->SYSTEM.PASCAL Dir Entry

                               Set up to read
                                 2D := 4 (Device)   ASSUMES Slot#6, Drive #1
                                 30.31 := $B4.B5 buffer (#$1000)
                                 32 := 0
                                 33 := 2 sectors (1 block)
                                 34.35 := First Block System.Pascal  34.35 = BLOCK addr for I/O
                                 C8.C9 := $B4.B5 buffer (#$1000)
                                 F8.F9 := First Block System.Pascal
                                          (Segment Dictionary)
                                 X := 0
[$BD82]
0982-   A9 02       LDA   #$02
0984-   85 33       STA   $33
0986-   A0 00       LDY   #$00
0988-   84 32       STY   $32
098A-   B1 C8       LDA   ($C8),Y
098C-   85 34       STA   $34
098E-   85 F8       STA   $F8
0990-   C8          INY
0991-   B1 C8       LDA   ($C8),Y
0993-   85 35       STA   $35
0995-   85 F9       STA   $F9
0997-   A5 B4       LDA   $B4
0999-   85 30       STA   $30
099B-   85 C8       STA   $C8
099D-   A5 B5       LDA   $B5
099F-   85 31       STA   $31
09A1-   85 C9       STA   $C9
09A3-   A9 04       LDA   #$04
09A5-   85 2D       STA   $2D
09A7-   A2 00       LDX   #$00
09A9-   20 F7 FF    JSR   $FFF7   JMP $E7FE to read block of sectors

09AC-   8A          TXA
09AD-   D0 A0       BNE   $094F   Error, $094F to give bad news

[$BDAF]
09AF-   A5 C8       LDA   $C8     $C8.C9 == I/O buffer
09B1-   A6 C9       LDX   $C9
09B3-   E8          INX
09B4-   85 CA       STA   $CA     $CA.CB == I/O buffer + $256
09B6-   86 CB       STX   $CB               (Next sector of Seg Dictionary)

09B8-   A2 0F       LDX   #$0F    16 Segments
09BA-   A0 3F       LDY   #$3F    Start with segment 15 and work back to 0
09BC-   84 04       STY   $04     ...and at the end of the 4 byte entry (Addr, Length)

09BE-   A4 04       LDY   $04     $0C19 is Active Segment Table (?) ($0C19(?))
```

```
09C0-   B1 C8       LDA   ($C8),Y
09C2-   9D 6A 0C    STA   $0C6A,X  Length (Low)
09C5-   88          DEY
09C6-   B1 C8       LDA   ($C8),Y
09C8-   9D 5A 0C    STA   $0C5A,X  Length (High)
09CB-   88          DEY
09CC-   88          DEY
09CD-   B1 C8       LDA   ($C8),Y
09CF-   18          CLC
09D0-   65 F8       ADC   $F8
09D2-   9D 3A 0C    STA   $0C3A,X  Addr (Low)
09D5-   C8          INY
09D6-   B1 C8       LDA   ($C8),Y
09D8-   65 F9       ADC   $F9
09DA-   9D 4A 0C    STA   $0C4A,X  Addr (High)

09DD-   88          DEY            Point to next Segment entry
09DE-   88          DEY
09DF-   84 04       STY   $04

[$BDE1]
09E1-   8A          TXA
09E2-   0A          ASL
09E3-   A8          TAY
09E4-   B1 CA       LDA   ($CA),Y  Segment Index
09E6-   A8          TAY
09E7-   8A          TXA
09E8-   99 19 0C    STA   $0C19,Y  $0C19,Y holds Segment Index
09EB-   CA          DEX
09EC-   10 D0       BPL   $09BE    Get information for all 16 Segments
09EE-   60          RTS
-------------------------------------------------------------------------------
                                   From $BC0B,
                                   After reading SYSTEM.PASCAL Seg dictionary into Active Seg Table
                                     $F3 = Boot Slot ($Cx), e.g. $C6
[BDEF]
09EF-   AC 83 C0    LDY   $C083    Read/Write RAM, Bank 2
09F2-   AC 83 C0    LDY   $C083    Read/Write RAM, Bank 2 (need 2 of these)
09F5-   A4 F3       LDY   $F3      Y := $F3 (#$C6) used in $BE1C
09F7-   A2 04       LDX   #$04     $07 := #$4 == Device
09F9-   86 07       STX   $07
09FB-   A2 0E       LDX   #$0E     $54 := #$E
09FD-   86 54       STX   $54
09FF-   A9 00       LDA   #$00     $52 := 0

[$BE00]
0901-   85 52       STA   $52     Identify Disk Peripherals
0903-   20 1C BE    JSR   $BE1C   Check boot slot (e.g., $C600)

0906-   A0 C7       LDY   #$C7    Check other slots

*******************************************
*****  Tracing this reveals a huge bug *****
*******************************************

0908-   C4 F3       CPY   $F3     Boot slot?  (e.g., #$C6)?
090A-   F0 03       BEQ   $090F     Yes, already checked
```

```
090C-   20 1C BE    JSR   $BE1C            ***** Y changes *****
090F-   88          DEY
0910-   C0 C1       CPY   #$C1      #$C1
0912-   B0 F4       BCS   $0908
0914-   AD 80 C0    LDA   $C080
0917-   60          RTS
---------------------------------
0918-   00          BRK             See $BEA3
0919-   80          ???
091A-   C0 40       CPY   #$40
---------------------------------
[BE1C]
                                    Check 1 slot for disk drives

091C-   84 53       STY   $53      ...First Time, $52.53 := #$C600
091E-   20 58 BF    JSR   $BF58    A := sum of 256 bytes at $52.53
0921-   85 50       STA   $50      X > 0 means sum > 255
0923-   65 F6       ADC   $F6
0925-   85 F6       STA   $F6        F6.F7 === Random Number (?)
0927-   8A          TXA
0928-   65 F7       ADC   $F7
092A-   85 F7       STA   $F7
092C-   E0 00       CPX   #$00     Anything in slot?
092E-   F0 2B       BEQ   $095B      No, (e.g., #$C700 BEQ)

0930-   86 51       STX   $51      $50.51 := sum at $52.53 -->
0932-   20 58 BF    JSR   $BF58    A := Sum again
0935-   C5 50       CMP   $50      See if we get same results as above
0937-   D0 22       BNE   $095B      to verify we have something
0939-   E4 51       CPX   $51        in the slot.
093B-   D0 1E       BNE   $095B
                                    Seems we have something in this slot
[$BE3D]
093D-   A0 01       LDY   #$01     Check for Disk Device
093F-   B1 52       LDA   ($52),Y
0941-   C9 20       CMP   #$20     $Cs01 = #$20?
0943-   D0 16       BNE   $095B      No, $BE5B
0945-   A0 03       LDY   #$03
0947-   B1 52       LDA   ($52),Y $Cs03 = #$00?
0949-   D0 10       BNE   $095B      No, $BE5B
094B-   A0 05       LDY   #$05
094D-   B1 52       LDA   ($52),Y $Cs05 = #$03?
094F-   C9 03       CMP   #$03
0951-   D0 08       BNE   $095B      No, $BE5B
0953-   A0 FF       LDY   #$FF
0955-   B1 52       LDA   ($52),Y
0957-   C9 FF       CMP   #$FF     $CsFF = #$FF?
0959-   D0 01       BNE   $095C      No, $BE5C    Disk Controller
095B-   60          RTS            Not a Disk Controller, RETURN

 --- To here if Disk Controller ---

[$BE5C]
095C-   AA          TAX            $CsFF byte (=0 for disk controller)
095D-   F0 6E       BEQ   $09CD       Disk II?  Yes, $BECD...First Time

[$BE5F]                             ---------------------------------
```

```
095F-  88          DEY             Can any of this code
0960-  B1 52       LDA   ($52),Y   ever execute with AppleWin?
0962-  4A          LSR
0963-  90 F6       BCC   $095B
0965-  4A          LSR
0966-  90 F3       BCC   $095B
0968-  4A          LSR
0969-  4A          LSR
096A-  29 03       AND   #$03
096C-  85 08       STA   $08
096E-  A5 53       LDA   $53
0970-  0A          ASL
0971-  0A          ASL
0972-  0A          ASL
0973-  0A          ASL
0974-  85 10       STA   $10
0976-  A0 FB       LDY   #$FB
0978-  B1 52       LDA   ($52),Y
097A-  4A          LSR
097B-  08          PHP
097C-  90 1C       BCC   $099A
097E-  A6 54       LDX   $54
0980-  E0 11       CPX   #$11
0982-  B0 D7       BCS   $095B
0984-  A4 FC       LDY   $FC
0986-  C0 40       CPY   #$40
0988-  B0 D1       BCS   $095B
098A-  98          TYA
098B-  D0 08       BNE   $0995
098D-  86 FF       STX   $FF
098F-  8E 24 FC    STX   $FC24
0992-  C8          INY
0993-  E6 FC       INC   $FC
0995-  8A          TXA
0996-  99 24 FC    STA   $FC24,Y
0999-  2C A6 07    BIT   $07A6     $099A::LDX $07
099C-  A0 00       LDY   #$00

099E-  A5 53       LDA   $53
09A0-  9D 63 FC    STA   $FC63,X   $FC63,X
09A3-  B9 18 BE    LDA   $BE18,Y   $BE18,Y
09A6-  18          CLC
09A7-  65 10       ADC   $10
09A9-  9D 71 FC    STA   $FC71,X   $FC71,X
09AC-  E0 05       CPX   #$05
09AE-  D0 02       BNE   $09B2
09B0-  A2 08       LDX   #$08
09B2-  E8          INX
09B3-  C8          INY
09B4-  C4 08       CPY   $08
09B6-  90 E6       BCC   $099E
09B8-  28          PLP
09B9-  B0 03       BCS   $09BE
09BB-  86 07       STX   $07
09BD-  60          RTS
09BE-  86 54       STX   $54
09C0-  98          TYA
```

```
09C1-   18          CLC
09C2-   65 FC       ADC   $FC
09C4-   C9 40       CMP   #$40
09C6-   90 02       BCC   $09CA
09C8-   A9 40       LDA   #$40
09CA-   85 FC       STA   $FC
09CC-   60          RTS           See note at $BE5F
        ------------------------------------------------------------------
                                  Process the DISK II controller

[$BECD]
09CD-   A5 53       LDA   $53     Slot address HIGH part ($C6 boot slot)
09CF-   0A          ASL
09D0-   0A          ASL
09D1-   0A          ASL
09D2-   0A          ASL
09D3-   85 0C       STA   $0C     $0C := Slot * #$60 (#$60 for slot 6, Drive 1)

                              ====Begin Loop to process up to 2 drives====
                                 $0C := $60, $61, $70, $71, $50, $51, ...
[$BED5]
09D5-   A6 FB       LDX   $FB       Wait for previous drive to stop spinning
09D7-   BD 8E C0    LDA   $C08E,X   C08E  Prepare Latch for Input
09DA-   BD 8C C0    LDA   $C08C,X   C08C  Strobe Data Latch
09DD-   A0 08       LDY   #$08
09DF-   DD 8C C0    CMP   $C08C,X   C08C  Strobe Data Latch
09E2-   D0 F6       BNE   $09DA     Drive is still spinning, $09DA
09E4-   88          DEY             Try to get 8 consecutive identical reads
09E5-   D0 F8       BNE   $09DF
09E7-   A0 07       LDY   #$07      Got them.   Drive should be stopped now.
09E9-   20 67 BF    JSR   $BF67     BF67 === Wait a bit
09EC-   88          DEY
09ED-   D0 FA       BNE   $09E9     Wait a bit longer

                                  Previous processed drive
                                    has stopped spinning

09EF-   A5 53       LDA   $53       Slot we are currently processing
09F1-   0A          ASL
09F2-   0A          ASL
09F3-   0A          ASL
09F4-   0A          ASL
09F5-   85 FB       STA   $FB       $FB := #$60, #$70, etc.
09F7-   AA          TAX
09F8-   BD 89 C0    LDA   $C089,X   MotorOn
09FB-   A6 FB       LDX   $FB
09FD-   BD 8E C0    LDA   $C08E,X   C08E  Prepare Latch for Input

[$BF00]
0900-   BD 8C C0    LDA   $C08C,X   Strobe Data Latch
0903-   A4 0C       LDY   $0C       Y := $0C  (e.g., #$60, #$61, #$50, #$51, etc.)
0905-   B9 8A C0    LDA   $C08A,Y   Engage Drive #1 or #2
0908-   A0 0D       LDY   #$0D      09 := #$0D
090A-   84 09       STY   $09
090C-   A0 30       LDY   #$30      0A := #$30
090E-   84 0A       STY   $0A
```

```
0910-   A0 09       LDY   #$09
0912-   BD 8C C0    LDA   $C08C,X   Strobe Data Latch
0915-   88          DEY
0916-   F0 24       BEQ   $093C [$BF3C] Eventually took this jump
0918-   48          PHA                 when 9 reads in a row
0919-   68          PLA                 differ from previous one
091A-   48          PHA
091B-   68          PLA
091C-   DD 8C C0    CMP   $C08C,X   Strobe Data Latch
091F-   D0 F1       BNE   $0912     Data changing?  Yes, $0912
0921-   C6 0A       DEC   $0A       ...First Time EQ
0923-   D0 EB       BNE   $0910
0925-   C6 09       DEC   $09
0927-   D0 E7       BNE   $0910     Retry a bunch of times.
                                    If still the same values, then fall through
                                    as though no drive there.

0929-   BD 88 C0    LDA   $C088,X   MotorOff

092C-   A4 0C       LDY   $0C       Processing drive #1?
092E-   C4 FB       CPY   $FB
0930-   D0 05       BNE   $0937       No, $0937
0932-   E6 0C       INC   $0C         Yes, try drive #2
0934-   4C D5 BE    JMP   $BED5

0937-   A0 01       LDY   #$01      To here after doing 2 drives
0939-   84 FA       STY   $FA
093B-   60          RTS
---------------------------------
                                  Found a diskette in a drive
[$BF3C]
093C-   BD 88 C0    LDA   $C088,X  MotorOff   ...First Time took this path
093F-   A6 07       LDX   $07      X := Device (4 = boot)
0941-   A5 FB       LDA   $FB      ******************************
0943-   9D 63 FC    STA   $FC63,X  *                            *
0946-   A5 0C       LDA   $0C      *                            *
0948-   29 01       AND   #$01     *                            *
094A-   9D 71 FC    STA   $FC71,X  *                            *
094D-   E0 05       CPX   #$05     ******************************
094F-   D0 02       BNE   $0953
0951-   A2 08       LDX   #$08     After 5, then do 9 (slot #4, Drive 1)
0953-   E8          INX            After 4, do 5
0954-   86 07       STX   $07
0956-   10 D4       BPL   $092C    4, 5, 9, 10, 11, 12
---------------------------------
[$BF58]
                                  Sum 256 bytes pointed to by:
                                    $52.53-->  (such as $C600)
0958-   18          CLC
0959-   A9 00       LDA   #$00     A=X=Y= 0
095B-   AA          TAX
095C-   A8          TAY
095D-   71 52       ADC   ($52),Y  A = Sum 256 bytes
095F-   90 02       BCC   $0963
0961-   E8          INX            X indicates sum is > 255
0962-   18          CLC
```

```
0963-   C8          INY
0964-   D0 F7       BNE   $095D
0966-   60          RTS
---------------------------------
[$BF67]
0967-   A2 11       LDX   #$11      Wait a bit
0969-   CA          DEX
096A-   D0 FD       BNE   $0969
096C-   E6 49       INC   $49
096E-   D0 02       BNE   $0972
0970-   E6 48       INC   $48
0972-   38          SEC
0973-   E9 01       SBC   #$01
0975-   D0 F0       BNE   $0967
0977-   60          RTS
---------------------------------
0978-   00          BRK
0979-   00          BRK
097A-   00          BRK
097B-   00          BRK
097C-   00          BRK
097D-   00          BRK
097E-   00          BRK
097F-   00          BRK
0980-   00          BRK
0981-   00          BRK
0982-   00          BRK
0983-   00          BRK
0984-   00          BRK
0985-   00          BRK
0986-   00          BRK
0987-   00          BRK
0988-   00          BRK
0989-   00          BRK
098A-   00          BRK
098B-   00          BRK
098C-   00          BRK
098D-   00          BRK
098E-   00          BRK
098F-   00          BRK
0990-   00          BRK
0991-   00          BRK
0992-   00          BRK
0993-   00          BRK
0994-   00          BRK
0995-   00          BRK
0996-   00          BRK
0997-   00          BRK
0998-   00          BRK
0999-   00          BRK
099A-   00          BRK
099B-   00          BRK
099C-   00          BRK
099D-   00          BRK
099E-   00          BRK
099F-   00          BRK
09A0-   00          BRK
```

	09A1- 00	BRK					
	09A2- 00	BRK					
	09A3- 00	BRK					
	09A4- 00	BRK					
	09A5- 00	BRK					
	09A6- 00	BRK					
	09A7- 00	BRK					
	09A8- 00	BRK					
	09A9- 00	BRK					
	09AA- 00	BRK					
	09AB- 00	BRK					
	09AC- 00	BRK					
	09AD- 00	BRK					
	09AE- 00	BRK					
	09AF- 00	BRK					
	09B0- 00	BRK					
	09B1- 00	BRK					
	09B2- 00	BRK					
	09B3- 00	BRK					
	09B4- 00	BRK					
	09B5- 00	BRK					
	09B6- 00	BRK					
	09B7- 00	BRK					
	09B8- 00	BRK					
	09B9- 00	BRK					
	09BA- 00	BRK					
	09BB- 00	BRK					
	09BC- 00	BRK					
	09BD- 00	BRK					
	09BE- 00	BRK					
	09BF- 00	BRK					
	09C0- 00	BRK					
	09C1- 00	BRK					
	09C2- 00	BRK					
	09C3- 00	BRK					
	09C4- 00	BRK					
	09C5- 00	BRK					
	09C6- 00	BRK					
	09C7- 00	BRK					
	09C8- 00	BRK					
	09C9- 00	BRK					
	09CA- 00	BRK					
	09CB- 00	BRK					
	09CC- 00	BRK					
	09CD- 00	BRK					
	09CE- 00	BRK					
	09CF- 00	BRK					
	09D0- 00	BRK					
	09D1- 00	BRK					
	09D2- 00	BRK					
	09D3- 00	BRK					
	09D4- 00	BRK					
	09D5- 00	BRK					
	09D6- 00	BRK					
	09D7- 00	BRK					
	09D8- 00	BRK					
	09D9- 00	BRK					

```
09DA-   00          BRK
09DB-   00          BRK
09DC-   00          BRK
09DD-   00          BRK
09DE-   00          BRK
09DF-   00          BRK
09E0-   00          BRK
09E1-   00          BRK
09E2-   00          BRK
09E3-   00          BRK
09E4-   00          BRK
09E5-   00          BRK
09E6-   00          BRK
09E7-   00          BRK
09E8-   00          BRK
09E9-   00          BRK
09EA-   00          BRK
09EB-   00          BRK
09EC-   00          BRK
09ED-   00          BRK

[$BFEE]
09EE-   EE BF                       $BFEE points to itself (!)

09F0-   F0 BF                       $BFF0 points to itself (!)
        F0 BF
        91 BC
        91 BC
        FC BF
09FA-   FF

[$BFFB]
09FB-   00          BRK             $BFFB first p-code executed by heartbeat
09FC-   00          BRK
09FD-   CD 01 01    CMP   $0101     $CD = CALL EXTERNAL PROCEDURE 1,1
```

Updated June 8, 2015 Note: "(*$I WIZ..." messes up line numbering in Pascal.

```
     2    1     1:D     1 (*$L PRINTER: *)
     3    1     1:D     1 (* "$S++" OPTION BEFORE "$L" *)
     4    1     1:D     1 (*$R-*)
     5    1     1:D     1 (*$I-*)
     6    1     1:D     1 (*$V-*)
     7    1     1:D     1
     8    1     1:D     1 (*
     9    1     1:D     1    WIZARDRY IV, SYSTEM.PASCAL
    10    1     1:D     1    REVERSE ENGINEERED BY:  THOMAS WILLIAM EWERS, 2014-2015
    11    1     1:D     1 *)
    12    1     1:D     1
    13    1     1:D     1 PROGRAM WIZARDRY;
    14    1     1:D     3
    15    1     1:D     3
    16    1     1:D     3   CONST
    17    1     1:D     3         DRIVE1 = 4;
    18    1     1:D     3         CRETURN = 13;
    19    1     1:D     3         ESCAPE  = 27;
    20    1     1:D     3
    21    1     1:D     3   TYPE
    22    1     1:D     3        PINT = ^INTEGER;
    23    1     1:D     3        TLONGSTR = STRING[ 255];
    24    1     1:D     3
    25    1     1:D     3        TCHARSTR = STRING[ 1];
    26    1     1:D     3
    27    1     1:D     3        TBUFFER = PACKED ARRAY[ 0..1023] OF 0..255;
    28    1     1:D     3
    29    1     1:D     3        TWIZLONG = RECORD
    30    1     1:D     3            LOW  : INTEGER;
    31    1     1:D     3            MID  : INTEGER;
    32    1     1:D     3            HIGH : INTEGER;
    33    1     1:D     3          END;
    34    1     1:D     3
    35    1     1:D     3        TRACE = (NORACE, HUMAN, ELF, DWARF, GNOME, HOBBIT);
    36    1     1:D     3
    37    1     1:D     3        TCLASS = (FIGHTER, MAGE, PRIEST, THIEF,
    38    1     1:D     3                  BISHOP, SAMURAI, LORD, NINJA);
    39    1     1:D     3
    40    1     1:D     3        TALIGN = (UNALIGN, GOOD, NEUTRAL, EVIL);
    41    1     1:D     3
    42    1     1:D     3        TSTATUS = (OK, AFRAID, ASLEEP, PLYZE,
    43    1     1:D     3                   STONED, DEAD, ASHES, LOST);
    44    1     1:D     3
    45    1     1:D     3        TATTRIB = (STRENGTH, IQ, PIETY, VITALITY, AGILITY, LUCK);
    46    1     1:D     3
    47    1     1:D     3        TSPELL7G = ARRAY[ 1..7] OF INTEGER;
    48    1     1:D     3
    49    1     1:D     3        THPREC = RECORD
    50    1     1:D     3            LEVEL   : INTEGER;
    51    1     1:D     3            HPFAC   : INTEGER;
    52    1     1:D     3            HPMINAD : INTEGER;
    53    1     1:D     3          END;
    54    1     1:D     3
    55    1     1:D     3        TCHAR = RECORD
    56    1     1:D     3            NAME     : STRING[ 15];             (* ^.0 *)
    57    1     1:D     3            PASSWORD : STRING[ 15];
```

```
    58  1   1:D   3            INMAZE   : BOOLEAN;
    59  1   1:D   3            RACE     : TRACE;
    60  1   1:D   3            CLASS    : TCLASS;
    61  1   1:D   3            AGE      : INTEGER;
    62  1   1:D   3            STATUS   : TSTATUS;
    63  1   1:D   3            ALIGN    : TALIGN;
    64  1   1:D   3            ATTRIB   : PACKED ARRAY[ STRENGTH..LUCK] OF 0..18;
    65  1   1:D   3            LUCKSKIL : PACKED ARRAY[ 0..4] OF 0..31;
    66  1   1:D   3            GOLD     : TWIZLONG;                (* ^.1A *)
    67  1   1:D   3            POSS     : RECORD                   (* ^.1D *)
    68  1   1:D   3                POSSCNT : INTEGER;
    69  1   1:D   3                POSSESS : ARRAY[ 1..8] OF RECORD
    70  1   1:D   3                    EQUIPED : BOOLEAN;
    71  1   1:D   3                    CURSED  : BOOLEAN;
    72  1   1:D   3                    IDENTIF : BOOLEAN;
    73  1   1:D   3                    EQINDEX : INTEGER;
    74  1   1:D   3                  END;
    75  1   1:D   3              END;
    76  1   1:D   3            EXP      : TWIZLONG;                (* ^.3E *)
    77  1   1:D   3            MAXLEVAC : INTEGER;
    78  1   1:D   3            CHARLEV  : INTEGER;
    79  1   1:D   3            HPLEFT   : INTEGER;                (* ^.43 *)
    80  1   1:D   3            HPMAX    : INTEGER;
    81  1   1:D   3            SPELLSKN : PACKED ARRAY[ 0..49] OF BOOLEAN;
    82  1   1:D   3            MAGESP   : TSPELL7G;               (* ^.49 *)
    83  1   1:D   3            PRIESTSP : TSPELL7G;               (* ???  *)
    84  1   1:D   3            HPCALCMD : INTEGER;
    85  1   1:D   3            ARMORCL  : INTEGER;
    86  1   1:D   3            HEALPTS  : INTEGER;
    87  1   1:D   3            CRITHITM : BOOLEAN;
    88  1   1:D   3            SWINGCNT : INTEGER;                (* ^.5B *)
    89  1   1:D   3            HPDAMRC  : THPREC;
    90  1   1:D   3            WEPVSTY2 : PACKED ARRAY[ 0..1, 0..13] OF BOOLEAN;
    91  1   1:D   3            WEPVSTY3 : PACKED ARRAY[ 0..1, 0..6] OF BOOLEAN;
    92  1   1:D   3            WEPVSTYP : PACKED ARRAY[ 0..13] OF BOOLEAN;
    93  1   1:D   3            LOSTXYL  : RECORD CASE INTEGER OF                (* ^.64 *)
    94  1   1:D   3                1:  (LOCATION : ARRAY[ 1..4] OF INTEGER);
    95  1   1:D   3                2:  (POISNAMT : ARRAY[ 1..4] OF INTEGER);
    96  1   1:D   3                3:  (AWARDS   : ARRAY[ 1..4] OF INTEGER);
    97  1   1:D   3              END;
    98  1   1:D   3          END;
    99  1   1:D   3
   100  1   1:D   3        (* LOOKS LIKE TSCNTOC = RECORD;  FROM WIZ III *)
   101  1   1:D   3        (* BASE47C, 484, 48C LOOK LIKE RECPER2B, RECPERDK, BLOFF (?) *)
   102  1   1:D   3        TSCENARI = PACKED RECORD
   103  1   1:D   3            GAMENAME : PACKED ARRAY[ 0..28]  OF INTEGER;
   104  1   1:D   3            RECPERDK : PACKED ARRAY[ 0..7]   OF INTEGER;
   105  1   1:D   3            RECSIZE  : PACKED ARRAY[ 0..7]   OF INTEGER;
   106  1   1:D   3            BLOFF    : PACKED ARRAY[ 0..7]   OF INTEGER;  (* BASE48C *)
   107  1   1:D   3            Z        : PACKED ARRAY[ 0..129] OF INTEGER;
   108  1   1:D   3
   109  1   1:D   3            SPELLHSH : PACKED ARRAY[ 0..50] OF INTEGER;  (* BASE516 *)
   110  1   1:D   3            Z3       : PACKED ARRAY[ 0..50] OF 0..7;     (* BASE549 *)
   111  1   1:D   3            Z4       : PACKED ARRAY[ 0..6] OF INTEGER;
   112  1   1:D   3          END;
   113  1   1:D   3
   114  1   1:D   3
```

```
115  1  1:D  3      TOBJTYPE = (WEAPON, ARMOR, SHIELD, HELMET, GAUNTLET,
116  1  1:D  3                  SPECIAL, MISC);
117  1  1:D  3
118  1  1:D  3      TOBJREC = RECORD
119  1  1:D  3        OBJTYPE  : TOBJTYPE;
120  1  1:D  3        ALIGN    : TALIGN;
121  1  1:D  3        CURSED   : BOOLEAN;
122  1  1:D  3        SPECIAL  : INTEGER;
123  1  1:D  3        CHANGETO : INTEGER;
124  1  1:D  3        CHGCHANC : INTEGER;
125  1  1:D  3        PRICE    : TWIZLONG;
126  1  1:D  3        BOLTACXX : INTEGER;
127  1  1:D  3        SPELLPWR : INTEGER;
128  1  1:D  3        CLASSUSE : PACKED ARRAY[ TCLASS] OF BOOLEAN;
129  1  1:D  3        HEALPTS  : INTEGER;
130  1  1:D  3        WEPVSTY2 : PACKED ARRAY[ 0..15] OF BOOLEAN;
131  1  1:D  3        WEPVSTY3 : PACKED ARRAY[ 0..15] OF BOOLEAN;
132  1  1:D  3        ARMORMOD : INTEGER;
133  1  1:D  3        WEPHITMD : INTEGER;
134  1  1:D  3        WEPHPDAM : THPREC;
135  1  1:D  3        XTRASWNG : INTEGER;
136  1  1:D  3        CRITHITM : BOOLEAN;
137  1  1:D  3        WEPVSTYP : PACKED ARRAY[ 0..13] OF BOOLEAN;
138  1  1:D  3      END;
139  1  1:D  3
140  1  1:D  3      TWALL = (OPEN, WALL, DOOR, HIDEDOOR);
141  1  1:D  3
142  1  1:D  3      TSQUARE = (NORMAL, STAIRS, PIT, CHUTE, SPINNER, DARK, TRANSFER,
143  1  1:D  3                 OUCHY, BUTTONZ, ROCKWATE, FIZZLE, SCNMSG, ENCOUNTE);
144  1  1:D  3
145  1  1:D  3      TMAZE = RECORD
146  1  1:D  3        W : PACKED ARRAY[ 0..19] OF PACKED ARRAY[ 0..19] OF TWALL;
147  1  1:D  3        S : PACKED ARRAY[ 0..19] OF PACKED ARRAY[ 0..19] OF TWALL;
148  1  1:D  3        E : PACKED ARRAY[ 0..19] OF PACKED ARRAY[ 0..19] OF TWALL;
149  1  1:D  3        N : PACKED ARRAY[ 0..19] OF PACKED ARRAY[ 0..19] OF TWALL;
150  1  1:D  3
151  1  1:D  3        FIGHTS : PACKED ARRAY[ 0..19] OF PACKED ARRAY[ 0..19] OF 0..1;
152  1  1:D  3
153  1  1:D  3        SQREXTRA : PACKED ARRAY[ 0..19] OF PACKED ARRAY[ 0..19] OF 0..15;
154  1  1:D  3
155  1  1:D  3        SQRETYPE : PACKED ARRAY[ 0..15] OF TSQUARE;
156  1  1:D  3
157  1  1:D  3        AUX0 : PACKED ARRAY[ 0..15] OF INTEGER;
158  1  1:D  3        AUX1 : PACKED ARRAY[ 0..15] OF INTEGER;
159  1  1:D  3        AUX2 : PACKED ARRAY[ 0..15] OF INTEGER;
160  1  1:D  3
161  1  1:D  3        ENMYCALC : PACKED ARRAY[ 1..3] OF RECORD
162  1  1:D  3                     MINENEMY : INTEGER;
163  1  1:D  3                     MULTWORS : INTEGER;
164  1  1:D  3                     WORSE01  : INTEGER;
165  1  1:D  3                     RANGE0N  : INTEGER;
166  1  1:D  3                     PERCWORS : INTEGER;
167  1  1:D  3                   END;
168  1  1:D  3      END;
169  1  1:D  3
170  1  1:D  3
171  1  1:D  3      TMEMORYP = ^TMEMORY;
```

```
172  1  1:D  3
173  1  1:D  3        TMEMORY = PACKED RECORD
174  1  1:D  3            NEXT  : TMEMORYP;
175  1  1:D  3            SIZE  : 0..32767;
176  1  1:D  3            INUSE : BOOLEAN;
177  1  1:D  3          END;
178  1  1:D  3
179  1  1:D  3        TCACHEP = ^TCACHE;
180  1  1:D  3
181  1  1:D  3        TCACHE = RECORD
182  1  1:D  3            KEY      : INTEGER;  (* DISK ADDR, OR MSG # *)
183  1  1:D  3            FORWRD   : TCACHEP;
184  1  1:D  3            BACKWRD  : TCACHEP;
185  1  1:D  3            DATA     : TLONGSTR;
186  1  1:D  3          END;
187  1  1:D  3
188  1  1:D  3        TWINDOWP = ^TWINDOW;
189  1  1:D  3
190  1  1:D  3        TARR = ARRAY[ 0..0] OF INTEGER;
191  1  1:D  3        TARRP = ^TARR;
192  1  1:D  3
193  1  1:D  3        TMEMOVER = RECORD CASE INTEGER OF
194  1  1:D  3            1: (I:      INTEGER);
195  1  1:D  3            2: (PWIN:   TWINDOWP);
196  1  1:D  3            3: (PMEM:   TMEMORYP);
197  1  1:D  3            4: (PCACHE: TCACHEP);
198  1  1:D  3            5: (PARR:   TARRP);
199  1  1:D  3          END;
200  1  1:D  3
201  1  1:D  3        TWINHEAD = PACKED RECORD
202  1  1:D  3            NEXTW   : TWINDOWP;
203  1  1:D  3            HPOS    : 0..255;
204  1  1:D  3            VPOS    : 0..255;
205  1  1:D  3            HSIZE   : 0..255;
206  1  1:D  3            VSIZE   : 0..255;
207  1  1:D  3            HCURSOR : 0..255;
208  1  1:D  3            VCURSOR : 0..255;
209  1  1:D  3            PRIORITY: 0..255;
210  1  1:D  3            TEMP    : 0..255;
211  1  1:D  3          END;
212  1  1:D  3
213  1  1:D  3        TWINDOW = PACKED RECORD
214  1  1:D  3            HEAD : TWINHEAD;
215  1  1:D  3            DATA : PACKED ARRAY[ 0..719] OF CHAR;
216  1  1:D  3          END;
217  1  1:D  3
218  1  1:D  3        TBCD = ARRAY[ 0..13] OF INTEGER;
219  1  1:D  3
220  1  1:D  3        DIRENTRY = RECORD
221  1  1:D  3          FIRSTBLK : INTEGER;
222  1  1:D  3          LASTBLK  : INTEGER;
223  1  1:D  3          FILEKIND : PACKED RECORD
224  1  1:D  3              FT : (VOLHEAD, BADBLK, MACH6502, TEXT, DEBUG,
225  1  1:D  3                    DATA, GRAFFILE, FOTOFILE, SUBDIR);
226  1  1:D  3            END;
227  1  1:D  3          FILENAME : STRING[ 7];
228  1  1:D  3          VOLLB    : INTEGER;
```

```
229  1  1:D   3          FILECNT  : INTEGER;
230  1  1:D   3          LOADTIM  : INTEGER;
231  1  1:D   3          BOOTDATE : INTEGER;
232  1  1:D   3          RES1     : INTEGER;
233  1  1:D   3          RES2     : INTEGER;
234  1  1:D   3        END;
235  1  1:D   3
236  1  1:D   3   VAR
237  1  1:D   3        INCHAR   : CHAR;
238  1  1:D   4        BASE04   : INTEGER;
239  1  1:D   5        DIRECTIO : INTEGER;
240  1  1:D   6        MAZELEV  : INTEGER;     (* BASE06 *)
241  1  1:D   7        MAZEY    : INTEGER;
242  1  1:D   8        MAZEX    : INTEGER;
243  1  1:D   9        BASE09   : INTEGER;
244  1  1:D  10        CHARSWIN : TWINDOWP;
245  1  1:D  11        CAMPTIT  : TWINDOWP;
246  1  1:D  12        BASE0C   : PINT;
247  1  1:D  13        BASE0D   : PINT;
248  1  1:D  14        WINCHAIN : TWINDOWP;     (* BASE0E *)
249  1  1:D  15        CAMPWIN  : TWINDOWP;
250  1  1:D  16        BASE10   : TWINDOWP;
251  1  1:D  17        MAINWIN  : TWINDOWP;
252  1  1:D  18        BASE12   : INTEGER;
253  1  1:D  19
254  1  1:D  19        BASE13   : INTEGER;
255  1  1:D  20        BASE14   : INTEGER;
256  1  1:D  21        ONECACBL : BOOLEAN;
257  1  1:D  22        CACHEWRI : BOOLEAN;
258  1  1:D  23        DOGARBMM : BOOLEAN;
259  1  1:D  24
260  1  1:D  24        BASE18   : RECORD CASE INTEGER OF
261  1  1:D  24                     1: (B: BOOLEAN);
262  1  1:D  24                     2: (I: INTEGER);
263  1  1:D  24                   END;
264  1  1:D  25
265  1  1:D  25        INFOWOFF : RECORD CASE INTEGER OF   (* BASE19 *)
266  1  1:D  25                     1: (B: BOOLEAN);
267  1  1:D  25                     2: (I: INTEGER);
268  1  1:D  25                   END;
269  1  1:D  26
270  1  1:D  26        CACHEBL  : INTEGER;  (* BASE1A *)
271  1  1:D  27        KANABLCK : INTEGER;  (* BASE1B *)
272  1  1:D  28        ASCIIKBL : INTEGER;
273  1  1:D  29        SCENBLK  : INTEGER;
274  1  1:D  30        TIMEDLAY : INTEGER;
275  1  1:D  31        ATTK012  : INTEGER;  (* BASE1F *)
276  1  1:D  32        FIZZLES  : INTEGER;  (* BASE20 *)
277  1  1:D  33        CHSTALRM : INTEGER;
278  1  1:D  34        IDMONSTR : INTEGER;  (* BASE22 *)
279  1  1:D  35        LIGHT    : INTEGER;  (* BASE23 *)
280  1  1:D  36        ACMOD2   : INTEGER;  (* BASE24   PROTECT *)
281  1  1:D  37        BASE25   : INTEGER;
282  1  1:D  38        BASE26   : INTEGER;
283  1  1:D  39        BASE27   : INTEGER;
284  1  1:D  40        SAVELEV  : INTEGER;  (* BASE28 *)
285  1  1:D  41        SAVEY    : INTEGER;
```

```
286  1  1:D   42      SAVEX    : INTEGER;
287  1  1:D   43      BASE2B   : INTEGER;
288  1  1:D   44      MEMHEAD  : ^TMEMORY;  (* BASE2C *)
289  1  1:D   45      BASE2D   : INTEGER;
290  1  1:D   46      HFBLKCNT : INTEGER;   (* HUFF BLOCK CNT (LESS 1)  *)
291  1  1:D   47      HUFFBLK  : INTEGER;
292  1  1:D   48      ASCIIHSH : TMEMOVER;
293  1  1:D   49      MSGCACHL : TCACHEP;     (* BASE31 *)
294  1  1:D   50      MSGCACHF : TCACHEP;
295  1  1:D   51      DSKCACHL : TCACHEP;     (* BASE33 *)
296  1  1:D   52      DSKCACHF : TCACHEP;
297  1  1:D   53
298  1  1:D   53      BASE35   : INTEGER;
299  1  1:D   54      BASE36   : BOOLEAN;
300  1  1:D   55      BASE37   : BOOLEAN;
301  1  1:D   56      BASE38   : ARRAY[ 0..6] OF INTEGER;
302  1  1:D   63
303  1  1:D   63      BASE3F : ARRAY[ 0..2] OF INTEGER;
304  1  1:D   66      BASE42 : ARRAY[ 0..2] OF INTEGER;
305  1  1:D   69
306  1  1:D   69      BASE45 : INTEGER;
307  1  1:D   70      BASE46 : INTEGER;
308  1  1:D   71
309  1  1:D   71      BASE47   : TLONGSTR; (* BASE47 *)
310  1  1:D  199      BASEC7   : TLONGSTR; (* BASEC7 *)
311  1  1:D  327
312  1  1:D  327
313  1  1:D  327      BASE147 : ARRAY[ 0..2] OF STRING[ 15];
314  1  1:D  351
315  1  1:D  351    (* BASE14A : ARRAY[ 0..39] OF INTEGER; *)
316  1  1:D  351
317  1  1:D  351        FIGHTMAP : PACKED ARRAY[ 0..19, 0..19] OF BOOLEAN; (* BASE15F *)
318  1  1:D  391        (* BASEXXX  : ARRAY[ 0..19] OF INTEGER; *)
319  1  1:D  391
320  1  1:D  391      CHARACTR : ARRAY[ 0..6] OF TCHAR;
321  1  1:D 1119
322  1  1:D 1119
323  1  1:D 1119      SCENARIO : TSCENARI;
324  1  1:D 1371
325  1  1:D 1371      BASE55B  : PACKED ARRAY[ 0..1023] OF 0..255;
326  1  1:D 1883
327  1  1:D 1883      BASE75B  : TLONGSTR;  (* NOT SURE IF STRING OR RECORD (?) *)
328  1  1:D 2011                            (* AND DEFINITELY NOT SURE SIZE (?) *)
329  1  1:D 2011
330  1  1:D 2011      BASE7DB : PACKED ARRAY[ 0..15] OF BOOLEAN;
331  1  1:D 2012      BASE7DC : ARRAY[ 0..4] OF INTEGER;
332  1  1:D 2017      BLACKBOX : ARRAY[ 0..18] OF INTEGER;  (* BASE7E1 *)
333  1  1:D 2036      BASE7F4 : ARRAY[ 0..18] OF INTEGER;
334  1  1:D 2055
335  1  1:D 2055      TREBORY : INTEGER;  (* BASE807 *)
336  1  1:D 2056      TREBORX : INTEGER;  (* BASE808 *)
337  1  1:D 2057      CHGMSGP2 : INTEGER; (* CHANGE MESSAGE POWER OF 2 FOR TREBOR *)
338  1  1:D 2058      TREBMSG : INTEGER;
339  1  1:D 2059      BASE80B : INTEGER;
340  1  1:D 2060
341  1  1:D 2060      BRELOC   : BOOLEAN;  (* BASE80C *)
342  1  1:D 2061      BASCIIHU : BOOLEAN;
```

```
343  1   1:D  2062       BSTROPS  : BOOLEAN;
344  1   1:D  2063       BLINES24 : BOOLEAN;
345  1   1:D  2064       BCACHE   : BOOLEAN;
346  1   1:D  2065       BAPPLE   : BOOLEAN;  (* BASE811 *)
347  1   1:D  2066       BKRNSRCH : BOOLEAN;
348  1   1:D  2067       MAZEBLCK : INTEGER;
349  1   1:D  2068
350  1   1:D  2068       BASE814  : PACKED ARRAY[ 0..511] OF 0..7;
351  1   1:D  2171       BASE87B  : PACKED ARRAY[ 0..511] OF BOOLEAN;
352  1   1:D  2203
353  1   1:D  2203       BASE89B  : STRING;  (* COMPLETE GUESSING *)
354  1   1:D  2244
355  1   1:D  2244       PADDING  : ARRAY[ 1..87] OF INTEGER;
356  1   1:D  2331
357  1   1:D  2331
358  1   2:D     1   PROCEDURE PRINTBEL; FORWARD;                          (* P010002 *)
359  1   2:D     1
360  1   3:D     3   FUNCTION GETADDR( VAR A:STRING) : INTEGER; FORWARD;  (* P010003 *)
361  1   3:D     4
362  1   4:D     3   FUNCTION GETREC( DATATYPE : INTEGER;                 (* P010004 *)
363  1   4:D     4                     DATAINDX : INTEGER;
364  1   4:D     5                     DATASIZE : INTEGER) : INTEGER; FORWARD;
365  1   4:D     6
366  1   5:D     3   FUNCTION GETRECW( DATATYPE : INTEGER;                (* P010005 *)
367  1   5:D     4                      DATAINDX : INTEGER;
368  1   5:D     5                      DATASIZE : INTEGER) : INTEGER; FORWARD;
369  1   5:D     6
370  1   6:D     1   PROCEDURE ADDLONGS( VAR FIRST  : TWIZLONG;           (* P010006 *)
371  1   6:D     2                       VAR SECOND : TWIZLONG); FORWARD;
372  1   6:D     3
373  1   7:D     1   PROCEDURE SUBLONGS( VAR FIRST  : TWIZLONG;           (* P010007 *)
374  1   7:D     2                       VAR SECOND : TWIZLONG); FORWARD;
375  1   7:D     3
376  1   8:D     1   PROCEDURE MULTLONG( VAR LONGNUM : TWIZLONG;          (* P010008 *)
377  1   8:D     2                       VAR INTNUM  : INTEGER); FORWARD;
378  1   8:D     3
379  1   9:D     1   PROCEDURE DIVLONG( VAR LONGNUM : TWIZLONG;           (* P010009 *)
380  1   9:D     2                     VAR INTNUM  : INTEGER); FORWARD;
381  1   9:D     3
382  1  10:D     3   FUNCTION TESTLONG( FIRST  : TWIZLONG;                (* P01000A *)
383  1  10:D     4                     SECOND : TWIZLONG) : INTEGER;  FORWARD;
384  1  10:D    11
385  1  11:D     1   PROCEDURE PRNTLONG( PRNTWIN : TWINDOWP;              (* P01000B *)
386  1  11:D     2                       LONGNUM : TWIZLONG); FORWARD;
387  1  11:D     6
388  1  12:D     1   PROCEDURE GETKEY; FORWARD;                          (* P01000C *)
389  1  13:D     1   PROCEDURE GETCR; FORWARD;                           (* P01000D *)
390  1  14:D     1   PROCEDURE CENTSTR( MP02: TWINDOWP;                  (* P01000E *)
391  1  14:D     2                     MP01: TLONGSTR); FORWARD;
392  1  14:D   131
393  1  15:D     1   PROCEDURE PAUSE1; FORWARD;                          (* P01000F *)
394  1  15:D     1
395  1  16:D     1   PROCEDURE PAUSE2; FORWARD;                          (* P010010 *)
396  1  16:D     1
397  1  17:D     3   FUNCTION GETWIN( HPOS:     INTEGER;                (* P010011 *)
398  1  17:D     4                    VPOS:     INTEGER;
399  1  17:D     5                    HSIZE:    INTEGER;
```

```
400  1  17:D   6                  VSIZE:    INTEGER;
401  1  17:D   7                  PRIORITY: INTEGER;
402  1  17:D   8                  SHOW:     BOOLEAN)  : TWINDOWP; FORWARD;
403  1  17:D   9
404  1  18:D   1  PROCEDURE DELWIN( VAR  WIN2DEL : TWINDOWP;          (* P010012 *)
405  1  18:D   2                        REDRAW  : BOOLEAN); FORWARD;
406  1  18:D   3
407  1  19:D   1  PROCEDURE SETWNPRI( PRIWIN   : TWINDOWP;           (* P010013 *)
408  1  19:D   2                      PRIORITY : INTEGER;
409  1  19:D   3                      REDRAW   : BOOLEAN); FORWARD;
410  1  19:D   4
411  1  20:D   1  PROCEDURE MVCURSOR( MVWIN: TWINDOWP;                (* P010014 *)
412  1  20:D   2                      HPOS:  INTEGER;
413  1  20:D   3                      VPOS:  INTEGER); FORWARD;
414  1  20:D   4
415  1  21:D   1  PROCEDURE PRINTNUM( PRWIN   : TWINDOWP;             (* P010015 *)
416  1  21:D   2                      NUM     : INTEGER;
417  1  21:D   3                      FIELDSZ : INTEGER); FORWARD;
418  1  21:D   4
419  1  22:D   1  PROCEDURE PRINTCR( MP01 : TWINDOWP); FORWARD;       (* P010016 *)
420  1  22:D   2
421  1  23:D   1  PROCEDURE CLEARWIN( MP02 : TWINDOWP;                (* P010017 *)
422  1  23:D   2                      MP01 : BOOLEAN); FORWARD;
423  1  23:D   3
424  1  24:D   1  PROCEDURE CLRRECT( CLRWIN : TWINDOWP;               (* P010018 *)
425  1  24:D   2                     HOR1   : INTEGER;
426  1  24:D   3                     VER1   : INTEGER;
427  1  24:D   4                     HOR2   : INTEGER;
428  1  24:D   5                     VER2   : INTEGER); FORWARD;
429  1  24:D   6
430  1  25:D   1  PROCEDURE PROTWIN( PROTWIN : TWINDOWP;              (* P010019 *)
431  1  25:D   2                     REDRAW  : BOOLEAN); FORWARD;
432  1  25:D   3
433  1  26:D   1  PROCEDURE UNPROWIN( MP02 : TWINDOWP;                (* P01001A *)
434  1  26:D   2                      MP01 : BOOLEAN); FORWARD;
435  1  26:D   3
436  1  27:D   1  PROCEDURE SOLICIT(     PRWIN    : TWINDOWP;        (* P01001B *)
437  1  27:D   2                     VAR REPLYSTR : TLONGSTR;
438  1  27:D   3                         MAXSTR   : INTEGER); FORWARD;
439  1  27:D   4
440  1  28:D   3  FUNCTION MENU( MENUWIN : TWINDOWP;                  (* P01001C *)
441  1  28:D   4                 MENUSTR : TLONGSTR) : INTEGER; FORWARD;
442  1  28:D 133
443  1  29:D   1  PROCEDURE P01001D( VAR MP02 : TLONGSTR;             (* P01001D *)
444  1  29:D   2                        MP01 : INTEGER); FORWARD;
445  1  29:D   3
446  1  30:D   3  FUNCTION P01001E( MP03 : INTEGER) : INTEGER; FORWARD;
447  1  30:D   4
448  1  31:D   3  FUNCTION KEYPRESS : BOOLEAN; FORWARD;               (* P01001F *)
449  1  31:D   3
450  1  32:D   3  FUNCTION RAND : INTEGER; FORWARD;                   (* P010020 *)
451  1  32:D   3
452  1  33:D   1  PROCEDURE PRPICCH( MYWIN  : TWINDOWP;               (* P010021 *)
453  1  33:D   2                     MYCHAR : CHAR); FORWARD;
454  1  33:D   3
455  1  34:D   1  PROCEDURE PRPICMB( MYWIN  : TWINDOWP;               (* P010022 *)
456  1  34:D   2                     MYMBCH : TLONGSTR); FORWARD;
```

```
457  1  34:D  131
458  1  35:D    1  PROCEDURE DRAWSCR2( UNUSED : TWINDOWP); FORWARD;     (* P010023 *)
459  1  35:D    2
460  1  36:D    1  PROCEDURE GETMSGTX( VAR MSGSTR : TLONGSTR;          (* P010024 *)
461  1  36:D    2                          MSGNUM : INTEGER); FORWARD;
462  1  36:D    3
463  1  37:D    1  PROCEDURE DISP1MSGP( MSGWIN : TWINDOWP;             (* P010025 *)
464  1  37:D    2                       MSGNUM : INTEGER); FORWARD;
465  1  37:D    3
466  1  38:D    1  PROCEDURE DISP1LIN( MSGWIN : TWINDOWP;
467  1  38:D    2                      MSGNUM : INTEGER); FORWARD;
468  1  38:D    3
469  1  39:D    3  FUNCTION MENUINDX( MP04 : TWINDOWP;                 (* P010027 *)
470  1  39:D    4                     MP03 : INTEGER) : INTEGER; FORWARD;
471  1  39:D    5
472  1  40:D    3  FUNCTION MBSTRLEN( MP01 : TLONGSTR) : INTEGER ; FORWARD; (* P010028 *)
473  1  40:D  132
474  1  41:D    3  FUNCTION GTTITWIN( MSGINDX  : INTEGER;              (* P010029 *)
475  1  41:D    4                     VPOS     : INTEGER;
476  1  41:D    5                     VSIZE    : INTEGER;
477  1  41:D    6                     PRIORITY : INTEGER) : TWINDOWP; FORWARD;
478  1  41:D    7
479  1  42:D    1  PROCEDURE POOLGOLD( CHARI : INTEGER); FORWARD;      (* P01002A *)
480  1  42:D    2
481  1  43:D    1  PROCEDURE PCONCAT( VAR A: STRING; B: TLONGSTR); FORWARD; (* P01002B *)
482  1  43:D  131
483  1  44:D    1  PROCEDURE PCONCAT2( VAR MP03 : TLONGSTR;            (* P01002C *)
484  1  44:D    2                          MP02 : TLONGSTR;
485  1  44:D    3                          MP01 : TLONGSTR); FORWARD;
486  1  44:D  260
487  1  45:D    1  PROCEDURE PCONCAT3( VAR MP04 : TLONGSTR;            (* P01002D *)
488  1  45:D    2                          MP03 : TLONGSTR;
489  1  45:D    3                          MP02 : TLONGSTR;
490  1  45:D    4                          MP01 : TLONGSTR); FORWARD;
491  1  45:D  389
492  1  46:D    1  PROCEDURE COPYSTR( VAR DEST  : TLONGSTR;            (* P01002E *)
493  1  46:D    2                         SRC   : TLONGSTR;
494  1  46:D    3                         FIRST : INTEGER;
495  1  46:D    4                         COUNT : INTEGER); FORWARD;
496  1  46:D  133
497  1  47:D    3  FUNCTION P01002F( MP04 : TLONGSTR;
498  1  47:D    4                    MP03 : TLONGSTR) : INTEGER; FORWARD;
499  1  47:D  261
500  1  48:D    3  FUNCTION POS( SUBSTR : TLONGSTR;                    (* P010030 *)
501  1  48:D    4                BIGSTR : TLONGSTR) : INTEGER; FORWARD;
502  1  48:D  261
503  1  49:D    1  PROCEDURE INT2STR(     ININT  : INTEGER;            (* P010031 *)
504  1  49:D    2                     VAR OUTSTR : TLONGSTR); FORWARD;
505  1  49:D    3
506  1  50:D    1  PROCEDURE P010032(     MP02 : TWIZLONG;
507  1  50:D    2                     VAR MP01 : TLONGSTR); FORWARD;
508  1  50:D    6
509  1  51:D    3  FUNCTION GTLNGSUB( MENUSTR : TLONGSTR) : INTEGER; FORWARD;
510  1  51:D  132                                                      (* P010033 *)
511  1  51:D  132
512  1  52:D    3  FUNCTION GREATER( NUM1 : INTEGER;                   (* P010034 *)
513  1  52:D    4                    NUM2 : INTEGER) : INTEGER; FORWARD;
```

```
514   1   52:D     5
515   1   53:D     3   FUNCTION GETWIN2( MBHORSIZ : INTEGER;  (* P010035 *)
516   1   53:D     4                     VPOS     : INTEGER;
517   1   53:D     5                     VSIZE    : INTEGER;
518   1   53:D     6                     PRIORITY : INTEGER;
519   1   53:D     7                     SHOW     : BOOLEAN) : TWINDOWP; FORWARD;
520   1   53:D     8
521   1   54:D     1   PROCEDURE INSERT2( VAR INSERTTX : TLONGSTR;          (* P010036 *)
522   1   54:D     2                      VAR WHOLESTR : TLONGSTR;
523   1   54:D     3                          INSERTCH : TCHARSTR); FORWARD;
524   1   54:D     5
525   1   55:D     1   PROCEDURE INSERTST( VAR INSERTTX : TLONGSTR;         (* P010037 *)
526   1   55:D     2                       VAR WHOLESTR : TLONGSTR); FORWARD;
527   1   55:D     3
528   1   56:D     1   PROCEDURE BEEPSPKR( MP01 : INTEGER); FORWARD;        (* P010038 *)
529   1   56:D     2
530   1   57:D     1   PROCEDURE P010039( MP06 : TLONGSTR;          (* P010039 *)
531   1   57:D     2                      MP05 : INTEGER;
532   1   57:D     3                      MP04 : INTEGER;
533   1   57:D     4                      MP03 : INTEGER;
534   1   57:D     5                      MP02 : INTEGER;
535   1   57:D     6                      MP01 : INTEGER); FORWARD;
536   1   57:D   135
537   1   58:D     1   PROCEDURE P01003A( MP01 : TLONGSTR); FORWARD;   (* P01003A *)
538   1   58:D   130
539   1   58:D   130
540   1   59:D     3   FUNCTION FINDFILE( DRIVE  : INTEGER;                (* P01003B *)
541   1   59:D     4                      FILENM : STRING) : INTEGER; FORWARD;
542   1   59:D    46
543   1   59:D    46
544   1   59:D    46 (* ================  SEGMENTS ================= *)
545   1   59:D    46
546   7    1:D     1   SEGMENT PROCEDURE P010101;
547   7    1:0     0     BEGIN
548   7    1:0     0     END;
549   7    1:0    12
550   8    1:D     1   SEGMENT PROCEDURE P010201;
551   8    1:0     0     BEGIN
552   8    1:0     0     END;
553   8    1:0    12
554   9    1:D     1   SEGMENT PROCEDURE P010301;
555   9    1:0     0     BEGIN
556   9    1:0     0     END;
557   9    1:0    12
558  10    1:D     1   SEGMENT PROCEDURE P010401;
559  10    1:0     0     BEGIN
560  10    1:0     0     END;
561  10    1:0    12
562  11    1:D     1   SEGMENT PROCEDURE P010501;
563  11    1:0     0     BEGIN
564  11    1:0     0     END;
565  11    1:0    12
565  11    1:0    12 (*$I WIZ4B:KANJIREA  *)
566  11    1:0    12
567  11    1:0    12
568  12    1:D     1   SEGMENT PROCEDURE KANJIREA(    KANMP03: TWINDOWP;      (* P010601 *)
569  12    1:D     2                               VAR KANMP02: STRING;
```

```
570  12   1:D    3                                KANMP01: INTEGER);
571  12   1:D    4
572  12   1:D    4                            (*   NIL, BASE47, BASE09 *)
573  12   1:D    4                            (*        ??????, 0      *)
574  12   1:D    4
575  12   1:D    4    VAR
576  12   1:D    4         KANMP04 : PACKED ARRAY[ 0..15] OF BOOLEAN;
577  12   1:D    5                   (*
578  12   1:D    5                      00 = ASCII.KRN
579  12   1:D    5                      01 = KANA.KRN
580  12   1:D    5                      02 = KANJI.KRN
581  12   1:D    5
582  12   1:D    5                      04 = 200.CHARSET
583  12   1:D    5                      05 = 200.MONSTERS
584  12   1:D    5                   *)
585  12   1:D    5
586  12   1:D    5
587  12   2:D    3    FUNCTION WRITEPRO : BOOLEAN;       (* P010602 *)
588  12   2:D    3
589  12   2:D    3      VAR
590  12   2:D    3           MP03 : INTEGER;
591  12   2:D    4           MP04 : INTEGER;
592  12   2:D    5           MP05 : INTEGER;
593  12   2:D    6           MP06 : PACKED ARRAY[ 0..511] OF CHAR;
594  12   2:D  262           MP106 : INTEGER;
595  12   2:D  263
596  12   2:0    0       BEGIN
597  12   2:1    0         UNITREAD( DRIVE1, MP06, 512, 0);
598  12   2:1   11         MP106 := ORD( MP06[ 0]);
599  12   2:1   18         IF MP106 <> 255 THEN
600  12   2:2   27           MP06[ 0] := CHR( MP106 + 1)
601  12   2:1   35         ELSE
602  12   2:2   38           MP06[ 0] := CHR( 0);
603  12   2:1   43         UNITWRITE( DRIVE1, MP06, 512, 0);
604  12   2:1   54         UNITREAD( DRIVE1, MP06, 512, 0);
605  12   2:1   65         WRITEPRO := ORD( MP06[ 0]) = MP106;
606  12   2:1   75         MP06[ 0] := CHR( MP106);
607  12   2:1   82         UNITWRITE( DRIVE1, MP06, 512, 0)
608  12   2:0   93       END;
609  12   2:0  106
610  12   2:0  106
611  12   3:D    1    PROCEDURE P010603;
612  12   3:D    1
613  12   3:0    0       BEGIN
614  12   3:1    0         EXIT( WIZARDRY)
615  12   3:0    4       END;
616  12   3:0   16
617  12   3:0   16
618  12   4:D    1    PROCEDURE GARBMEM;   (* P010604 *)
619  12   4:D    1
620  12   4:D    1      VAR
621  12   4:D    1           CHAIN2   : ^TMEMORY;
622  12   4:D    2           B4NOTINU : ^TMEMORY;
623  12   4:D    3           CHAINPTR : ^TMEMORY;
624  12   4:D    4
625  12   4:D    4
626  12   4:0    0       BEGIN
```

```
627  12   4:1    0        DOGARBMM := FALSE;
628  12   4:1    3        CHAINPTR := MEMHEAD;
629  12   4:1    7        B4NOTINU := NIL;
630  12   4:1   10        WHILE CHAINPTR <> NIL DO
631  12   4:2   15          BEGIN
632  12   4:3   15            IF NOT CHAINPTR^.INUSE THEN
633  12   4:4   24              BEGIN
634  12   4:5   24                IF CHAINPTR^.NEXT <> NIL THEN
635  12   4:6   30                  BEGIN
636  12   4:7   30                    CHAIN2 := CHAINPTR^.NEXT;
637  12   4:7   34                    WHILE (CHAIN2 <> NIL) AND (NOT CHAIN2^.INUSE) DO
638  12   4:8   47                      BEGIN
639  12   4:9   47                        CHAINPTR^.NEXT := CHAIN2^.NEXT;
640  12   4:9   51                        CHAINPTR^.SIZE := CHAINPTR^.SIZE + CHAIN2^.SIZE;
641  12   4:9   70                        CHAIN2 := CHAIN2^.NEXT
642  12   4:8   71                      END
643  12   4:6   74                  END;
644  12   4:5   76                IF CHAINPTR^.NEXT = NIL THEN
645  12   4:6   82                  BEGIN
646  12   4:7   82                    IF B4NOTINU = NIL THEN
647  12   4:8   87                      MEMHEAD := NIL
648  12   4:7   87                    ELSE
649  12   4:8   92                      B4NOTINU^.NEXT := NIL;
650  12   4:7   95                    RELEASE( CHAINPTR)
651  12   4:6   97                  END
652  12   4:4   99              END;
653  12   4:3   99            B4NOTINU := CHAINPTR;
654  12   4:3  102            CHAINPTR := CHAINPTR^.NEXT
655  12   4:2  103          END;
656  12   4:0  108      END;
657  12   4:0  124
658  12   4:0  124
659  12   5:D    3    FUNCTION GETMEM( MEMSIZE: INTEGER) : TWINDOWP;  (* P010605 *)
660  12   5:D    4
661  12   5:D    4      VAR
662  12   5:D    4           SPLITBUF : ^TMEMORY;
663  12   5:D    5           MEMCHAIN : ^TMEMORY;
664  12   5:D    6           MP06     : TMEMOVER;
665  12   5:D    7           PTR2INT  : TMEMOVER;
666  12   5:D    8
667  12   5:0    0      BEGIN
668  12   5:1    0        IF DOGARBMM THEN
669  12   5:2    4          GARBMEM;
670  12   5:1    6        MEMSIZE := MEMSIZE + SIZEOF( TMEMORY);
671  12   5:1   11        MEMCHAIN :=MEMHEAD;
672  12   5:1   15        WHILE MEMCHAIN <> NIL DO
673  12   5:2   20          BEGIN
674  12   5:3   20            IF NOT MEMCHAIN^.INUSE THEN
675  12   5:4   29              IF MEMCHAIN^.SIZE >= MEMSIZE THEN
676  12   5:5   39                BEGIN
677  12   5:6   39                  IF (MEMCHAIN^.SIZE - MEMSIZE) > SIZEOF( TMEMORY) THEN
678  12   5:7   51                    BEGIN
679  12   5:8   51                      PTR2INT.PMEM := MEMCHAIN;
680  12   5:8   54                      PTR2INT.I := PTR2INT.I + MEMSIZE;
681  12   5:8   59                      SPLITBUF := PTR2INT.PMEM;
682  12   5:8   62                      SPLITBUF^.NEXT := MEMCHAIN^.NEXT;
683  12   5:8   66                      SPLITBUF^.SIZE := MEMCHAIN^.SIZE - MEMSIZE;
```

```
684  12   5:8   80                      SPLITBUF^.INUSE := FALSE;
685  12   5:8   87                      MEMCHAIN^.SIZE := MEMSIZE;
686  12   5:8   94                      MEMCHAIN^.NEXT := SPLITBUF
687  12   5:7   95                    END;
688  12   5:6   97                  MEMCHAIN^.INUSE := TRUE;
689  12   5:6  104                  PTR2INT.PMEM := MEMCHAIN;
690  12   5:6  107                  MP06.I := PTR2INT.I + SIZEOF( TMEMORY);
691  12   5:6  112                  GETMEM := MP06.PWIN;
692  12   5:6  115                  EXIT( GETMEM)
693  12   5:5  119                END;
694  12   5:3  119              SPLITBUF := MEMCHAIN;
695  12   5:3  122              MEMCHAIN := MEMCHAIN^.NEXT
696  12   5:2  123            END;
697  12   5:1  128          MARK( MEMCHAIN);
698  12   5:1  132          IF MEMHEAD = NIL THEN
699  12   5:2  138            MEMHEAD := MEMCHAIN
700  12   5:1  138          ELSE
701  12   5:2  143            SPLITBUF^.NEXT := MEMCHAIN;
702  12   5:1  146          MEMCHAIN^.NEXT := NIL;
703  12   5:1  149          MEMCHAIN^.SIZE := MEMSIZE;
704  12   5:1  156          MEMCHAIN^.INUSE := TRUE;
705  12   5:1  163          PTR2INT.PMEM := MEMCHAIN;
706  12   5:1  166          MP06.I := PTR2INT.I + SIZEOF( TMEMORY);
707  12   5:1  171          GETMEM := MP06.PWIN;
708  12   5:1  174          PTR2INT.I := PTR2INT.I + MEMSIZE;
709  12   5:1  179          RELEASE( PTR2INT.PMEM)
710  12   5:0  181        END;
711  12   5:0  198
712  12   5:0  198
713  12   6:D    1      PROCEDURE RELMEM( MP01: TWINDOWP);
714  12   6:D    2
715  12   6:D    2        VAR
716  12   6:D    2            MP02 : TMEMOVER;
717  12   6:D    3            MP03 : ^TMEMORY;
718  12   6:D    4            MP04 : TMEMOVER;
719  12   6:D    5            MP05 : INTEGER;
720  12   6:D    6
721  12   6:0    0        BEGIN
722  12   6:1    0          IF MEMHEAD = NIL THEN
723  12   6:2    6            EXIT( RELMEM);
724  12   6:1   10          MP03 := MEMHEAD;
725  12   6:1   14          MP04.PWIN := MP01;
726  12   6:1   17          MP05 := MP04.I - SIZEOF( TMEMORY);
727  12   6:1   22          MP02.I := MP05;
728  12   6:1   25          WHILE MP03 <> MP02.PMEM DO
729  12   6:2   30            BEGIN
730  12   6:3   30              MP03 := MP03^.NEXT;
731  12   6:3   34              IF MP03 = NIL THEN
732  12   6:4   39                EXIT( RELMEM);
733  12   6:2   43            END;
734  12   6:1   45          MP03^.INUSE := FALSE;
735  12   6:1   52          DOGARBMM := TRUE
736  12   6:0   52        END;
737  12   6:0   70
738  12   6:0   70
739  12   7:D    1      PROCEDURE P010607( MP01: BOOLEAN);
740  12   7:D    2
```

```
741  12    7:D     2       VAR
742  12    7:D     2           MP02 : INTEGER;
743  12    7:D     3
744  12    7:D     3
745  12    8:D     3       FUNCTION P010608( MP04 : TCACHEP;
746  12    8:D     4                         MP03 : INTEGER) : TCACHEP;
747  12    8:D     5
748  12    8:D     5         VAR
749  12    8:D     5             MP05 : TWINDOWP;
750  12    8:D     6
751  12    8:D     6             MP06 : TCACHEP;
752  12    8:D     7
753  12    8:D     7             MP07 : TCACHEP;
754  12    8:D     8
755  12    8:0     0         BEGIN (* P010608 *)
756  12    8:1     0           MP05 := GETMEM( MP03);
757  12    8:1     7           MOVELEFT( MP05, MP06, 2);
758  12    8:1    16           MP07 := MP06;
759  12    8:1    19           MP07^.FORWRD := NIL;
760  12    8:1    24           MP07^.BACKWRD := MP04;
761  12    8:1    29           IF MP04 <> NIL THEN
762  12    8:2    34             MP04^.FORWRD := MP06;
763  12    8:1    39           MP07^.KEY := -1;
764  12    8:1    43           P010608 := MP06
765  12    8:0    43         END;  (* P010608 *)
766  12    8:0    58
767  12    8:0    58
768  12    7:0     0       BEGIN (* P010607 *)
769  12    7:1     0         IF MP01 THEN
770  12    7:2     3           BEGIN
771  12    7:3     3             WHILE  MEMAVAIL > 10500 DO
772  12    7:4    11               BEGIN
773  12    7:5    11                 DSKCACHL := P010608( DSKCACHL, 518);
774  12    7:5    22                 FOR MP02 := 0 TO 3 DO
775  12    7:6    33                   MSGCACHL := P010608( MSGCACHL, 26)
776  12    7:4    36               END
777  12    7:2    49           END
778  12    7:1    51         ELSE
779  12    7:2    53           BEGIN
780  12    7:3    53             DSKCACHF  := P010608( NIL, 518);
781  12    7:3    63             DSKCACHL  := DSKCACHF;
782  12    7:3    67             MSGCACHF := P010608( NIL, 26);
783  12    7:3    75             MSGCACHL := MSGCACHF;
784  12    7:3    79             DSKCACHL  := P010608( DSKCACHL, 518);
785  12    7:3    90             FOR MP02 := 0 TO 23 DO
786  12    7:4   101               MSGCACHL := P010608( MSGCACHL, 26)
787  12    7:2   104           END
788  12    7:0   117       END;  (* P010607 *)
789  12    7:0   136
790  12    9:D     1     PROCEDURE P010609( MP02: STRING; MP01: INTEGER);
791  12    9:D    44                       (* 'ASCII.KRN', 0 *)
792  12    9:D    44       VAR
793  12    9:D    44           MP2C : INTEGER;
794  12    9:D    45           BUFFER : ARRAY[ 0..511] OF INTEGER;  (* GUESSING *)
795  12    9:D   557           MP22D : TWINDOWP;
796  12    9:D   558
797  12    9:D   558
```

```
798  12  10:D    1      PROCEDURE P01060A( MP02 : INTEGER;  MP01 : INTEGER);
799  12  10:D    3
800  12  10:D    3        VAR
801  12  10:D    3             UNUSED1 : INTEGER;
802  12  10:D    4             UNUSED2 : INTEGER;
803  12  10:D    5
804  12  10:0    0        BEGIN (* P01060A *)
805  12  10:1    0          MP22D := GETMEM( MP01);
806  12  10:1    9          UNITREAD( DRIVE1, MP22D^, MP01, ASCIIKBL + MP02)
807  12  10:0   23        END;  (* P01060A *)
808  12  10:0   36
809  12  10:0   36
810  12   9:0    0      BEGIN (* P010609 *)
811  12   9:1    0        BASE75B[ 0] := CHR( MP01);
812  12   9:1   11        IF ASCIIHSH.PMEM <> NIL THEN
813  12   9:2   17          BEGIN
814  12   9:3   17            MOVELEFT( ASCIIHSH.I, MP22D, 2);
815  12   9:3   27            RELMEM( MP22D);
816  12   9:2   32          END;
817  12   9:1   32        FOR MP2C := 0 TO 2 DO
818  12   9:2   47          KANMP04[ MP2C] := (MP2C = MP01);
819  12   9:2   68          (*
820  12   9:2   68             SET T/F IF WE HAVE THE FILE:
821  12   9:2   68
822  12   9:2   68                 [ 00]  ASCII.KRN
823  12   9:2   68                 [ 01]  KANA.KRN
824  12   9:2   68                 [ 02]  KANJI.KRN
825  12   9:2   68          *)
826  12   9:1   68        ASCIIKBL := FINDFILE( DRIVE1, MP02);
827  12   9:1   78        UNITREAD( DRIVE1, BUFFER, 512, ASCIIKBL);
828  12   9:1   90        HUFFBLK := 0;
829  12   9:1   93        HFBLKCNT := (BUFFER[ 1] DIV 2) - 1;
830  12   9:1  105        P01060A( BUFFER[ 0], BUFFER[ 1]);
831  12   9:1  119        MOVELEFT( MP22D, ASCIIHSH.I, 2);
832  12   9:1  129        COPYSTR( MP02, MP02, 1, LENGTH( MP02) - 3);
833  12   9:1  143        PCONCAT( MP02, 'HUFF');
834  12   9:1  155        MP2C := FINDFILE( DRIVE1, MP02);
835  12   9:1  165        BASCIIHU := MP2C > 0;
836  12   9:1  172        IF BASCIIHU THEN
837  12   9:2  177          BEGIN
838  12   9:3  177            UNITREAD( DRIVE1, BUFFER, 1024, MP2C);
839  12   9:3  189            UNITWRITE( BASE12, BUFFER, 32, 0)
840  12   9:2  199          END;
841  12   9:0  199      END;  (* P010609 *)
842  12   9:0  214
843  12  11:D    1    PROCEDURE INIT1;  (* P01060B *)
844  12  11:D    1
845  12  11:D    1      VAR
846  12  11:D    1           MP01 : INTEGER;
847  12  11:D    2           MP02 : INTEGER;
848  12  11:D    3           MP03 : INTEGER;
849  12  11:D    4           MP04B: INTEGER;
850  12  11:D    5           MP05 : RECORD CASE INTEGER OF
851  12  11:D    5                    0: (I : INTEGER);
852  12  11:D    5                    1: (P : PINT);
853  12  11:D    5                  END;
854  12  11:D    6
```

```
855  12  12:D   1  PROCEDURE CHARSET( A: STRING);  (* P01060C *)
856  12  12:D  43
857  12  12:D  43    VAR
858  12  12:D  43          FIRSTBLK: INTEGER;
859  12  12:D  44
860  12  12:0   0    BEGIN (* CHARSET *)
861  12  12:1   0      BASE47 := A;
862  12  12:1  11      PCONCAT( BASE47, '.CHARSET');
863  12  12:1  27      FIRSTBLK := FINDFILE( DRIVE1, BASE47);
864  12  12:1  37      IF FIRSTBLK < 0 THEN
865  12  12:2  43        CHARSET( '200')               (* THIS LOOKS HORRIBLE!!! *)
866  12  12:1  49      ELSE
867  12  12:2  53        UNITWRITE( BASE12, FIRSTBLK, 14, FIRSTBLK)
868  12  12:0  64    END; (* CHARSET *)
869  12  12:0  76
870  12  12:0  76
871  12  13:D   1  PROCEDURE MONSTERS( A: STRING;  MP01: INTEGER);  (* P01060D *)
872  12  13:D  44
873  12  13:D  44    VAR
874  12  13:D  44         MP2C : INTEGER;
875  12  13:D  45
876  12  13:0   0    BEGIN (* MONSTERS *)
877  12  13:1   0      BASE75B[ 1] := CHR( MP01);
878  12  13:1  11      BASE47 := A;
879  12  13:1  17      PCONCAT( BASE47, '.MONSTERS');
880  12  13:1  34      MP2C := FINDFILE( DRIVE1, BASE47);
881  12  13:1  44      IF MP2C < 0 THEN
882  12  13:2  50        MONSTERS( '200', 0)
883  12  13:1  57      ELSE
884  12  13:2  61        UNITWRITE( BASE12, MP2C, 18, MP2C)
885  12  13:0  72    END;  (* MONSTERS *)
886  12  13:0  84
887  12  13:0  84
888  12  11:0   0  BEGIN (* INIT1 *)
889  12  11:1   0    MEMHEAD := NIL;
890  12  11:1   3    DOGARBMM := FALSE;
891  12  11:1   6    ASCIIHSH.PMEM := NIL;
892  12  11:1   9    CACHEBL := -1;
893  12  11:1  13    CACHEWRI := FALSE;
894  12  11:1  16    ONECACBL := FALSE;
895  12  11:1  19    FILLCHAR( KANMP04, 2, 0);
896  12  11:1  27    UNITWRITE( BASE12, KANMP04, 1, 0); (* CLEAR SCREENS *)
897  12  11:1  38                       (*   := #$31 === [5] [4] [0] SET        *)
898  12  11:1  38                       (*    [ 0] = ASCII.KRN
899  12  11:1  38                            [ 1] = KANA.KRN  (FALSE)
900  12  11:1  38                            [ 2] = KANJI.KRN (FALSE)
901  12  11:1  38                            [ 3] = ?         (FALSE)
902  12  11:1  38                            [ 4] = 200.CHARSET
903  12  11:1  38                            [ 5] = 200.MONSTERS
904  12  11:1  38
905  12  11:1  38                          SETS $F1 AND $E6
906  12  11:1  38                       *)
907  12  11:1  38    IF KANMP04[ 4] THEN
908  12  11:2  48      CHARSET( '200')     (* 200.CHARSET TO $D000 BANK 1 *)
909  12  11:1  54    ELSE
910  12  11:2  58      CHARSET( '400');
911  12  11:2  66
```

```
912  12   11:1    66         IF KANMP04[ 5] THEN
913  12   11:2    76           MONSTERS( '200', 0)  (* 200.MONSTERS;  $66.67 := $34.35
914  12   11:2    83                                   CLEARS $74.75.76.77            *)
915  12   11:1    83         ELSE
916  12   11:2    87           MONSTERS( '400', 1);
917  12   11:2    96
918  12   11:1    96         WINCHAIN := NIL;
919  12   11:1    99         IF BAPPLE THEN
920  12   11:2   104           BEGIN
921  12   11:3   104             MARK( MP05.P);
922  12   11:3   108             IF MP05.I <= 8182 THEN               (*  8182 == $1FF6 *)
923  12   11:4   115               MAINWIN := GETMEM( 8184 - MP05.I)
924  12   11:3   120             ELSE
925  12   11:4   128               MAINWIN := NIL;
926  12   11:3   131             MARK( MP05.P);
927  12   11:3   135             BASE10 := GETMEM( 16380 - MP05.I);  (* 16380 == $3FFC *)
928  12   11:3   146             IF MAINWIN <> NIL THEN
929  12   11:4   152               RELMEM( MAINWIN);
930  12   11:2   156           END;
931  12   11:1   156         MAINWIN := GETWIN( 1, 1, 38, 22, 0, TRUE);
932  12   11:1   169         BASE0D := NIL;
933  12   11:1   172         BASE0C := NIL;
934  12   11:1   175         CAMPTIT := NIL;
935  12   11:1   178         BASE10 := NIL;
936  12   11:1   181         CHARSWIN := NIL;
937  12   11:1   184         CAMPWIN  := NIL;
938  12   11:1   187         INFOWOFF.B := FALSE;
939  12   11:1   190         BASE18.B := TRUE;
940  12   11:1   193         P010609( 'ASCII.KRN', 0);
941  12   11:1   208         P010607( FALSE);
942  12   11:1   211         KANMP01 := 1;
943  12   11:0   215       END;  (* INIT1 *)
944  12   11:0   228
945  12   11:0   228
946  12   14:D     1     PROCEDURE SOLICSM;                    (* P01060E *)
947  12   14:D     1
948  12   14:D     1       VAR
949  12   14:D     1           MP01 : INTEGER;
950  12   14:D     2           MP02 : INTEGER;
951  12   14:D     3
952  12   14:D     3
953  12   15:D     1       PROCEDURE DISPXLIN( DISPWIN : TWINDOWP;  (* P01060F *)
954  12   15:D     2                           VPOS    : INTEGER;
955  12   15:D     3                           FIRSTMSG : INTEGER;
956  12   15:D     4                           MSGCNT   : INTEGER);
957  12   15:D     5
958  12   15:D     5         VAR
959  12   15:D     5             MSGNUM : INTEGER;
960  12   15:D     6
961  12   15:0     0         BEGIN (* DISPXLIN *)
962  12   15:1     0           MVCURSOR( DISPWIN, 0, VPOS);
963  12   15:1     6           FOR MSGNUM := FIRSTMSG TO FIRSTMSG + MSGCNT - 1 DO
964  12   15:2    21             DISP1LIN( DISPWIN, MSGNUM)
965  12   15:0    23         END;  (* DISPXLIN *)
966  12   15:0    48
967  12   15:0    48
968  12   16:D     1       PROCEDURE SCRLENMY;          (* P010610 *)
```

```
 969  12  16:D     1
 970  12  16:D     1          VAR
 971  12  16:D     1               MP01 : INTEGER;
 972  12  16:D     2               MP02 : INTEGER;
 973  12  16:D     3               MP03 : INTEGER;
 974  12  16:D     4               MP04 : INTEGER;
 975  12  16:D     5               MP05 : INTEGER;
 976  12  16:D     6
 977  12  16:0     0          BEGIN (* SCRLENMY *)
 978  12  16:0     0            (* CALC # OF BYTES IN 1 ROW OF WINMAIN FOR HI-RES PICTURES *)
 979  12  16:1     0            MP01 := 2 * ( MAINWIN^.HEAD.HSIZE - 2);
 980  12  16:1    13
 981  12  16:1    13            (* SCROLL THE PICS DOWNWARD AND OFF THE SCREEN *)
 982  12  16:1    13            FOR MP05 := 9 TO 13 DO
 983  12  16:2    24              BEGIN
 984  12  16:3    24                FOR MP04 := 13 DOWNTO MP05 + 1 DO
 985  12  16:4    37                  MOVELEFT( MAINWIN^.DATA[ MP01 * (MP04 - 1)],
 986  12  16:4    46                            MAINWIN^.DATA[ MP01 * MP04],
 987  12  16:4    53                            MP01);
 988  12  16:4    63
 989  12  16:4    63                (* CLEAR EACH 16 BIT VALUE TO #$20 #$20 *)
 990  12  16:3    63                FILLCHAR( MAINWIN^.DATA[ MP01 * MP05],
 991  12  16:3    70                          MP01,
 992  12  16:3    71                          ' ');
 993  12  16:3    74                UNITWRITE( BASE12, MAINWIN^, 3, 0)
 994  12  16:2    84              END;
 995  12  16:0    91          END;  (* SCRLENMY *)
 996  12  16:0   108
 997  12  16:0   108
 998  12  14:0     0        BEGIN  (* SOLICSM *)
 999  12  14:1     0          IF BLINES24 THEN
1000  12  14:2     5            BASE10 := GETWIN( 0, 17, 40, 7, 30, FALSE)
1001  12  14:1    11          ELSE
1002  12  14:2    20            BASE10 := GETWIN( 0, 18, 40, 7, 30, FALSE);
1003  12  14:1    33          CLEARWIN( MAINWIN, TRUE);
1004  12  14:1    39          IF FINDFILE( DRIVE1, 'IS.TITLE') > 0 THEN
1005  12  14:2    60            BEGIN
1006  12  14:3    60              IF KANMP04[ 5] THEN
1007  12  14:4    70                BASE47 := '200.TITLE'
1008  12  14:3    72              ELSE
1009  12  14:4    88                BASE47 := '400.TITLE';
1010  12  14:3   104              MP02 := FINDFILE( DRIVE1, BASE47);
1011  12  14:3   114              UNITWRITE( BASE12, MP02, 23, MP02);
1012  12  14:2   124            END;
1013  12  14:2   124
1014  12  14:1   124          DISPXLIN( MAINWIN, 0, 2040, 6); (* VPOS = 0; MSG 2040 - 2045 *)
1015  12  14:1   133                                            (* "TITLE" AND CREDITS       *)
1016  12  14:1   133
1017  12  14:1   133          UNITREAD( DRIVE1, BASEC7, 19, 5);    (* READ 19 BYTES FROM BLOCK #5 *)
1018  12  14:1   143                                            (* LAST BLOCK OF DIRECTORY     *)
1019  12  14:1   143
1020  12  14:1   143          DISPXLIN( BASE10, 2, 2021, 2);  (* VPOS = 2; MSG 2021 - 2022   *)
1021  12  14:1   151                                            (* S)TART GAME, M)AKE SCENARIO *)
1022  12  14:1   151
1023  12  14:1   151          REPEAT
1024  12  14:1   151
1025  12  14:2   151            WHILE NOT KEYPRESS DO
```

```
1026  12   14:3   159            BEGIN
1027  12   14:4   159              FILLCHAR( BASE47, 4, CHR( 255) );  (* BASE47 === KANMP02 *)
1028  12   14:4   168              UNITWRITE( BASE12, KANMP02, 17, ORD( MAINWIN));
1029  12   14:4   180              UNITWRITE( BASE12, MAINWIN^, 3, 0);
1030  12   14:4   190              FILLCHAR( BASE47, 4, CHR( 255) );
1031  12   14:4   199              MP01 := RAND MOD 4;
1032  12   14:4   208              FOR MP02 := 0 TO MP01 DO
1033  12   14:5   219                KANMP02[ MP02] := CHR( 1 + RAND MOD 24);
1034  12   14:4   240              UNITWRITE( BASE12, KANMP02, 13, ORD( MAINWIN));
1035  12   14:4   252              UNITWRITE( BASE12, MAINWIN^, 3, 0);
1036  12   14:4   262              MP02 := 0;
1037  12   14:4   265              WHILE (MP02 < 30)  AND (NOT KEYPRESS) DO
1038  12   14:5   277                BEGIN
1039  12   14:6   277                  MP02 := MP02 + 1;
1040  12   14:6   282                  MP01 := 2;
1041  12   14:6   285                  UNITWRITE( BASE12, MP01, 19, MP01)
1042  12   14:5   295                END;
1043  12   14:4   297              SCRLENMY
1044  12   14:3   297            END;
1045  12   14:3   301
1046  12   14:3   301          (* AFTER WHILE LOOPS *)
1047  12   14:3   301
1048  12   14:2   301          GETKEY;                  (* INCHAR := KEY *)
1049  12   14:2   304          GETMSGTX( BASE47, 2021); (*  "S)TART GAME"    *)
1050  12   14:2   312          GETMSGTX( BASEC7, 2022); (*  "M)AKE SCENARIO" *)
1051  12   14:2   321
1052  12   14:1   321        UNTIL (INCHAR = BASE47[ 1]) OR   (*  "S"  *)
1053  12   14:1   327              (INCHAR = BASEC7[ 1]);     (*  "M"  *)
1054  12   14:1   337
1055  12   14:1   337        DELWIN( BASE10, TRUE);
1056  12   14:1   343        IF INCHAR = BASEC7[ 1] THEN
1057  12   14:2   352          KANMP01 := 2  (* M)AKE SCENARIO *)
1058  12   14:1   352        ELSE
1059  12   14:2   358          KANMP01 := 3; (* S)TART GAME    *)
1060  12   14:1   362        MP02 := 0;
1061  12   14:1   365        UNITWRITE( BASE12, MP02, 23, MP02)
1062  12   14:1   375
1063  12   14:0   375      END;   (* SOLICSM *)
1064  12   14:0   398
1065  12   14:0   398
1066  12   17:D     1    PROCEDURE P010611;
1067  12   17:D     1
1068  12   17:D     1      VAR
1069  12   17:D     1           MP01 : INTEGER;
1070  12   17:D     2           MP02 : INTEGER;
1071  12   17:D     3           MP03 : INTEGER;
1072  12   17:D     4           MP04 : INTEGER;
1073  12   17:D     5
1074  12   17:0     0      BEGIN (* P010611 *)
1075  12   17:1     0        FILLCHAR( KANMP04, 2, 0);
1076  12   17:1     8        UNITWRITE( BASE12, KANMP04, 1, 0);  (* JUST A NOP ????? *)
1077  12   17:1    19        BASE36 := KANMP04[ 2];
1078  12   17:1    29        IF KANMP04[ 1] OR KANMP04[ 2] THEN
1079  12   17:2    48          BEGIN
1080  12   17:2    48
1081  12   17:2    48            (* KANA.KRN OR KANJI.KRN PROCESSING  (?)  *)
1082  12   17:2    48
```

```
1083  12   17:3    48            CLEARWIN( MAINWIN, TRUE);
1084  12   17:3    54            PRINTCR( MAINWIN);
1085  12   17:3    59            DISP1LIN( MAINWIN, 2024);   (* 2024 = SELECT LANGUAGE *)
1086  12   17:3    67            PRINTCR( MAINWIN);
1087  12   17:3    72            DISP1LIN( MAINWIN, 2025);   (* 2025 = ENGLISH *)
1088  12   17:3    80            IF KANMP04[ 1] THEN
1089  12   17:4    90              DISP1LIN( MAINWIN, 2026); (* 2026 = KANA    *)
1090  12   17:3    98            IF KANMP04[ 2] THEN
1091  12   17:4   108              DISP1LIN( MAINWIN, 2027); (* 2027 = KANJI   *)
1092  12   17:3   116            MVCURSOR( MAINWIN, 0, 9);
1093  12   17:3   123            BASE09 := 1700;                  (* 1700 = CREDITS+NEC 9801 (C) *)
1094  12   17:3   128            GETMSGTX( BASE47, BASE09);       (*        "TOFU" MSG            *)
1095  12   17:3   134            WHILE BASE47 <> '**ERR**' DO
1096  12   17:4   150              BEGIN
1097  12   17:5   150                CENTSTR( MAINWIN, BASE47);
1098  12   17:5   157                BASE09 :=  BASE09 + 1;
1099  12   17:5   162                GETMSGTX( BASE47, BASE09)
1100  12   17:4   165              END;
1101  12   17:3   170            REPEAT
1102  12   17:4   170              REPEAT
1103  12   17:5   170                GETKEY;
1104  12   17:5   173                MP03 := ORD( INCHAR) - 48
1105  12   17:4   174              UNTIL (MP03 > 0) AND (MP03 <= 3);
1106  12   17:3   187            UNTIL KANMP04[ MP03 - 1];
1107  12   17:3   199
1108  12   17:3   199            CASE MP03 OF
1109  12   17:3   202             1: P010609( 'ASCII.KRN', MP03 - 1);
1110  12   17:3   221             2: P010609( 'KANA.KRN',  MP03 - 1);
1111  12   17:3   239             3: P010609( 'KANJI.KRN', MP03 - 1);
1112  12   17:3   258            END;
1113  12   17:2   272          END;
1114  12   17:1   272        CLEARWIN( MAINWIN, TRUE);
1115  12   17:1   278        IF KANMP04[ 3] THEN
1116  12   17:2   288          BEGIN
1117  12   17:3   288            MVCURSOR( MAINWIN, 0, 9);
1118  12   17:3   295            DISP1LIN( MAINWIN, 5);
1119  12   17:3   301            GETMSGTX( BASE47, 31);
1120  12   17:3   307            GETMSGTX( BASEC7, 32);
1121  12   17:3   314            REPEAT
1122  12   17:4   314              GETKEY;
1123  12   17:3   317            UNTIL (INCHAR = BASE47[ 1]) OR
1124  12   17:3   323                  (INCHAR = BASEC7[ 1]);
1125  12   17:3   333            KANMP04[ 3] := INCHAR = BASE47[ 1];
1126  12   17:3   347            CLEARWIN( MAINWIN, TRUE)
1127  12   17:2   350          END;
1128  12   17:1   353        UNITWRITE( BASE12, KANMP04, 1, 0);  (* JUST A NOP ???? *)
1129  12   17:1   364        P010607( TRUE);
1130  12   17:1   367        KANMP01 := 4
1131  12   17:0   367      END;  (* P010611 *)
1132  12   17:0   394
1133  12   17:0   394
1134  12   18:D     1    PROCEDURE P010612;
1135  12   18:D     1
1136  12   18:D     1      VAR
1137  12   18:D     1          MP01 : INTEGER;
1138  12   18:D     2          MP02 : INTEGER;
1139  12   18:D     3          MP03 : INTEGER;
```

```
1140  12  18:D   4          MP04 : INTEGER;
1141  12  18:D   5          MP05 : INTEGER;
1142  12  18:D   6          MP06 : INTEGER;
1143  12  18:D   7          MP07 : INTEGER;
1144  12  18:D   8          MP08 : INTEGER;
1145  12  18:D   9          MP09 : INTEGER;
1146  12  18:D  10          MP0A : INTEGER;
1147  12  18:D  11          MP0B : INTEGER;
1148  12  18:D  12          MP0C : INTEGER;
1149  12  18:D  13
1150  12  18:0   0      BEGIN (* P010612 *)
1151  12  18:1   0        MP03 := 0;
1152  12  18:1   3        UNITWRITE( BASE12, MP03, 22, MP03);
1153  12  18:1  13        WHILE WRITEPRO DO
1154  12  18:2  19          BEGIN
1155  12  18:3  19            CLEARWIN( MAINWIN, TRUE);
1156  12  18:3  25            MVCURSOR( MAINWIN, 0, 10);
1157  12  18:3  32            DISP1LIN( MAINWIN, 13);  (* PLACE SCENARIO DISK "A" *)
1158  12  18:3  38            DISP1LIN( MAINWIN, 14);  (* IN DRIVE 1, PRESS RTN   *)
1159  12  18:3  44            GETCR
1160  12  18:2  44          END;
1161  12  18:1  49        SCENBLK := FINDFILE( DRIVE1, 'SCENARIO.DATA');
1162  12  18:1  73        UNITREAD( DRIVE1, BASE55B, 512, 5);  (* SERIALBL *)
1163  12  18:1  85        MVCURSOR( MAINWIN, 0, 0);
1164  12  18:1  92
1165  12  18:1  92        (* DO IT AGAIN!? *)
1166  12  18:1  92        SCENBLK := FINDFILE( DRIVE1, 'SCENARIO.DATA');
1167  12  18:1 116        UNITREAD( DRIVE1, BASE55B, 512, 5);
1168  12  18:1 128
1169  12  18:1 128        TIMEDLAY := 20;
1170  12  18:1 131        CACHEWRI := FALSE;
1171  12  18:1 134        CACHEBL := SCENBLK;
1172  12  18:1 138        ONECACBL := FALSE;
1173  12  18:1 141        UNITREAD( DRIVE1, BASE55B, 1024, SCENBLK);
1174  12  18:1 154        MOVELEFT( BASE55B, SCENARIO, 504);
1175  12  18:1 167        KANABLCK := FINDFILE( DRIVE1, 'KANA.KEYMAP');
1176  12  18:1 189        BASE13 := 1;
1177  12  18:1 192        MAZEX := 0;
1178  12  18:1 195        MAZEY := 0;
1179  12  18:1 198        MAZELEV := 0;
1180  12  18:1 201        BASE04 := 0;
1181  12  18:1 204        DIRECTIO := 0;
1182  12  18:1 207        ACMOD2 := 0;
1183  12  18:1 210        MP03 := 1;
1184  12  18:1 213        UNITWRITE( BASE12, MP03, 22, MP03);
1185  12  18:1 223        IF BCACHE THEN
1186  12  18:2 228          BASE09 := 2
1187  12  18:1 228        ELSE
1188  12  18:2 233          BASE09 := 0;
1189  12  18:1 236        EXIT( KANJIREA)
1190  12  18:0 240      END;  (* P010612 *)
1191  12  18:0 254
1192  12  18:0 254
1193  12  19:D   1    PROCEDURE MOREWIZ;  (* P010613 *)
1194  12  19:D   1
1195  12  19:0   0      BEGIN
1196  12  19:1   0        GETMSGTX( BASE47, 20);  (* RTN FOR MORE WIZARDRY *)
```

```
1197  12   19:1     6         BASE10 := GETWIN2( MBSTRLEN( BASE47), 21, 3, 1, TRUE);
1198  12   19:1    24         CENTSTR( BASE10, BASE47);
1199  12   19:1    30         DELWIN( BASE10, FALSE);
1200  12   19:1    36         GETCR;
1201  12   19:1    39         BASE13 := 1;
1202  12   19:1    42         MAZEX := 0;
1203  12   19:1    45         MAZEY := 0;
1204  12   19:1    48         MAZELEV := 0;
1205  12   19:1    51         BASE04 := 0;
1206  12   19:1    54         DIRECTIO := 0;
1207  12   19:1    57         ACMOD2 := 0;
1208  12   19:1    60         BASE09 := 0;
1209  12   19:1    63         EXIT( KANJIREA)
1210  12   19:0    67       END;  (* MOREWIZ *)
1211  12   19:0    80
1212  12   19:0    80 (*$I WIZ4B:KANJIREA  *)
1212  12   19:0    80 (*$I WIZ4B:KANJIREA2 *)
1213  12   19:0    80
1214  12   19:0    80
1215  12   19:0    80 (* KANJIREA2.TEXT *)
1216  12   19:0    80
1217  12   20:D     1     PROCEDURE P010614;
1218  12   20:D     1
1219  12   20:D     1       VAR
1220  12   20:D     1            P14MP01 : INTEGER;
1221  12   20:D     2            P14MP02 : INTEGER;
1222  12   20:D     3            P14MP03 : INTEGER;
1223  12   20:D     4            MP04 : TWINDOWP;
1224  12   20:D     5            MP05 : BOOLEAN;
1225  12   20:D     6            MP06 : BOOLEAN;
1226  12   20:D     7            MP07 : CHAR;
1227  12   20:D     8            MP08 : CHAR;
1228  12   20:D     9            MP09 : INTEGER;
1229  12   20:D    10            MP0A : INTEGER;
1230  12   20:D    11            MP0B : INTEGER;
1231  12   20:D    12
1232  12   20:D    12            MP0C : STRING[ 2];
1233  12   20:D    14            MP0E : INTEGER;
1234  12   20:D    15            MP0F : INTEGER;
1235  12   20:D    16
1236  12   20:D    16            MP10 : STRING[ 62];
1237  12   20:D    48            MP30 : PACKED ARRAY[ 0..255] OF CHAR;
1238  12   20:D   176
1239  12   20:D   176
1240  12   21:D     1       PROCEDURE P010615;
1241  12   21:D     1
1242  12   21:0     0         BEGIN
1243  12   21:1     0           P14MP01 := P14MP01 - 1;
1244  12   21:1     8           IF (KANMP02[ P14MP01] >= CHR( 129) ) AND
1245  12   21:1    19              (KANMP02[ P14MP01] <= CHR( 159) ) THEN
1246  12   21:2    33             P14MP01 := P14MP01 - 1;
1247  12   21:1    41           P14MP03 := P14MP03 - 1;
1248  12   21:1    49           MVCURSOR( KANMP03, P14MP03, P14MP02);
1249  12   21:1    61           PRPICCH( KANMP03, ' ');
1250  12   21:1    68           MVCURSOR( KANMP03, P14MP03, P14MP02)
1251  12   21:0    77         END;
1252  12   21:0    92
```

```
1253  12   21:0    92
1254  12   22:D     1       PROCEDURE P010616( VAR MP01 : CHAR);
1255  12   22:D     2
1256  12   22:D     2         VAR
1257  12   22:D     2              (* GOOFY CONSTRUCT BUT IT WORKS! *)
1258  12   22:D     2
1259  12   22:D     2             MP02 : PACKED RECORD CASE INTEGER OF
1260  12   22:D     2                 1 : (I: INTEGER);
1261  12   22:D     2                 2 : (S: STRING[ 1]);
1262  12   22:D     2               END;
1263  12   22:D     3
1264  12   22:0     0         BEGIN
1265  12   22:1     0           UNITREAD( 2, MP02, 1, 0);  (* KEYBOARD *)
1266  12   22:1     9           MP01 := MP02.S[ 0];
1267  12   22:0    15         END;
1268  12   22:0    28
1269  12   22:0    28
1270  12   23:D     1       PROCEDURE P010617( MP02 : INTEGER;
1271  12   23:D     2                          MP01 : INTEGER);
1272  12   23:D     3
1273  12   23:D     3         VAR
1274  12   23:D     3             MP03 : CHAR;
1275  12   23:D     4
1276  12   23:0     0         BEGIN
1277  12   23:1     0           IF P14MP01 > 0 THEN
1278  12   23:2     7             IF MP10[ P14MP01] <> CHR( 255) THEN
1279  12   23:3    20               IF (ORD( MP30[ 64 + ORD( MP10[ P14MP01])])) > MP02 THEN
1280  12   23:4    37                 BEGIN
1281  12   23:5    37                   MP03 := CHR( ORD( MP10[ P14MP01 - 1]) + MP01);
1282  12   23:5    50                   INCHAR := MP03;
1283  12   23:5    53                   MP0C[ 1] := KANMP02[ P14MP01 - 1];
1284  12   23:5    67                   MP0C[ 2] := INCHAR;
1285  12   23:5    73                   KANMP02[ P14MP01] := INCHAR;
1286  12   23:5    81                   MVCURSOR( KANMP03, P14MP03 - 1, P14MP02);
1287  12   23:5    95                   PRPICMB( KANMP03, MP0C)
1288  12   23:4   101                 END
1289  12   23:0   104         END;
1290  12   23:0   116
1291  12   23:0   116
1292  12   24:D     1       PROCEDURE P010618( VAR MP03 : CHAR;
1293  12   24:D     2                          VAR MP02 : CHAR;
1294  12   24:D     3                              MP01 : INTEGER);
1295  12   24:D     4
1296  12   24:D     4         VAR
1297  12   24:D     4             MP04 : INTEGER;
1298  12   24:D     5             MP05 : INTEGER;
1299  12   24:D     6
1300  12   24:0     0         BEGIN
1301  12   24:1     0           MP05 := ORD( MP03);
1302  12   24:1     4           MP04 := ORD( MP02);
1303  12   24:1     8
1304  12   24:1     8           MP04 := MP04 + MP01;
1305  12   24:1    13           IF MP04 = 127 THEN
1306  12   24:2    18             IF MP01 < 0 THEN
1307  12   24:3    23               MP04 := MP04 - 1
1308  12   24:2    24             ELSE
1309  12   24:3    30               MP04 := MP04 + 1
```

```
1310  12   24:1    31           ELSE
1311  12   24:2    37             IF MP04 > 252 THEN
1312  12   24:3    44               BEGIN
1313  12   24:4    44                 MP04 := MP04 - 189;
1314  12   24:4    51                 MP05 := MP05 + 1
1315  12   24:3    52               END
1316  12   24:2    56             ELSE
1317  12   24:3    58               IF MP04 < 64 THEN
1318  12   24:4    63                 IF MP05 = 129 THEN
1319  12   24:5    70                   BEGIN
1320  12   24:6    70                     MP04 := MP04 + 51;
1321  12   24:6    75                     MP05 := 152
1322  12   24:5    75                   END
1323  12   24:4    80                 ELSE
1324  12   24:5    82                   BEGIN
1325  12   24:6    82                     MP04 := MP04 + 189;
1326  12   24:6    89                     MP05 := MP05 - 1
1327  12   24:5    90                   END
1328  12   24:3    94               ELSE
1329  12   24:4    96                 IF (MP05 = 152) AND (MP04 > 114) THEN
1330  12   24:5   107                   BEGIN
1331  12   24:6   107                     MP05 := 129;
1332  12   24:6   112                     MP04 := MP04 - 51
1333  12   24:5   113                   END;
1334  12   24:5   117
1335  12   24:1   117           MP03 := CHR( MP05);
1336  12   24:1   120           MP02 := CHR( MP04)
1337  12   24:1   122
1338  12   24:0   122         END;
1339  12   24:0   136
1340  12   24:0   136
1341  12   25:D     1       PROCEDURE P010619;
1342  12   25:D     1
1343  12   25:D     1         VAR
1344  12   25:D     1              MP01 : INTEGER;
1345  12   25:D     2              MP02 : CHAR;
1346  12   25:D     3              MP03 : CHAR;
1347  12   25:D     4
1348  12   25:0     0         BEGIN
1349  12   25:1     0           MP03 := MP08;
1350  12   25:1     5           MP02 := MP07;
1351  12   25:1    10           P010618( MP03, MP02, -MP0B);
1352  12   25:1    20           FOR MP01 := 0 TO 9 DO
1353  12   25:2    31             BEGIN
1354  12   25:3    31               MVCURSOR( MP04, MP01 + MP01 + 1, 3);
1355  12   25:3    43               MP0C[ 1] := MP03;
1356  12   25:3    49               MP0C[ 2] := MP02;
1357  12   25:3    55               PRPICMB( MP04, MP0C);
1358  12   25:3    64               P010618( MP03, MP02, 1)
1359  12   25:2    69             END
1360  12   25:0    71         END;
1361  12   25:0    92
1362  12   25:0    92
1363  12   26:D     1       PROCEDURE P01061A( MP02 : INTEGER;
1364  12   26:D     2                          MP01 : CHAR);
1365  12   26:D     3
1366  12   26:0     0         BEGIN
```

```
1367  12  26:1   0          MVCURSOR( MP04, MP02 + MP02 + 1, 4);
1368  12  26:1   12         PRPICCH( MP04, MP01)
1369  12  26:0   16       END;
1370  12  26:0   32
1371  12  26:0   32
1372  12  27:D   1      PROCEDURE P01061B( MP01 : INTEGER);
1373  12  27:D   2
1374  12  27:0   0        BEGIN
1375  12  27:1   0          IF MP0B <> -1 THEN
1376  12  27:2   8             P01061A( MP0B, ' ');
1377  12  27:1   14         MP0B := MP01;
1378  12  27:1   18         P01061A( MP0B, '{')   (* 123 *)
1379  12  27:0   22       END;
1380  12  27:0   36
1381  12  27:0   36
1382  12  28:D   1      PROCEDURE P01061C( MP01 : INTEGER);
1383  12  28:D   2
1384  12  28:D   2        VAR
1385  12  28:D   2            MP02 : INTEGER;
1386  12  28:D   3
1387  12  28:0   0        BEGIN
1388  12  28:1   0          IF MP06 THEN
1389  12  28:2   5            BEGIN
1390  12  28:3   5              MP02 := MP0B + MP01;
1391  12  28:3   12             P010618( MP08, MP07, MP01);
1392  12  28:3   21             IF MP02 < 0 THEN
1393  12  28:4   26               BEGIN
1394  12  28:5   26                 P01061B( MP02 + 10);
1395  12  28:5   31                 P010619
1396  12  28:4   31               END
1397  12  28:3   33             ELSE
1398  12  28:4   35               IF MP02 > 9 THEN
1399  12  28:5   40                 BEGIN
1400  12  28:6   40                   P01061B( MP02 - 10);
1401  12  28:6   45                   P010619
1402  12  28:5   45                 END
1403  12  28:4   47               ELSE
1404  12  28:5   49                 P01061B( MP02)
1405  12  28:2   50           END
1406  12  28:0   52       END;
1407  12  28:0   64
1408  12  28:0   64
1409  12  20:0   0      BEGIN (* P010614 *)
1410  12  20:1   0        UNITREAD( DRIVE1, MP30, 256, KANABLCK);
1411  12  20:1   12       FILLCHAR( KANMP02, KANMP01 + 1, 0);
1412  12  20:1   24       P14MP01 := 0;
1413  12  20:1   27       MP05 := TRUE;
1414  12  20:1   30       MP06 := FALSE;
1415  12  20:1   33       MP09 := 130;  (* $82 *)
1416  12  20:1   38       MP0A := 158;  (* $1E *)
1417  12  20:1   43       MP0C := '  ';
1418  12  20:1   52       MP08 := CHR( 129);  (* $81 *)
1419  12  20:1   57       MP07 := CHR( 64);
1420  12  20:1   60       MP0B := -1;
1421  12  20:1   64       MP04 := NIL;
1422  12  20:1   67       MP0F := KANMP03^.HEAD.HCURSOR;  (* (3)  8, 0 RTBIT;#;^ *)
1423  12  20:1   77       IF MP0F + 23 > 39 THEN
```

```
1424  12   20:2    84          MP0F := 39 - 23;
1425  12   20:2    89
1426  12   20:1    89        REPEAT
1427  12   20:2    89          P14MP03 := KANMP03^.HEAD.HCURSOR;
1428  12   20:2    99          P14MP02 := KANMP03^.HEAD.VCURSOR;
1429  12   20:2   109          MVCURSOR( KANMP03, P14MP03, P14MP02);
1430  12   20:2   117          IF MP05 THEN
1431  12   20:3   120            PRPICCH( KANMP03, '_')
1432  12   20:2   124          ELSE
1433  12   20:3   129            PRPICCH( KANMP03, CHR( 127));
1434  12   20:2   136          GETKEY;
1435  12   20:2   139          MVCURSOR( KANMP03, P14MP03, P14MP02);
1436  12   20:2   147          PRPICCH( KANMP03, ' ');
1437  12   20:2   154          MVCURSOR( KANMP03, P14MP03, P14MP02);
1438  12   20:2   162
1439  12   20:2   162          IF (INCHAR = CHR( 8)) AND (P14MP01 > 0) THEN
1440  12   20:3   171            P010615
1441  12   20:2   171          ELSE IF BASE36 AND (INCHAR = CHR( 2)) THEN
1442  12   20:4   183            BEGIN
1443  12   20:5   183              MP06 := NOT MP06;
1444  12   20:5   187              IF MP06 THEN
1445  12   20:6   190                IF MP04 = NIL THEN
1446  12   20:7   195                  BEGIN
1447  12   20:8   195                    MP04 :=
1448  12   20:8   195                      GETWIN(KANMP03^.HEAD.HPOS + MP0F,
1449  12   20:8   205                             KANMP03^.HEAD.VPOS +
1450  12   20:8   213                               KANMP03^.HEAD.VCURSOR + 3,
1451  12   20:8   224                             23, 7, 30, TRUE);
1452  12   20:8   235
1453  12   20:8   235                    MVCURSOR( MP04, 0, 0);
1454  12   20:8   241                    DISP1LIN( MP04, 2);
1455  12   20:8   246                      (* INSERT DISK 'A' CONTAINING *)
1456  12   20:8   246                    DISP1LIN( MP04, 3);
1457  12   20:8   251                      (* SAVE GAME YOU WISH TO MOVE INTO *)
1458  12   20:8   251                    P01061B( 0);
1459  12   20:8   254                    P010619
1460  12   20:7   254                  END
1461  12   20:6   256                ELSE
1462  12   20:7   258                  UNPROWIN( MP04, TRUE)
1463  12   20:5   260              ELSE
1464  12   20:6   265                  PROTWIN( MP04, TRUE);
1465  12   20:4   270            END (* IF BASE36 *)
1466  12   20:4   270
1467  12   20:3   270          ELSE IF INCHAR = CHR( 1) THEN
1468  12   20:5   277            BEGIN
1469  12   20:6   277              MP05 := NOT MP05;
1470  12   20:6   281              IF MP05 THEN
1471  12   20:7   284                BEGIN
1472  12   20:8   284                  MP09 := 130;
1473  12   20:8   289                  MP0A := 158
1474  12   20:7   289                END
1475  12   20:6   294              ELSE
1476  12   20:7   296                BEGIN
1477  12   20:8   296                  MP09 := 131;
1478  12   20:8   301                  MP0A := 64
1479  12   20:7   301                END
1480  12   20:5   304            END
```

```
1481  12   20:5   304
1482  12   20:4   304          ELSE IF INCHAR = CHR( 11) THEN
1483  12   20:6   311              P01061C( -10)
1484  12   20:6   313
1485  12   20:5   313          ELSE IF INCHAR = CHR( 10) THEN
1486  12   20:7   322              P01061C( 10)
1487  12   20:7   323
1488  12   20:6   323          ELSE IF INCHAR = CHR( 12) THEN
1489  12   20:8   332              P01061C( 1)
1490  12   20:8   333
1491  12   20:7   333          ELSE IF INCHAR = CHR( 9) THEN
1492  12   20:9   342              P01061C( -1)
1493  12   20:9   344
1494  12   20:8   344          ELSE IF (INCHAR >= CHR( 166)) AND
1495  12   20:9   353                  (INCHAR <= CHR( 221)) AND
1496  12   20:9   359                  (P14MP03 < KANMP03^.HEAD.HSIZE - 3) AND
1497  12   20:9   372                  (P14MP01 < KANMP01 - 1) THEN
1498  12   20:0   382            BEGIN
1499  12   20:1   382              IF MP06 THEN
1500  12   20:2   385                BEGIN
1501  12   20:3   385                  MP0E := ORD( INCHAR) - 166;
1502  12   20:3   392                  MP0E := MP0E + MP0E + 128;
1503  12   20:3   401                  MP08 := MP30[ MP0E];
1504  12   20:3   407                  MP07 := MP30[ MP0E + 1];
1505  12   20:3   415                  P01061B( 0);
1506  12   20:3   418                  P010619
1507  12   20:2   418                END
1508  12   20:1   420              ELSE
1509  12   20:2   422                BEGIN
1510  12   20:3   422                  P14MP01 := P14MP01 + 1;
1511  12   20:3   427                  KANMP02[ P14MP01] := CHR( MP09);
1512  12   20:3   433                  MP0C[ 1] :=  CHR( MP09);
1513  12   20:3   438                  P14MP01 := P14MP01 + 1;
1514  12   20:3   443                  MP0E := ORD( INCHAR) - 166;
1515  12   20:3   450                  MP10[ P14MP01] := CHR( MP0E);
1516  12   20:3   455                  MP0E := ORD( MP30[ MP0E]);
1517  12   20:3   461                  IF MP0E > 127 THEN
1518  12   20:4   466                    BEGIN
1519  12   20:5   466                      P14MP01 := P14MP01 - 1;
1520  12   20:5   471                      INCHAR := CHR( MP0E - 128);
1521  12   20:5   478                      KANMP02[ P14MP01] := INCHAR;
1522  12   20:5   484                      MP10[ P14MP01] :=  CHR( 255);
1523  12   20:5   491                      PRPICCH( KANMP03, INCHAR)
1524  12   20:4   495                    END
1525  12   20:3   498                  ELSE
1526  12   20:4   500                    BEGIN
1527  12   20:5   500                      MP0E := MP0E + MP0A;
1528  12   20:5   505                      IF MP0E >= 127 THEN
1529  12   20:6   510                        MP0E := MP0E + 1;
1530  12   20:5   515                      INCHAR := CHR( MP0E);
1531  12   20:5   518                      KANMP02[ P14MP01] := INCHAR;
1532  12   20:5   524                      MP10[ P14MP01 - 1] := INCHAR;
1533  12   20:5   531                      MP0C[ 2] := INCHAR;
1534  12   20:5   536                      PRPICMB( KANMP03, MP0C)
1535  12   20:4   541                    END
1536  12   20:2   544                END
1537  12   20:0   544            END
```

```
1538  12  20:0  544
1539  12  20:9  544          ELSE IF INCHAR = CHR( 222) THEN
1540  12  20:1  553              P010617( 0, 1)
1541  12  20:1  555
1542  12  20:0  555          ELSE IF INCHAR = CHR( 223) THEN
1543  12  20:2  566              P010617( 1, 2)
1544  12  20:2  568
1545  12  20:1  568          ELSE IF (INCHAR > CHR( 31)) AND
1546  12  20:2  575                  (INCHAR <= CHR( 127)) AND
1547  12  20:2  579                  (P14MP03 < (KANMP03^.HEAD.HSIZE - 3)) AND
1548  12  20:2  592                  (P14MP01 < KANMP01) THEN
1549  12  20:3  600                BEGIN
1550  12  20:3  600
1551  12  20:4  600                  IF (INCHAR = CHR( 42)) OR
1552  12  20:4  603                     (INCHAR < CHR( 33)) OR
1553  12  20:4  607                     (INCHAR > CHR( 47)) THEN
1554  12  20:5  613                   BEGIN
1555  12  20:6  613                     P14MP01 := P14MP01 + 1;
1556  12  20:6  618                     KANMP02[ P14MP01] := INCHAR;
1557  12  20:6  624                     MP10[ P14MP01] := CHR( 255);
1558  12  20:6  631                     PRPICCH( KANMP03, INCHAR)
1559  12  20:5  635                   END
1560  12  20:5  638
1561  12  20:3  638                END
1562  12  20:3  638
1563  12  20:2  638          ELSE IF (INCHAR = CHR( 13)) OR
1564  12  20:3  643                  (INCHAR = CHR( 27)) THEN
1565  12  20:4  649              BEGIN
1566  12  20:5  649                IF MP06 THEN
1567  12  20:6  652                  BEGIN
1568  12  20:7  652                    IF (P14MP03  < KANMP03^.HEAD.HSIZE - 3) AND
1569  12  20:7  664                       (P14MP01 < KANMP01 - 1) THEN
1570  12  20:8  674                     BEGIN
1571  12  20:9  674                       P14MP01 := P14MP01 + 1;
1572  12  20:9  679                       KANMP02[ P14MP01] := MP08;
1573  12  20:9  685                       P14MP01 := P14MP01 + 1;
1574  12  20:9  690                       KANMP02[ P14MP01] := MP07;
1575  12  20:9  696                       MP10[ P14MP01] := CHR( 255);
1576  12  20:9  703                       MP0C[ 1] := MP08;
1577  12  20:9  708                       MP0C[ 2] := MP07;
1578  12  20:9  713                       PRPICMB( KANMP03, MP0C)
1579  12  20:8  718                     END
1580  12  20:6  721                  END
1581  12  20:5  721                ELSE
1582  12  20:6  723                  BEGIN
1583  12  20:7  723                    PRPICCH( KANMP03, ' ');
1584  12  20:7  730                    IF INCHAR = CHR( 27) THEN
1585  12  20:8  735                        WHILE P14MP01 > 0 DO
1586  12  20:9  740                          P010615;
1587  12  20:7  744                    KANMP02[ 0] := CHR( P14MP01);
1588  12  20:7  750                    IF MP04 <> NIL THEN
1589  12  20:8  755                      DELWIN( MP04, TRUE);
1590  12  20:7  761                    EXIT( KANJIREA)
1591  12  20:6  765                  END
1592  12  20:4  765              END
1593  12  20:4  765
1594  12  20:3  765          ELSE
```

```
1595  12   20:4   767            PRINTBEL
1596  12   20:4   767
1597  12   20:1   767        UNTIL FALSE;
1598  12   20:1   773
1599  12   20:0   773      END;  (* P010614 *)
1600  12   20:0   814
1601  12   20:0   814
1602  12    1:0     0    BEGIN  (* KANJIREA *)
1603  12    1:0     0
1604  12    1:1     0      IF KANMP03 = NIL THEN
1605  12    1:2     5        REPEAT
1606  12    1:3     5          BEGIN
1607  12    1:4     5            CASE KANMP01 OF
1608  12    1:4     8              0: INIT1;
1609  12    1:4    12              1: SOLICSM;  (* S)TART GAME; M)AKE SCENARIO *)
1610  12    1:4    16              2: BEGIN
1611  12    1:4    16                   (* MAKE SCENARIO *)
1612  12    1:6    16                   BASE09 := 1;
1613  12    1:6    19                   EXIT( KANJIREA);
1614  12    1:5    23                 END;
1615  12    1:4    25              3: P010611;  (* START GAME *)
1616  12    1:4    29              4: P010612;
1617  12    1:4    33              5: MOREWIZ;
1618  12    1:4    37            END
1619  12    1:3    56          END
1620  12    1:2    56        UNTIL FALSE
1621  12    1:1    56      ELSE
1622  12    1:2    61        P010614
1623  12    1:2    61
1624  12    1:0    61    END;  (* KANJIREA *)
1625  12    1:0    78
1626  12    1:0    78 (*$I WIZ4B:KANJIREA2 *)
1627  12    1:0    78
1627  12    1:0    78 (*$I WIZ4B:UTILITIE  *)
1628  12    1:0    78
1629  12    1:0    78
1630  13    1:D     1 SEGMENT PROCEDURE UTILITIE;  (* P010701 *)
1631  13    1:D     1
1632  13    1:D     1
1633  13    1:D     1  VAR
1634  13    1:D     1       UTILMP01 : INTEGER;
1635  13    1:D     2       UTILMP02 : INTEGER;
1636  13    1:D     3       UTILMP03 : INTEGER;
1637  13    1:D     4
1638  13    1:D     4
1639  13    2:D     1  PROCEDURE P010702;
1640  13    2:D     1
1641  13    2:D     1    VAR
1642  13    2:D     1         P02MP01 : INTEGER;
1643  13    2:D     2         P02MP02 : INTEGER;
1644  13    2:D     3         P02MP03 : INTEGER;
1645  13    2:D     4         BFIGHT  : BOOLEAN;
1646  13    2:D     5         UTILWIN : TWINDOWP;  (* MP05 *)
1647  13    2:D     6         MAZE    : TMAZE;
1648  13    2:D   453
1649  13    2:D   453
1650  13    3:D     1    PROCEDURE DISPMSG( MSGNUM : INTEGER;  (* P010703 *)
```

```
1651  13    3:D     2                          VPOS : INTEGER);
1652  13    3:D     3
1653  13    3:0     0       BEGIN  (* DISPMSG *)
1654  13    3:1     0         GETMSGTX( BASE47, MSGNUM);
1655  13    3:1     6         UTILWIN := GETWIN2( MBSTRLEN( BASE47), VPOS, 3, 31, TRUE);
1656  13    3:1    25         CENTSTR( UTILWIN, BASE47);
1657  13    3:1    33         DELWIN( UTILWIN, FALSE)
1658  13    3:0    37       END;   (* DISPMSG *)
1659  13    3:0    52
1660  13    3:0    52
1661  13    4:D     1     PROCEDURE SETFIGHT;  (* P010704 *)
1662  13    4:D     1
1663  13    4:D     1       VAR
1664  13    4:D     1           YPOS     : INTEGER;
1665  13    4:D     2           XPOS     : INTEGER;
1666  13    4:D     3           YPOS2    : INTEGER;
1667  13    4:D     4           XPOS2    : INTEGER;
1668  13    4:D     5           FIGHTXXX : ARRAY[ 0..19, 0..19] OF BOOLEAN;
1669  13    4:D   405                      (* NOT PACKED (!) *)
1670  13    4:D   405
1671  13    5:D     1       PROCEDURE FNDFIGHT; (* P010705 *)
1672  13    5:D     1
1673  13    5:D     1         VAR
1674  13    5:D     1             ALL20X20 : INTEGER;
1675  13    5:D     2
1676  13    5:0     0         BEGIN  (* FNDFIGHT *)
1677  13    5:1     0           XPOS := RAND MOD 20;
1678  13    5:1    10           YPOS := RAND MOD 20;
1679  13    5:1    20           FOR ALL20X20 := 1 TO 400 DO
1680  13    5:2    33             BEGIN
1681  13    5:3    33               IF MAZE.FIGHTS[ XPOS][ YPOS] = 1 THEN
1682  13    5:4    53                 BEGIN
1683  13    5:5    53                   IF NOT FIGHTXXX[ XPOS][ YPOS] THEN
1684  13    5:6    70                     BEGIN
1685  13    5:7    70                       BFIGHT := TRUE;
1686  13    5:7    74                       EXIT( FNDFIGHT)
1687  13    5:6    78                     END
1688  13    5:4    78                 END;
1689  13    5:4    78
1690  13    5:3    78               XPOS := XPOS + 1;
1691  13    5:3    86               IF XPOS = 20 THEN
1692  13    5:4    93                 BEGIN
1693  13    5:5    93                   XPOS := XPOS - 20;
1694  13    5:5   101                   YPOS := YPOS + 1;
1695  13    5:5   109                   IF YPOS = 20 THEN
1696  13    5:6   116                     YPOS := YPOS - 20
1697  13    5:4   119                 END;
1698  13    5:4   124
1699  13    5:2   124             END;
1700  13    5:2   131
1701  13    5:1   131           BFIGHT := FALSE
1702  13    5:1   131
1703  13    5:0   131         END;   (* FNDFIGHT *)
1704  13    5:0   150
1705  13    5:0   150
1706  13    6:D     1       PROCEDURE SETFIGH2( XPOSORIG : INTEGER;  (* P010706 *)
1707  13    6:D     2                           YPOSORIG : INTEGER);
```

```
1708  13    6:D     3
1709  13    6:D     3         VAR
1710  13    6:D     3              PREVLOCI : INTEGER;
1711  13    6:D     4              CURRLOCI : INTEGER;
1712  13    6:D     5              DIST2GO  : INTEGER;
1713  13    6:D     6              FROM     : INTEGER;
1714  13    6:D     7              PREVY    : INTEGER;
1715  13    6:D     8              PREVX    : INTEGER;
1716  13    6:D     9
1717  13    6:D     9              DISTTAB  : STRING[ 100];
1718  13    6:D    60              FROMTAB  : STRING[ 100];
1719  13    6:D   111              PREVYTAB : STRING[ 100];
1720  13    6:D   162              PREVXTAB : STRING[ 100];
1721  13    6:D   213
1722  13    6:D   213
1723  13    7:D     1         PROCEDURE CHECKLOC( NEWX    : INTEGER;  (* P010707 *)
1724  13    7:D     2                             NEWY    : INTEGER;
1725  13    7:D     3                             WALL    : TWALL;
1726  13    7:D     4                             FROMDIR : INTEGER;
1727  13    7:D     5                             DIST2GO : INTEGER);
1728  13    7:D     6
1729  13    7:0     0           BEGIN  (* CHECKLOC *)
1730  13    7:1     0             IF WALL <> OPEN THEN
1731  13    7:2     5               EXIT( CHECKLOC);
1732  13    7:1     9             IF DIST2GO = 1 THEN  (* STEP FROM 4 DOWN TO 1 AWAY *)
1733  13    7:2    14               EXIT( CHECKLOC);
1734  13    7:1    18             NEWX := (NEWX + 20) MOD 20;
1735  13    7:1    25             NEWY := (NEWY + 20) MOD 20;
1736  13    7:1    32             IF (MAZE.FIGHTS[ NEWX][ NEWY] <> 1) OR
1737  13    7:1    46                (FIGHTXXX[ NEWX][ NEWY]) THEN
1738  13    7:2    59               EXIT( CHECKLOC);
1739  13    7:2    63
1740  13    7:1    63             PREVXTAB[ CURRLOCI] := CHR( NEWX);
1741  13    7:1    72             PREVYTAB[ CURRLOCI] := CHR( NEWY);
1742  13    7:1    80             FROMTAB[  CURRLOCI] := CHR( FROMDIR);
1743  13    7:1    88             DISTTAB[  CURRLOCI] := CHR( DIST2GO - 1);
1744  13    7:1    98             CURRLOCI := (CURRLOCI + 1) MOD 100
1745  13    7:0   103           END;   (* CHECKLOC *)
1746  13    7:0   120
1747  13    7:0   120
1748  13    6:0     0         BEGIN  (* SETFIGH2 *)
1749  13    6:0     0
1750  13    6:0     0           (* WE'VE FOUND A LOCATION FOR A FIGHT *)
1751  13    6:0     0           (* NOW SET UP SURROUNDING LOCATIONS   *)
1752  13    6:0     0           (* NEVER MORE THAN 3 STEPS AWAY FROM ORIGIN *)
1753  13    6:0     0
1754  13    6:1     0           CURRLOCI := 0;
1755  13    6:1     3           PREVLOCI := 0;
1756  13    6:1     6           PREVXTAB[ CURRLOCI] := CHR( XPOSORIG);
1757  13    6:1    12           PREVYTAB[ CURRLOCI] := CHR( YPOSORIG);
1758  13    6:1    17           FROMTAB[  CURRLOCI] := CHR( -1);
1759  13    6:1    23           DISTTAB[  CURRLOCI] := CHR( 4);
1760  13    6:1    28           CURRLOCI := 1;
1761  13    6:1    31           WHILE CURRLOCI <> PREVLOCI DO
1762  13    6:2    36             BEGIN
1763  13    6:3    36               PREVX   := ORD( PREVXTAB[ PREVLOCI]);
1764  13    6:3    43               PREVY   := ORD( PREVYTAB[ PREVLOCI]);
```

```
1765  13   6:3   49              FROM    := ORD( FROMTAB[ PREVLOCI]);
1766  13   6:3   55              DIST2GO := ORD( DISTTAB[ PREVLOCI]);
1767  13   6:3   61              PREVLOCI := (PREVLOCI + 1) MOD 100;
1768  13   6:3   68              FIGHTXXX[ PREVX][ PREVY] := TRUE;
1769  13   6:3   79              IF FROM <> 3 THEN
1770  13   6:4   84                CHECKLOC( PREVX + 1, PREVY, MAZE.E[ PREVX][ PREVY],
1771  13   6:4   99                          1, DIST2GO);
1772  13   6:3  103              IF FROM <> 1 THEN
1773  13   6:4  108                CHECKLOC( PREVX - 1, PREVY, MAZE.W[ PREVX][ PREVY],
1774  13   6:4  123                          3, DIST2GO);
1775  13   6:3  127              IF FROM <> 0 THEN
1776  13   6:4  132                CHECKLOC( PREVX, PREVY - 1, MAZE.S[ PREVX][ PREVY],
1777  13   6:4  147                          2, DIST2GO);
1778  13   6:3  151              IF FROM <> 2 THEN
1779  13   6:4  156                CHECKLOC( PREVX, PREVY + 1, MAZE.N[ PREVX][ PREVY],
1780  13   6:4  172                          0, DIST2GO)
1781  13   6:2  174            END;
1782  13   6:0  178          END;   (* SETFIGH2 *)
1783  13   6:0  194
1784  13   6:0  194
1785  13   4:0    0        BEGIN  (* SETFIGHT *)
1786  13   4:1    0          DISPMSG( 211, 12);  (* LAYING OUT PATROL AREAS! *)
1787  13   4:1    6          FIZZLES := 0;
1788  13   4:1    9          FILLCHAR( FIGHTXXX, 800, 0);
1789  13   4:1   18          BFIGHT := TRUE;
1790  13   4:1   22          FOR XPOS2 := 1 TO 16 DO
1791  13   4:2   36            BEGIN
1792  13   4:3   36              IF BFIGHT THEN
1793  13   4:4   41                BEGIN
1794  13   4:5   41                  FNDFIGHT;
1795  13   4:5   43                  IF BFIGHT THEN
1796  13   4:6   48                    SETFIGH2( XPOS, YPOS)
1797  13   4:4   50                END;
1798  13   4:2   52            END;
1799  13   4:2   59
1800  13   4:1   59          DISPMSG( 212, 18);  (* SETTING UP THE TRAPS! *)
1801  13   4:1   65
1802  13   4:1   65          FOR XPOS2 := 0 TO 19 DO
1803  13   4:2   79            BEGIN
1804  13   4:3   79              FOR YPOS2 := 0 TO 19 DO
1805  13   4:4   93                BEGIN
1806  13   4:5   93                  IF MAZE.SQRETYPE[ MAZE.SQREXTRA[ XPOS2][ YPOS2]]
1807  13   4:5  112                     = ENCOUNTE THEN
1808  13   4:6  117                    BEGIN
1809  13   4:7  117                      IF NOT FIGHTXXX[ XPOS2][ YPOS2] THEN
1810  13   4:8  129                        SETFIGH2( XPOS2, YPOS2)
1811  13   4:6  131                    END
1812  13   4:5  133                  ELSE
1813  13   4:6  135                    BEGIN
1814  13   4:7  135                      IF MAZE.SQRETYPE[ MAZE.SQREXTRA[ XPOS2][ YPOS2]]
1815  13   4:7  154                         = SCNMSG THEN
1816  13   4:8  159                        BEGIN
1817  13   4:9  159                          IF MAZE.AUX2[ MAZE.SQREXTRA[ XPOS2][ YPOS2]]
1818  13   4:9  177                             = 600 THEN
1819  13   4:0  184                            IF NOT FIGHTXXX[ XPOS2][ YPOS2] THEN
1820  13   4:1  196                              SETFIGH2( XPOS2, YPOS2)
1821  13   4:8  198                        END;
```

```
1822  13    4:6   200                  END;
1823  13    4:4   200              END;
1824  13    4:2   207          END;
1825  13    4:2   214
1826  13    4:1   214          FOR XPOS2 := 0 TO 19 DO
1827  13    4:2   228            FOR YPOS2 := 0 TO 19 DO
1828  13    4:3   242              FIGHTMAP[ XPOS2][ YPOS2] :=  FIGHTXXX[ XPOS2][ YPOS2];
1829  13    4:3   276
1830  13    4:0   276       END;   (* SETFIGHT *)
1831  13    4:0   300
1832  13    4:0   300
1833  13    8:D     1     PROCEDURE INIT;  (* P010708 *)
1834  13    8:D     1
1835  13    8:D     1       VAR
1836  13    8:D     1            UNUSED  : INTEGER;
1837  13    8:D     2            UNUSED2 : INTEGER;
1838  13    8:D     3            INDX    : INTEGER;
1839  13    8:D     4            LEVCHK  : BOOLEAN;
1840  13    8:D     5
1841  13    8:0     0       BEGIN  (* INIT *)
1842  13    8:1     0         LEVCHK := (MAZELEV = 1) OR (MAZELEV >= 12); (* MAZELEV STARTS AT 11 *)
1843  13    8:1     9         DISPMSG( 210, 6);           (* SUMMONING THE GUARDIANS! *)
1844  13    8:1    15         FOR INDX := 0 TO 511 DO
1845  13    8:2    28           BEGIN
1846  13    8:3    28             BASE814[ INDX] := 0;
1847  13    8:3    37             BASE87B[ INDX] := FALSE
1848  13    8:2    44           END;
1849  13    8:1    53         IF LEVCHK THEN
1850  13    8:2    56           BEGIN
1851  13    8:3    56             BASE814[ 0] := 5;
1852  13    8:3    65             BASE87B[ 0] := TRUE
1853  13    8:2    72           END
1854  13    8:0    74       END;   (* INIT *)
1855  13    8:0    88
1856  13    8:0    88
1857  13    9:D     1     PROCEDURE EXT2SHOP;  (* P010709 *)
1858  13    9:D     1
1859  13    9:0     0       BEGIN  (* EXT2SHOP *)
1860  13    9:1     0         BASE13 := 8;
1861  13    9:1     3         EXIT( UTILITIE)
1862  13    9:0     7       END;   (* EXT2SHOP *)
1863  13    9:0    20
1864  13    9:0    20
1865  13   10:D     1     PROCEDURE TICKETS;  (* P01070A *)
1866  13   10:D     1
1867  13   10:D     1       VAR
1868  13   10:D     1            QUADCNT : INTEGER;
1869  13   10:D     2            VALCODE : INTEGER;
1870  13   10:D     3            MSG3R   : INTEGER;
1871  13   10:D     4            MSG2R   : INTEGER;
1872  13   10:D     5            MSG1R   : INTEGER;
1873  13   10:D     6            MSG3L   : INTEGER;
1874  13   10:D     7            MSG2L   : INTEGER;
1875  13   10:D     8            MSG1L   : INTEGER;
1876  13   10:D     9            MSG3    : INTEGER;
1877  13   10:D    10            MSG2    : INTEGER;
1878  13   10:D    11            MSG1    : INTEGER;
```

```
1879  13   10:D    12
1880  13   10:D    12
1881  13   11:D     3       FUNCTION STR2NUM( VAR NUMSTR : TLONGSTR) : INTEGER;  (* P01070B *)
1882  13   11:D     4
1883  13   11:D     4         VAR
1884  13   11:D     4             RESULT : INTEGER;
1885  13   11:D     5             IPOS   : INTEGER;
1886  13   11:D     6
1887  13   11:0     0         BEGIN  (* STR2NUM *)
1888  13   11:1     0           STR2NUM := -1;
1889  13   11:1     4           RESULT := 0;
1890  13   11:1     7           FOR IPOS := 1 TO LENGTH( NUMSTR) DO
1891  13   11:2    20             IF (NUMSTR[ IPOS] >= '0') AND (NUMSTR[ IPOS] <= '9') THEN
1892  13   11:3    33               RESULT := 10 * RESULT + ORD( NUMSTR[ IPOS]) - ORD( '0')
1893  13   11:2    41             ELSE
1894  13   11:3    46               EXIT( STR2NUM);
1895  13   11:1    57           STR2NUM := RESULT
1896  13   11:0    57         END;   (* STR2NUM *)
1897  13   11:0    74
1898  13   11:0    74
1899  13   12:D     1       PROCEDURE GET2NUMS(    MSGOFF   : INTEGER;  (* P01070C *)
1900  13   12:D     2                              VAR FIRSTNUM : INTEGER;
1901  13   12:D     3                              VAR SECONDNM : INTEGER);
1902  13   12:D     4
1903  13   12:0     0         BEGIN  (* GET2NUMS *)
1904  13   12:1     0           GETMSGTX( BASE47, 28001 + MSGOFF);  (* SPECIAL 28000 +  *)
1905  13   12:1    10           MOVELEFT( BASE47[ 1], FIRSTNUM, 2);
1906  13   12:1    18           MOVELEFT( BASE47[ 3], SECONDNM, 2)
1907  13   12:0    26         END;   (* GET2NUMS *)
1908  13   12:0    38
1909  13   12:0    38
1910  13   13:D     3       FUNCTION MULTMOD( NUM1 : INTEGER;  (* P01070D *)
1911  13   13:D     4                         NUM2 : INTEGER) : INTEGER;
1912  13   13:D     5
1913  13   13:D     5         (* MULTIPLY 2 NUMBERS, AND MOD BY 10000 *)
1914  13   13:D     5
1915  13   13:D     5         VAR
1916  13   13:D     5             ANSWER : INTEGER;
1917  13   13:D     6
1918  13   13:0     0         BEGIN  (* MULTMOD *)
1919  13   13:1     0           ANSWER := 0;
1920  13   13:1     3           WHILE NUM2 > 0 DO        (* MULTIPLY BY SUCCESSIVE ADDITIONS *)
1921  13   13:2     8             BEGIN
1922  13   13:3     8               ANSWER := (ANSWER + NUM1) MOD 10000;
1923  13   13:3    17               NUM2 := NUM2 - 1
1924  13   13:2    18             END;
1925  13   13:1    24           MULTMOD := ANSWER
1926  13   13:0    24         END;   (* MULTMOD *)
1927  13   13:0    42
1928  13   13:0    42
1929  13   10:0     0       BEGIN  (* TICKETS *)
1930  13   10:1     0         GETMSGTX( BASE47, 28000);           (* SPECIAL 28000 MSG!     *)
1931  13   10:1     8         MOVELEFT( BASE47[  1], QUADCNT, 2); (* 28000:  $02 $3C $00     *)
1932  13   10:1    17                                             (* SEE HUFFMAN1TO32766.TXT *)
1933  13   10:1    17                                             (* $3C = 60.               *)
1934  13   10:1    17         QUADCNT := QUADCNT DIV 3;           (* 60 DIV 3 = 20.          *)
1935  13   10:1    22         MSG1 := RAND MOD QUADCNT;  (* 0..19 *)
```

```
1936  13   10:1    31         MSG2 := RAND MOD QUADCNT;  (* 0..19 *)
1937  13   10:1    40         MSG3 := RAND MOD QUADCNT;  (* 0..19 *)
1938  13   10:1    49         GET2NUMS(     3 * MSG1, MSG1L, MSG1R);  (* MSG#, [1], [2] RETURNED *)
1939  13   10:1    58         GET2NUMS( 1 + 3 * MSG2, MSG2L, MSG2R);
1940  13   10:1    69         GET2NUMS( 2 + 3 * MSG3, MSG3L, MSG3R);
1941  13   10:1    80         VALCODE := 1000 +
1942  13   10:1    83           (MULTMOD( MSG1R, 11) + MULTMOD( MSG2R, 7) + MSG3R) MOD 9000;
1943  13   10:1   105
1944  13   10:1   105         REPEAT
1945  13   10:2   105           CLEARWIN( MAINWIN, TRUE);
1946  13   10:2   111           MVCURSOR( MAINWIN, 0, 4);
1947  13   10:2   118           DISP1LIN( MAINWIN, 550);  (* TICKETS...TICKETS, PLEASE!        *)
1948  13   10:2   126           DISP1LIN( MAINWIN, 551);  (* ENTER THE VALIDATION CODE OF THIS *)
1949  13   10:2   134           DISP1LIN( MAINWIN, 552);  (* MORDORCHARGE CARD TO CONTINUE     *)
1950  13   10:2   142           MVCURSOR( MAINWIN, 9, 8);
1951  13   10:2   149           PRINTNUM( MAINWIN, MSG1L, 5);
1952  13   10:2   156           PRINTNUM( MAINWIN, MSG2L, 5);
1953  13   10:2   163           PRINTNUM( MAINWIN, MSG3L, 5);
1954  13   10:2   170           PRINTCR( MAINWIN);
1955  13   10:2   175           PRINTCR( MAINWIN);
1956  13   10:2   180           DISP1MSG( MAINWIN, 553);   (* VALIDATION CODE > *)
1957  13   10:2   188           SOLICIT( MAINWIN, BASE47, 40);
1958  13   10:2   196           IF VALCODE = STR2NUM( BASE47) THEN
1959  13   10:3   206             EXIT( TICKETS);
1960  13   10:2   210           PRINTCR( MAINWIN);
1961  13   10:2   215           PRINTCR( MAINWIN);
1962  13   10:2   220           DISP1LIN( MAINWIN, 554);  (* THE NUMBER YOU HAVE ENTERED IS NOT *)
1963  13   10:2   228           DISP1LIN( MAINWIN, 555);  (* CORRECT.  PRESS ~ TO TRY AGAIN.    *)
1964  13   10:2   236           GETCR
1965  13   10:1   236         UNTIL FALSE
1966  13   10:1   239
1967  13   10:0   239       END;   (* TICKETS *)
1968  13   10:0   256
1969  13   10:0   256
1970  13    2:0     0     BEGIN  (* P010702 *)
1971  13    2:1     0       TREBORX := (RAND MOD 100) - 50;
1972  13    2:1    12       TREBORY := (RAND MOD 100) - 50;
1973  13    2:1    24       TREBMSG := 2500;  (* W..E..R..D..N..A.., AND THE OTHERS *)
1974  13    2:1    30       CHGMSGP2 := 16;
1975  13    2:1    34       IF MAZELEV = 0 THEN  (* MAZELEV STARTS AT 11 *)
1976  13    2:2    39         EXT2SHOP;
1977  13    2:2    41
1978  13    2:1    41       IF MAZELEV < 0 THEN
1979  13    2:2    46         BEGIN
1980  13    2:3    46           MAZELEV := -(1 * MAZELEV);
1981  13    2:3    52           BASE13 := 11
1982  13    2:2    52         END
1983  13    2:1    55       ELSE
1984  13    2:2    57         BASE13 := 5;
1985  13    2:2    60
1986  13    2:1    60       IF CHARACTR[ 6].CHARLEV < 12 - MAZELEV THEN
1987  13    2:2    74         IF CHARACTR[ 6].CHARLEV <> 0 THEN
1988  13    2:3    86           IF MAZELEV > 4 THEN
1989  13    2:4    91             TICKETS;
1990  13    2:4    93
1991  13    2:1    93       MOVELEFT( BASE55B[ GETREC( 1, MAZELEV - 1, SIZEOF( MAZE))],
1992  13    2:1   108                 MAZE,
```

```
1993  13    2:1   111                SIZEOF( MAZE));
1994  13    2:1   116
1995  13    2:1   116        CLEARWIN( MAINWIN, TRUE);
1996  13    2:1   122        INIT;
1997  13    2:1   124        SETFIGHT;
1998  13    2:1   126        CLEARWIN( MAINWIN, TRUE);
1999  13    2:1   132        EXIT( UTILITIE)
2000  13    2:1   136
2001  13    2:0   136      END;   (* P010702 *)
2002  13    2:0   148
2003  13    2:0   148
2004  13   14:D     1    PROCEDURE P01070E;
2005  13   15:D     1      PROCEDURE P01070F;
2006  13   16:D     1        PROCEDURE P010710;
2007  13   17:D     1          PROCEDURE P010711;
2008  13   17:0     0            BEGIN  (* P010711 *)
2009  13   17:0     0            END;   (* P010711 *)
2010  13   18:D     1          PROCEDURE P010712;
2011  13   18:0     0            BEGIN  (* P010712 *)
2012  13   18:0     0            END;   (* P010712 *)
2013  13   19:D     1          PROCEDURE P010713;
2014  13   19:0     0            BEGIN  (* P010713 *)
2015  13   19:0     0            END;   (* P010713 *)
2016  13   20:D     1          PROCEDURE P010714;
2017  13   20:0     0            BEGIN  (* P010714 *)
2018  13   20:0     0            END;   (* P010714 *)
2019  13   21:D     1          PROCEDURE P010715;
2020  13   21:0     0            BEGIN  (* P010715 *)
2021  13   21:0     0            END;   (* P010715 *)
2022  13   22:D     1          PROCEDURE P010716;
2023  13   22:0     0            BEGIN  (* P010716 *)
2024  13   22:0     0            END;   (* P010716 *)
2025  13   23:D     1          PROCEDURE P010717;
2026  13   23:0     0            BEGIN  (* P010717 *)
2027  13   23:0     0            END;   (* P010717 *)
2028  13   16:0     0          BEGIN  (* P010710 *)
2029  13   16:0     0          END;   (* P010710 *)
2030  13   15:0     0        BEGIN  (* P01070F *)
2031  13   15:0     0        END;   (* P01070F *)
2032  13   24:D     1      PROCEDURE P010718;
2033  13   24:0     0        BEGIN  (* P010718 *)
2034  13   24:0     0        END;   (* P010718 *)
2035  13   25:D     1      PROCEDURE P010719;
2036  13   25:0     0        BEGIN  (* P010719 *)
2037  13   25:0     0        END;   (* P010719 *)
2038  13   26:D     1      PROCEDURE P01071A;
2039  13   26:0     0        BEGIN  (* P01071A *)
2040  13   26:0     0        END;   (* P01071A *)
2041  13   27:D     1      PROCEDURE P01071B;
2042  13   28:D     1        PROCEDURE P01071C;
2043  13   28:0     0          BEGIN  (* P01071C *)
2044  13   28:0     0          END;   (* P01071C *)
2045  13   27:0     0        BEGIN  (* P010718 *)
2046  13   27:0     0        END;   (* P010718 *)
2047  13   29:D     1      PROCEDURE P01071D;
2048  13   29:0     0        BEGIN  (* P01071D *)
2049  13   29:0     0        END;   (* P01071D *)
```

```
2050  13   30:D     1     PROCEDURE P01071E;
2051  13   30:0     0       BEGIN  (* P01071E *)
2052  13   30:0     0       END;   (* P01071E *)
2053  13   14:0     0     BEGIN  (* P01070E *)
2054  13   14:0     0     END;   (* P01070E *)
2055  13   31:D     1   PROCEDURE P01071F;
2056  13   31:0     0     BEGIN  (* P01071F *)
2057  13   31:0     0     END;   (* P01071F *)
2058  13   32:D     1   PROCEDURE P010720( MP01 : INTEGER);
2059  13   32:0     0     BEGIN  (* P010720 *)
2060  13   32:0     0     END;   (* P010720 *)
2061  13   32:0    12
2062  13   32:0    12
2063  13    1:0     0   BEGIN (* UTILITIE MAIN LINE P010701 *)
2064  13    1:0     0
2065  13    1:1     0     CASE BASE13 OF
2066  13    1:1     4            7: P010702;
2067  13    1:1     8       11, 18: P01071F;
2068  13    1:1    12           12: P010720( 6);
2069  13    1:1    17     END
2070  13    1:1    48
2071  13    1:0    48   END;  (* UTILITIE MAIN LINE P010701 *)
2072  13    1:0    60
2073  13    1:0    60 (*$I WIZ4B:UTILITIE  *)
2074  13    1:0    60
2074  13    1:0    60 (*$I WIZ4B:SHOPS *)
2075  13    1:0    60
2076  14    1:D     1 SEGMENT PROCEDURE SHOPS;  (* P010801 *)
2077  14    1:D     1
2078  14    2:D     1   PROCEDURE WERDNA00( MSGNUM1 : INTEGER;  (* P010802 *)
2079  14    2:D     2                        MSGNUM2 : INTEGER);
2080  14    2:D     3
2081  14    2:D     3       (* WERDNA00( 0, 510)    "OH NO!!! THE NIGHTMARES ...                *)
2082  14    2:D     3       (* WERDNA00( 520, 530)  "AT LAST YOU AWAKEN" "BUT FIRST YOU HAVE TO" *)
2083  14    2:D     3
2084  14    2:D     3
2085  14    2:D     3     VAR
2086  14    2:D     3          UNUSED  : INTEGER;
2087  14    2:D     4          LINECNT : INTEGER;
2088  14    2:D     5          MSGLEN  : INTEGER;
2089  14    2:D     6
2090  14    2:D     6
2091  14    3:D     1     PROCEDURE PIC2MAIN( CHAR1 : INTEGER;  (* P010803 *)
2092  14    3:D     2                         CHAR2 : INTEGER;
2093  14    3:D     3                         CHAR3 : INTEGER;
2094  14    3:D     4                         CHAR4 : INTEGER);
2095  14    3:D     5
2096  14    3:D     5       (* INDEXES INTO 200.MONSTERS *)
2097  14    3:D     5
2098  14    3:D     5       (* 41, 42, 43, 44 == WERDNA ON PLATFORM *)
2099  14    3:D     5       (* 45, 46, 47, 48 == TREBOR'S GHOST     *)
2100  14    3:D     5
2101  14    3:D     5       VAR
2102  14    3:D     5            INDX     : INTEGER;
2103  14    3:D     6            CURRLINE : INTEGER;
2104  14    3:D     7            MBHSIZE  : INTEGER;
2105  14    3:D     8            CHAR2XB  : INTEGER;
```

```
2106  14   3:D    9            CHAR2XA  : INTEGER;
2107  14   3:D   10            UNUSED   : INTEGER;
2108  14   3:D   11            CHARX    : INTEGER;
2109  14   3:D   12            CHAR2XPS : INTEGER;
2110  14   3:D   13            LINEX    : INTEGER;
2111  14   3:D   14            MBSTR    : STRING[ 78];  (* MULTI-BYTE STRING *)
2112  14   3:D   54
2113  14   3:D   54
2114  14   3:0    0        BEGIN  (* PIC2MAIN *)
2115  14   3:1    0          BASE47[ 0] := CHR( CHAR1);
2116  14   3:1    5          BASE47[ 1] := CHR( CHAR2);
2117  14   3:1   10          BASE47[ 2] := CHR( CHAR3);
2118  14   3:1   15          BASE47[ 3] := CHR( CHAR4);
2119  14   3:1   20
2120  14   3:1   20          (* PREPARE WINDOW BUFFER FOR MB CHARS (#$01, #$20) *)
2121  14   3:1   20          UNITWRITE( BASE12, BASE47, 17, ORD( MAINWIN) );
2122  14   3:1   31
2123  14   3:1   31          (*
2124  14   3:1   31             DISPLAY 'PICTURE' ON MAIN WINDOW:
2125  14   3:1   31               WERDNA DEAD LAYING ON PLATFORM
2126  14   3:1   31                  -OR-
2127  14   3:1   31               GHOST OF TREBOR
2128  14   3:1   31          *)
2129  14   3:1   31          UNITWRITE( BASE12, BASE47, 13, ORD( MAINWIN) );
2130  14   3:1   42
2131  14   3:1   42          MBHSIZE := 2 * (MAINWIN^.HEAD.HSIZE - 2);  (* EXCLUDE 'FRAME' *)
2132  14   3:1   55          CURRLINE := 9 * MBHSIZE;
2133  14   3:1   60
2134  14   3:1   60          FOR LINEX := 9 TO 13 DO
2135  14   3:2   72            BEGIN
2136  14   3:3   72              CHAR2XPS := 0;
2137  14   3:3   75              FOR CHARX := 0 TO MAINWIN^.HEAD.HSIZE - 3 DO
2138  14   3:4   95                BEGIN
2139  14   3:5   95                  CHAR2XA := CURRLINE + CHARX + CHARX;
2140  14   3:5  102                  CHAR2XB := CHAR2XA + 1;
2141  14   3:5  107                  IF        MAINWIN^.DATA[ CHAR2XB] <  CHR( 128) THEN
2142  14   3:6  119                    IF      MAINWIN^.DATA[ CHAR2XB] <> CHR( 0)   THEN
2143  14   3:7  129                       IF   MAINWIN^.DATA[ CHAR2XB] <> ' '       THEN
2144  14   3:8  139                         IF (NOT BAPPLE) OR
2145  14   3:8  143                            (MAINWIN^.DATA[ CHAR2XB] <> CHR( 1))  THEN
2146  14   3:9  154                           BEGIN
2147  14   3:0  154                             MBSTR[ CHAR2XPS]     := MAINWIN^.DATA[ CHAR2XA];
2148  14   3:0  164                             MBSTR[ CHAR2XPS + 1] := MAINWIN^.DATA[ CHAR2XB];
2149  14   3:0  176                             CHAR2XPS := CHAR2XPS + 2
2150  14   3:9  177                           END;
2151  14   3:4  181                END;
2152  14   3:4  188
2153  14   3:3  188              FOR INDX := 0 TO (MBHSIZE DIV 2) - 1 DO
2154  14   3:4  204                BEGIN
2155  14   3:5  204                  MAINWIN^.DATA[ CURRLINE + INDX + INDX] := ' ';
2156  14   3:5  215                  MAINWIN^.DATA[ CURRLINE + INDX + INDX + 1] := CHR( 0)
2157  14   3:4  227                END;
2158  14   3:4  235
2159  14   3:4  235              (* 'CENTER' THE PICTURE IN THE WINDOW *)
2160  14   3:3  235              MOVELEFT( MBSTR,
2161  14   3:3  238                        MAINWIN^.DATA[
2162  14   3:3  242                          (CURRLINE +
```

```
2163  14   3:3   243                          (2 * (MBHSIZE DIV 4))) - (2 * (CHAR2XPS DIV 4))],
2164  14   3:3   255                            CHAR2XPS
2165  14   3:3   255                          );
2166  14   3:3   258              CURRLINE := CURRLINE + MBHSIZE
2167  14   3:2   259            END;
2168  14   3:1   270          UNITWRITE( BASE12, MAINWIN^, 3, 0)  (* DISPLAY MAIN WINDOW *)
2169  14   3:0   280        END;   (* PIC2MAIN *)
2170  14   3:0   300
2171  14   3:0   300
2172  14   2:0     0      BEGIN (* WERDNA00 *)
2173  14   2:1     0        IF (MAZEX = TREBORX) AND (MAZEY = TREBORY) THEN
2174  14   2:2    13          BEGIN
2175  14   2:3    13            CLEARWIN( MAINWIN, TRUE);
2176  14   2:3    19            PIC2MAIN( 45, 46, 47, 48);  (* GHOST OF TREBOR *)
2177  14   2:3    25            FOR MSGLEN := 0 TO 5 DO
2178  14   2:3    36              (* CHECK KEYBOARD FOR INPUT.  WAIT A BIT.  UPDATE RNG *)
2179  14   2:4    36              PAUSE2
2180  14   2:2    36          END;
2181  14   2:1    46        CLEARWIN( MAINWIN, TRUE);
2182  14   2:1    52        PIC2MAIN( 41, 42, 43, 44);      (* DEAD WERDNA LAYING ON PLATFORM *)
2183  14   2:1    58
2184  14   2:1    58        IF MSGNUM1 <> 0 THEN            (* AT LAST YOU AWAKEN *)
2185  14   2:2    63          BEGIN
2186  14   2:3    63            MSGLEN := 0;
2187  14   2:3    66            GETMSGTX( BASE47, MSGNUM1);
2188  14   2:3    72            WHILE BASE47 <> '**ERR**' DO
2189  14   2:4    88              BEGIN
2190  14   2:5    88                MVCURSOR( MAINWIN, 0, MSGLEN);
2191  14   2:5    95                CENTSTR(  MAINWIN, BASE47);
2192  14   2:5   102                MSGLEN := MSGLEN + 1;
2193  14   2:5   107                GETMSGTX( BASE47, MSGNUM1 + MSGLEN)
2194  14   2:4   112              END
2195  14   2:2   115          END;
2196  14   2:2   117
2197  14   2:1   117        IF MSGNUM2 <> 0 THEN           (* ...BUT FIRST YOU HAVE TO REGAIN...  *)
2198  14   2:2   122          BEGIN                        (* -OR- "OH NO!!! THE NIGHTMARES AND...*)
2199  14   2:3   122            MSGLEN := 0;
2200  14   2:3   125              GETMSGTX( BASE47, MSGNUM2);
2201  14   2:3   131              WHILE BASE47 <> '**ERR**' DO
2202  14   2:4   147                BEGIN
2203  14   2:5   147                  MSGLEN := MSGLEN + 1;
2204  14   2:5   152                  GETMSGTX( BASE47, MSGNUM2 + MSGLEN)
2205  14   2:4   157                END;
2206  14   2:3   162              FOR LINECNT := 0 TO MSGLEN - 1 DO
2207  14   2:4   175                BEGIN
2208  14   2:5   175                  MVCURSOR( MAINWIN, 0,
2209  14   2:5   178                            MAINWIN^.HEAD.VSIZE - 2 - MSGLEN + LINECNT);
2210  14   2:5   194                  DISP1LIN( MAINWIN, MSGNUM2 + LINECNT)
2211  14   2:4   199                END;
2212  14   2:2   209          END;
2213  14   2:2   209
2214  14   2:1   209        GETCR;
2215  14   2:1   212        CLEARWIN( MAINWIN, TRUE)
2216  14   2:0   215      END;  (* WERDNA00 *)
2217  14   2:0   238
2218  14   2:0   238
2219  14   4:D     1    PROCEDURE OHNO;  (* P010804 *)
```

```
2220  14    4:D     1
2221  14    4:D     1     VAR
2222  14    4:D     1          UNUSED  : INTEGER;
2223  14    4:D     2          UNUSED2 : ARRAY[ 0..5] OF INTEGER;
2224  14    4:D     8
2225  14    4:D     8
2226  14    5:D     1     PROCEDURE WERDDEAD;  (* P010805 *)
2227  14    5:D     1
2228  14    5:D     1       VAR
2229  14    5:D     1            POSSX : INTEGER;
2230  14    5:D     2
2231  14    5:0     0       BEGIN (* WERDDEAD *)
2232  14    5:1     0         WITH CHARACTR[ 6] DO
2233  14    5:2     8           BEGIN
2234  14    5:3     8             FOR POSSX := 1 TO POSS.POSSCNT DO
2235  14    5:4    21               IF POSS.POSSESS[ POSSX].EQINDEX > 4 THEN
2236  14    5:5    34                  POSS.POSSESS[ POSSX].EQINDEX := 0;
2237  14    5:3    53             STATUS := DEAD
2238  14    5:2    56           END;
2239  14    5:0    58       END;  (* WERDDEAD *)
2240  14    5:0    72
2241  14    5:0    72
2242  14    4:0     0     BEGIN (* OHNO *)
2243  14    4:1     0       WERDDEAD;
2244  14    4:1     2       PROTWIN( CHARSWIN, FALSE);
2245  14    4:1     7       CLEARWIN( CHARSWIN, FALSE);
2246  14    4:1    12       CLEARWIN( MAINWIN, TRUE);
2247  14    4:1    18       FILLCHAR( UNUSED2, 12, 0);
2248  14    4:1    25       WERDNA00( 0, 510);     (* "OH NO!!! THE NIGHTMARES AND THE TORMENTS...*)
2249  14    4:1    31       BASE13 := 23;
2250  14    4:1    34       EXIT( SHOPS)
2251  14    4:0    38     END;  (* OHNO *)
2252  14    4:0    50
2253  14    4:0    50
2254  14    6:D     1   PROCEDURE SAVRESTR( SAVEFLAG : BOOLEAN);  (* P010806 *)
2255  14    6:D     2
2256  14    6:D     2     (* SAVEFLAG = FALSE === NEW GAME OR RESTORE GAME 1-8 *)
2257  14    6:D     2     (* SAVEFLAG = TRUE  === SAVE GAME *)
2258  14    6:D     2     (* BASE38[ 6] = 0 FOR NEW GAME, OTHERWISE 1-8 *)
2259  14    6:D     2
2260  14    6:D     2     TYPE
2261  14    6:D     2          TSRREC = RECORD
2262  14    6:D     2              SRCHARTR : TCHAR;
2263  14    6:D     2              MP16F : TWIZLONG;
2264  14    6:D     2              MP172 : TWIZLONG;
2265  14    6:D     2              MP175 : ARRAY[ 0..23] OF INTEGER;
2266  14    6:D     2              MP18D : ARRAY[ 0..70] OF INTEGER;
2267  14    6:D     2              MP1D4 : INTEGER;
2268  14    6:D     2            END;
2269  14    6:D     2
2270  14    6:D     2     VAR
2271  14    6:D     2          WERDNABL : INTEGER;
2272  14    6:D     3          UNUSED   : INTEGER;  (* MP03 *)
2273  14    6:D     4          UNUSED2  : INTEGER;  (* MP04 *)
2274  14    6:D     5          INDX     : INTEGER;
2275  14    6:D     6          SRBUFF   : PACKED ARRAY[ 0..511] OF CHAR;
2276  14    6:D   262          SPELLI   : INTEGER;
```

```
2277  14    6:D   263          SRREC    : TSRREC; (* MP107 *)
2278  14    6:D   469          SRBUFF2X : PACKED ARRAY[ 0..1023] OF CHAR;
2279  14    6:D   981
2280  14    6:D   981
2281  14    7:D     1     PROCEDURE SRGLOBAL;  (* P010807 *)
2282  14    7:D     1
2283  14    7:D     1       VAR
2284  14    7:D     1            UNUSED : INTEGER;
2285  14    7:D     2            FTINDX   : INTEGER;
2286  14    7:D     3
2287  14    7:D     3
2288  14    8:D     1         PROCEDURE FROMTO1( VAR FROMTO : INTEGER;  (* P010808 *)
2289  14    8:D     2                                TOFROM : INTEGER);
2290  14    8:D     3
2291  14    8:0     0           BEGIN (* FROMTO1 *)
2292  14    8:1     0             IF SAVEFLAG THEN
2293  14    8:2     5               MOVELEFT( FROMTO, SRREC.MP18D[ TOFROM], 2)
2294  14    8:1    18             ELSE
2295  14    8:2    20               MOVELEFT( SRREC.MP18D[ TOFROM], FROMTO, 2)
2296  14    8:0    33           END;  (* FROMTO1 *)
2297  14    8:0    46
2298  14    8:0    46
2299  14    9:D     1         PROCEDURE FROMTO2( VAR FROMTO : INTEGER;  (* P010809 *)
2300  14    9:D     2                                TOFROM : INTEGER);
2301  14    9:D     3
2302  14    9:0     0           BEGIN (* FROMTO2 *)
2303  14    9:1     0             IF SAVEFLAG THEN
2304  14    9:2     5               SRREC.MP18D[ TOFROM] := FROMTO
2305  14    9:1    12             ELSE
2306  14    9:2    17               FROMTO := SRREC.MP18D[ TOFROM]
2307  14    9:0    25           END;  (* FROMTO2 *)
2308  14    9:0    40
2309  14    9:0    40
2310  14    7:0     0       BEGIN (* SRGLOBAL *)
2311  14    7:1     0         IF SAVEFLAG THEN
2312  14    7:2     5           BEGIN
2313  14    7:3     5             MOVELEFT( BASE42,  SRREC.MP172, 6);
2314  14    7:3    16             MOVELEFT( BASE3F,  SRREC.MP16F, 6);
2315  14    7:3    27             MOVELEFT( BASE147, SRREC.MP175, 48)
2316  14    7:2    39           END
2317  14    7:1    39         ELSE
2318  14    7:2    41           BEGIN
2319  14    7:3    41             MOVELEFT( SRREC.MP172, BASE42,  6);
2320  14    7:3    52             MOVELEFT( SRREC.MP16F, BASE3F,  6);
2321  14    7:3    63             MOVELEFT( SRREC.MP175, BASE147, 48);
2322  14    7:2    75           END;
2323  14    7:2    75
2324  14    7:1    75         FROMTO2( MAZEX, 1);
2325  14    7:1    80         FROMTO2( MAZEY, 2);
2326  14    7:1    85         FROMTO2( MAZELEV, 3);
2327  14    7:1    90         FROMTO2( DIRECTIO, 4);
2328  14    7:1    95         FROMTO1( INFOWOFF.I, 5);
2329  14    7:1   100         FROMTO1( BASE18.I, 6);
2330  14    7:1   105         FROMTO2( SAVEX, 7);
2331  14    7:1   110         FROMTO2( SAVEY, 8);
2332  14    7:1   115         FROMTO2( SAVELEV, 9);
2333  14    7:1   120         FROMTO2( ACMOD2, 10);
```

```
2334  14    7:1   125          FROMTO2( LIGHT, 11);
2335  14    7:1   130          FROMTO2( IDMONSTR, 12);
2336  14    7:1   135          FROMTO2( TIMEDLAY, 13);
2337  14    7:1   140          FROMTO2( BASE45, 14);
2338  14    7:1   145          FROMTO2( BASE46, 15);
2339  14    7:1   150          FROMTO2( BASE80B, 16);
2340  14    7:1   156
2341  14    7:1   156          FOR FTINDX := 1 TO 5 DO
2342  14    7:2   167            FROMTO2( BASE7DC[ FTINDX - 1], 19 + FTINDX);
2343  14    7:2   187
2344  14    7:1   187          FOR FTINDX := 1 TO 19 DO
2345  14    7:2   198            FROMTO2( BLACKBOX[ FTINDX - 1], 30 + FTINDX);
2346  14    7:2   218
2347  14    7:1   218          FOR FTINDX := 1 TO 19 DO
2348  14    7:2   229            FROMTO2( BASE7F4[ FTINDX - 1], 50 + FTINDX)
2349  14    7:2   240
2350  14    7:0   240        END;  (* SRGLOBAL *)
2351  14    7:0   268
2352  14    7:0   268
2353  14    6:0     0      BEGIN (* SAVRESTR *)
2354  14    6:1     0        CLEARWIN( MAINWIN, TRUE);
2355  14    6:1     6        WERDNABL := FINDFILE( DRIVE1, 'WERDNA.DATA');
2356  14    6:1    28
2357  14    6:1    28        WITH CHARACTR[ 6] DO  (* WERDNA CHARACTER *)
2358  14    6:2    37          BEGIN
2359  14    6:3    37            IF SAVEFLAG THEN
2360  14    6:4    40              BEGIN
2361  14    6:4    40
2362  14    6:4    40                (* SAVING A GAME *)
2363  14    6:4    40
2364  14    6:5    40                IF BASE38[ 6] <> 0 THEN
2365  14    6:6    50                  BEGIN
2366  14    6:7    50                    SRREC.SRCHARTR := CHARACTR[ 6];
2367  14    6:7    61                    SRGLOBAL;
2368  14    6:7    63                    FILLCHAR( SRBUFF, 512, 0);
2369  14    6:7    72                    SRREC.MP1D4 := 1787;
2370  14    6:7    78                    MOVELEFT( SRREC, SRBUFF, SIZEOF( TSRREC));
2371  14    6:7    90                    UNITWRITE( DRIVE1, SRBUFF, 512,
2372  14    6:7    97                               (WERDNABL + BASE38[ 6] - 1));
2373  14    6:7   110                    IF MAZELEV = 5 THEN
2374  14    6:8   115                      BEGIN
2375  14    6:9   115                        MOVELEFT( BASE55B[ GETREC( 1, 4, 1024)],
2376  14    6:9   128                                  SRBUFF2X,
2377  14    6:9   132                                  1024);
2378  14    6:9   137
2379  14    6:9   137                        UNITWRITE( DRIVE1, SRBUFF2X, 1024,
2380  14    6:9   145                                   (WERDNABL + 6) + (2 * BASE38[ 6]));
2381  14    6:8   160                      END;
2382  14    6:6   160                  END
2383  14    6:4   160              END
2384  14    6:4   160
2385  14    6:3   160            ELSE
2386  14    6:3   162
2387  14    6:4   162              BEGIN
2388  14    6:4   162
2389  14    6:4   162                (* NEW GAME OR RESTORING A GAME *)
2390  14    6:4   162
```

```
2391  14    6:5   162              IF BASE38[ 6] = 0 THEN
2392  14    6:5   172
2393  14    6:5   172                (* NEW GAME *)
2394  14    6:5   172
2395  14    6:6   172                NAME := ''
2396  14    6:5   175              ELSE
2397  14    6:6   182                BEGIN
2398  14    6:6   182
2399  14    6:6   182                  (* RESTORING GAME BASE38[ 6] === 1..8 *)
2400  14    6:6   182
2401  14    6:7   182                  UNITREAD( DRIVE1, SRBUFF, 512, WERDNABL + BASE38[ 6] - 1);
2402  14    6:7   202                  MOVELEFT( SRBUFF, SRREC, SIZEOF( TSRREC));
2403  14    6:7   214                  IF SRREC.MP1D4 <> 1787 THEN
2404  14    6:8   223                    BEGIN
2405  14    6:9   223                      SRREC.MP1D4 := 1787;
2406  14    6:9   229                      FOR INDX := 1 TO 8 DO
2407  14    6:0   243                        SRREC.SRCHARTR.POSS.POSSESS[ INDX].IDENTIF := FALSE;
2408  14    6:8   262                    END;
2409  14    6:8   262
2410  14    6:7   262                  CHARACTR[ 6] := SRREC.SRCHARTR;
2411  14    6:7   273                  CHARACTR[ 6].LOSTXYL.POISNAMT[ 4] := 0;
2412  14    6:7   288                  SRGLOBAL;
2413  14    6:7   290                  IF MAZELEV = 5 THEN
2414  14    6:8   295                    BEGIN
2415  14    6:9   295                      UNITREAD( DRIVE1, SRBUFF2X, 1024,
2416  14    6:9   303                                ((WERDNABL + 6) + (2 * BASE38[ 6])));
2417  14    6:9   318
2418  14    6:9   318                      MOVELEFT( SRBUFF2X,
2419  14    6:9   322                                BASE55B[ GETRECW( 1, 4, 1024)],
2420  14    6:9   335                                1024)
2421  14    6:8   340                    END
2422  14    6:6   340                END;
2423  14    6:6   340
2424  14    6:5   340              IF (NAME = '') OR (STATUS <> OK) THEN
2425  14    6:6   358                BEGIN
2426  14    6:6   358
2427  14    6:6   358                  (* NEW GAME; OR RESTORE GAME AND WERDNA WAS NOT OK *)
2428  14    6:6   358
2429  14    6:7   358                  MAZEX := 10;
2430  14    6:7   361                  MAZEY := 9;
2431  14    6:7   364                  MAZELEV := 11;
2432  14    6:7   367                  DIRECTIO := 3;
2433  14    6:7   370                  INFOWOFF.B := FALSE;
2434  14    6:7   373                  BASE18.B := TRUE;
2435  14    6:7   376                  SAVEX := 10;
2436  14    6:7   379                  SAVEY := 9;
2437  14    6:7   382                  SAVELEV := 11;
2438  14    6:7   385                  ACMOD2 := 0;
2439  14    6:7   388                  LIGHT := 0;
2440  14    6:7   391                  IDMONSTR := 0;
2441  14    6:7   394                  TIMEDLAY := 1;
2442  14    6:7   397                  BASE45 := 0;
2443  14    6:7   400                  BASE46 := 0;
2444  14    6:7   403                  BASE80B := 0;
2445  14    6:7   407
2446  14    6:7   407                  FOR INDX := 1 TO 5 DO
2447  14    6:8   421                    BASE7DC[ INDX - 1] := 0;
```

```
2448  14    6:8   438
2449  14    6:7   438                  FOR INDX := 1 TO 19 DO
2450  14    6:8   452                    BLACKBOX[ INDX - 1] := 0;
2451  14    6:8   469
2452  14    6:7   469                  FOR INDX := 1 TO 19 DO
2453  14    6:8   483                    BASE7F4[ INDX - 1] := 0;
2454  14    6:8   500
2455  14    6:7   500                  FOR INDX := 0 TO 2 DO
2456  14    6:8   514                    BEGIN
2457  14    6:9   514                      BASE42[ INDX] := -1;
2458  14    6:9   522                      BASE3F[ INDX] := 0;
2459  14    6:9   529                      BASE147[ INDX] := ''
2460  14    6:8   535                    END;
2461  14    6:8   547
2462  14    6:7   547                  NAME := 'WERDNA';
2463  14    6:7   561
2464  14    6:7   561                  PASSWORD := '';
2465  14    6:7   571
2466  14    6:7   571                  INMAZE := FALSE;
2467  14    6:7   578                  RACE   := HUMAN;
2468  14    6:7   585                  CLASS  := MAGE;
2469  14    6:7   592                  AGE    := 5200;
2470  14    6:7   601                  STATUS := OK;
2471  14    6:7   608                  ALIGN  := EVIL;
2472  14    6:7   615
2473  14    6:7   615                  ATTRIB[ STRENGTH] := 8;
2474  14    6:7   626                  ATTRIB[ IQ      ] := 8;
2475  14    6:7   637                  ATTRIB[ PIETY   ] := 8;
2476  14    6:7   648                  ATTRIB[ VITALITY] := 8;
2477  14    6:7   659                  ATTRIB[ AGILITY ] := 8;
2478  14    6:7   670                  ATTRIB[ LUCK    ] := 8;
2479  14    6:7   681
2480  14    6:7   681                  LUCKSKIL[ 0] := 16;
2481  14    6:7   692                  LUCKSKIL[ 1] := 16;
2482  14    6:7   703                  LUCKSKIL[ 2] := 16;
2483  14    6:7   714                  LUCKSKIL[ 3] := 16;
2484  14    6:7   725                  LUCKSKIL[ 4] := 16;
2485  14    6:7   736
2486  14    6:7   736                  FILLCHAR( GOLD, 6, 0);   (* ^.1A *)
2487  14    6:7   746                  FILLCHAR( POSS, 66, 0);  (* ^.1D *)
2488  14    6:7   756                  FILLCHAR( EXP, 6, 0);    (* ^.3E *)
2489  14    6:7   766
2490  14    6:7   766                  EXP.MID  := 100;
2491  14    6:7   773                  MAXLEVAC := 0;
2492  14    6:7   780                  CHARLEV  := 0;
2493  14    6:7   787                  HPLEFT   := 1;           (* ^.43 *)
2494  14    6:7   794                  HPMAX    := 1;           (* ^.44 *)
2495  14    6:7   801
2496  14    6:7   801                  FOR SPELLI := 0 TO 50 DO (* 50 IS ONE TOO MANY ! *)
2497  14    6:8   818                    SPELLSKN[ SPELLI]  :=  (SPELLI > 0) AND (SPELLI < 22);
2498  14    6:8   851                                           (* ^.45 *)
2499  14    6:7   851                  HPCALCMD := 1;           (* ^.57 *)
2500  14    6:7   858                  ARMORCL   := 10;
2501  14    6:7   865                  HEALPTS  := 0;
2502  14    6:7   872                  CRITHITM := FALSE;
2503  14    6:7   879                  SWINGCNT := 1;           (* ^.5B *)
2504  14    6:7   886
```

```
2505  14    6:7   886                  HPDAMRC.LEVEL := 1;
2506  14    6:7   893                  HPDAMRC.HPFAC := 4;
2507  14    6:7   900                  HPDAMRC.HPMINAD := 0;
2508  14    6:7   907
2509  14    6:7   907                  FILLCHAR( MAGESP,   SIZEOF( TSPELL7G), 0);  (* ^.49 *)
2510  14    6:7   917                  FILLCHAR( PRIESTSP, SIZEOF( TSPELL7G), 0);  (* ^.49 *)
2511  14    6:7   927
2512  14    6:7   927                  FILLCHAR( WEPVSTY2, 2 * 2, 0);
2513  14    6:7   939                  FILLCHAR( WEPVSTY3, 2 * 2, 0);
2514  14    6:7   951                  FILLCHAR( WEPVSTYP, 2 * 2, 0);
2515  14    6:7   963                    (* I HAVE WEPVSTYP AS 0..13 OF BOOLEAN *)
2516  14    6:7   963
2517  14    6:7   963                  LOSTXYL.LOCATION[ 1] := 0;           (* ^.64 *)
2518  14    6:7   975                  LOSTXYL.LOCATION[ 2] := 10 * 100 + 9;
2519  14    6:7   991                  LOSTXYL.LOCATION[ 3] := 11;
2520  14    6:7  1003                  LOSTXYL.LOCATION[ 4] := 0;
2521  14    6:7  1015
2522  14    6:7  1015                  TREBORX := -32767;
2523  14    6:7  1022
2524  14    6:7  1022                  WERDNA00( 520, 530)  (* AT LAST YOU AWAKEN FROM ENDLESS...*)
2525  14    6:7  1028                                       (* ...                              *)
2526  14    6:7  1028                                       (* BUT FIRST YOU HAVE TO REGAIN...   *)
2527  14    6:6  1028                END;
2528  14    6:4  1030              END;
2529  14    6:2  1030          END;
2530  14    6:2  1030
2531  14    6:0  1030      END;  (* SAVRESTR *)
2532  14    6:0  1060
2533  14    6:0  1060
2534  14   10:D     3    FUNCTION SOLGAM18( MSGNUM : INTEGER) : INTEGER;  (* P01080A *)
2535  14   10:D     4
2536  14   10:D     4
2537  14   10:D     4      (* RETURNS:   -1  <ESC> PRESSED.
2538  14   10:D     4                    0  <ENTER> PRESSED FOR NEW GAME.
2539  14   10:D     4                 1..8  GAME NUMBER.                 *)
2540  14   10:D     4
2541  14   10:0     0      BEGIN (* SOLGAM18 *)
2542  14   10:1     0        BASE10 := GETWIN( 0, 2, 40, 4, 6, TRUE);
2543  14   10:1    13        DISP1LIN( BASE10, MSGNUM);
2544  14   10:1    18        DISP1LIN( BASE10, MSGNUM + 1);
2545  14   10:1    25        DELWIN( BASE10, FALSE);
2546  14   10:1    31
2547  14   10:1    31        REPEAT
2548  14   10:2    31          GETKEY;
2549  14   10:2    34          IF INCHAR = CHR( 13) THEN
2550  14   10:3    39            INCHAR := '0';
2551  14   10:2    42          IF INCHAR = CHR( 27) THEN
2552  14   10:3    47            INCHAR := CHR( ORD( '0') - 1);  (* '/' !? *)
2553  14   10:1    52        UNTIL (ORD( INCHAR) >= (ORD( '0') - 1)) AND
2554  14   10:1    57              (INCHAR <= '8');
2555  14   10:1    63
2556  14   10:1    63        SOLGAM18 := ORD( INCHAR) - ORD( '0')
2557  14   10:1    65
2558  14   10:0    65      END;  (* SOLGAM18 *)
2559  14   10:0    82
2560  14   10:0    82
2561  14   11:D     1    PROCEDURE PLAYLVMV;  (* P01080B *)
```

```
2562  14   11:D     1
2563  14   11:D     1
2564  14   12:D     1     PROCEDURE PLAYGAME;  (* P01080C *)
2565  14   12:D     1
2566  14   12:D     1       VAR
2567  14   12:D     1           UNUSED : ARRAY[ 0..106] OF INTEGER;
2568  14   12:D   108
2569  14   12:D   108
2570  14   12:0     0       BEGIN (* PLAYGAME *)
2571  14   12:1     0         BASE38[ 6] := SOLGAM18( 503); (* PLAY WHICH SAVE GAME (1-8)?   *)
2572  14   12:1    13                                       (* ~ FOR A NEW GAME, (ESC) EXITS *)
2573  14   12:1    13         IF BASE38[ 6] < 0 THEN  (* <ESC> *)
2574  14   12:2    23           BEGIN
2575  14   12:3    23             BASE09 := -1;
2576  14   12:3    27             EXIT( PLAYGAME)
2577  14   12:2    31           END;
2578  14   12:1    31         SAVRESTR( FALSE);  (* NEW / RESTORE *)
2579  14   12:1    34                            (* BASE38[ 6] = 0 FOR NEW GAME,
2580  14   12:1    34                                       = 1..8 TO PLAY A SAVED GAME *)
2581  14   12:1    34
2582  14   12:1    34         MOVELEFT( BASE55B[ GETREC( 1,  MAZELEV - 1, 1024)],
2583  14   12:1    49                   BASE09, 2);
2584  14   12:1    55
2585  14   12:1    55         UNITREAD( DRIVE1, BASE55B, 512, CACHEBL);
2586  14   12:1    68         CLEARWIN( MAINWIN, FALSE);
2587  14   12:1    74         BASE10 := GETWIN( 0, 10, 40, 4, 6, TRUE);
2588  14   12:1    87         DISP1LIN( BASE10, 500);   (* "ENTERING" *)
2589  14   12:1    94         DISP1LIN( BASE10, 501);   (* "THE RETURN OF WERDNA" *)
2590  14   12:1   101         DELWIN( BASE10, FALSE);
2591  14   12:1   107         BASE13 := 7;
2592  14   12:1   110         EXIT( SHOPS)
2593  14   12:1   114
2594  14   12:0   114       END;  (* PLAYGAME *)
2595  14   12:0   126
2596  14   12:0   126
2597  14   13:D     1     PROCEDURE MOVESAVE;  (* P01080D *)
2598  14   13:D     1
2599  14   13:D     1       VAR
2600  14   13:D     1           UNUSED  : INTEGER;
2601  14   13:D     2           MP02    : INTEGER;
2602  14   13:D     3           UNUSED2 : INTEGER;
2603  14   13:D     4           NUM07   : INTEGER;
2604  14   13:D     5           MP05 : PACKED ARRAY[ 0..511] OF CHAR;
2605  14   13:D   261
2606  14   13:D   261           MP105 : PACKED ARRAY[ 0..1023] OF CHAR;
2607  14   13:D   773
2608  14   13:D   773           UWINOUT : INTEGER;
2609  14   13:D   774
2610  14   13:D   774
2611  14   14:D     1       PROCEDURE MSG123( MSG1 : INTEGER;  (* P01080E *)
2612  14   14:D     2                         MSG2 : INTEGER;
2613  14   14:D     3                         MSG3 : INTEGER);
2614  14   14:D     4
2615  14   14:D     4         VAR
2616  14   14:D     4             UNUSED : INTEGER;
2617  14   14:D     5             VSIZE  : INTEGER;
2618  14   14:D     6
```

```
2619  14   14:D     6
2620  14   14:0     0         BEGIN (* MSG123 *)
2621  14   14:1     0           VSIZE := 3;
2622  14   14:1     3           IF MSG2 > 0 THEN
2623  14   14:2     8             VSIZE := 4;
2624  14   14:1    11           IF MSG3 > 0 THEN
2625  14   14:2    16             VSIZE := 5;
2626  14   14:1    19           BASE10 := GETWIN( 0, 10, 40, VSIZE, 30, TRUE);
2627  14   14:1    32           DISP1LIN( BASE10, MSG1);
2628  14   14:1    37           IF MSG2 > 0 THEN
2629  14   14:2    42             DISP1LIN( BASE10, MSG2);
2630  14   14:1    47           IF MSG3 > 0 THEN
2631  14   14:2    52             DISP1LIN( BASE10, MSG3);
2632  14   14:1    57           DELWIN( BASE10, FALSE)
2633  14   14:0    60         END;  (* MSG123 *)
2634  14   14:0    76
2635  14   14:0    76
2636  14   15:D     1       PROCEDURE INSERTA;  (* P01080F *)
2637  14   15:D     1
2638  14   15:0     0         BEGIN (* INSERTA *)
2639  14   15:1     0           MSG123( 9, 6, 0);
2640  14   15:1     5             (*
2641  14   15:1     5                 INSERT YOUR ORIGINAL DISKETTE "A" INTO
2642  14   15:1     5                 DISK DRIVE 1, THEN PRESS ~
2643  14   15:1     5             *)
2644  14   15:1     5
2645  14   15:1     5           GETCR;
2646  14   15:1     8           IF BRELOC THEN
2647  14   15:2    13             BEGIN
2648  14   15:3    13               UWINOUT := 1;  (* RESET THE $F2 FLAG TO "VIRTUAL DISK" *)
2649  14   15:3    18               UNITWRITE( BASE12, UWINOUT, 31, 0)
2650  14   15:2    30             END;
2651  14   15:1    30           EXIT( MOVESAVE)
2652  14   15:0    34         END;  (* INSERTA *)
2653  14   15:0    46
2654  14   15:0    46
2655  14   16:D     1       PROCEDURE GETKEY18;  (* P010810 *)
2656  14   16:D     1
2657  14   16:0     0         BEGIN (* GETKEY18 *)
2658  14   16:0     0
2659  14   16:1     0           REPEAT
2660  14   16:2     0             GETKEY;
2661  14   16:2     3             IF (INCHAR = CHR( 13)) OR (INCHAR = CHR( 27)) THEN
2662  14   16:3    12               INSERTA
2663  14   16:1    12           UNTIL (INCHAR >= '1') AND (INCHAR <= '8')
2664  14   16:1    20
2665  14   16:0    20         END;  (* GETKEY18 *)
2666  14   16:0    38
2667  14   16:0    38
2668  14   17:D     1       PROCEDURE CHKDSKA;  (* P010811 *)
2669  14   17:D     1
2670  14   17:0     0         BEGIN (* CHKDSKA *)
2671  14   17:1     0           MP02 := FINDFILE( DRIVE1, 'WERDNA.DATA');
2672  14   17:1    23           IF MP02 < 0 THEN
2673  14   17:2    30             BEGIN
2674  14   17:3    30               MSG123( 10, 0, 0);
2675  14   17:3    35                 (*
```

```
2676  14   17:3    35                    THAT IS NOT A DISKETTE "A" - PRESS ~
2677  14   17:3    35                 *)
2678  14   17:3    35
2679  14   17:3    35               INSERTA
2680  14   17:2    35             END
2681  14   17:0    37          END;  (* CHKDSKA *)
2682  14   17:0    50
2683  14   17:0    50
2684  14   13:0     0        BEGIN (* MOVESAVE *)
2685  14   13:1     0          CLEARWIN( MAINWIN, TRUE);
2686  14   13:1     6          UWINOUT := -1;
2687  14   13:1    11          MSG123( 2, 3, 6);
2688  14   13:1    16            (*
2689  14   13:1    16                INSERT DISKETTE "A" CONTAINING THE
2690  14   13:1    16                SAVE GAME YOU WISH TO MOVE INTO
2691  14   13:1    16                DISK DRIVE 1, THEN PRESS ~
2692  14   13:1    16            *)
2693  14   13:1    16
2694  14   13:1    16          IF BRELOC THEN
2695  14   13:2    21            BEGIN
2696  14   13:3    21              UWINOUT := 2;   (* RESET $F2 TO NO "VIRTUAL DISK" *)
2697  14   13:3    25              UNITWRITE( BASE12, UWINOUT, 31, 0)
2698  14   13:2    36            END;
2699  14   13:1    36          GETCR;
2700  14   13:1    39          CHKDSKA;
2701  14   13:1    41          MSG123( 7, 0, 0);
2702  14   13:1    46            (*
2703  14   13:1    46                MOVE WHICH SAVE GAME (1-8, ~ EXITS)
2704  14   13:1    46            *)
2705  14   13:1    46
2706  14   13:1    46          GETKEY18;
2707  14   13:1    48          NUM07 := ORD( INCHAR) - ORD( '1');
2708  14   13:1    53          UNITREAD( DRIVE1, MP05, 512, MP02 + NUM07);
2709  14   13:1    66          UNITREAD( DRIVE1, MP105, 1024, MP02 + 8 + NUM07 + NUM07);
2710  14   13:1    84          MSG123( 4, 5, 6);
2711  14   13:1    89            (*
2712  14   13:1    89                INSERT DISKETTE "A" ONTO WHICH
2713  14   13:1    89                THE SAVE GAME IS TO BE MOVED INTO
2714  14   13:1    89                DISK DRIVE 1, THEN PRESS ~
2715  14   13:1    89            *)
2716  14   13:1    89
2717  14   13:1    89          GETCR;
2718  14   13:1    92          CHKDSKA;
2719  14   13:1    94          MSG123( 8, 0, 0);
2720  14   13:1    99            (*
2721  14   13:1    99                MOVE TO WHAT SAVE GAME (1-8, ~ EXITS
2722  14   13:1    99            *)
2723  14   13:1    99
2724  14   13:1    99          GETKEY18;
2725  14   13:1   101          NUM07 := ORD( INCHAR) - ORD( '1');
2726  14   13:1   106          UNITWRITE( DRIVE1, MP05, 512, MP02 + NUM07);
2727  14   13:1   119          UNITWRITE( DRIVE1, MP105, 1024, MP02 + 8 + NUM07 + NUM07);
2728  14   13:1   137          INSERTA
2729  14   13:1   137
2730  14   13:0   137        END;  (* MOVESAVE *)
2731  14   13:0   152
2732  14   13:0   152
```

```
2733  14   11:0     0     BEGIN (* PLAYLVMV *)
2734  14   11:0     0
2735  14   11:0     0       (*
2736  14   11:0     0          WELCOME TO:
2737  14   11:0     0          THE RETURN OF WERDNA!
2738  14   11:0     0
2739  14   11:0     0          YOU ARE WERDNA, THE EVIL WIZARD.
2740  14   11:0     0          YEARS AGO...
2741  14   11:0     0       *)
2742  14   11:0     0
2743  14   11:1     0       CLEARWIN( MAINWIN, TRUE);
2744  14   11:1     6       FOR BASE09 := 3 TO 19 DO
2745  14   11:2    17         BEGIN
2746  14   11:3    17           MVCURSOR( MAINWIN, 0, BASE09);
2747  14   11:3    24           DISP1LIN( MAINWIN, 220 + BASE09);
2748  14   11:2    34         END;
2749  14   11:2    41
2750  14   11:1    41       REPEAT
2751  14   11:1    41
2752  14   11:2    41         GETMSGTX( BASE47, 502);  (* P)LAY GAME/L)EAVE GAME/M)OVE SAVE GAMES *)
2753  14   11:2    49         BASE10 := GETWIN2( GTLNGSUB( BASE47), 0, 5, 4, TRUE);
2754  14   11:2    67         BASE09 := MENU( BASE10, BASE47);
2755  14   11:2    77         DELWIN( BASE10, TRUE);
2756  14   11:2    83
2757  14   11:2    83         CASE BASE09 OF
2758  14   11:2    86           0:  PLAYGAME;
2759  14   11:2    90           1:  BASE13 := 0;  (* LEAVE GAME *)
2760  14   11:2    95           2:  MOVESAVE;
2761  14   11:2    99         END;
2762  14   11:2   112
2763  14   11:1   112       UNTIL BASE09 >= 0;
2764  14   11:1   117
2765  14   11:1   117       EXIT( SHOPS)
2766  14   11:1   121
2767  14   11:0   121     END;  (* PLAYLVMV *)
2768  14   11:0   138
2769  14   11:0   138
2770  14   18:D     1   PROCEDURE SAVEGAME;  (* P010812 *)
2771  14   18:D     1
2772  14   18:0     0     BEGIN (* SAVEGAME *)
2773  14   18:1     0       PROTWIN( CHARSWIN, TRUE);
2774  14   18:1     5       BASE38[ 6] := SOLGAM18( 505);  (* "SAVE TO WHICH SAVE GAME (1-8) ?" *)
2775  14   18:1    18       IF BASE38[ 6] = -1 THEN
2776  14   18:2    29         BEGIN
2777  14   18:3    29           BASE13 := 5;
2778  14   18:3    32           EXIT( SHOPS)
2779  14   18:2    36         END;
2780  14   18:2    36
2781  14   18:1    36       WITH CHARACTR[ 6] DO
2782  14   18:2    44         BEGIN
2783  14   18:3    44           LOSTXYL.LOCATION[ 2] :=  (100 * MAZEX) + MAZEY;
2784  14   18:3    58           LOSTXYL.LOCATION[ 3] := MAZELEV;
2785  14   18:3    68           SAVRESTR( TRUE);
2786  14   18:3    71           BASE13 := 23;
2787  14   18:3    74           EXIT( SHOPS)
2788  14   18:2    78         END;
2789  14   18:2    78
```

```
2790  14   18:0    78     END;  (* SAVEGAME *)
2791  14   18:0    90
2792  14   18:0    90
2793  14   19:D     1   PROCEDURE SHOPINIT;  (* P010813 *)
2794  14   19:D     1
2795  14   19:0     0     BEGIN (* SHOPINIT *)
2796  14   19:1     0       ACMOD2 := 0;
2797  14   19:1     3       LIGHT := 0;
2798  14   19:1     6       IDMONSTR := 0;
2799  14   19:1     9       CHSTALRM := 0;
2800  14   19:1    12       ATTK012 := 0;
2801  14   19:1    15       FIZZLES := 0;
2802  14   19:1    18       BASE37 := FALSE;
2803  14   19:1    21       BASE38[ 6] := -1;
2804  14   19:1    29       BASE13 := 23;
2805  14   19:0    32     END;  (* SHOPINIT *)
2806  14   19:0    44
2807  14   19:0    44
2808  14    1:0     0   BEGIN  (* SHOPS *)
2809  14    1:0     0
2810  14    1:1     0     CASE BASE13 OF
2811  14    1:1     4
2812  14    1:1     4       1:  SHOPINIT;
2813  14    1:1     8      14:  OHNO;     (* OH NO!!! *)
2814  14    1:1    12       8:  SAVEGAME;
2815  14    1:1    16      23:  PLAYLVMV;
2816  14    1:1    20
2817  14    1:1    20     END;
2818  14    1:1    74
2819  14    1:0    74   END;   (* SHOPS *)
2820  14    1:0    86
2821  14    1:0    86 (*$I WIZ4B:SHOPS *)
2822  14    1:0    86
2823  15    1:D     1   SEGMENT PROCEDURE P010901;
2824  15    1:0     0     BEGIN
2825  15    1:0     0     END;
2826  15    1:0    12
2826  15    1:0    12 (*$I WIZ4C:CAMP  *)
2827  15    1:0    12
2828  16    1:D     1 SEGMENT PROCEDURE CAMP;  (* P010A01 *)
2829  16    1:D     1
2830  16    2:D     1   PROCEDURE P010A02;
2831  16    2:D     1
2832  16    2:D     1     VAR
2833  16    2:D     1         MP01 : INTEGER;
2834  16    2:D     2         MP02 : INTEGER;
2835  16    2:D     3         MP03 : INTEGER;
2836  16    2:D     4         MP04 : INTEGER;
2837  16    2:D     5         MP05 : INTEGER;
2838  16    2:D     6         MP06 : INTEGER;
2839  16    2:D     7         MP07 : INTEGER;
2840  16    2:D     8
2841  16    2:D     8
2842  16    3:D     1     PROCEDURE P010A03;
2843  16    3:D     1
2844  16    3:D     1       VAR
2845  16    3:D     1           DEADOBJS : INTEGER;
```

```
2846  16    3:D     2            OBJCOUNT : INTEGER;
2847  16    3:D     3            P03MP03 : PACKED ARRAY[ 0..2] OF INTEGER;  (* GUESS *)
2848  16    3:D     6
2849  16    3:D     6            FOUNDOBJ : PACKED ARRAY[ 1..8] OF INTEGER;
2850  16    3:D    14
2851  16    3:D    14            DEADOBJL : PACKED ARRAY[ 1..100] OF INTEGER; (* GUESS *)
2852  16    3:D   114
2853  16    3:D   114            LOOTLIST : PACKED ARRAY[ 0..250] OF BOOLEAN;  (* GUESS *)
2854  16    3:D   130            UNIDENT : PACKED ARRAY[ 0..250] OF BOOLEAN;  (* MP82 *)
2855  16    3:D   146
2856  16    3:D   146
2857  16    4:D     3        FUNCTION RAND1TOX( MP03 : INTEGER) : INTEGER;  (* P010A04 *)
2858  16    4:D     4
2859  16    4:0     0          BEGIN (* RAND1TOX *)
2860  16    4:1     0            RAND1TOX := 1 + (RAND MOD MP03)
2861  16    4:0     8          END;  (* RAND1TOX *)
2862  16    4:0    24
2863  16    4:0    24
2864  16    5:D     1        PROCEDURE P010A05;
2865  16    5:D     1
2866  16    5:D     1          (* CREATE FOUNDOBJ[] LIST TO DISPLAY AS "RUMMAGING AROUND...YOU FIND *)
2867  16    5:D     1
2868  16    5:D     1          VAR
2869  16    5:D     1               LVLCALC : INTEGER;
2870  16    5:D     2               TEMPX   : INTEGER;  (* MULTIPLE USES *)
2871  16    5:D     3               COUNT   : INTEGER;
2872  16    5:D     4
2873  16    5:D     4
2874  16    6:D     1          PROCEDURE SHUFFLE;  (* P010A06 *)
2875  16    6:D     1
2876  16    6:D     1            (* SHUFFLE OBJECTS IN FOUNDOBJ[] FROM 1 TO OBJCOUNT *)
2877  16    6:D     1
2878  16    6:D     1            VAR
2879  16    6:D     1                 SAVE1 : INTEGER;
2880  16    6:D     2                 OBJ2  : INTEGER;
2881  16    6:D     3                 OBJ1  : INTEGER;
2882  16    6:D     4                 DO100 : INTEGER;
2883  16    6:D     5
2884  16    6:0     0            BEGIN (* SHUFFLE *)
2885  16    6:1     0              IF OBJCOUNT > 1 THEN
2886  16    6:2     7                FOR DO100 := 1 TO 100 DO
2887  16    6:3    18                  BEGIN
2888  16    6:4    18                    OBJ1 := RAND1TOX( OBJCOUNT);
2889  16    6:4    27                    OBJ2 := RAND1TOX( OBJCOUNT);
2890  16    6:4    36                    SAVE1 := FOUNDOBJ[ OBJ1];
2891  16    6:4    47                    FOUNDOBJ[ OBJ1] := FOUNDOBJ[ OBJ2];
2892  16    6:4    65                    FOUNDOBJ[ OBJ2] := SAVE1
2893  16    6:3    73                  END;
2894  16    6:3    82
2895  16    6:1    82              EXIT( P010A05);
2896  16    6:1    86
2897  16    6:0    86            END;  (* SHUFFLE *)
2898  16    6:0   100
2899  16    6:0   100
2900  16    7:D     3          FUNCTION P010A07( MP03 : INTEGER) : BOOLEAN;
2901  16    7:D     4
2902  16    7:0     0            BEGIN (* P010A07 *)
```

```
2903  16    7:1     0            IF UNIDENT[ MP03] THEN
2904  16    7:2    11              BEGIN
2905  16    7:3    11                P010A07 := NOT LOOTLIST[ MP03];
2906  16    7:3    22                LOOTLIST[ MP03] := TRUE
2907  16    7:2    29              END
2908  16    7:1    31            ELSE
2909  16    7:2    33              P010A07 := TRUE
2910  16    7:0    33          END;  (* P010A07 *)
2911  16    7:0    48
2912  16    7:0    48
2913  16    5:0     0        BEGIN (* P010A05 *)
2914  16    5:1     0          FILLCHAR( LOOTLIST, 32, 0);
2915  16    5:1     8          WITH CHARACTR[ 6].POSS DO   (* MP04 *)
2916  16    5:2    18            BEGIN
2917  16    5:3    18              FOR COUNT := 1 TO POSSCNT DO
2918  16    5:4    30                BEGIN
2919  16    5:5    30                  WITH POSSESS[ COUNT] DO   (* MP06 *)
2920  16    5:6    40                    BEGIN
2921  16    5:7    40                      IF UNIDENT[ EQINDEX] THEN
2922  16    5:8    52                        LOOTLIST[ EQINDEX] := TRUE;
2923  16    5:6    62                    END; (* WITH POSSESS *)
2924  16    5:4    62                END;
2925  16    5:4    69
2926  16    5:2    69            END; (* WITH *)
2927  16    5:2    69
2928  16    5:1    69          FOR COUNT := 1 TO 19 DO  (* 19 ITEMS IN BLACK BOX *)
2929  16    5:2    80            BEGIN
2930  16    5:3    80              IF UNIDENT[ BLACKBOX[ COUNT - 1]] THEN
2931  16    5:4    99                IF BLACKBOX[ COUNT - 1] <> 104 THEN    (* 104 = BLACK BOX(?) *)
2932  16    5:5   112                  LOOTLIST[ BLACKBOX[ COUNT - 1]] := TRUE;
2933  16    5:2   129            END;
2934  16    5:1   136          DEADOBJS := 0;
2935  16    5:1   140          LVLCALC := 12 - MAZELEV;
2936  16    5:1   145          IF LVLCALC < 1 THEN
2937  16    5:2   150            LVLCALC := 11 + LVLCALC;
2938  16    5:2   155
2939  16    5:1   155          FOR COUNT := 0 TO 5 DO
2940  16    5:2   166            BEGIN
2941  16    5:3   166              IF BASE7DB[ COUNT] THEN
2942  16    5:4   176                BEGIN
2943  16    5:5   176                  WITH CHARACTR[ COUNT] DO    (* MP05 *)
2944  16    5:6   184                    BEGIN
2945  16    5:7   184                      WITH GOLD DO           (* MP06 *)
2946  16    5:8   189                        BEGIN
2947  16    5:9   189                          IF LOW > 0 THEN
2948  16    5:0   195                            BEGIN
2949  16    5:1   195                              LOW := RAND1TOX( LOW);
2950  16    5:1   203                              MID  := 0;
2951  16    5:1   208                              HIGH := 0;
2952  16    5:1   213
2953  16    5:1   213                              FOR TEMPX := 1 TO LVLCALC DO
2954  16    5:2   224                                ADDLONGS( CHARACTR[ 6].GOLD,
2955  16    5:2   232                                          GOLD);
2956  16    5:0   245                            END;
2957  16    5:8   245                        END;
2958  16    5:8   245
2959  16    5:7   245                      WITH POSS DO           (* MP06 *)
```

```
2960  16    5:8   250                        BEGIN
2961  16    5:9   250                          FOR TEMPX := 1 TO POSSCNT DO
2962  16    5:0   262                            BEGIN
2963  16    5:1   262                              WITH POSSESS[ TEMPX] DO   (* MP08 *)
2964  16    5:2   272                                BEGIN
2965  16    5:3   272                                  IF P010A07( EQINDEX) THEN
2966  16    5:4   280                                    BEGIN
2967  16    5:5   280                                      DEADOBJS := DEADOBJS + 1;
2968  16    5:5   288                                      DEADOBJL[ DEADOBJS] := EQINDEX
2969  16    5:4   298                                    END
2970  16    5:2   301                                END;
2971  16    5:2   301
2972  16    5:0   301                            END;
2973  16    5:8   308                        END; (* WITH POSS *)
2974  16    5:8   308
2975  16    5:6   308                    END; (* WITH CHARACTR *)
2976  16    5:6   308
2977  16    5:4   308                END;
2978  16    5:2   308              END;
2979  16    5:2   315
2980  16    5:2   315            (* 51E7 *)
2981  16    5:2   315
2982  16    5:1   315            FOR COUNT := 1 TO 8 DO
2983  16    5:2   326              FOUNDOBJ[ COUNT] := -1;
2984  16    5:2   344
2985  16    5:1   344            OBJCOUNT := 0;
2986  16    5:1   348            IF DEADOBJS = 0 THEN
2987  16    5:2   355              SHUFFLE;  (* AND EXIT P010A05 *)
2988  16    5:1   357            FOR COUNT := 1 TO DEADOBJS DO
2989  16    5:2   370              BEGIN
2990  16    5:3   370                IF UNIDENT[ DEADOBJL[ COUNT]] THEN
2991  16    5:4   389                  IF DEADOBJL[ COUNT] <> -1 THEN
2992  16    5:5   403                    BEGIN
2993  16    5:6   403                      OBJCOUNT := OBJCOUNT + 1;
2994  16    5:6   411                      FOUNDOBJ[ OBJCOUNT] := DEADOBJL[ COUNT];
2995  16    5:6   431                      DEADOBJL[ COUNT] := -1
2996  16    5:5   439                    END
2997  16    5:2   442              END;
2998  16    5:2   449
2999  16    5:1   449            WHILE OBJCOUNT < 8 DO
3000  16    5:2   456              BEGIN
3001  16    5:3   456                COUNT := RAND1TOX( DEADOBJS);
3002  16    5:3   465                TEMPX := COUNT;
3003  16    5:3   468                WHILE DEADOBJL[ COUNT] = -1 DO
3004  16    5:4   482                  BEGIN
3005  16    5:5   482                    COUNT := COUNT + 1;
3006  16    5:5   487                    IF COUNT > DEADOBJS THEN
3007  16    5:6   494                      COUNT := 1;
3008  16    5:5   497                    IF COUNT = TEMPX THEN
3009  16    5:6   502                      SHUFFLE
3010  16    5:4   502                  END;
3011  16    5:4   506
3012  16    5:3   506                OBJCOUNT := OBJCOUNT + 1;
3013  16    5:3   514                FOUNDOBJ[ OBJCOUNT] := DEADOBJL[ COUNT];
3014  16    5:3   534                DEADOBJL[ COUNT] := -1
3015  16    5:2   542              END;
3016  16    5:2   547
```

```
3017  16   5:1   547          SHUFFLE
3018  16   5:1   547
3019  16   5:0   547        END;  (* P010A05 *)
3020  16   5:0   584
3021  16   5:0   584
3022  16   8:D     1      PROCEDURE GETLOOT;  (* P010A08 *)
3023  16   8:D     1
3024  16   8:D     1        VAR
3025  16   8:D     1            UNUSED  : INTEGER;
3026  16   8:D     2            OBJNAMI : INTEGER;
3027  16   8:D     3            OBJX    : INTEGER;
3028  16   8:D     4
3029  16   8:D     4
3030  16   8:0     0        BEGIN (* GETLOOT *)
3031  16   8:1     0          PROTWIN( CHARSWIN, FALSE);
3032  16   8:1     5          BASE10 := GETWIN( 0, 0, 40, 18, 25, TRUE);
3033  16   8:1    18          DISP1LIN( BASE10, 800);          (* RUMMAGING AROUND IN THE PILE *)
3034  16   8:1    25          DISP1LIN( BASE10, 801);          (* OF DEAD, YOU FIND...         *)
3035  16   8:1    32          FOR OBJX := 1 TO OBJCOUNT DO
3036  16   8:2    45            BEGIN
3037  16   8:3    45              MVCURSOR( BASE10, 10, 2 + OBJX);
3038  16   8:3    53              PRINTNUM( BASE10, OBJX, 1);
3039  16   8:3    59              PRPICMB(  BASE10, ') ');
3040  16   8:3    68              OBJNAMI := 14001 + (FOUNDOBJ[ OBJX] * 2);  (* A STONE, ... *)
3041  16   8:3    85              IF UNIDENT[ OBJX] THEN
3042  16   8:4    96                OBJNAMI := OBJNAMI - 1;
3043  16   8:3   101              GETMSGTX( BASE47, OBJNAMI);  (* OBJECT NAME *)
3044  16   8:3   107              PRPICMB(  BASE10, BASE47)
3045  16   8:2   110            END;
3046  16   8:2   120
3047  16   8:1   120          MVCURSOR( BASE10, 0, 12);
3048  16   8:1   126          DISP1LIN( BASE10, 802);  (* PRESS 1-8 TO TAKE AN ITEM        *)
3049  16   8:1   133          DISP1LIN( BASE10, 803);  (* ~ WHEN FINISHED STRIPPING BODIES *)
3050  16   8:1   140
3051  16   8:1   140          WITH CHARACTR[ 6].POSS DO    (* MP04 *)
3052  16   8:2   150            BEGIN
3053  16   8:2   150
3054  16   8:3   150            REPEAT
3055  16   8:3   150
3056  16   8:4   150              CLRRECT( BASE10, 0, 15, 38, 1);
3057  16   8:4   158              MVCURSOR( BASE10, 0, 15);
3058  16   8:4   164              IF POSSCNT > 6 THEN
3059  16   8:5   170                DISP1LIN( BASE10, 813 - POSSCNT)
3060  16   8:5   177                          (* YOU CAN'T CARRY ANY MORE ITEMS *)
3061  16   8:5   177                          (* YOU CAN CARRY ONLY 1 MORE ITEM *)
3062  16   8:4   177              ELSE
3063  16   8:5   182                BEGIN
3064  16   8:6   182                  GETMSGTX( BASE47, 804);  (* (YOU CAN CARRY ^ MORE ITEMS) *)
3065  16   8:6   190                  INT2STR( 8 - POSSCNT, BASEC7);
3066  16   8:6   200                  INSERTST( BASEC7, BASE47);
3067  16   8:6   208                  CENTSTR(  BASE10, BASE47);
3068  16   8:5   214                END;
3069  16   8:5   214
3070  16   8:4   214              GETKEY;
3071  16   8:4   217              IF INCHAR = CHR( CRETURN) THEN
3072  16   8:5   222                BEGIN
3073  16   8:6   222                  DELWIN( BASE10, FALSE);
```

```
3074  16    8:6   228                    UNPROWIN( CHARSWIN, FALSE);
3075  16    8:6   233                    EXIT( P010A03)
3076  16    8:5   237                  END
3077  16    8:4   237                ELSE
3078  16    8:5   239                  BEGIN
3079  16    8:6   239                    IF (INCHAR >= '1') AND (INCHAR <= '8') AND (POSSCNT < 8) THEN
3080  16    8:7   253                      BEGIN
3081  16    8:8   253                        OBJX := ORD( INCHAR) - ORD( '0');
3082  16    8:8   258                        IF FOUNDOBJ[ OBJX] <> -1 THEN
3083  16    8:9   272                          BEGIN
3084  16    8:0   272                            POSSCNT := POSSCNT + 1;
3085  16    8:0   278                            WITH POSSESS[ POSSCNT] DO   (* MP05 *)
3086  16    8:1   289                              BEGIN
3087  16    8:2   289                                EQUIPED := FALSE;
3088  16    8:2   292                                IDENTIF := FALSE;
3089  16    8:2   297                                CURSED  := FALSE;
3090  16    8:2   302                                EQINDEX := FOUNDOBJ[ OBJX];
3091  16    8:2   315                                FOUNDOBJ[ OBJX] := -1;
3092  16    8:2   326                                CLRRECT( BASE10, 0, OBJX + 2, 38, 1)
3093  16    8:1   333                              END;
3094  16    8:9   336                          END;
3095  16    8:7   336                      END;
3096  16    8:5   336                  END;
3097  16    8:5   336
3098  16    8:3   336                UNTIL FALSE
3099  16    8:3   336
3100  16    8:2   336              END;  (* WITH CHARACTR[ 6] *)
3101  16    8:2   339
3102  16    8:0   339          END;  (* GETLOOT *)
3103  16    8:0   356
3104  16    8:0   356
3105  16    9:D     1        PROCEDURE P010A09( CHARXX : INTEGER);
3106  16    9:D     2
3107  16    9:D     2          VAR
3108  16    9:D     2               EQPCNT : INTEGER;
3109  16    9:D     3               POSSX  : INTEGER;
3110  16    9:D     4
3111  16    9:0     0          BEGIN (* P010A09 *)
3112  16    9:1     0            FOR POSSX := 1 TO 5 DO
3113  16    9:2    11              BASE7DC[ POSSX - 1] := -BASE7DC[ POSSX - 1];
3114  16    9:1    37            EQPCNT := 1;
3115  16    9:1    40            WITH CHARACTR[ CHARXX].POSS DO                (* MP04 *)
3116  16    9:2    50              BEGIN
3117  16    9:3    50                FOR POSSX := 1 TO POSSCNT DO
3118  16    9:4    62                  BEGIN
3119  16    9:5    62                    POSSESS[ EQPCNT] := POSSESS[ POSSX];
3120  16    9:5    80                    WITH POSSESS[ EQPCNT] DO      (* MP06 *)
3121  16    9:6    90                      BEGIN
3122  16    9:7    90                        IF EQINDEX <> 0 + 0 THEN
3123  16    9:8    98                          BEGIN
3124  16    9:9    98                            IF EQINDEX <= 5 THEN
3125  16    9:0   104                              BEGIN
3126  16    9:1   104                                IF BASE7DC[ EQINDEX - 1] = 0 THEN
3127  16    9:2   118                                  BASE7DC[ EQINDEX - 1] := 1
3128  16    9:1   127                                ELSE
3129  16    9:2   131                                  BASE7DC[ EQINDEX - 1] :=
3130  16    9:2   140                                    ABS(  BASE7DC[ EQINDEX - 1])
```

```
3131  16    9:0   151                          END;
3132  16    9:9   152                        EQPCNT := EQPCNT + 1
3133  16    9:8   153                      END;
3134  16    9:6   157                    END; (* WITH POSSESS[] *)
3135  16    9:6   157
3136  16    9:4   157                  END;
3137  16    9:4   164
3138  16    9:4   164              (* 54FC *)
3139  16    9:3   164                  EQPCNT := EQPCNT - 1;
3140  16    9:3   169                  IF EQPCNT < POSSCNT THEN
3141  16    9:4   175                    POSSCNT := EQPCNT;
3142  16    9:4   178
3143  16    9:2   178                END; (* WITH *)
3144  16    9:2   178
3145  16    9:1   178                FOR POSSX := 1 TO 5 DO
3146  16    9:2   189                  IF BASE7DC[ POSSX - 1] = -1 THEN
3147  16    9:3   203                    BASE7DC[ POSSX - 1] := 0
3148  16    9:2   211                  ELSE
3149  16    9:3   215                    BASE7DC[ POSSX - 1] := ABS( BASE7DC[ POSSX - 1])
3150  16    9:3   233
3151  16    9:0   233          END;  (* P010A09 *)
3152  16    9:0   260
3153  16    9:0   260
3154  16   10:D     1        PROCEDURE SETUNID( OBJX : INTEGER);  (* P010A0A *)
3155  16   10:D     2
3156  16   10:0     0          BEGIN (* SETUNID *)
3157  16   10:1     0            UNIDENT[ OBJX] := TRUE
3158  16   10:0     8          END;  (* SETUNID *)
3159  16   10:0    22
3160  16   10:0    22
3161  16   11:D     1        PROCEDURE SETUNID2( OBJX : INTEGER;  (* P010A0B *)
3162  16   11:D     2                            OBJY : INTEGER);
3163  16   11:D     3
3164  16   11:0     0          BEGIN (* SETUNID2 *)
3165  16   11:1     0            WHILE OBJX <= OBJY DO
3166  16   11:2     5              BEGIN
3167  16   11:3     5                UNIDENT[ OBJX] := TRUE;
3168  16   11:3    15                OBJX := OBJX + 1
3169  16   11:2    16              END
3170  16   11:0    20          END;  (* SETUNID2 *)
3171  16   11:0    36
3172  16   11:0    36
3173  16    3:0     0        BEGIN (* P010A03 *)
3174  16    3:1     0          FILLCHAR( UNIDENT, 32, 0);  (* UNIDENT[] SET TO FALSE        *)
3175  16    3:1     8                                      (* OBJECTS IDENTIFIED BY DEFAULT *)
3176  16    3:1     8          SETUNID2( 1, 20);
3177  16    3:1    12          SETUNID(  46);
3178  16    3:1    15          SETUNID2( 56, 58);
3179  16    3:1    19          SETUNID2( 61, 62);
3180  16    3:1    23          SETUNID(  65);
3181  16    3:1    26          SETUNID2( 75, 77);
3182  16    3:1    30          SETUNID2( 87, 90);
3183  16    3:1    34          SETUNID2( 95, 96);
3184  16    3:1    38          SETUNID2( 100, 119);
3185  16    3:1    42
3186  16    3:1    42          P010A09( 6);
3187  16    3:1    45          P010A05;
```

```
3188  16    3:1    47         IF OBJCOUNT = 0 THEN
3189  16    3:2    52           EXIT( P010A03);
3190  16    3:1    56         GETLOOT
3191  16    3:1    56
3192  16    3:0    56       END;  (* P010A03 *)
3193  16    3:0    70
3194  16    3:0    70
3195  16   12:D     1     PROCEDURE RDSPELLS;  (* P010A0C *)
3196  16   12:D     1
3197  16   12:D     1       VAR
3198  16   12:D     1            SPELLWIN : TWINDOWP;
3199  16   12:D     2            SPELLGRP : INTEGER;
3200  16   12:D     3            CHARX    : INTEGER;
3201  16   12:D     4
3202  16   12:D     4
3203  16   13:D     1       PROCEDURE PRSPELL( MAGPRST  : INTEGER;  (* P010A0D *)
3204  16   13:D     2                          SPELFRST : INTEGER;
3205  16   13:D     3                          SPELLAST : INTEGER);
3206  16   13:D     4
3207  16   13:D     4         VAR
3208  16   13:D     4              POSITION : INTEGER;
3209  16   13:D     5              SPELVERT : INTEGER;
3210  16   13:D     6              SPELHORZ : INTEGER;
3211  16   13:D     7              SPELLI   : INTEGER;
3212  16   13:D     8
3213  16   13:D     8
3214  16   13:0     0         BEGIN (* PRSPELL *)
3215  16   13:1     0           CLEARWIN( SPELLWIN, TRUE);
3216  16   13:1     7           POSITION := 0;
3217  16   13:1    10           FOR SPELLI := SPELFRST TO SPELLAST DO
3218  16   13:2    21             BEGIN
3219  16   13:3    21               GETMSGTX( BASE47, 5000 + SPELLI); (* *HALITO, MOGREF, KATINO...*)
3220  16   13:3    31               IF BASE47[ 1] = '*' THEN
3221  16   13:4    39                 BEGIN
3222  16   13:5    39                   SPELHORZ := 9 * (POSITION MOD 4);
3223  16   13:5    46                   SPELVERT := 7 * (POSITION DIV 4);
3224  16   13:5    53                   POSITION := POSITION + 1;
3225  16   13:5    58                   IF POSITION = 4 THEN           (* IS THIS A BUG!? *)
3226  16   13:6    63                     SPELHORZ := SPELHORZ + 1;
3227  16   13:5    68                   COPYSTR( BASE47, BASE47, 2, LENGTH( BASE47) - 1);
3228  16   13:4    82                 END;
3229  16   13:4    82
3230  16   13:3    82               IF CHARACTR[ CHARX].SPELLSKN[ SPELLI] THEN
3231  16   13:4    99                 BEGIN
3232  16   13:5    99                   MVCURSOR( SPELLWIN, SPELHORZ, SPELVERT);
3233  16   13:5   107                   PRPICMB(  SPELLWIN, BASE47);
3234  16   13:5   115                   SPELVERT := SPELVERT + 1
3235  16   13:4   116                 END;
3236  16   13:4   120
3237  16   13:2   120             END;
3238  16   13:2   127
3239  16   13:1   127           SPELVERT := SPELLWIN^.HEAD.VSIZE - 2;
3240  16   13:1   139           SPELHORZ := SPELLWIN^.HEAD.HSIZE - 2;
3241  16   13:1   151           GETMSGTX( BASE47, MAGPRST);
3242  16   13:1   157           MVCURSOR( SPELLWIN, SPELHORZ - MBSTRLEN( BASE47), SPELVERT - 2);
3243  16   13:1   175           PRPICMB( SPELLWIN, BASE47);
3244  16   13:1   183           GETMSGTX( BASE47, 900);  (* PRESS ~ *)
```

```
3245  16   13:1   191          MVCURSOR( SPELLWIN, SPELHORZ - MBSTRLEN( BASE47), SPELVERT - 1);
3246  16   13:1   209          PRPICMB( SPELLWIN, BASE47);
3247  16   13:1   217          GETCR;
3248  16   13:1   220          INCHAR := CHR( 0)
3249  16   13:0   221        END;  (* PRSPELL *)
3250  16   13:0   238
3251  16   13:0   238
3252  16   14:D     1      PROCEDURE PR7GSPEL( MP02 : INTEGER;  (* P010A0E *)
3253  16   14:D     2                          MP01 : TSPELL7G);
3254  16   14:D    10
3255  16   14:0     0        BEGIN (* PR7GSPEL *)
3256  16   14:1     0          DISP1MSG( SPELLWIN, MP02);
3257  16   14:1    12          DISP1MSG( SPELLWIN, 901);  (* SPELLS = *)
3258  16   14:1    21          PRINTNUM( SPELLWIN, MP01[ 1], 1);
3259  16   14:1    36          SPELLGRP := 1 + 1;
3260  16   14:1    42          WHILE SPELLGRP <= 7 DO
3261  16   14:2    49            BEGIN
3262  16   14:3    49              PRPICCH( SPELLWIN, '/');
3263  16   14:3    56              PRINTNUM( SPELLWIN, MP01[ SPELLGRP], 1);
3264  16   14:3    73              SPELLGRP := SPELLGRP + 1
3265  16   14:2    76            END;
3266  16   14:1    83          PRINTCR( SPELLWIN)
3267  16   14:0    86        END;  (* PR7GSPEL *)
3268  16   14:0   104
3269  16   14:0   104
3270  16   12:0     0      BEGIN (* RDSPELLS *)
3271  16   12:1     0        SPELLWIN := GETWIN( 0, 7, 40, 17, 15, FALSE);
3272  16   12:1    13        CHARX := BASE09;
3273  16   12:1    16        WITH CHARACTR[ CHARX] DO      (* MP04 *)
3274  16   12:2    24          BEGIN
3275  16   12:2    24
3276  16   12:3    24            REPEAT
3277  16   12:4    24              CLEARWIN( SPELLWIN, TRUE);
3278  16   12:4    29              PR7GSPEL( 902, MAGESP);    (*  MAGE *)
3279  16   12:4    37              PR7GSPEL( 903, PRIESTSP);  (* PRIEST *)
3280  16   12:4    45              MVCURSOR( SPELLWIN, 8, 4);
3281  16   12:4    51              SPELLGRP := MENUINDX( SPELLWIN, 904);
3282  16   12:4    62                 (* M)AGE SPELL BOOKS/P)RIEST SPELL BOOKS/L)EAVE *)
3283  16   12:4    62              CASE SPELLGRP OF
3284  16   12:4    65                0 : PRSPELL( 905,  1, 21);  (* MAGE SPELLS *)
3285  16   12:4    74                1 : PRSPELL( 906, 22, 50);  (* PRIEST SPELLS *)
3286  16   12:4    83              END;
3287  16   12:4    94
3288  16   12:3    94            UNTIL SPELLGRP = 2;  (* "L" *)
3289  16   12:3    99
3290  16   12:3    99            INCHAR := CHR( 0);
3291  16   12:3   102            DELWIN( SPELLWIN, FALSE);
3292  16   12:3   108            BASE13 := 25;
3293  16   12:3   111            BASE09 := CHARX;
3294  16   12:3   114            EXIT( P010A02)
3295  16   12:3   118
3296  16   12:2   118          END;
3297  16   12:2   118
3298  16   12:0   118      END;  (* RDSPELLS *)
3299  16   12:0   132
3300  16   12:0   132
3301  16   15:D     1    PROCEDURE KANDIFND;  (* P010A0F *)
```

```
3302  16   15:D     1
3303  16   15:D     1       VAR
3304  16   15:D     1            MP01 : INTEGER;
3305  16   15:D     2            MP02 : INTEGER;
3306  16   15:D     3            MP03 : INTEGER;
3307  16   15:D     4            MP04 : INTEGER;
3308  16   15:D     5            MP05 : INTEGER;
3309  16   15:D     6            SVBASE09 : INTEGER;
3310  16   15:D     7            MP07 : ARRAY[ 1..100] OF INTEGER;
3311  16   15:D   107            MP6B : ARRAY[ 1..4] OF INTEGER;
3312  16   15:D   111
3313  16   16:D     1       PROCEDURE EXITCAMP;  (* P010A10 *)
3314  16   16:D     1
3315  16   16:0     0         BEGIN (* EXITCAMP *)
3316  16   16:1     0           DELWIN( BASE10, FALSE);
3317  16   16:1     6           INCHAR := 'A';
3318  16   16:1     9           BASE09 := SVBASE09;
3319  16   16:1    14           BASE13 := 25;
3320  16   16:1    17           EXIT( P010A02)
3321  16   16:0    21         END;  (* EXITCAMP *)
3322  16   16:0    34
3323  16   16:0    34
3324  16   17:D     1       PROCEDURE EXTCMPCR;  (* P010A11 *)
3325  16   17:D     1
3326  16   17:0     0         BEGIN (* EXTCMPCR *)
3327  16   17:1     0           MVCURSOR( BASE10, 0, BASE10^.HEAD.VSIZE - 3);
3328  16   17:1    13           DISP1LIN( BASE10, 907);  (* PRESS ~ TO LEAVE *)
3329  16   17:1    20           GETCR;
3330  16   17:1    23           EXITCAMP
3331  16   17:0    23         END;  (* EXTCMPCR *)
3332  16   17:0    38
3333  16   17:0    38
3334  16   18:D     1       PROCEDURE FNDTREBR;  (* P010A12 *)
3335  16   18:D     1
3336  16   18:0     0         BEGIN (* FNDTREBR *)
3337  16   18:1     0           GETMSGTX( BASE47, 910);  (* IN THE *)
3338  16   18:1     8           MP02 := TREBORX;
3339  16   18:1    14           MP01 := TREBORY;
3340  16   18:1    20           IF MP01 > 9 THEN
3341  16   18:2    27             GETMSGTX( BASEC7, 911)   (* NORTH- *)
3342  16   18:1    33           ELSE
3343  16   18:2    38             GETMSGTX( BASEC7, 912);  (* SOUTH- *)
3344  16   18:2    47
3345  16   18:1    47           PCONCAT( BASE47, BASEC7);
3346  16   18:1    55
3347  16   18:1    55           IF MP02 > 9 THEN
3348  16   18:2    62             GETMSGTX( BASEC7, 913)   (* EAST   *)
3349  16   18:1    68           ELSE
3350  16   18:2    73             GETMSGTX( BASEC7, 914);  (* WEST   *)
3351  16   18:2    82
3352  16   18:1    82           PCONCAT( BASE47, BASEC7);
3353  16   18:1    90
3354  16   18:1    90           GETMSGTX( BASEC7, 915);    (* OF THIS LEVEL. *)
3355  16   18:1    99           PCONCAT( BASE47, BASEC7);
3356  16   18:1   107
3357  16   18:1   107           INT2STR( MP6B[ 3], BASEC7);
3358  16   18:1   122           INSERTST( BASEC7, BASE47);
```

```
3359  16   18:1   130           CENTSTR( BASE10, BASE47);
3360  16   18:1   136           EXTCMPCR
3361  16   18:0   136         END;  (* FNDTREBR *)
3362  16   18:0   150
3363  16   18:0   150
3364  16   15:0     0       BEGIN (* KANDIFND *)
3365  16   15:1     0         SVBASE09 := BASE09;
3366  16   15:1     3         BASE10 := GETWIN( 1, 7, 38, 6, 17, TRUE);
3367  16   15:1    16         DISP1LIN( BASE10, 916);    (* LOCATE SOMEONE *)
3368  16   15:1    23         PRINTCR( BASE10);
3369  16   15:1    27         DISP1MSG( BASE10, 917);    (* FIND WHOM ? > *)
3370  16   15:1    34         SOLICIT( BASE10, BASE47, 15);
3371  16   15:1    41         IF BASE47 = '' THEN
3372  16   15:2    50           EXITCAMP;
3373  16   15:1    52         CLEARWIN( BASE10, TRUE);
3374  16   15:1    57         GETMSGTX( BASEC7, 918);    (* ^ IS ... *)
3375  16   15:1    66         INSERTST( BASE47, BASEC7);
3376  16   15:1    74         CENTSTR( BASE10, BASEC7);
3377  16   15:1    81         IF BASE47 = 'TREBOR' THEN
3378  16   15:2    96           FNDTREBR
3379  16   15:1    96         ELSE
3380  16   15:2   100           BEGIN
3381  16   15:3   100             MP04 := 29000;  (* WERDNA, KADORTO, ADAMS, WOODHEAD, GREENBERG, *)
3382  16   15:3   105                             (* KILROY, WARTOW *)
3383  16   15:3   105             GETMSGTX( BASEC7, MP04);
3384  16   15:3   112             WHILE BASEC7 <> '**ERR**' DO
3385  16   15:4   129               BEGIN
3386  16   15:5   129                 IF BASEC7 = BASE47 THEN
3387  16   15:6   138                   BEGIN
3388  16   15:7   138                     DISP1LIN( BASE10, MP04 + 1);
3389  16   15:7   145                     EXTCMPCR
3390  16   15:6   145                   END;
3391  16   15:5   147                   MP04 := MP04 + 2;
3392  16   15:5   152                   GETMSGTX( BASEC7, MP04)
3393  16   15:4   156               END;
3394  16   15:2   161           END;
3395  16   15:1   161         DISP1LIN( BASE10, 908);   (* HUNTING YOU DOWN! *)
3396  16   15:1   168         EXTCMPCR
3397  16   15:0   168       END;  (* KANDIFND *)
3398  16   15:0   184
3399  16   15:0   184
3400  16   15:0   184
3401  16   19:D     1     PROCEDURE MALOR;  (* P010A13 *)
3402  16   19:D     1
3403  16   19:D     1       VAR
3404  16   19:D     1             DELTAUD : INTEGER;
3405  16   19:D     2             DELTANS : INTEGER;
3406  16   19:D     3             DELTAEW : INTEGER;
3407  16   19:D     4             MP04 : INTEGER;
3408  16   19:D     5             NSEWUD : INTEGER;
3409  16   19:D     6             SAVECAST : INTEGER;
3410  16   19:D     7             MALORWIN : TWINDOWP;
3411  16   19:D     8
3412  16   19:D     8
3413  16   20:D     1       PROCEDURE TELEPORT;  (* P010A14 *)
3414  16   20:D     1
3415  16   21:D     1         PROCEDURE ROCK;  (* P010A15 *)
```

```
3416  16   21:D     1
3417  16   21:D     1          VAR
3418  16   21:D     1              UNUSED : INTEGER;
3419  16   21:D     2
3420  16   21:0     0          BEGIN (* ROCK *)
3421  16   21:1     0            DISP1LIN( BASE10, 200);  (* YOU TELEPORTED INTO ROCK! *)
3422  16   21:1     7            PAUSE2;
3423  16   21:1    10            DELWIN( BASE10, FALSE);
3424  16   21:1    16            BASE13 := 14;
3425  16   21:1    19            EXIT( P010A02)
3426  16   21:0    23          END;  (* ROCK *)
3427  16   21:0    36
3428  16   21:0    36
3429  16   22:D     1        PROCEDURE BOUNCE;  (* P010A16 *)
3430  16   22:D     1
3431  16   22:0     0          BEGIN (* BOUNCE *)
3432  16   22:1     0            DISP1LIN( BASE10, 206);  (* YOU BOUNCED BACK TO WHERE YOU WERE! *)
3433  16   22:1     7            PAUSE2;
3434  16   22:1    10            DELWIN( BASE10, FALSE);
3435  16   22:1    16            EXIT( P010A02)
3436  16   22:0    20          END;  (* BOUNCE *)
3437  16   22:0    32
3438  16   22:0    32
3439  16   23:D     1        PROCEDURE P010A17( MP05 : INTEGER;
3440  16   23:D     2                           MP04 : INTEGER;
3441  16   23:D     3                           MP03 : INTEGER;
3442  16   23:D     4                           MP02 : INTEGER;
3443  16   23:D     5                           MP01 : INTEGER);
3444  16   23:D     6
3445  16   23:0     0          BEGIN (* P010A17 *)
3446  16   23:1     0            IF (MP05 = MP04) OR (MP05 = MP03) OR
3447  16   23:1     7               (MP05 = MP02) OR (MP05 = MP01)    THEN
3448  16   23:2    17              EXIT( P010A17);
3449  16   23:1    21            BOUNCE
3450  16   23:0    21          END;  (* P010A17 *)
3451  16   23:0    36
3452  16   23:0    36
3453  16   24:D     1        PROCEDURE OBLIVION;  (* P010A18 *)
3454  16   24:D     1
3455  16   24:D     1          VAR
3456  16   24:D     1               WERDNABL : INTEGER;
3457  16   24:D     2               DISKBLK  : INTEGER;
3458  16   24:D     3               BUFFER   : PACKED ARRAY[ 0..511] OF CHAR;
3459  16   24:D   259
3460  16   24:D   259
3461  16   25:D     1          PROCEDURE DISPMSGX( MSGX : INTEGER);  (* P010A19 *)
3462  16   25:D     2
3463  16   25:0     0            BEGIN (* DISPMSGX *)
3464  16   25:1     0              CLEARWIN( BASE10, TRUE);
3465  16   25:1     5              REPEAT
3466  16   25:2     5                GETMSGTX( BASE47, MSGX);
3467  16   25:2    11                IF BASE47 = '**ERR**' THEN
3468  16   25:3    27                  EXIT( DISPMSGX);
3469  16   25:2    31                CENTSTR( BASE10, BASE47);
3470  16   25:2    37                MSGX := MSGX + 1
3471  16   25:1    38              UNTIL FALSE
3472  16   25:0    42            END;  (* DISPMSGX *)
```

```
3473  16   25:0    60
3474  16   25:0    60
3475  16   24:0     0          BEGIN (* OBLIVION *)
3476  16   24:1     0            DELWIN( BASE10, FALSE);
3477  16   24:1     6            BASE10 := GETWIN( 0, 6, 40, 8, 17, TRUE);
3478  16   24:1    19            DISPMSGX( 2100);  (* ARE YOU REALLY SEEKING WHAT IS BELOW *)
3479  16   24:1    24                              (* LIFE ITSELF?  BEWARE, YOUR CHOICE IS *)
3480  16   24:1    24                              (* IRREVOCABLE. (A REAL SERIOUS HINT!)  *)
3481  16   24:1    24                              (* ---MAKE A BACKUP SAVE GAME DISKETTE  *)
3482  16   24:1    24                              (* ^BEFORE ATTEMPTING THIS!!!           *)
3483  16   24:1    24            GETMSGTX( BASE47, 2110);  (* YN *)
3484  16   24:1    32
3485  16   24:1    32            REPEAT
3486  16   24:2    32              GETKEY
3487  16   24:1    32            UNTIL (INCHAR = BASE47[ 1]) OR (INCHAR = BASE47[ 2]);
3488  16   24:1    50
3489  16   24:1    50            IF INCHAR = BASE47[ 2] THEN
3490  16   24:2    58              BEGIN
3491  16   24:3    58                DELWIN( BASE10, TRUE);
3492  16   24:3    64                EXIT( P010A02)
3493  16   24:2    68              END;
3494  16   24:2    68
3495  16   24:1    68            FILLCHAR( BUFFER, 512, 0);
3496  16   24:1    77            WERDNABL := FINDFILE( DRIVE1, 'WERDNA.DATA');
3497  16   24:1    99            FOR DISKBLK := WERDNABL TO WERDNABL + 7 DO
3498  16   24:2   115              UNITWRITE( DRIVE1, BUFFER, 512, DISKBLK);
3499  16   24:1   133            DISPMSGX( 2120);  (* YOU HAVE FOUND THE TOTAL OBLIVION *)
3500  16   24:1   138                              (* YOU HAVE FOR SO LONG BEEN SEARCH- *)
3501  16   24:1   138                              (* ING.  ALL TRACES OF YOUR EXISTENCE *)
3502  16   24:1   138                              (* HAVE BEEN OBLITERATED!            *)
3503  16   24:1   138                              (* PRESS ~ TO REST FOREVER!          *)
3504  16   24:1   138            GETCR;
3505  16   24:1   141            DELWIN( BASE10, TRUE);
3506  16   24:1   147            BASE13 := 23;
3507  16   24:1   150            EXIT( P010A02)
3508  16   24:0   154          END;  (* OBLIVION *)
3509  16   24:0   170
3510  16   24:0   170
3511  16   20:0     0        BEGIN (* TELEPORT *)
3512  16   20:1     0          DELWIN( MALORWIN, FALSE);
3513  16   20:1     7          DELWIN( CAMPWIN, FALSE);
3514  16   20:1    13          DELWIN( CAMPTIT, FALSE);
3515  16   20:1    19          CLEARWIN( BASE10, TRUE);
3516  16   20:1    24          BASE13 := 7;
3517  16   20:1    27          BASE09 := MAZELEV;
3518  16   20:1    30          IF BASE09 > 12 THEN
3519  16   20:2    35            BASE09 := 13 - BASE09;
3520  16   20:1    40          MP04 := BASE09 - DELTAUD;
3521  16   20:1    48          IF MP04 < 1 THEN
3522  16   20:2    55            MP04 := 13 - MP04;
3523  16   20:1    63          IF BASE80B = 1 THEN
3524  16   20:2    70            BEGIN
3525  16   20:3    70              CASE MAZELEV OF
3526  16   20:3    73
3527  16   20:3    73                11 :
3528  16   20:4    73                     BEGIN
3529  16   20:5    73                       IF (MP04 < 1) OR (MP04 > 12) THEN
```

```
3530  16   20:6    86                        BOUNCE
3531  16   20:5    86                      ELSE
3532  16   20:6    90                        IF MP04 = 12 THEN
3533  16   20:7    97                          IF (MAZEX = 10) AND (MAZEY = 9) THEN
3534  16   20:8   106                            BEGIN
3535  16   20:9   106                              DELTAEW := 0;
3536  16   20:9   110                              DELTANS := -9
3537  16   20:8   110                            END
3538  16   20:7   115                          ELSE
3539  16   20:8   117                            BOUNCE
3540  16   20:4   117                  END;
3541  16   20:4   121
3542  16   20:3   121              2, 3, 4, 5, 6, 7, 8, 9, 10 :
3543  16   20:4   121                  BEGIN
3544  16   20:5   121                    IF (MP04 < 1) OR (MP04 > 11) THEN
3545  16   20:6   134                      BOUNCE
3546  16   20:4   134                  END;
3547  16   20:4   138
3548  16   20:3   138              1 :
3549  16   20:4   138                  BEGIN
3550  16   20:5   138                    IF (MP04 < 2) OR (MP04 > 11) THEN
3551  16   20:6   151                      BOUNCE
3552  16   20:4   151                  END;
3553  16   20:4   155
3554  16   20:3   155              12 :
3555  16   20:4   155                  BEGIN
3556  16   20:5   155                    IF MP04 > 12 THEN
3557  16   20:6   162                      OBLIVION
3558  16   20:5   162                    ELSE
3559  16   20:6   166                      IF MP04 = 11 THEN
3560  16   20:7   173                        BEGIN
3561  16   20:8   173                          DELTAEW := - MAZEX + 10;
3562  16   20:8   180                          DELTANS := - MAZEY + 9
3563  16   20:7   182                        END
3564  16   20:6   187                      ELSE
3565  16   20:7   189                        BOUNCE
3566  16   20:4   189                  END;
3567  16   20:4   193
3568  16   20:3   193              13, 14 :
3569  16   20:4   193                  BEGIN
3570  16   20:5   193                    BOUNCE
3571  16   20:4   193                  END;
3572  16   20:4   197
3573  16   20:3   197            END; (* CASE *)
3574  16   20:3   232
3575  16   20:2   232          END
3576  16   20:1   232        ELSE
3577  16   20:1   234          (* 5B84 *)
3578  16   20:2   234          BEGIN
3579  16   20:3   234            CASE MAZELEV OF
3580  16   20:3   237
3581  16   20:3   237              11 :
3582  16   20:4   237                  BEGIN
3583  16   20:5   237                    IF MP04 <> 12 THEN
3584  16   20:6   244                      BOUNCE
3585  16   20:5   244                    ELSE
3586  16   20:6   248                      IF (MAZEX = 10) AND (MAZEY = 9) THEN
```

```
3587  16   20:7   257                          BEGIN
3588  16   20:8   257                            DELTAEW := 0;
3589  16   20:8   261                            DELTANS := - 9
3590  16   20:7   261                          END
3591  16   20:6   266                        ELSE
3592  16   20:7   268                          BOUNCE
3593  16   20:4   268                    END;
3594  16   20:4   272
3595  16   20:3   272                10 :
3596  16   20:4   272                    BEGIN
3597  16   20:5   272                      IF MP04 <> 11 THEN
3598  16   20:6   279                        BOUNCE
3599  16   20:4   279                    END;
3600  16   20:4   283
3601  16   20:3   283                5, 6, 7, 8, 9 :
3602  16   20:4   283                    BEGIN
3603  16   20:5   283                      IF (MP04 <> 11) AND (MP04 <> 10) THEN
3604  16   20:6   296                        BOUNCE
3605  16   20:4   296                    END;
3606  16   20:4   300
3607  16   20:3   300                2, 3, 4, 13, 14 :
3608  16   20:4   300                    BEGIN
3609  16   20:5   300                      BOUNCE
3610  16   20:4   300                    END;
3611  16   20:4   304
3612  16   20:3   304                12 :
3613  16   20:4   304                    BEGIN
3614  16   20:5   304                      IF MP04 > 12 THEN
3615  16   20:6   311                        OBLIVION
3616  16   20:5   311                      ELSE
3617  16   20:6   315                        IF MP04 = 11 THEN
3618  16   20:7   322                          BEGIN
3619  16   20:8   322                            DELTAEW := - MAZEX + 10;
3620  16   20:8   329                            DELTANS := - MAZEY + 9
3621  16   20:7   331                          END
3622  16   20:6   336                        ELSE
3623  16   20:7   338                          BOUNCE
3624  16   20:4   338                    END;
3625  16   20:4   342
3626  16   20:3   342              END;  (* CASE *)
3627  16   20:3   376
3628  16   20:2   376            END;
3629  16   20:2   376
3630  16   20:2   376          (* 5C12 *)
3631  16   20:1   376            IF (MAZELEV - DELTAUD) = 12 THEN
3632  16   20:2   385              BEGIN
3633  16   20:3   385                SAVEX   := MAZEX;
3634  16   20:3   388                SAVEY   := MAZEY;
3635  16   20:3   391                SAVELEV := MAZELEV
3636  16   20:2   391              END;
3637  16   20:1   394            MAZEX   := MAZEX + DELTAEW;
3638  16   20:1   401            MAZEY   := MAZEY + DELTANS;
3639  16   20:1   408            MAZELEV := MP04;
3640  16   20:1   413
3641  16   20:1   413            IF MAZELEV <> 12 THEN
3642  16   20:2   418              BEGIN
3643  16   20:3   418                SAVEX   := MAZEX;
```

```
3644  16   20:3   421              SAVEY   := MAZEY;
3645  16   20:3   424              SAVELEV := MAZELEV
3646  16   20:2   424            END;
3647  16   20:2   427
3648  16   20:1   427          IF MAZELEV = 1 THEN
3649  16   20:2   432            BEGIN
3650  16   20:3   432              MAZEX := 0;
3651  16   20:3   435              MAZEY := 0
3652  16   20:2   435            END;
3653  16   20:2   438
3654  16   20:1   438          IF ((MAZEX < 0) OR (MAZEX > 19) OR
3655  16   20:1   445              (MAZEY < 0) OR (MAZEY > 19) OR
3656  16   20:1   453              (MAZELEV > SCENARIO.RECPERDK[ 1] - 2))
3657  16   20:1   465
3658  16   20:1   465                AND
3659  16   20:1   465
3660  16   20:1   465              (MAZELEV > 0) THEN
3661  16   20:1   471
3662  16   20:2   471            ROCK
3663  16   20:1   471          ELSE
3664  16   20:2   475            IF MAZELEV < 1 THEN
3665  16   20:3   480              ROCK;
3666  16   20:3   482
3667  16   20:1   482          DELWIN( BASE10, TRUE);
3668  16   20:1   488          EXIT( P010A02)
3669  16   20:1   492
3670  16   20:0   492        END;  (* TELEPORT *)
3671  16   20:0   508
3672  16   20:0   508
3673  16   19:0     0      BEGIN (* MALOR *)
3674  16   19:1     0        SAVECAST := BASE09;
3675  16   19:1     3        BASE10 := GETWIN( 0, 7, 40, 6, 16, TRUE);
3676  16   19:1    16        DISP1LIN( BASE10, 207);   (* TELEPORT *)
3677  16   19:1    23        PRINTCR( BASE10);
3678  16   19:1    27        DISP1MSG( BASE10, 208);   (* EAST =  0 NORTH =  0 UP =  0 *)
3679  16   19:1    34        MALORWIN := GETWIN( 5, 12, 12, 10, 17, TRUE);
3680  16   19:1    47        DELTAEW := 0;
3681  16   19:1    50        DELTANS := 0;
3682  16   19:1    53        DELTAUD := 0;
3683  16   19:1    56
3684  16   19:1    56        REPEAT
3685  16   19:2    56          MVCURSOR( BASE10, 8, 2);
3686  16   19:2    62          PRINTNUM( BASE10, DELTAEW, 4);
3687  16   19:2    68          MVCURSOR( BASE10, 22, 2);
3688  16   19:2    74          PRINTNUM( BASE10, DELTANS, 4);
3689  16   19:2    80          MVCURSOR( BASE10, 33, 2);
3690  16   19:2    86          PRINTNUM( BASE10, DELTAUD, 4);
3691  16   19:2    92          MVCURSOR( MALORWIN, 0, 0);
3692  16   19:2    98          NSEWUD := MENUINDX( MALORWIN,  209);
3693  16   19:2   109                  (* E)AST/W)EST/N)ORTH/S)OUTH/U)P/D)OWN/T)ELEPORT/Q)UIT *)
3694  16   19:2   109
3695  16   19:2   109          CASE NSEWUD OF
3696  16   19:2   112            2 : DELTANS := DELTANS + 1;
3697  16   19:2   119            3 : DELTANS := DELTANS - 1;
3698  16   19:2   126            0 : DELTAEW := DELTAEW + 1;
3699  16   19:2   133            1 : DELTAEW := DELTAEW - 1;
3700  16   19:2   140            5 : DELTAUD := DELTAUD - 1;
```

```
3701  16   19:2   147            4 : DELTAUD := DELTAUD + 1;
3702  16   19:2   154            6 : TELEPORT;
3703  16   19:2   158          END;
3704  16   19:2   180
3705  16   19:1   180        UNTIL NSEWUD > 5;
3706  16   19:1   185
3707  16   19:1   185        DELWIN( MALORWIN, FALSE);
3708  16   19:1   191        DELWIN( BASE10, TRUE);
3709  16   19:1   197        BASE13 := 25;
3710  16   19:1   200        BASE09 := SAVECAST;
3711  16   19:1   203        EXIT( P010A02)
3712  16   19:1   207
3713  16   19:0   207      END;  (* MALOR *)
3714  16   19:0   222
3715  16   19:0   222
3716  16   26:D     1    PROCEDURE DUMAPIC;  (* P010A1A *)
3717  16   26:D     1
3718  16   26:D     1      VAR
3719  16   26:D     1          LEVLCALC : INTEGER;
3720  16   26:D     2          SAVBASE9 : INTEGER;
3721  16   26:D     3          UNUSED   : ARRAY[ 1..41] OF INTEGER;
3722  16   26:D    44
3723  16   26:0     0      BEGIN (* DUMAPIC *)
3724  16   26:1     0        BASE13 := 25;
3725  16   26:1     3        SAVBASE9 := BASE09;
3726  16   26:1     6        BASE10 := GETWIN( 1, 7, 38, 9, 16, TRUE);
3727  16   26:1    19        DISP1LIN( BASE10, 21);  (* YOUR POSITION:  *)
3728  16   26:1    24        PRINTCR( BASE10);
3729  16   26:1    28        GETMSGTX( BASE47, 22);  (* FACING ^ *)
3730  16   26:1    34        GETMSGTX( BASEC7, 23 + DIRECTIO);
3731  16   26:1    43        INSERTST( BASEC7, BASE47);
3732  16   26:1    51        CENTSTR( BASE10, BASE47);
3733  16   26:1    57        PRINTCR( BASE10);
3734  16   26:1    61        GETMSGTX( BASE47, 27);  (* ! EAST " NORTH # DOWN. *)
3735  16   26:1    67        INT2STR( MAZEX, BASEC7);
3736  16   26:1    74        INSERT2( BASEC7, BASE47, '!');
3737  16   26:1    83        INT2STR( MAZEY, BASEC7);
3738  16   26:1    90        INSERT2( BASEC7, BASE47, '"');
3739  16   26:1    99
3740  16   26:1    99        IF MAZELEV = 13 THEN
3741  16   26:2   104          LEVLCALC := -1
3742  16   26:1   104        ELSE IF MAZELEV = 14 THEN
3743  16   26:3   115          LEVLCALC := -2
3744  16   26:2   115        ELSE
3745  16   26:3   121          LEVLCALC := MAZELEV - 1;
3746  16   26:3   126
3747  16   26:1   126        INT2STR( LEVLCALC, BASEC7);
3748  16   26:1   133        INSERT2( BASEC7, BASE47, '#');
3749  16   26:1   142        CENTSTR( BASE10, BASE47);
3750  16   26:1   148        PRINTCR( BASE10);
3751  16   26:1   152        DISP1LIN( BASE10, 250);  (* PRESS ~ *)
3752  16   26:1   159        DELWIN( BASE10, FALSE);
3753  16   26:1   165        GETCR;
3754  16   26:1   168        INCHAR := 'A';
3755  16   26:1   171        BASE09 := SAVBASE9;
3756  16   26:1   174        EXIT( P010A02)
3757  16   26:0   178      END;  (* DUMAPIC *)
```

```
3758  16   26:0   190
3759  16   26:0   190
3760  16    2:0     0     BEGIN (* P010A02 *)
3761  16    2:0     0
3762  16    2:1     0       CASE BASE13 OF
3763  16    2:1     4
3764  16    2:1     4         22:  CASE BASE26 OF
3765  16    2:2     8                0:  RDSPELLS;
3766  16    2:2    12                2:  KANDIFND;
3767  16    2:2    16                3:  DUMAPIC;
3768  16    2:2    20                4:  MALOR;
3769  16    2:2    24              END;
3770  16    2:2    44
3771  16    2:1    44          9: WITH CHARACTR[ 6] DO
3772  16    2:3    52               BEGIN
3773  16    2:4    52                 IF STATUS <= ASLEEP THEN
3774  16    2:5    59                   BEGIN
3775  16    2:6    59                     STATUS := OK;
3776  16    2:6    64                     BASE13 := 5;
3777  16    2:6    67                     P010A03
3778  16    2:5    67                   END
3779  16    2:4    69                 ELSE
3780  16    2:5    71                   BASE13 := 14
3781  16    2:3    71               END;
3782  16    2:1    76         20: BEGIN
3783  16    2:3    76               BASE09 := 0;
3784  16    2:3    79               BASE13 := 21
3785  16    2:2    79             END;
3786  16    2:2    84
3787  16    2:1    84       END; (* CASE *)
3788  16    2:1   120
3789  16    2:0   120     END;  (* P010A02 *)
3790  16    2:0   132
3791  16    2:0   132
3792  16    2:0   132 (*$I WIZ4C:CAMP  *)
3792  16    2:0   132 (*$I WIZ4C:CAMP2 *)
3793  16    2:0   132
3794  16    2:0   132 (* CAMP2 *)
3795  16    2:0   132
3796  16   27:D     1   PROCEDURE P010A1B;
3797  16   27:D     1
3798  16   27:D     1     VAR
3799  16   27:D     1          P1BMP01  : INTEGER;
3800  16   27:D     2          OBJIDS   : ARRAY[ 0..8] OF INTEGER;
3801  16   27:D    11          CURSEDXX : ARRAY[ 0..8] OF BOOLEAN;
3802  16   27:D    20          CANUSE   : ARRAY[ 0..8] OF BOOLEAN;
3803  16   27:D    29          OBJNAMES : ARRAY[ 0..8] OF ARRAY[ FALSE..TRUE] OF STRING[ 15];
3804  16   27:D   173
3805  16   28:D     1     PROCEDURE P010A1C( MP03 : INTEGER;
3806  16   28:D     2                        MP02 : INTEGER;
3807  16   28:D     3                        MP01 : BOOLEAN);
3808  16   28:D     4
3809  16   28:0     0       BEGIN (* P010A1C *)
3810  16   28:1     0         BASE09 := MP03;
3811  16   28:1     3         BASE26 := MP02;
3812  16   28:1     6         BASE13 := 22;
3813  16   28:1     9         IF MP01 THEN
```

```
3814  16   28:2    12           DELWIN( BASE10, FALSE);
3815  16   28:1    18         EXIT( P010A1B)
3816  16   28:0    22       END;  (* P010A1C *)
3817  16   28:0    34
3818  16   28:0    34
3819  16   29:D     1     PROCEDURE P010A1D( MP01 : TWINDOWP);
3820  16   29:D     2
3821  16   29:0     0       BEGIN (* P010A1D *)
3822  16   29:1     0         DISP1LIN( MP01, 600)    (* ~ TO LEAVE *)
3823  16   29:0     4       END;  (* P010A1D *)
3824  16   29:0    20
3825  16   29:0    20
3826  16   30:D     1     PROCEDURE P010A1E( MP01 : STRING);
3827  16   30:D    43
3828  16   30:0     0       BEGIN (* P010A1E *)
3829  16   30:1     0         IF (MP01 = ' ') OR (MP01 = '') THEN
3830  16   30:2    20           EXIT( P010A1E);
3831  16   30:1    24         CLEARWIN( BASE10, TRUE);
3832  16   30:1    29         MVCURSOR( BASE10, 0, 1);
3833  16   30:1    35         BASE47 := '* ';
3834  16   30:1    44         PCONCAT2( BASE47, MP01, ' *');
3835  16   30:1    56         CENTSTR( BASE10, BASE47);
3836  16   30:1    62         PAUSE2
3837  16   30:0    62       END;  (* P010A1E *)
3838  16   30:0    78
3839  16   30:0    78
3840  16   31:D     1     PROCEDURE INSPECT;  (* P010A1F *)
3841  16   31:D     1
3842  16   31:D     1       VAR
3843  16   31:D     1           UNUSED : INTEGER;
3844  16   31:D     2           CAMPCHAR : INTEGER;
3845  16   31:D     3           MP03 : INTEGER;
3846  16   31:D     4
3847  16   31:D     4
3848  16   32:D     1       PROCEDURE DSPSPELS;  (* P010A20 *)
3849  16   32:D     1
3850  16   32:D     1         VAR
3851  16   32:D     1             INDX : INTEGER;
3852  16   32:D     2
3853  16   32:0     0         BEGIN (* DSPSPELS *)
3854  16   32:1     0           WITH CHARACTR[ CAMPCHAR] DO
3855  16   32:2    10             BEGIN
3856  16   32:3    10               MVCURSOR( CAMPWIN, 0, 9);
3857  16   32:3    16               DISP1MSG( CAMPWIN, 603);  (* MAGE *)
3858  16   32:3    23               FOR INDX := 1 TO 7 DO
3859  16   32:4    34                 BEGIN
3860  16   32:5    34                   PRINTNUM( CAMPWIN, MAGESP[ INDX], 1);
3861  16   32:5    48                   IF INDX < 7 THEN
3862  16   32:6    53                     PRPICCH( CAMPWIN, '/')
3863  16   32:4    55                 END;
3864  16   32:4    65
3865  16   32:3    65               DISP1MSG( CAMPWIN, 604);  (* PRST *)
3866  16   32:3    72               FOR INDX := 1 TO 7 DO
3867  16   32:4    83                 BEGIN
3868  16   32:5    83                   PRINTNUM( CAMPWIN, PRIESTSP[ INDX], 1);
3869  16   32:5    97                   IF INDX < 7 THEN
3870  16   32:6   102                     PRPICCH( CAMPWIN, '/')
```

```
3871  16  32:4   104                END
3872  16  32:2   107             END;
3873  16  32:0   114          END;  (* DSPSPELS *)
3874  16  32:0   130
3875  16  32:0   130
3876  16  33:D     1        PROCEDURE DSPITEMS;  (* P010A21 *)
3877  16  33:D     1
3878  16  33:D     1          VAR
3879  16  33:D     1               ITEMX   : INTEGER;
3880  16  33:D     2               HPOSNUM : INTEGER;
3881  16  33:D     3               OBJECT  : TOBJREC;
3882  16  33:D    26
3883  16  33:D    26
3884  16  34:D     1          PROCEDURE P010A22( MP01 : INTEGER);
3885  16  34:D     2
3886  16  34:D     2            VAR
3887  16  34:D     2                 MP02 : INTEGER;
3888  16  34:D     3                 MP03 : INTEGER;
3889  16  34:D     4
3890  16  34:0     0            BEGIN (* P010A22 *)
3891  16  34:1     0              FOR MP03 := 1 TO 5 DO
3892  16  34:2    11                BASE7DC[ MP03 - 1] :=  - BASE7DC[ MP03 - 1];
3893  16  34:1    37              MP02 := 1;
3894  16  34:1    40              WITH CHARACTR[ MP01].POSS DO
3895  16  34:2    50                BEGIN
3896  16  34:3    50                  FOR MP03 := 1 TO POSSCNT DO
3897  16  34:4    62                    BEGIN
3898  16  34:5    62                      POSSESS[ MP02] := POSSESS[ MP03];
3899  16  34:5    80                      WITH POSSESS[ MP02] DO
3900  16  34:6    90                        BEGIN
3901  16  34:7    90                          IF EQINDEX <> (0 + 0) THEN
3902  16  34:8    98                            BEGIN
3903  16  34:9    98                              IF EQINDEX <= 5 THEN
3904  16  34:0   104                                 IF BASE7DC[ EQINDEX - 1] = 0 THEN
3905  16  34:1   118                                   BASE7DC[ EQINDEX - 1] := 1
3906  16  34:0   127                                 ELSE
3907  16  34:1   131                                   BASE7DC[ EQINDEX - 1] :=
3908  16  34:1   140                                     ABS( BASE7DC[ EQINDEX - 1]);
3909  16  34:9   152                               MP02 := MP02 + 1
3910  16  34:8   153                             END;
3911  16  34:6   157                        END; (* WITH *)
3912  16  34:4   157                    END;
3913  16  34:4   164
3914  16  34:3   164                  MP02 := MP02 - 1;
3915  16  34:3   169                  IF MP02 < POSSCNT THEN
3916  16  34:4   175                    BEGIN
3917  16  34:5   175                      FOR MP03 := 1 TO 8 DO
3918  16  34:6   186                        OBJIDS[ MP03] := -1;
3919  16  34:5   202                      POSSCNT := MP02
3920  16  34:4   203                    END;
3921  16  34:4   205
3922  16  34:2   205                END;  (* WITH *)
3923  16  34:2   205
3924  16  34:1   205                FOR MP03 := 1 TO 5 DO
3925  16  34:2   216                  IF BASE7DC[ MP03 - 1] = -1 THEN
3926  16  34:3   230                    BASE7DC[ MP03 - 1] := 0
3927  16  34:2   238                  ELSE
```

```
3928  16   34:3   242                    BASE7DC[ MP03 - 1] := ABS( BASE7DC[ MP03 - 1])
3929  16   34:3   260
3930  16   34:0   260            END;  (* P010A22 *)
3931  16   34:0   288
3932  16   34:0   288
3933  16   33:0     0          BEGIN (* DSPITEMS *)
3934  16   33:1     0            P010A22( CAMPCHAR);
3935  16   33:1     5
3936  16   33:1     5            WITH CHARACTR[ CAMPCHAR] DO
3937  16   33:2    15            BEGIN
3938  16   33:2    15
3939  16   33:3    15            FOR ITEMX := 1 TO 8 DO
3940  16   33:4    27              BEGIN
3941  16   33:5    27                HPOSNUM := 19 + 19 * ((ITEMX - 1) MOD 2);
3942  16   33:5    38                MVCURSOR( CAMPWIN, HPOSNUM - 19, 11 + (ITEMX - 1) DIV 2);
3943  16   33:5    52                IF ITEMX <= POSS.POSSCNT THEN
3944  16   33:6    60                  BEGIN
3945  16   33:6    60
3946  16   33:7    60                    WITH POSS.POSSESS[ ITEMX] DO
3947  16   33:8    71                    BEGIN
3948  16   33:8    71
3949  16   33:9    71                    IF EQINDEX >= 0 THEN
3950  16   33:0    78                      BEGIN
3951  16   33:1    78                        IF OBJIDS[ ITEMX] <> EQINDEX THEN
3952  16   33:2    91                          BEGIN
3953  16   33:2    91
3954  16   33:3    91                            MOVELEFT( BASE55B[
3955  16   33:3    94                                        GETREC( 4,
3956  16   33:3    95                                                POSS.POSSESS[ ITEMX].EQINDEX - 0,
3957  16   33:3   107                                                 SIZEOF( TOBJREC))],
3958  16   33:3   113                                        OBJECT,
3959  16   33:3   116                                        SIZEOF( TOBJREC));
3960  16   33:3   119
3961  16   33:3   119                            IF BASCIIHU THEN
3962  16   33:4   124                              BEGIN
3963  16   33:5   124                                UNITWRITE( BASE12, OBJECT, 33,
3964  16   33:5   130                                           GETADDR(BASE89B));
3965  16   33:5   141                                MOVELEFT( BASE89B[ 1], OBJECT, SIZEOF( TOBJREC))
3966  16   33:4   151                              END;
3967  16   33:4   151
3968  16   33:3   151                            BASE09 := EQINDEX;
3969  16   33:3   156                            OBJIDS[ ITEMX] := BASE09;
3970  16   33:3   164
3971  16   33:3   164                            BASE09 := 14000 + BASE09 + BASE09 - 0;
3972  16   33:3   175                              (* "BLOODSTONE", "LANDER'S TURQ.", ETC.) *)
3973  16   33:3   175
3974  16   33:3   175                              (* KNOWN *)
3975  16   33:3   175                            GETMSGTX( OBJNAMES[ ITEMX][ TRUE], BASE09 + 1);
3976  16   33:3   192                              (* UNKNOWN *)
3977  16   33:3   192                            GETMSGTX( OBJNAMES[ ITEMX][ FALSE], BASE09);
3978  16   33:3   207
3979  16   33:3   207                            CANUSE[ ITEMX] := OBJECT.CLASSUSE[ CLASS];
3980  16   33:3   224                            CURSEDXX[ ITEMX] := OBJECT.CURSED
3981  16   33:2   230                          END;
3982  16   33:2   232
3983  16   33:1   232                        PRINTNUM( CAMPWIN, ITEMX, 1);
3984  16   33:1   238                        PRPICCH( CAMPWIN, ')');
```

```
3985  16   33:1   243                      IF EQUIPED THEN
3986  16   33:2   248                        IF CURSEDXX[ ITEMX] THEN
3987  16   33:3   257                          PRPICCH( CAMPWIN, '-')
3988  16   33:2   259                        ELSE
3989  16   33:3   264                          PRPICCH( CAMPWIN, '*')
3990  16   33:1   266                      ELSE
3991  16   33:2   271                        IF CANUSE[ ITEMX] THEN
3992  16   33:3   280                          PRPICCH( CAMPWIN, ' ')
3993  16   33:2   282                        ELSE
3994  16   33:3   287                          PRPICCH( CAMPWIN, '#');  (* #$23 *)
3995  16   33:3   292
3996  16   33:1   292                      PRPICMB( CAMPWIN,
3997  16   33:1   293                               OBJNAMES[ ITEMX][ TRUE])
3998  16   33:0   304                    END;
3999  16   33:0   307
4000  16   33:8   307                  END; (* WITH *)
4001  16   33:6   307                END;
4002  16   33:6   307
4003  16   33:5   307              IF CAMPWIN^.HEAD.HCURSOR = 0 THEN
4004  16   33:6   317                PRPICCH( CAMPWIN, ' ');
4005  16   33:6   322
4006  16   33:5   322              WHILE (CAMPWIN^.HEAD.HCURSOR <> 0) AND
4007  16   33:5   330                    (CAMPWIN^.HEAD.HCURSOR < HPOSNUM) DO
4008  16   33:6   341                PRPICCH( CAMPWIN, ' ')
4009  16   33:6   343
4010  16   33:4   343            END;  (* FOR *)
4011  16   33:4   355
4012  16   33:2   355          END; (* WITH *)
4013  16   33:2   355
4014  16   33:0   355        END;  (* DSPITEMS *)
4015  16   33:0   380
4016  16   33:0   380
4017  16   35:D     1      PROCEDURE DSPSTATS;  (* P010A23 *)
4018  16   35:D     1
4019  16   35:0     0        BEGIN (* DSPSTATS *)
4020  16   35:1     0          WITH CHARACTR[ CAMPCHAR] DO
4021  16   35:2    10            BEGIN
4022  16   35:3    10              MVCURSOR( CAMPWIN, 0, 2);
4023  16   35:3    16              DISP1MSG( CAMPWIN, 605);  (* STRENGTH *)
4024  16   35:3    23              PRINTNUM( CAMPWIN, ATTRIB[ STRENGTH], 3);
4025  16   35:3    36              DISP1MSG( CAMPWIN, 606);  (* GOLD     *)
4026  16   35:3    43              PRNTLONG( CAMPWIN, GOLD);
4027  16   35:3    50              DISP1MSG( CAMPWIN, 607);  (* LEV      *)
4028  16   35:3    57              PRINTNUM( CAMPWIN, CHARLEV, 4);
4029  16   35:3    65              DISP1MSG( CAMPWIN, 608);  (*  I.Q.    *)
4030  16   35:3    72              PRINTNUM( CAMPWIN, ATTRIB[ IQ], 3);
4031  16   35:3    85              DISP1MSG( CAMPWIN, 609);  (* KEYS     *)
4032  16   35:3    92              PRNTLONG( CAMPWIN, EXP);  (* ????     *)
4033  16   35:3    99              DISP1MSG( CAMPWIN, 610);  (* AGE      *)
4034  16   35:3   106              PRINTNUM( CAMPWIN, AGE DIV 52, 4);
4035  16   35:3   116              DISP1MSG( CAMPWIN, 611);  (*  PIETY   *)
4036  16   35:3   123              PRINTNUM( CAMPWIN, ATTRIB[ PIETY], 3);
4037  16   35:3   136              PRINTCR(  CAMPWIN);
4038  16   35:3   140
4039  16   35:3   140              DISP1MSG( CAMPWIN, 612);  (* VITALITY *)
4040  16   35:3   147              PRINTNUM( CAMPWIN, ATTRIB[ VITALITY], 3);
4041  16   35:3   160              DISP1MSG( CAMPWIN, 613);  (*  H.P.    *);
```

```
4042  16   35:3   167              PRINTNUM( CAMPWIN, HPLEFT, 5);
4043  16   35:3   175              PRPICCH(  CAMPWIN, '/');
4044  16   35:3   180              PRINTNUM( CAMPWIN, HPMAX, 4);
4045  16   35:3   188              DISP1MSG( CAMPWIN, 614);  (*   A.C.   *);
4046  16   35:3   195              PRINTNUM( CAMPWIN, ARMORCL - ACMOD2, 3);
4047  16   35:3   206              DISP1MSG( CAMPWIN, 615);  (* AGILITY  *)
4048  16   35:3   213              PRINTNUM( CAMPWIN, ATTRIB[ AGILITY], 3);
4049  16   35:3   226              PRINTCR ( CAMPWIN);
4050  16   35:3   230
4051  16   35:3   230              DISP1MSG( CAMPWIN, 616);  (*  LUCK    *)
4052  16   35:3   237              PRINTNUM( CAMPWIN, ATTRIB[ LUCK], 3);
4053  16   35:3   250              DISP1MSG( CAMPWIN, 617);  (* STATUS   *)
4054  16   35:3   257              DISP1MSG( CAMPWIN, 10040 + ORD( STATUS));  (* OK,...,LOST *)
4055  16   35:3   268              IF LOSTXYL.POISNAMT[ 1] > 0 THEN
4056  16   35:4   281                DISP1MSG( CAMPWIN, 618);  (* (POISON) *)
4057  16   35:4   288
4058  16   35:3   288              WHILE CAMPWIN^.HEAD.HCURSOR <> 0 DO
4059  16   35:4   298                 PRPICCH( CAMPWIN, ' ')
4060  16   35:4   300
4061  16   35:2   300            END;
4062  16   35:2   305
4063  16   35:0   305          END;  (* DSPSTATS *)
4064  16   35:0   320
4065  16   35:0   320
4066  16   35:0   320
4067  16   35:0   320
4068  16   35:0   320
4069  16   36:D     1      PROCEDURE CASTWHAT( SPELHASH : INTEGER);  (* P010A24 *)
4070  16   36:D     2
4071  16   36:D     2      {
4072  16   36:D     2        CONST
4073  16   36:D     2
4074  16   36:D     2              HALITO   =  4178;
4075  16   36:D     2              MOGREF   =  2409;
4076  16   36:D     2              KATINO   =  3983;
4077  16   36:D     2              DUMAPIC  =  3245;
4078  16   36:D     2
4079  16   36:D     2              DILTO    =  3340;
4080  16   36:D     2              SOPIC    =  1953;
4081  16   36:D     2
4082  16   36:D     2              MAHALITO =  6181;
4083  16   36:D     2              MOLITO   =  4731;
4084  16   36:D     2
4085  16   36:D     2              MORLIS   =  4744;
4086  16   36:D     2              DALTO    =  3180;
4087  16   36:D     2              LAHALITO =  6156;
4088  16   36:D     2
4089  16   36:D     2              MAMORLIS =  7525;
4090  16   36:D     2              MAKANITO =  6612;
4091  16   36:D     2              MADALTO  =  4925;
4092  16   36:D     2
4093  16   36:D     2              LAKANITO =  6587;
4094  16   36:D     2              ZILWAN   =  4573;
4095  16   36:D     2              MASOPIC  =  3990;
4096  16   36:D     2              HAMAN    =  1562;
4097  16   36:D     2
4098  16   36:D     2              MALOR    =  3128;
```

```
4099  16  36:D   2              MAHAMAN  =  2597;
4100  16  36:D   2              TILTOWAI = 11157;
4101  16  36:D   2
4102  16  36:D   2
4103  16  36:D   2              KALKI    =  1449;
4104  16  36:D   2              DIOS     =  2301;
4105  16  36:D   2              BADIOS   =  3675;
4106  16  36:D   2              MILWA    =  2889;
4107  16  36:D   2              PORFIC   =  2287;
4108  16  36:D   2
4109  16  36:D   2              MATU     =  3139;
4110  16  36:D   2              CALFO    =     0;  (* 1717 *)
4111  16  36:D   2              MANIFO   =  2619;
4112  16  36:D   2              MONTINO  =  5970;
4113  16  36:D   2
4114  16  36:D   2              LOMILWA  =  5333;
4115  16  36:D   2              DIALKO   =  2718;
4116  16  36:D   2              LATUMAPI =  6491;
4117  16  36:D   2              BAMATU   =  5169;
4118  16  36:D   2
4119  16  36:D   2              DIAL     =   761;
4120  16  36:D   2              BADIAL   =  1253;
4121  16  36:D   2              LATUMOFI =  9463;
4122  16  36:D   2              MAPORFIC =  4322;
4123  16  36:D   2
4124  16  36:D   2              DIALMA   =  1614;
4125  16  36:D   2              BADIALMA =  2446;
4126  16  36:D   2              LITOKAN  =  4396;
4127  16  36:D   2              KANDI    =  1185; (* 1885 *)
4128  16  36:D   2              DI       =   180;
4129  16  36:D   2              BADI     =   382;
4130  16  36:D   2
4131  16  36:D   2              LORTO    =  4296;
4132  16  36:D   2              MADI     =   547;
4133  16  36:D   2              MABADI   =   759;
4134  16  36:D   2              LOKTOFEI =  8330;
4135  16  36:D   2
4136  16  36:D   2              MALIKTO  =  5514;
4137  16  36:D   2              KADORTO  =  6673;
4138  16  36:D   2 }
4139  16  36:D   2
4140  16  36:D   2         VAR
4141  16  36:D   2              USEITEM : BOOLEAN;
4142  16  36:D   3              MP03 : INTEGER;
4143  16  36:D   4              MP04 : INTEGER;
4144  16  36:D   5              MP05 : INTEGER;
4145  16  36:D   6              P24MP06 : INTEGER;
4146  16  36:D   7              MP07 : INTEGER;
4147  16  36:D   8              MP08 : INTEGER;
4148  16  36:D   9              MP09 : INTEGER;
4149  16  36:D  10              MP0A : INTEGER;
4150  16  36:D  11              SPELNAME : STRING[ 80];
4151  16  36:D  52
4152  16  36:D  52
4153  16  37:D   1         PROCEDURE EXITCAST( MP01 : INTEGER);  (* P010A25 *)
4154  16  37:D   2
4155  16  37:0   0           BEGIN (* EXITCAST *)
```

```
4156  16   37:1     0            GETMSGTX( BASE47, MP01);
4157  16   37:1     6            P010A1E( BASE47);
4158  16   37:1    10            DELWIN( BASE10, TRUE);
4159  16   37:1    16            DSPSPELS;
4160  16   37:1    18            EXIT( CASTWHAT)
4161  16   37:0    22          END;  (* EXITCAST *)
4162  16   37:0    34
4163  16   37:0    34
4164  16   38:D     1        PROCEDURE CANTCAST;  (* P010A26 *)
4165  16   38:D     1
4166  16   38:0     0          BEGIN (* CANTCAST *)
4167  16   38:1     0            EXITCAST( 619)     (* YOU CAN'T CAST THAT *)
4168  16   38:0     3          END;  (* CANTCAST *)
4169  16   38:0    18
4170  16   38:0    18
4171  16   39:D     1        PROCEDURE CHKFIZZ;  (* P010A27 *)
4172  16   39:D     1
4173  16   39:0     0          BEGIN (* CHKFIZZ *)
4174  16   39:1     0            IF FIZZLES > 0 THEN
4175  16   39:2     6              EXITCAST( 620)    (* FIZZLE! *)
4176  16   39:0     9          END;  (* CHKFIZZ *)
4177  16   39:0    24
4178  16   39:0    24
4179  16   40:D     1        PROCEDURE CLRWIN;  (* P010A28 *)
4180  16   40:D     1
4181  16   40:0     0          BEGIN (* CLRWIN *)
4182  16   40:1     0            CLEARWIN( BASE10, FALSE);
4183  16   40:1     5            P24MP06 := 6
4184  16   40:0     5          END;  (* CLRWIN *)
4185  16   40:0    22
4186  16   40:0    22
4187  16   41:D     1        PROCEDURE CHKSPCNT( MP02 : INTEGER;  (* P010A29 *)
4188  16   41:D     2                            MP01 : INTEGER);
4189  16   41:D     3
4190  16   41:0     0          BEGIN (* CHKSPCNT *)
4191  16   41:1     0            IF USEITEM THEN
4192  16   41:2     5              EXIT( CHKSPCNT);
4193  16   41:1     9            WITH CHARACTR[ CAMPCHAR] DO
4194  16   41:2    19              BEGIN
4195  16   41:3    19                IF (PRIESTSP[ MP02] <= 0) OR
4196  16   41:3    30                   (NOT SPELLSKN[ MP01])          THEN
4197  16   41:4    42                  CANTCAST
4198  16   41:2    42              END;
4199  16   41:2    44
4200  16   41:0    44          END;  (* CHKSPCNT *)
4201  16   41:0    56
4202  16   41:0    56
4203  16   42:D     1        PROCEDURE DECPRIEST( MP01 : INTEGER);  (* P010A2A *)
4204  16   42:D     2
4205  16   42:0     0          BEGIN (* DECPRIEST *)
4206  16   42:1     0            IF NOT USEITEM THEN
4207  16   42:2     6              BEGIN
4208  16   42:3     6                WITH CHARACTR[ CAMPCHAR] DO
4209  16   42:4    16                  BEGIN
4210  16   42:5    16                    PRIESTSP[ MP01] := PRIESTSP[ MP01] - 1
4211  16   42:4    33                  END;
4212  16   42:2    36              END;
```

```
4213  16  42:1   36            DSPSPELS;
4214  16  42:1   38            CHKFIZZ
4215  16  42:0   38          END;  (* DECPRIEST *)
4216  16  42:0   52
4217  16  42:0   52
4218  16  43:D    1        PROCEDURE DOHEAL( MP04 : INTEGER;  (* P010A2B *)
4219  16  43:D    2                          MP03 : INTEGER;
4220  16  43:D    3                          MP02 : INTEGER;
4221  16  43:D    4                          MP01 : INTEGER);
4222  16  43:D    5
4223  16  43:D    5          VAR
4224  16  43:D    5               MP05 : INTEGER;
4225  16  43:D    6
4226  16  43:0    0          BEGIN (* DOHEAL *)
4227  16  43:1    0            P24MP06 := 6;
4228  16  43:1    4            WITH CHARACTR[ P24MP06] DO
4229  16  43:2   14              BEGIN
4230  16  43:3   14                CHKSPCNT( MP02, MP01);
4231  16  43:3   18                CLRWIN;
4232  16  43:3   20                DECPRIEST( MP02);
4233  16  43:3   23                MP05 := 0;
4234  16  43:3   26                IF MP04 = -1 THEN
4235  16  43:4   32                  BEGIN
4236  16  43:5   32                    MP05 := HPMAX;
4237  16  43:5   37                    LOSTXYL.POISNAMT[ 1] := 0;
4238  16  43:5   47                    IF STATUS < DEAD THEN
4239  16  43:6   54                      STATUS := OK
4240  16  43:4   57                  END
4241  16  43:3   59                ELSE
4242  16  43:4   61                  WHILE MP04 > 0 DO
4243  16  43:5   66                    BEGIN
4244  16  43:6   66                      MP05 := MP05 + (RAND MOD MP03) + 1;
4245  16  43:6   79                      MP04 := MP04 - 1
4246  16  43:5   80                    END;
4247  16  43:3   86                HPLEFT := HPLEFT + MP05;
4248  16  43:3   95                IF HPLEFT > HPMAX THEN
4249  16  43:4  104                  HPLEFT := HPMAX;
4250  16  43:3  111                CLEARWIN( BASE10, TRUE);
4251  16  43:3  116                PRINTCR( BASE10);
4252  16  43:3  120                DISP1MSG( BASE10, 623);  (* NOW *)
4253  16  43:3  127                PRINTNUM( BASE10, HPLEFT, 5);
4254  16  43:3  135                PRPICCH( BASE10, '/');
4255  16  43:3  140                PRINTNUM( BASE10, HPMAX, 4);
4256  16  43:3  148                PAUSE2;
4257  16  43:3  151                DSPSPELS;
4258  16  43:3  153                IF CAMPCHAR = P24MP06 THEN
4259  16  43:4  162                  DSPSTATS;
4260  16  43:3  164                DELWIN( BASE10, TRUE);
4261  16  43:3  170                EXIT( CASTWHAT)
4262  16  43:2  174              END;
4263  16  43:0  174          END;  (* DOHEAL *)
4264  16  43:0  188
4265  16  43:0  188
4266  16  44:D    1        PROCEDURE DOKANDI;  (* P010A2C *)
4267  16  44:D    1
4268  16  44:0    0          BEGIN (* DOKANDI *)
4269  16  44:1    0            CHKSPCNT( 5, 42);
```

```
4270  16   44:1     4              DECPRIEST( 5);
4271  16   44:1     7              P010A1C( CAMPCHAR, 2, TRUE)
4272  16   44:0    12            END;  (* DOKANDI *)
4273  16   44:0    26
4274  16   44:0    26
4275  16   45:D     1          PROCEDURE DODIKADO( DIKADOXX : INTEGER);  (* P010A2D *)
4276  16   45:D     2
4277  16   45:D     2
4278  16   46:D     1            PROCEDURE DIKADORT;  (* P010A2E *)
4279  16   46:D     1
4280  16   46:0     0              BEGIN (* DIKADORT *)
4281  16   46:1     0                P24MP06 := 6;
4282  16   46:1     4                WITH CHARACTR[ P24MP06] DO
4283  16   46:2    14                  BEGIN
4284  16   46:2    14
4285  16   46:3    14                    IF (RAND MOD 25) <= ATTRIB[ VITALITY] THEN
4286  16   46:4    32                      BEGIN
4287  16   46:5    32                        STATUS := OK;
4288  16   46:5    37                        IF DIKADOXX = 5 THEN
4289  16   46:6    44                          HPLEFT := 1
4290  16   46:5    47                        ELSE
4291  16   46:6    51                          HPLEFT := HPMAX;
4292  16   46:6    58
4293  16   46:5    58                        IF ATTRIB[ VITALITY] = 3 THEN
4294  16   46:6    70                          STATUS := LOST
4295  16   46:5    73                        ELSE
4296  16   46:6    77                          ATTRIB[ VITALITY] := ATTRIB[ VITALITY] - 1
4297  16   46:4    92                      END;
4298  16   46:4    95
4299  16   46:3    95                    IF STATUS = OK THEN
4300  16   46:4   102                      EXITCAST( 624)               (* OK *)
4301  16   46:3   105                    ELSE
4302  16   46:4   109                      BEGIN
4303  16   46:5   109                        STATUS := SUCC( STATUS);
4304  16   46:5   118                        EXITCAST( 625)             (* OOPS! *)
4305  16   46:4   121                      END
4306  16   46:4   123
4307  16   46:2   123                  END  (* WITH *)
4308  16   46:2   123
4309  16   46:0   123              END;  (* DIKADORT *)
4310  16   46:0   136
4311  16   46:0   136
4312  16   45:0     0            BEGIN (* DODIKADO *)
4313  16   45:1     0              IF DIKADOXX = 5 THEN
4314  16   45:2     5                CHKSPCNT( DIKADOXX, 43)
4315  16   45:1     7              ELSE
4316  16   45:2    11                CHKSPCNT( DIKADOXX, 50);
4317  16   45:2    15
4318  16   45:1    15              CLRWIN;
4319  16   45:1    17
4320  16   45:1    17              WITH CHARACTR[ P24MP06] DO
4321  16   45:2    27                BEGIN
4322  16   45:2    27
4323  16   45:3    27                IF STATUS < DEAD THEN
4324  16   45:4    34                  EXITCAST( 626);   (* NOT DEAD *)
4325  16   45:4    39
4326  16   45:3    39                IF STATUS = LOST THEN
```

```
4327  16   45:4    46                EXITCAST( 627);   (* LOST *)
4328  16   45:4    51
4329  16   45:3    51              DECPRIEST( DIKADOXX);
4330  16   45:3    54              IF DIKADOXX = 5 THEN
4331  16   45:4    59                BEGIN
4332  16   45:5    59                  IF STATUS = DEAD THEN
4333  16   45:6    66                    DIKADORT
4334  16   45:5    66                  ELSE
4335  16   45:6    70                    IF STATUS = ASHES THEN
4336  16   45:7    77                      EXITCAST( 628)      (* TRY KADORTO *)
4337  16   45:4    80                END
4338  16   45:3    82              ELSE
4339  16   45:4    84                DIKADORT
4340  16   45:4    84
4341  16   45:2    84            END (* WITH *)
4342  16   45:2    86
4343  16   45:0    86          END;  (* DODIKADO *)
4344  16   45:0    98
4345  16   45:0    98
4346  16   47:D     1        PROCEDURE DODUMAPI;  (* P010A2F *)
4347  16   47:D     1
4348  16   47:0     0          BEGIN (* DODUMAPI *)
4349  16   47:1     0            WITH CHARACTR[ CAMPCHAR] DO
4350  16   47:2    10              BEGIN
4351  16   47:3    10                IF NOT USEITEM THEN
4352  16   47:4    16                  IF (MAGESP[ 1] = 0) OR (NOT SPELLSKN[ 4]) THEN
4353  16   47:5    39                    CANTCAST;
4354  16   47:3    41                CHKFIZZ;
4355  16   47:3    43                IF NOT USEITEM THEN
4356  16   47:4    49                  MAGESP[ 1] := MAGESP[ 1] - 1;
4357  16   47:3    69                DSPSPELS;
4358  16   47:3    71                P010A1C( CAMPCHAR, 3, TRUE)
4359  16   47:2    76              END; (* WITH *)
4360  16   47:0    78          END;  (* DODUMAPI *)
4361  16   47:0    90
4362  16   47:0    90
4363  16   48:D     1        PROCEDURE DOMALOR;  (* P010A30 *)
4364  16   48:D     1
4365  16   48:0     0          BEGIN (* DOMALOR *)
4366  16   48:1     0            WITH CHARACTR[ CAMPCHAR] DO
4367  16   48:2    10              BEGIN
4368  16   48:3    10                IF NOT USEITEM THEN
4369  16   48:4    16                  IF (MAGESP[ 7] = 0) OR (NOT SPELLSKN[ 19]) THEN
4370  16   48:5    39                    CANTCAST;
4371  16   48:3    41                CHKFIZZ;
4372  16   48:3    43                IF NOT USEITEM THEN
4373  16   48:4    49                  MAGESP[ 7] := MAGESP[ 7] - 1;
4374  16   48:3    69                P010A1C( CAMPCHAR, 4, TRUE)
4375  16   48:2    74              END  (* WITH *)
4376  16   48:2    76
4377  16   48:0    76          END;  (* DOMALOR *)
4378  16   48:0    88
4379  16   48:0    88
4380  16   49:D     3        FUNCTION CASTSPEL : BOOLEAN;  (* P010A31 *)
4381  16   49:D     3
4382  16   49:0     0          BEGIN (* CASTSPEL *)
4383  16   49:1     0            CASTSPEL := TRUE;
```

```
4384  16   49:1     3            IF SPELHASH = 6491 THEN  (* LATUMAPI *)
4385  16   49:2    12              BEGIN
4386  16   49:3    12                CHKSPCNT( 3, 33);
4387  16   49:3    16                DECPRIEST( 3);
4388  16   49:3    19                IDMONSTR := 10000
4389  16   49:2    19              END
4390  16   49:1    24            ELSE IF SPELHASH = 2301 THEN  (* DIOS *)
4391  16   49:3    35                DOHEAL( 1, 8, 1, 23)
4392  16   49:2    39            ELSE IF SPELHASH = 2889 THEN  (* MILWA *)
4393  16   49:4    52              BEGIN
4394  16   49:5    52                CHKSPCNT( 1, 25);
4395  16   49:5    56                DECPRIEST( 1);
4396  16   49:5    59                LIGHT := LIGHT + 30 + (RAND MOD 15)
4397  16   49:4    70              END
4398  16   49:3    73            ELSE IF SPELHASH = 3245 THEN  (* DUMAPIC *)
4399  16   49:5    84              DODUMAPI
4400  16   49:4    84            ELSE IF SPELHASH = 1185 THEN  (* KANDI 1185 1885 *)
4401  16   49:6    97              DOKANDI
4402  16   49:5    97            ELSE IF SPELHASH = 5333 THEN  (* LOMILWA *)
4403  16   49:7   110              BEGIN
4404  16   49:8   110                CHKSPCNT( 3, 31);
4405  16   49:8   114                DECPRIEST( 3);
4406  16   49:8   117                LIGHT := 10000
4407  16   49:7   117              END
4408  16   49:6   122            ELSE IF SPELHASH = 9463 THEN  (* LATUMOFI *)
4409  16   49:8   133              BEGIN
4410  16   49:9   133                CHKSPCNT( 4, 37);
4411  16   49:9   137                CLRWIN;
4412  16   49:9   139                DECPRIEST( 4);
4413  16   49:9   142                CHARACTR[ P24MP06].LOSTXYL.POISNAMT[ 1] := 0
4414  16   49:8   157              END
4415  16   49:7   159            ELSE IF SPELHASH = 2718 THEN  (* DIALKO *)
4416  16   49:9   170              BEGIN
4417  16   49:0   170                CHKSPCNT( 3, 32);
4418  16   49:0   174                CLRWIN;
4419  16   49:0   176                DECPRIEST( 3);
4420  16   49:0   179                WITH CHARACTR[ P24MP06] DO
4421  16   49:1   189                  BEGIN
4422  16   49:2   189                    IF (STATUS = PLYZE) OR (STATUS = ASLEEP) THEN
4423  16   49:3   202                      STATUS := OK
4424  16   49:1   205                  END (* WITH *)
4425  16   49:9   207              END
4426  16   49:8   207            ELSE IF SPELHASH = 761 THEN  (* DIAL *)
4427  16   49:0   218              DOHEAL( 2, 8, 4, 35)
4428  16   49:9   222            ELSE IF SPELHASH = 4322 THEN  (* MAPORFIC *)
4429  16   49:1   235              BEGIN
4430  16   49:2   235                CHKSPCNT( 4, 38);
4431  16   49:2   239                DECPRIEST( 4);
4432  16   49:2   242                ACMOD2 := 2
4433  16   49:1   242              END
4434  16   49:0   245            ELSE IF SPELHASH = 1614 THEN  (* DIALMA *)
4435  16   49:2   256              DOHEAL( 3,8, 5, 39)
4436  16   49:1   260            ELSE IF SPELHASH = 180 THEN   (* DI *)
4437  16   49:3   273              DODIKADO( 5)
4438  16   49:2   274            ELSE IF SPELHASH = 547 THEN  (* MADI *)
4439  16   49:4   287              DOHEAL( -1, -1, 6, 46)
4440  16   49:3   293            ELSE IF SPELHASH = 6673 THEN  (* KADORTO *)
```

```
4441  16   49:5   306              DODIKADO( 7)
4442  16   49:4   307            ELSE IF SPELHASH = 3128 THEN  (* MALOR *)
4443  16   49:6   320              DOMALOR
4444  16   49:5   320            ELSE
4445  16   49:6   324              CASTSPEL := FALSE
4446  16   49:6   324
4447  16   49:0   324          END;  (* CASTSPEL *)
4448  16   49:0   354
4449  16   49:0   354
4450  16   50:D     1        PROCEDURE P010A32( VAR P32MP01 : TLONGSTR);
4451  16   50:D     2
4452  16   50:D     2          VAR
4453  16   50:D     2            P32MP02 : INTEGER;
4454  16   50:D     3            P32MP03 : STRING[30];
4455  16   50:D    19
4456  16   50:D    19            MP13    : INTEGER;
4457  16   50:D    20            MP14    : INTEGER;
4458  16   50:D    21
4459  16   50:D    21            MP15    : INTEGER;
4460  16   50:D    22            MP16    : ARRAY[ 1..50] OF INTEGER;
4461  16   50:D    72
4462  16   50:D    72
4463  16   51:D     1          PROCEDURE CALCHASH( VAR SPELNAME : STRING);  (* P010A33 *)
4464  16   51:D     2
4465  16   51:D     2            VAR
4466  16   51:D     2              HASHCALC : INTEGER;
4467  16   51:D     3              SPELLI   : INTEGER;
4468  16   51:D     4
4469  16   51:0     0            BEGIN (* CALCHASH *)
4470  16   51:1     0              SPELHASH := LENGTH( SPELNAME);
4471  16   51:1     6              FOR SPELLI := 1 TO LENGTH( SPELNAME) DO
4472  16   51:2    19                BEGIN
4473  16   51:3    19                  IF SPELHASH < 20000 THEN
4474  16   51:4    28                    BEGIN
4475  16   51:5    28                      HASHCALC := ORD( SPELNAME[ SPELLI]) - 64;
4476  16   51:5    35                                                  (* 64 = ORD('A') - 1 *)
4477  16   51:5    35                      SPELHASH := SPELHASH + HASHCALC * HASHCALC * SPELLI
4478  16   51:4    41                    END
4479  16   51:2    47                END
4480  16   51:0    47            END;  (* CALCHASH *)
4481  16   51:0    68
4482  16   51:0    68
4483  16   52:D     1          PROCEDURE GTSPELNM( MP01 : INTEGER);  (* P010A34 *)
4484  16   52:D     2
4485  16   52:0     0            BEGIN (* GTSPELNM *)
4486  16   52:1     0              GETMSGTX( P32MP03, 5000 + MP01); (* HALITO..KADORTO *)
4487  16   52:1    11              IF P32MP03[ 1] = '*' THEN
4488  16   52:2    20                COPYSTR( P32MP03, P32MP03, 2, LENGTH( P32MP03) - 1)
4489  16   52:0    34            END;  (* GTSPELNM *)
4490  16   52:0    50
4491  16   52:0    50
4492  16   53:D     1          PROCEDURE P010A35( MP01 : INTEGER);
4493  16   53:D     2
4494  16   53:0     0            BEGIN (* P010A35 *)
4495  16   53:1     0              IF MP01 = 0 THEN
4496  16   53:2     5                P32MP03 := ''
4497  16   53:1     8              ELSE
```

```
4498  16  53:2   15                GTSPELNM( MP01);
4499  16  53:1   18              P32MP01 := P32MP03
4500  16  53:0   21            END;  (* P010A35 *)
4501  16  53:0   38
4502  16  53:0   38
4503  16  54:D    1          PROCEDURE P010A36;
4504  16  54:D    1
4505  16  54:0    0            BEGIN (* P010A36 *)
4506  16  54:1    0              CALCHASH( P32MP01);
4507  16  54:1    5              FOR P32MP02 := 1 TO 50 DO
4508  16  54:2   19                BEGIN
4509  16  54:3   19                  IF SPELHASH = SCENARIO.SPELLHSH[ P32MP02] THEN
4510  16  54:4   34                    BEGIN
4511  16  54:5   34                      GTSPELNM( P32MP02);
4512  16  54:5   39                      IF P32MP03 = P32MP01 THEN
4513  16  54:6   49                        EXIT( P010A32)
4514  16  54:5   53                      ELSE
4515  16  54:6   55                        EXIT( P010A36)
4516  16  54:4   59                    END;
4517  16  54:2   59                END;
4518  16  54:0   69            END;  (* P010A36 *)
4519  16  54:0   84
4520  16  54:0   84
4521  16  55:D    1          PROCEDURE P010A37;
4522  16  55:D    1
4523  16  55:D    1            VAR
4524  16  55:D    1                P37MP01 : TWINDOWP;
4525  16  55:D    2                MP02 : INTEGER;
4526  16  55:D    3                MP03 : INTEGER;
4527  16  55:D    4                MP04 : INTEGER;
4528  16  55:D    5
4529  16  55:D    5
4530  16  56:D    1            PROCEDURE P010A38( MP01 : INTEGER);
4531  16  56:D    2
4532  16  56:0    0              BEGIN (* P010A38 *)
4533  16  56:1    0                P010A35( MP01);
4534  16  56:1    3                DELWIN( P37MP01, TRUE);
4535  16  56:1   10                EXIT( P010A37)
4536  16  56:0   14              END;  (* P010A38 *)
4537  16  56:0   26
4538  16  56:0   26
4539  16  55:0    0            BEGIN (* P010A37 *)
4540  16  55:1    0              P37MP01 := GETWIN( 7, 9 - (MP15 + 1) DIV 2,
4541  16  55:1   10                                 26, 6 + MP15, 31, TRUE);
4542  16  55:1   25              DISP1LIN( P37MP01, 1080);   (* CAST WHICH SPELL ? *)
4543  16  55:1   32              PRINTCR( P37MP01);
4544  16  55:1   36              FOR MP04 := 1 TO MP15 DO
4545  16  55:2   49                BEGIN
4546  16  55:3   49                  MVCURSOR( P37MP01, 0, MP04 + 1);
4547  16  55:3   57                  PRPICCH( P37MP01, CHR( ORD( 'A') + MP04 - 1) );
4548  16  55:3   66                  PRPICMB( P37MP01, ') ');
4549  16  55:3   75                  GTSPELNM( MP16[ MP04]);
4550  16  55:3   86                  PRPICMB( P37MP01, P32MP03);
4551  16  55:3   93                  WHILE P37MP01^.HEAD.HCURSOR < 21 DO
4552  16  55:4  103                    PRPICCH( P37MP01, ' ');
4553  16  55:3  110                  PRPICMB( P37MP01, '(');
4554  16  55:3  115                  MP02 := SCENARIO.Z3[  MP16[ MP04]];
```

```
4555  16   55:3   133                    IF MP16[ MP04] < 22 THEN
4556  16   55:4   146                      MP03 := CHARACTR[ CAMPCHAR].MAGESP[ MP02]
4557  16   55:3   161                    ELSE
4558  16   55:4   166                      MP03 := CHARACTR[ CAMPCHAR].PRIESTSP[ MP02];
4559  16   55:3   184                    PRINTNUM( P37MP01, MP03, 1);
4560  16   55:3   190                    PRPICCH( P37MP01, ')')
4561  16   55:2   192                  END;
4562  16   55:2   202
4563  16   55:1   202                MVCURSOR( P37MP01, 0, MP15 + 3);
4564  16   55:1   212                DISP1LIN( P37MP01, 1081);        (* OR PRESS ~ TO NOT CAST *)
4565  16   55:1   219                SPELHASH := -1;
4566  16   55:1   224
4567  16   55:1   224                REPEAT
4568  16   55:2   224                  GETKEY;
4569  16   55:2   227                  IF INCHAR = CHR( CRETURN) THEN
4570  16   55:3   232                    P010A38( 0);
4571  16   55:2   235                  MP04 := ORD( INCHAR) - ORD( 'A') + 1;
4572  16   55:2   242                  IF (MP04 >= 1) AND (MP04 <= MP15) THEN
4573  16   55:3   253                    BEGIN
4574  16   55:4   253                      SPELHASH := SCENARIO.SPELLHSH[ MP16[ MP04]];
4575  16   55:4   271                      P010A38( MP16[ MP04])
4576  16   55:3   280                    END;
4577  16   55:3   282
4578  16   55:2   282                  PRINTBEL
4579  16   55:1   282                UNTIL FALSE
4580  16   55:1   285
4581  16   55:0   285              END;  (* P010A37 *)
4582  16   55:0   308
4583  16   55:0   308
4584  16   50:0     0            BEGIN (* P010A32 *)
4585  16   50:1     0              P010A36;
4586  16   50:1     2              MP15 := 0;
4587  16   50:1     5              FOR P32MP02 := 1 TO 50 DO
4588  16   50:2    17                BEGIN
4589  16   50:3    17                  GTSPELNM( P32MP02);
4590  16   50:3    20                  IF P01002F( P32MP01, P32MP03) = 1 THEN
4591  16   50:4    32                    IF CHARACTR[ CAMPCHAR].SPELLSKN[ P32MP02] THEN
4592  16   50:5    49                      BEGIN
4593  16   50:6    49                        MP15 := MP15 + 1;
4594  16   50:6    55                        MP16[ MP15] := P32MP02
4595  16   50:5    63                      END;
4596  16   50:2    65                END;
4597  16   50:1    72              SPELHASH := -1;
4598  16   50:1    77              P32MP01 := '';
4599  16   50:1    83              IF MP15 = 1 THEN
4600  16   50:2    89                BEGIN
4601  16   50:3    89                  SPELHASH := SCENARIO.SPELLHSH[ MP16[ 1]];
4602  16   50:3   106                  P010A35( MP16[ 1])
4603  16   50:2   114                END
4604  16   50:1   116              ELSE
4605  16   50:2   118                IF MP15 > 1 THEN
4606  16   50:3   124                  P010A37
4607  16   50:3   124
4608  16   50:0   124            END;  (* P010A32 *)
4609  16   50:0   140
4610  16   50:0   140
4611  16   36:0     0          BEGIN (* CASTWHAT *)
```

```
4612  16   36:1     0          SPELNAME := '';
4613  16   36:1     7          IF BLINES24 THEN
4614  16   36:2    12            BASE10 := GETWIN( 13, 7, 26, 5, 15, TRUE)
4615  16   36:1    18          ELSE
4616  16   36:2    27            BASE10 := GETWIN( 13, 8, 26, 5, 15, TRUE);
4617  16   36:1    40          USEITEM := SPELHASH > 0;
4618  16   36:1    45          IF SPELHASH = -1 THEN
4619  16   36:2    51            BEGIN
4620  16   36:3    51              DISP1LIN( BASE10, 629);  (* CAST WHAT SPELL ? *)
4621  16   36:3    58              P010A1D( BASE10);
4622  16   36:3    61              DISP1MSG( BASE10, 630);  (*    > *)
4623  16   36:3    68              MP08 := BASE10^.HEAD.HCURSOR;
4624  16   36:3    76              MP07 := BASE10^.HEAD.VCURSOR;
4625  16   36:3    84              SOLICIT( BASE10, SPELNAME, 15);
4626  16   36:3    91              IF SPELNAME = '' THEN
4627  16   36:4   100                EXITCAST( 9999);
4628  16   36:3   105              P010A32( SPELNAME);
4629  16   36:3   109              MVCURSOR( BASE10, MP08, MP07);
4630  16   36:3   115              PRPICMB( BASE10, SPELNAME);
4631  16   36:3   121              MP08 := 2;
4632  16   36:3   124              UNITWRITE( BASE12, MP08, 19, MP08); (* LOOP FOR KEYPRESS *)
4633  16   36:3   134              IF SPELHASH = -1 THEN
4634  16   36:4   140                EXITCAST( 9999);
4635  16   36:2   145            END;
4636  16   36:2   145
4637  16   36:1   145          IF CASTSPEL THEN
4638  16   36:2   151            EXITCAST( 632);  (* DONE!  *)
4639  16   36:1   156          EXITCAST( 631)     (* WHAT ? *)
4640  16   36:1   159
4641  16   36:0   159        END;  (* CASTWHAT *)
4642  16   36:0   174
4643  16   36:0   174 (*$I WIZ4C:CAMP2 *)
4643  16   36:0   174 (*$I WIZ4C:CAMP3 *)
4644  16   36:0   174
4645  16   36:0   174 (* CAMP3 *)
4646  16   36:0   174
4647  16   57:D     1       PROCEDURE USEITEM;  (* P010A39 *)
4648  16   57:D     1
4649  16   57:D     1         VAR
4650  16   57:D     1             THEITEM : TOBJREC;
4651  16   57:D    24             MP18 : INTEGER;
4652  16   57:D    25             MP19 : INTEGER;
4653  16   57:D    26             MP1A : INTEGER;
4654  16   57:D    27             MP1B : INTEGER;
4655  16   57:D    28
4656  16   57:D    28
4657  16   58:D     1         PROCEDURE P010A3A( MP01 : INTEGER);
4658  16   58:D     2
4659  16   58:0     0          BEGIN (* P010A3A *)
4660  16   58:1     0            GETMSGTX( BASE47, MP01);
4661  16   58:1     6            P010A1E( BASE47)
4662  16   58:0     8          END;  (* P010A3A *)
4663  16   58:0    22
4664  16   58:0    22
4665  16   59:D     1         PROCEDURE EXITUSE( MP01 : INTEGER);  (* P010A3B *)
4666  16   59:D     2
4667  16   59:0     0          BEGIN (* EXITUSE *)
```

```
4668  16   59:1     0              P010A3A( MP01);
4669  16   59:1     3              DELWIN( BASE10, TRUE);
4670  16   59:1     9              EXIT( USEITEM)
4671  16   59:0    13            END;  (* EXITUSE *)
4672  16   59:0    26
4673  16   59:0    26
4674  16   60:D     1          PROCEDURE P010A3C( MP02 : INTEGER;
4675  16   60:D     2                             MP01 : INTEGER);
4676  16   60:D     3
4677  16   60:D     3            VAR
4678  16   60:D     3                 MP03 : TWINDOWP;
4679  16   60:D     4
4680  16   60:0     0            BEGIN (* P010A3C *)
4681  16   60:1     0              IF CHARACTR[ CAMPCHAR].POSS.POSSESS[ MP1B].EQINDEX - 0 <>
4682  16   60:1    20                 MP02 THEN
4683  16   60:2    24                EXIT( P010A3C);
4684  16   60:1    28              MP03 := GETWIN( 0, 3, 40, 20, 31, TRUE);
4685  16   60:1    41              GETMSGTX( BASE47, MP01);
4686  16   60:1    47              WHILE BASE47 <> '**ERR**' DO
4687  16   60:2    63                BEGIN
4688  16   60:3    63                  CENTSTR( MP03, BASE47);
4689  16   60:3    69                  MP01 := MP01 + 1;
4690  16   60:3    74                  GETMSGTX( BASE47, MP01)
4691  16   60:2    77                END;
4692  16   60:1    82              GETCR;
4693  16   60:1    85              DELWIN( MP03, TRUE);
4694  16   60:1    91              DELWIN( BASE10, TRUE);
4695  16   60:1    97              EXIT( USEITEM)
4696  16   60:1   101
4697  16   60:0   101            END;  (* P010A3C *)
4698  16   60:0   116
4699  16   60:0   116
4700  16   61:D     1          PROCEDURE P010A3D;
4701  16   61:D     1
4702  16   61:D     1            VAR
4703  16   61:D     1                 MP01 : INTEGER;
4704  16   61:D     2                 MP02 : INTEGER;
4705  16   61:D     3                 MP03 : INTEGER;
4706  16   61:D     4                 MP04 : TWINDOWP;
4707  16   61:D     5
4708  16   61:D     5
4709  16   62:D     1            PROCEDURE P010A3E;
4710  16   62:D     1
4711  16   62:0     0              BEGIN (* P010A3E *)
4712  16   62:1     0                CHARACTR[ 6].POSS.POSSCNT := 8;
4713  16   62:1    10                DELWIN( MP04, TRUE);
4714  16   62:1    17                DSPITEMS;
4715  16   62:1    19                EXIT( USEITEM)
4716  16   62:0    23              END;  (* P010A3E *)
4717  16   62:0    36
4718  16   62:0    36
4719  16   63:D     1            PROCEDURE P010A3F;
4720  16   63:D     1
4721  16   63:0     0              BEGIN (* P010A3F *)
4722  16   63:1     0                PRPICMB( MP04, '                ')
4723  16   63:0    22              END;  (* P010A3F *)
4724  16   63:0    38
```

```
4725  16   63:0    38
4726  16   64:D     1          PROCEDURE P010A40( MP01 : INTEGER);
4727  16   64:D     2
4728  16   64:0     0            BEGIN (* P010A40 *)
4729  16   64:1     0              MVCURSOR( MP04, 19, MP01 - 1);
4730  16   64:1    10              PRPICCH( MP04, CHR( ORD( 'A') + MP01 - 1));
4731  16   64:1    21              PRPICMB( MP04, ') ');
4732  16   64:1    32              IF BLACKBOX[ MP01 - 1] = 0 THEN
4733  16   64:2    45                P010A3F
4734  16   64:1    45              ELSE
4735  16   64:2    49                DISP1MSG( MP04, 14001 + (2 * BLACKBOX[ MP01 - 1]))
4736  16   64:2    67
4737  16   64:2    67              (* BLOODSTONE, LANDER'S TURQ, AMBER DRAGON, ... *)
4738  16   64:2    67
4739  16   64:0    67            END;  (* P010A40 *)
4740  16   64:0    82
4741  16   64:0    82
4742  16   65:D     1          PROCEDURE P010A41( MP01 : INTEGER);
4743  16   65:D     2
4744  16   65:0     0            BEGIN (* P010A41 *)
4745  16   65:1     0              WITH CHARACTR[ 6].POSS.POSSESS[ MP01] DO
4746  16   65:2    15                BEGIN
4747  16   65:3    15                  MVCURSOR( MP04, 0, MP01 - 1);
4748  16   65:3    25                  PRINTNUM( MP04, MP01, 1);
4749  16   65:3    33                  PRPICCH(  MP04, ')');
4750  16   65:3    40                  IF EQINDEX = 0 THEN
4751  16   65:4    46                    P010A3F
4752  16   65:3    46                  ELSE
4753  16   65:4    50                    BEGIN
4754  16   65:5    50                      IF CURSED THEN
4755  16   65:6    54                        PRPICCH( MP04, '-')
4756  16   65:5    58                      ELSE
4757  16   65:6    63                        IF EQUIPED THEN
4758  16   65:7    67                          PRPICCH( MP04, '*')
4759  16   65:6    71                        ELSE
4760  16   65:7    76                          PRPICCH( MP04, ' ');
4761  16   65:5    83                      DISP1MSG( MP04, 14001 + (2 * EQINDEX))
4762  16   65:5    94
4763  16   65:4    94                    END;
4764  16   65:4    97
4765  16   65:2    97                END; (* WITH *)
4766  16   65:2    97
4767  16   65:0    97            END;  (* P010A41 *)
4768  16   65:0   110
4769  16   65:0   110
4770  16   66:D     1          PROCEDURE P010A42( MP02 : INTEGER;
4771  16   66:D     2                             MP01 : INTEGER);
4772  16   66:D     3
4773  16   66:D     3            VAR
4774  16   66:D     3                 MP03 : INTEGER;
4775  16   66:D     4
4776  16   66:0     0            BEGIN (* P010A42 *)
4777  16   66:1     0              FOR MP03 := MP02 TO MP01 DO
4778  16   66:2    11                BEGIN
4779  16   66:3    11                  DISP1MSG( MP04, MP03);
4780  16   66:3    18                  PRINTCR(  MP04)
4781  16   66:2    21                END;
```

```
4782  16   66:1    31              PRINTCR( MP04)
4783  16   66:0    34            END;  (* P010A42 *)
4784  16   66:0    52
4785  16   66:0    52
4786  16   67:D     1          PROCEDURE P010A43( MP01 : INTEGER);
4787  16   67:D     2
4788  16   67:D     2            VAR
4789  16   67:D     2                 MP02 : INTEGER;
4790  16   67:D     3                 MP03 : INTEGER;
4791  16   67:D     4                 MP04 : INTEGER;
4792  16   67:D     5                 MP05 : INTEGER;
4793  16   67:D     6
4794  16   67:0     0            BEGIN (* P010A43 *)
4795  16   67:1     0              WITH CHARACTR[ 6].POSS.POSSESS[ MP01] DO
4796  16   67:2    15                BEGIN
4797  16   67:3    15                  IF (EQINDEX = 0) OR (CURSED) OR (EQUIPED) THEN
4798  16   67:4    27                    EXIT( P010A43);
4799  16   67:3    31                  MP02 := 0;
4800  16   67:3    34                  FOR MP05 := 1 TO 19 DO
4801  16   67:4    45                    IF MP02 = 0 THEN
4802  16   67:5    50                      IF BLACKBOX[ MP05 - 1] = 0 THEN
4803  16   67:6    63                        MP02 := MP05;
4804  16   67:3    73                  IF MP02 = 0 THEN
4805  16   67:4    78                    EXIT( P010A43);
4806  16   67:3    82                  BLACKBOX[ MP02 - 1] := EQINDEX;
4807  16   67:3    93                  EQINDEX := 0;
4808  16   67:3    98                  P010A41( MP01);
4809  16   67:3   101                  P010A40( MP02)
4810  16   67:3   102
4811  16   67:2   102                END;  (* WITH *)
4812  16   67:2   104
4813  16   67:0   104            END;  (* P010A43 *)
4814  16   67:0   118
4815  16   67:0   118
4816  16   68:D     1          PROCEDURE P010A44( MP01 : INTEGER);
4817  16   68:D     2
4818  16   68:D     2            VAR
4819  16   68:D     2                 MP02 : INTEGER;
4820  16   68:D     3                 MP03 : INTEGER;
4821  16   68:D     4                 MP04 : INTEGER;
4822  16   68:D     5                 MP05 : INTEGER;
4823  16   68:D     6
4824  16   68:0     0            BEGIN (* P010A44 *)
4825  16   68:1     0              IF BLACKBOX[ MP01 - 1] = 0 THEN
4826  16   68:2    13                EXIT( P010A44);
4827  16   68:1    17              MP02 := 0;
4828  16   68:1    20              FOR MP05 := 1 TO 8 DO
4829  16   68:2    31                IF MP02 = 0 THEN
4830  16   68:3    36                  IF CHARACTR[ 6].POSS.POSSESS[ MP05].EQINDEX = 0 THEN
4831  16   68:4    54                    MP02 := MP05;
4832  16   68:1    64              IF MP02 = 0 THEN
4833  16   68:2    69                EXIT( P010A44);
4834  16   68:1    73              CHARACTR[ 6].POSS.POSSESS[ MP02].EQINDEX :=
4835  16   68:1    88                BLACKBOX[ MP01 - 1];
4836  16   68:1    98              BLACKBOX[ MP01 - 1] := 0;
4837  16   68:1   108              P010A41( MP02);
4838  16   68:1   111              P010A40( MP01)
```

```
4839  16   68:1   112
4840  16   68:1   112
4841  16   68:0   112            END;  (* P010A44 *)
4842  16   68:0   128
4843  16   68:0   128
4844  16   61:0     0          BEGIN (* P010A3D *)
4845  16   61:1     0            DELWIN( BASE10, FALSE);
4846  16   61:1     6            IF BLINES24 THEN
4847  16   61:2    11              MP04 := GETWIN( 0, 3, 40, 21, 16, TRUE)
4848  16   61:1    17            ELSE
4849  16   61:2    26              MP04 := GETWIN( 0, 4, 40, 21, 16, TRUE);
4850  16   61:2    39
4851  16   61:1    39            WITH CHARACTR[ 6].POSS DO
4852  16   61:2    49              BEGIN
4853  16   61:3    49                FOR MP03 := POSSCNT + 1 TO 8 DO
4854  16   61:4    63                  BEGIN
4855  16   61:5    63                    WITH POSSESS[ MP03] DO
4856  16   61:6    73                      BEGIN
4857  16   61:7    73                        EQINDEX := 0;
4858  16   61:7    78                        CURSED  := FALSE;
4859  16   61:7    83                        EQUIPED := FALSE;
4860  16   61:7    86                        IDENTIF := FALSE
4861  16   61:6    89                      END;
4862  16   61:4    91                  END;
4863  16   61:4    98
4864  16   61:2    98              END; (* WITH POSS *)
4865  16   61:2    98
4866  16   61:1    98            FOR MP03 := 1 TO 8 DO
4867  16   61:2   109              P010A41( MP03);
4868  16   61:1   119            MVCURSOR( MP04, 0, 9);
4869  16   61:1   125            P010A42( 280, 282);
4870  16   61:1   133            P010A42( 283, 285);
4871  16   61:1   141            P010A42( 286, 287);
4872  16   61:1   149            FOR MP03 := 1 TO 19 DO
4873  16   61:2   160              P010A40( MP03);
4874  16   61:2   170
4875  16   61:1   170            REPEAT
4876  16   61:2   170              GETKEY;
4877  16   61:2   173              IF INCHAR = CHR( CRETURN) THEN
4878  16   61:3   178                P010A3E
4879  16   61:2   178              ELSE IF (INCHAR >= '1') AND (INCHAR <= '8') THEN
4880  16   61:4   191                P010A43( ORD( INCHAR) - ORD( '0'))
4881  16   61:3   194              ELSE IF (INCHAR >= 'A') AND (INCHAR <= 'S') THEN
4882  16   61:5   207                P010A44( ORD( INCHAR) - ORD( 'A') + 1)
4883  16   61:1   212            UNTIL FALSE
4884  16   61:1   214
4885  16   61:0   214          END;  (* P010A3D *)
4886  16   61:0   238
4887  16   61:0   238
4888  16   57:0     0        BEGIN (* USEITEM *)
4889  16   57:1     0          IF BLINES24 THEN
4890  16   57:2     5            BASE10 := GETWIN( 13, 7, 26, 5, 15, TRUE)
4891  16   57:1    11          ELSE
4892  16   57:2    20            BASE10 := GETWIN( 13, 8, 26, 5, 15, TRUE);
4893  16   57:1    33          DISP1LIN( BASE10, 633);                    (* USE ITEM (#) ? *)
4894  16   57:1    40          P010A1D( BASE10);
4895  16   57:1    43
```

```
4896  16   57:1    43           WITH CHARACTR[ CAMPCHAR].POSS DO
4897  16   57:2    55             BEGIN
4898  16   57:3    55               MP1B := P01001E( POSSCNT);
4899  16   57:3    65               IF MP1B = 0 THEN
4900  16   57:4    71                 EXITUSE( 9999);
4901  16   57:4    76
4902  16   57:3    76               WITH POSSESS[ MP1B] DO
4903  16   57:4    88                 BEGIN
4904  16   57:5    88                   MOVELEFT( BASE55B[
4905  16   57:5    91                                 GETREC( 4, EQINDEX - 0, SIZEOF( TOBJREC))],
4906  16   57:5   103                             THEITEM,
4907  16   57:5   106                             SIZEOF( TOBJREC));
4908  16   57:5   109                   IF BASCIIHU THEN
4909  16   57:6   114                     BEGIN
4910  16   57:7   114                       UNITWRITE( BASE12, THEITEM, 33, GETADDR( BASE89B));
4911  16   57:7   131                       MOVELEFT( BASE89B[ 1], THEITEM, SIZEOF( TOBJREC))
4912  16   57:6   141                     END;
4913  16   57:5   141                   IF THEITEM.SPELLPWR = 50 THEN
4914  16   57:6   146                     P010A3D;
4915  16   57:5   148                   IF THEITEM.SPELLPWR = 28 THEN
4916  16   57:6   153                     EXITUSE( 290 + RAND MOD 3);  (* HEE! HEE! HEE! *)
4917  16   57:6   166                                                  (* HAA! HAA! HAA! *)
4918  16   57:6   166                                                  (* GOTCHA, FOOL!! *)
4919  16   57:5   166                   IF (EQINDEX - 0) = 20 THEN
4920  16   57:6   175                     EXITUSE( 293);               (* CLAP! CLAP!    *)
4921  16   57:5   180                   P010A3C(  90, 410);
4922  16   57:5   186                   P010A3C( 108, 440);
4923  16   57:5   192                   P010A3C( 109, 420);
4924  16   57:5   198                   IF (EQINDEX - 0) = 87 THEN
4925  16   57:6   207                     BEGIN
4926  16   57:7   207                       BASE7F4[ 8 - 1] := 1;
4927  16   57:7   217                       EXITUSE( 294)     (* G'WAN! GEDDOUDAHERE! *)
4928  16   57:6   220                     END;
4929  16   57:5   222                   IF THEITEM.SPELLPWR = 0 THEN
4930  16   57:6   227                     EXITUSE( 634);      (* POWERLESS *)
4931  16   57:5   232                   IF THEITEM.OBJTYPE <> SPECIAL THEN
4932  16   57:6   237                     IF NOT EQUIPED THEN
4933  16   57:7   243                       EXITUSE( 635);    (* NOT EQUIPPED *)
4934  16   57:5   248                   IF (RAND MOD 4000) < (40 * THEITEM.CHGCHANC) THEN
4935  16   57:6   263                     EQINDEX := THEITEM.CHANGETO + 0;
4936  16   57:5   271                   DELWIN( BASE10, FALSE);
4937  16   57:5   277                   CASTWHAT( SCENARIO.SPELLHSH[ THEITEM.SPELLPWR]);
4938  16   57:5   286                   DSPITEMS
4939  16   57:4   286                 END; (* WITH POSSESS *)
4940  16   57:4   288
4941  16   57:2   288             END;  (* WITH *)
4942  16   57:2   288
4943  16   57:0   288         END;  (* USEITEM *)
4944  16   57:0   300
4945  16   57:0   300
4946  16   69:D     1       PROCEDURE P010A45;  (* DROP ITEM *)
4947  16   69:D     1
4948  16   69:D     1         VAR
4949  16   69:D     1             MP01 : INTEGER;
4950  16   69:D     2             MP02 : INTEGER;
4951  16   69:D     3             MP03 : INTEGER;
4952  16   69:D     4             MP04 : INTEGER;
```

```
4953  16   69:D     5
4954  16   69:D     5
4955  16   70:D     1         PROCEDURE P010A46( MP01 : INTEGER);
4956  16   70:D     2
4957  16   70:0     0           BEGIN (* P010A46 *)
4958  16   70:1     0             GETMSGTX( BASE47, MP01);
4959  16   70:1     6             P010A1E( BASE47);
4960  16   70:1    10             DELWIN( BASE10, TRUE);
4961  16   70:1    16             EXIT( P010A45)
4962  16   70:0    20           END;  (* P010A46 *)
4963  16   70:0    32
4964  16   70:0    32
4965  16   69:0     0         BEGIN (* P010A45 *)
4966  16   69:1     0           IF BLINES24 THEN
4967  16   69:2     5             BASE10 := GETWIN( 13, 7, 26, 5, 15, TRUE)
4968  16   69:1    11           ELSE
4969  16   69:2    20             BASE10 := GETWIN( 13, 8, 26, 5, 15, TRUE);
4970  16   69:1    33           DISP1LIN( BASE10, 636);  (* DROP ITEM (#) ? *)
4971  16   69:1    40           P010A1D( BASE10);
4972  16   69:1    43           WITH CHARACTR[ CAMPCHAR].POSS DO
4973  16   69:2    55             BEGIN
4974  16   69:3    55               MP04 := P01001E( POSSCNT);
4975  16   69:3    64               IF MP04 = 0 THEN
4976  16   69:4    69                 P010A46( 9999);
4977  16   69:3    74               WITH POSSESS[ MP04] DO
4978  16   69:4    84                 BEGIN
4979  16   69:5    84                   IF CURSED THEN
4980  16   69:6    88                     P010A46( 637);  (* CURSED *)
4981  16   69:5    93                   IF EQUIPED THEN
4982  16   69:6    97                     P010A46( 638);  (* EQUIPPED *)
4983  16   69:5   102                   IF EQINDEX = 4 THEN
4984  16   69:6   108                     IF BASE7F4[ 6 - 1] = - 1 THEN
4985  16   69:7   122                       BEGIN
4986  16   69:8   122                         BASE7F4[  6 - 1] := 5;
4987  16   69:8   132                         BASE7F4[ 10 - 1] := MAZEX;
4988  16   69:8   142                         BASE7F4[ 11 - 1] := MAZEY;
4989  16   69:8   152                         BASE7F4[ 12 - 1] := MAZELEV
4990  16   69:7   160                       END;
4991  16   69:7   162
4992  16   69:4   162                 END; (* WITH POSSESS *)
4993  16   69:4   162
4994  16   69:3   162               FOR MP03 := MP04 + 1 TO POSSCNT DO
4995  16   69:4   176                 POSSESS[ MP03 - 1] := POSSESS[ MP03];
4996  16   69:3   203               POSSCNT := POSSCNT - 1;
4997  16   69:3   209               DSPITEMS;
4998  16   69:3   211               P010A46( 639)   (* DROPPED *)
4999  16   69:3   214
5000  16   69:2   214             END; (* WITH *)
5001  16   69:2   216
5002  16   69:0   216         END;  (* P010A45 *)
5003  16   69:0   230
5004  16   69:0   230
5005  16   71:D     1       PROCEDURE CAMPDO;  (* P010A47 *)
5006  16   71:D     1
5007  16   71:D     1         VAR
5008  16   71:D     1             MP01 : INTEGER;
5009  16   71:D     2
```

```
5010  16   71:D     2
5011  16   71:0     0         BEGIN (* CAMPDO *)
5012  16   71:0     0
5013  16   71:1     0           MP01 := 2;
5014  16   71:1     3           IF BLINES24 THEN
5015  16   71:2     8             BASE10 := GETWIN( 0, 20, 40, 4, 8, TRUE)
5016  16   71:1    14           ELSE
5017  16   71:2    23             BASE10 := GETWIN( 0, 21, 40, 4, 8, TRUE);
5018  16   71:1    36           MP03 := MENUINDX( BASE10, 649);
5019  16   71:1    48                                   (* R)EAD/E)QUIP/D)ROP/S)PELL/U)SE/L)EAVE *)
5020  16   71:1    48           IF MP03 = 5 THEN
5021  16   71:2    55             MP03 := -1;
5022  16   71:1    60           DELWIN( BASE10, FALSE);
5023  16   71:1    66
5024  16   71:1    66           CASE MP03 OF
5025  16   71:1    71             -1:  EXIT( CAMPDO);
5026  16   71:1    77              1:  BEGIN
5027  16   71:3    77                    BASE13 := 12;
5028  16   71:3    80                    BASE09 := CAMPCHAR;
5029  16   71:3    85                    EXIT( P010A1B)
5030  16   71:2    89                  END;
5031  16   71:1    91              0:  P010A1C( CAMPCHAR, 0, FALSE);
5032  16   71:1   100              2:  P010A45;
5033  16   71:1   104              3:  CASTWHAT( -1);
5034  16   71:1   110              4:  USEITEM;
5035  16   71:1   114           END;
5036  16   71:1   134
5037  16   71:0   134         END;  (* CAMPDO *)
5038  16   71:0   146
5039  16   71:0   146
5040  16   72:D     1       PROCEDURE CHEVRONS;  (* P010A48 *)
5041  16   72:D     1
5042  16   72:D     1         VAR
5043  16   72:D     1             AWARDX : INTEGER;
5044  16   72:D     2             BAWARDS : PACKED ARRAY[ 0..15] OF BOOLEAN;
5045  16   72:D     3
5046  16   72:D     3
5047  16   72:0     0         BEGIN (* CHEVRONS *)
5048  16   72:1     0           MOVELEFT( CHARACTR[ CAMPCHAR].LOSTXYL.AWARDS[ 4], BAWARDS, 2);
5049  16   72:1    22           BASE47 := '>!$#&*<?BCPCODG@';
5050  16   72:1    45           FOR AWARDX := 0 TO 15 DO
5051  16   72:2    56             IF BAWARDS[ AWARDX] THEN
5052  16   72:3    65               PRPICCH( CAMPWIN, BASE47[ AWARDX + 1]);
5053  16   72:3    82
5054  16   72:0    82         END;  (* CHEVRONS *)
5055  16   72:0    96
5056  16   72:0    96
5057  16   31:0     0       BEGIN (* INSPECT *)
5058  16   31:0     0
5059  16   31:1     0         CAMPCHAR := BASE09;   (* CHARACTR INDEX 1..6 (?) *)
5060  16   31:1     3         IF CAMPWIN = NIL THEN
5061  16   31:2     8           BEGIN
5062  16   31:3     8             WITH CHARACTR[ CAMPCHAR] DO
5063  16   31:4    16               BEGIN
5064  16   31:5    16                 IF BLINES24 THEN
5065  16   31:6    21                   CAMPWIN := GETWIN( 0, 3, 40, 17, 7, TRUE)
5066  16   31:5    27                 ELSE
```

```
5067  16   31:6    36                  CAMPWIN := GETWIN( 0, 4, 40, 17, 7, TRUE);
5068  16   31:5    49                PRPICMB(  CAMPWIN, NAME);
5069  16   31:5    54                PRPICCH(  CAMPWIN, ' ');
5070  16   31:5    59                PRPICCH(  CAMPWIN, 'L');
5071  16   31:5    64                PRINTNUM( CAMPWIN, CHARLEV, 3);
5072  16   31:5    72                PRPICCH(  CAMPWIN, ' ');
5073  16   31:5    77                GETMSGTX( BASE47, 10009 + ORD( ALIGN)); (* GOOD,...,EVIL *)
5074  16   31:5    89                COPYSTR(  BASE47, BASE47, 1, 1);
5075  16   31:5    98                PRPICMB(  CAMPWIN, BASE47);
5076  16   31:5   104                PRPICCH(  CAMPWIN, '-');
5077  16   31:5   109                GETMSGTX( BASE47, 10030 + ORD( CLASS)); (* FIGHTER,...,NINJA *)
5078  16   31:5   121                COPYSTR(  BASE47, BASE47, 1, 3);
5079  16   31:5   130                PRPICMB(  CAMPWIN, BASE47);
5080  16   31:5   136                PRPICCH(  CAMPWIN, ' ');
5081  16   31:5   141                GETMSGTX( BASE47, 9999 + ORD( RACE));   (* HUMAN,...,HOBBIT *)
5082  16   31:5   153                PRPICMB(  CAMPWIN, BASE47);
5083  16   31:5   159                PRPICCH(  CAMPWIN, ' ');
5084  16   31:5   164
5085  16   31:5   164                  CHEVRONS;
5086  16   31:5   166
5087  16   31:4   166                END;
5088  16   31:2   166              END;
5089  16   31:2   166
5090  16   31:1   166              DSPSTATS;
5091  16   31:1   168              DSPSPELS;
5092  16   31:1   170              DSPITEMS;
5093  16   31:1   172
5094  16   31:1   172              BASE2B := BASE13;
5095  16   31:1   176
5096  16   31:1   176              REPEAT
5097  16   31:2   176                CAMPDO
5098  16   31:1   176              UNTIL MP03 = -1;
5099  16   31:1   184              DELWIN( CAMPWIN, FALSE)
5100  16   31:1   187
5101  16   31:0   187          END;  (* INSPECT *)
5102  16   31:0   206
5103  16   31:0   206
5104  16   73:D     1        PROCEDURE P010A49;
5105  16   73:D     1
5106  16   73:0     0          BEGIN (* P010A49 *)
5107  16   73:1     0            DRAWSCR2( WINCHAIN);
5108  16   73:1     4            DELWIN( CAMPTIT, FALSE);
5109  16   73:1    10            BASE13 := 18;
5110  16   73:1    13            EXIT( P010A1B)
5111  16   73:0    17          END;  (* P010A49 *)
5112  16   73:0    30
5113  16   73:0    30
5114  16   27:0     0        BEGIN (* P010A1B *)
5115  16   27:1     0          FOR P1BMP01 := 1 TO 8 DO
5116  16   27:2    14            OBJIDS[ P1BMP01] := -1;
5117  16   27:1    29          IF (BASE13 = 16) OR (BASE13 = 25) THEN
5118  16   27:2    40            BEGIN
5119  16   27:3    40              BASE13 := BASE2B;
5120  16   27:3    44              IF BASE13 = 10 THEN
5121  16   27:4    50                BEGIN
5122  16   27:5    50                  INSPECT;
5123  16   27:5    52                  P010A49
```

```
5124  16   27:4    52            END;
5125  16   27:2    54          END;
5126  16   27:1    54        IF CAMPTIT = NIL THEN
5127  16   27:2    59          CAMPTIT := GTTITWIN( 652, 0, 1, 1);  (* CAMP *)
5128  16   27:1    72        FOR P1BMP01 := 1 TO 8 DO
5129  16   27:2    86          OBJIDS[ P1BMP01] := -1;
5130  16   27:1   101        BASE09 := 6;  (* CHARACTR INDEX *)
5131  16   27:1   104        INSPECT;
5132  16   27:1   106        P010A49;
5133  16   27:1   108
5134  16   27:0   108      END;  (* P010A1B *)
5135  16   27:0   124
5136  16   27:0   124
5137  16    1:0     0    BEGIN (* CAMP MAIN LINE P010A01 *)
5138  16    1:1     0      IF (BASE13 = 22) OR (BASE13 = 9) OR (BASE13 = 20) THEN
5139  16    1:2    16        P010A02
5140  16    1:1    16      ELSE
5141  16    1:2    20        REPEAT
5142  16    1:2    20
5143  16    1:2    20          (* BASE09 IS CHARACTER INDEX 1..6 (?) *)
5144  16    1:2    20
5145  16    1:3    20          P010A1B;
5146  16    1:3    22          IF BASE13 = 22 THEN
5147  16    1:4    28            P010A02;
5148  16    1:2    30        UNTIL BASE13 <> 22;
5149  16    1:2    36
5150  16    1:0    36    END;  (* CAMP MAIN LINE P010A01 *)
5151  16    1:0    50
5152  16    1:0    50
5153  16    1:0    50 (*$I WIZ4C:CAMP3 *)
5154  16    1:0    50
5154  16    1:0    50 (*$I WIZ4B:RUNNER  *)
5155  16    1:0    50
5156  17    1:D     1 SEGMENT PROCEDURE RUNNER;  (* P010B01 *)
5157  17    1:D     1
5158  17    1:D     1    CONST
5159  17    1:D     1          NORTH = 0;
5160  17    1:D     1          EAST  = 1;
5161  17    1:D     1          SOUTH = 2;
5162  17    1:D     1          WEST  = 3;
5163  17    1:D     1
5164  17    1:D     1    TYPE
5165  17    1:D     1          TMAZEDAT = PACKED ARRAY[ 0..3003] OF 0..255;
5166  17    1:D     1
5167  17    1:D     1    VAR
5168  17    1:D     1          INITTURN : BOOLEAN;
5169  17    1:D     2          NEEDDRMZ : BOOLEAN;
5170  17    1:D     3          NEEDDRAW : BOOLEAN;
5171  17    1:D     4          BSPIN    : BOOLEAN;
5172  17    1:D     5          LPIWIN   : TWINDOWP;
5173  17    1:D     6          CSIOWIN  : TWINDOWP;
5174  17    1:D     7
5175  17    1:D     7          MAZE     : TMAZE;
5176  17    1:D   454
5177  17    1:D   454          BMVTREB  : BOOLEAN;
5178  17    1:D   455          BMVORACL : BOOLEAN;
5179  17    1:D   456          SPINCNT  : INTEGER;
```

```
5180  17   1:D   457        UNUSED   : INTEGER;
5181  17   1:D   458
5182  17   1:D   458        ORACLEY  : INTEGER;
5183  17   1:D   459        ORACLEX  : INTEGER;
5184  17   1:D   460        MZLVORIG : INTEGER;
5185  17   1:D   461        COMMAND  : INTEGER;
5186  17   1:D   462        CHOICES  : STRING[ 30];  (* MP1CE *)
5187  17   1:D   478
5188  17   1:D   478        MAZEBLK0 : PACKED ARRAY[ 0..61] OF     (* MAZE.INFO *)
5189  17   1:D   478                     PACKED RECORD
5190  17   1:D   478                       A: 0..255;
5191  17   1:D   478                       B: 0..255;
5192  17   1:D   478                       C: INTEGER;
5193  17   1:D   478                     END;
5194  17   1:D   602
5195  17   1:D   602        MAZEDATA : TMAZEDAT;
5196  17   1:D  2104
5197  17   1:D  2104        MAZDAT00 : INTEGER;  (* INTERNAL ADDRESS FOR MAZEDATA *)
5198  17   1:D  2105
5199  17   1:D  2105
5200  17   2:D     1   PROCEDURE DRAWMAZE;  (* P010B02 *)
5201  17   2:D     1
5202  17   2:D     1     TYPE
5203  17   2:D     1          ARECORD = RECORD
5204  17   2:D     1                      A : INTEGER;
5205  17   2:D     1                      B : INTEGER;
5206  17   2:D     1                      C : INTEGER;
5207  17   2:D     1                      D : INTEGER;
5208  17   2:D     1                      E : INTEGER;
5209  17   2:D     1                      F : INTEGER;
5210  17   2:D     1                      G : INTEGER;
5211  17   2:D     1                    END;
5212  17   2:D     1     VAR
5213  17   2:D     1          RIGHTCOL : INTEGER;
5214  17   2:D     2          LEFTCOL  : INTEGER;
5215  17   2:D     3          Y4DRAW   : INTEGER;
5216  17   2:D     4          X4DRAW   : INTEGER;
5217  17   2:D     5          UNUSED   : INTEGER;
5218  17   2:D     6          NOTDARK  : BOOLEAN;
5219  17   2:D     7          GOTLIGHT : BOOLEAN;
5220  17   2:D     8          P02MP08  : ARECORD;
5221  17   2:D    15          P02MP0F  : ARECORD;
5222  17   2:D    22
5223  17   2:D    22
5224  17   3:D     1     PROCEDURE EXITDRMZ;  (* P010B03 *)
5225  17   3:D     1
5226  17   3:0     0       BEGIN (* EXITDRMZ *)
5227  17   3:1     0         UNITWRITE( BASE12, MAINWIN^, 3, 0);  (* DISPLAY WINDOW *)
5228  17   3:1    10         EXIT( DRAWMAZE)
5229  17   3:0    14       END;  (* EXITDRMZ *)
5230  17   3:0    26
5231  17   3:0    26
5232  17   4:D     1     PROCEDURE DRMZTHNG( MP01 : INTEGER);  (* P010B04 *)
5233  17   4:D     2
5234  17   4:0     0       BEGIN (* DRMZTHNG *)
5235  17   4:1     0         IF MP01 < 0 THEN
5236  17   4:2     5           EXIT( DRMZTHNG);
```

```
5237  17   4:1    9         P02MP08.A := LEFTCOL;
5238  17   4:1   15         P02MP08.B := RIGHTCOL;
5239  17   4:1   21         P02MP08.E := MAZEBLK0[ MP01].A;
5240  17   4:1   34         P02MP08.F := MAZEBLK0[ MP01].B;
5241  17   4:1   47         P02MP08.G := MAZDAT00 + MAZEBLK0[ MP01].C;
5242  17   4:1   63         UNITWRITE( BASE12, P02MP08, 34, ORD( MAINWIN))
5243  17   4:0   75       END;  (* DRMZTHNG *)
5244  17   4:0   88
5245  17   4:0   88
5246  17   5:D    1     PROCEDURE DRMZTHN2( MP03 : INTEGER;  (* P010B05 *)
5247  17   5:D    2                         MP02 : INTEGER;
5248  17   5:D    3                         MP01 : INTEGER);
5249  17   5:D    4
5250  17   5:0    0       BEGIN (* DRMZTHN2 *)
5251  17   5:1    0         IF MP03 < 0 THEN
5252  17   5:2    5           EXIT( DRMZTHN2);
5253  17   5:1    9         P02MP0F.A := LEFTCOL;
5254  17   5:1   15         P02MP0F.B := RIGHTCOL;
5255  17   5:1   21         P02MP0F.E := MAZEBLK0[ MP03].A;
5256  17   5:1   34         P02MP0F.F := MAZEBLK0[ MP03].B;
5257  17   5:1   47         P02MP0F.C := MP02;
5258  17   5:1   51         P02MP0F.D := MP01;
5259  17   5:1   55         P02MP0F.G := MAZDAT00 + MAZEBLK0[ MP03].C;
5260  17   5:1   71         UNITWRITE( BASE12, P02MP0F, 34, ORD( MAINWIN))
5261  17   5:0   83       END;  (* DRMZTHN2 *)
5262  17   5:0   96
5263  17   5:0   96
5264  17   6:D    1     PROCEDURE SHFTPOS( VAR X : INTEGER;  (* P010B06 *)
5265  17   6:D    2                        VAR Y : INTEGER;
5266  17   6:D    3                            RIGHSHFT : INTEGER;
5267  17   6:D    4                            FRWDSHFT : INTEGER);
5268  17   6:D    5
5269  17   6:0    0       BEGIN (* SHFTPOS *)
5270  17   6:1    0         CASE DIRECTIO OF
5271  17   6:1    3           NORTH:
5272  17   6:2    3               BEGIN
5273  17   6:3    3                 X := X + RIGHSHFT;
5274  17   6:3    9                 Y := Y + FRWDSHFT
5275  17   6:2   12               END;
5276  17   6:1   17           EAST:
5277  17   6:2   17               BEGIN
5278  17   6:3   17                 X := X + FRWDSHFT;
5279  17   6:3   23                 Y := Y - RIGHSHFT
5280  17   6:2   26               END;
5281  17   6:1   31           SOUTH:
5282  17   6:2   31               BEGIN
5283  17   6:3   31                 X := X - RIGHSHFT;
5284  17   6:3   37                 Y := Y - FRWDSHFT
5285  17   6:2   40               END;
5286  17   6:1   45           WEST:
5287  17   6:2   45               BEGIN
5288  17   6:3   45                 X := X - FRWDSHFT;
5289  17   6:3   51                 Y := Y + RIGHSHFT
5290  17   6:2   54               END;
5291  17   6:1   59         END;
5292  17   6:1   74
5293  17   6:1   74         X := (X + 20) MOD 20;
```

```
5294  17   6:1   82         Y := (Y + 20) MOD 20
5295  17   6:1   87
5296  17   6:0   87       END;  (* SHFTPOS *)
5297  17   6:0  102
5298  17   6:0  102
5299  17   7:D    3     FUNCTION ISLIGHT( RIGHSHFT : INTEGER;  (* P010B07 *)
5300  17   7:D    4                       FRWDSHFT : INTEGER) : BOOLEAN;
5301  17   7:D    5
5302  17   7:D    5       VAR
5303  17   7:D    5           SQARTYPE : INTEGER;
5304  17   7:D    6           Y : INTEGER;
5305  17   7:D    7           X : INTEGER;
5306  17   7:D    8
5307  17   7:D    8
5308  17   7:0    0       BEGIN (* ISLIGHT *)
5309  17   7:1    0         X := X4DRAW;
5310  17   7:1    5         Y := Y4DRAW;
5311  17   7:1   10         SHFTPOS( X, Y, RIGHSHFT, FRWDSHFT);
5312  17   7:1   18         SQARTYPE := MAZE.SQREXTRA[ X][ Y];
5313  17   7:1   32         IF MAZE.SQRETYPE[ SQARTYPE] = DARK THEN
5314  17   7:2   45           ISLIGHT := MAZE.AUX2[ SQARTYPE] <> 0
5315  17   7:1   53         ELSE
5316  17   7:2   59           ISLIGHT := TRUE
5317  17   7:0   59       END;  (* ISLIGHT *)
5318  17   7:0   74
5319  17   7:0   74
5320  17   8:D    3     FUNCTION FRWDVIEW( DELTAR : INTEGER) : TWALL;  (* P010B08 *)
5321  17   8:D    4
5322  17   8:D    4       VAR
5323  17   8:D    4           Y : INTEGER;
5324  17   8:D    5           X : INTEGER;
5325  17   8:D    6
5326  17   8:0    0       BEGIN (* FRWDVIEW *)
5327  17   8:1    0         X := X4DRAW;
5328  17   8:1    5         Y := Y4DRAW;
5329  17   8:1   10         SHFTPOS( X, Y, DELTAR, 0);
5330  17   8:1   18         CASE DIRECTIO OF
5331  17   8:1   21           NORTH:  FRWDVIEW := MAZE.N[ X][ Y];
5332  17   8:1   37            EAST:  FRWDVIEW := MAZE.E[ X][ Y];
5333  17   8:1   52           SOUTH:  FRWDVIEW := MAZE.S[ X][ Y];
5334  17   8:1   67            WEST:  FRWDVIEW := MAZE.W[ X][ Y];
5335  17   8:1   82         END;
5336  17   8:0   98       END;  (* FRWDVIEW *)
5337  17   8:0  110
5338  17   8:0  110
5339  17   9:D    3     FUNCTION LEFTVIEW( DELTAR : INTEGER) : TWALL;  (* P010B09 *)
5340  17   9:D    4
5341  17   9:D    4       VAR
5342  17   9:D    4           Y : INTEGER;
5343  17   9:D    5           X : INTEGER;
5344  17   9:D    6
5345  17   9:0    0       BEGIN (* LEFTVIEW *)
5346  17   9:1    0         X := X4DRAW;
5347  17   9:1    5         Y := Y4DRAW;
5348  17   9:1   10         SHFTPOS( X, Y, DELTAR, 0);
5349  17   9:1   18         CASE DIRECTIO OF
5350  17   9:1   21           NORTH:  LEFTVIEW := MAZE.W[ X][ Y];
```

```
5351  17    9:1    36            EAST:  LEFTVIEW := MAZE.N[ X][ Y];
5352  17    9:1    52           SOUTH:  LEFTVIEW := MAZE.E[ X][ Y];
5353  17    9:1    67            WEST:  LEFTVIEW := MAZE.S[ X][ Y];
5354  17    9:1    82          END;
5355  17    9:1    98
5356  17    9:0    98        END;  (* LEFTVIEW *)
5357  17    9:0   110
5358  17    9:0   110
5359  17   10:D     3     FUNCTION RIGHVIEW( DELTAR : INTEGER) : TWALL;  (* P010B0A *)
5360  17   10:D     4
5361  17   10:D     4       VAR
5362  17   10:D     4            Y : INTEGER;
5363  17   10:D     5            X : INTEGER;
5364  17   10:D     6
5365  17   10:0     0       BEGIN (* RIGHVIEW *)
5366  17   10:1     0         X := X4DRAW;
5367  17   10:1     5         Y := Y4DRAW;
5368  17   10:1    10         SHFTPOS( X, Y, DELTAR, 0);
5369  17   10:1    18         CASE DIRECTIO OF
5370  17   10:1    21           NORTH:  RIGHVIEW := MAZE.E[ X][ Y];
5371  17   10:1    36            EAST:  RIGHVIEW := MAZE.S[ X][ Y];
5372  17   10:1    51           SOUTH:  RIGHVIEW := MAZE.W[ X][ Y];
5373  17   10:1    66            WEST:  RIGHVIEW := MAZE.N[ X][ Y];
5374  17   10:1    82         END;
5375  17   10:0    98       END;  (* RIGHVIEW *)
5376  17   10:0   110
5377  17   10:0   110
5378  17   11:D     3     FUNCTION CHKHIDEN( WALL2CHK : TWALL) : TWALL;  (* P010B0B *)
5379  17   11:D     4
5380  17   11:D     4
5381  17   11:0     0       BEGIN (* CHKHIDEN *)
5382  17   11:1     0         IF WALL2CHK = HIDEDOOR THEN
5383  17   11:2     5           IF NOTDARK AND GOTLIGHT THEN
5384  17   11:3    14             WALL2CHK := DOOR
5385  17   11:2    14           ELSE
5386  17   11:3    19             WALL2CHK := WALL;
5387  17   11:1    22         CHKHIDEN := WALL2CHK
5388  17   11:0    22       END;  (* CHKHIDEN *)
5389  17   11:0    38
5390  17   11:0    38
5391  17   12:D     1     PROCEDURE STEPFRWD;  (* P010B0C *)
5392  17   12:D     1
5393  17   12:D     1       VAR
5394  17   12:D     1            SQREDESC : INTEGER;
5395  17   12:D     2
5396  17   12:0     0       BEGIN (* STEPFRWD *)
5397  17   12:1     0         SHFTPOS( X4DRAW, Y4DRAW, 0, 1);
5398  17   12:1    10         NOTDARK := TRUE;
5399  17   12:1    14         SQREDESC := MAZE.SQREXTRA[ X4DRAW][ Y4DRAW];
5400  17   12:1    32         IF MAZE.SQRETYPE[ SQREDESC] = DARK THEN
5401  17   12:2    45           NOTDARK := MAZE.AUX0[ SQREDESC] = 1
5402  17   12:1    53         ELSE
5403  17   12:2    60           IF MAZE.SQRETYPE[ SQREDESC] = TRANSFER THEN
5404  17   12:3    73             IF MAZE.AUX0[ SQREDESC] = MAZELEV THEN
5405  17   12:4    85               BEGIN
5406  17   12:5    85                 X4DRAW := MAZE.AUX2[ SQREDESC];
5407  17   12:5    96                 Y4DRAW := MAZE.AUX1[ SQREDESC]
```

```
5408  17  12:4   103              END
5409  17  12:0   107      END;  (* STEPFRWD *)
5410  17  12:0   120
5411  17  12:0   120
5412  17  13:D     1    PROCEDURE DRAWROW( MP18 : INTEGER;  (* P010B0D *)
5413  17  13:D     2                       MP17 : INTEGER;
5414  17  13:D     3                       MP16 : INTEGER;
5415  17  13:D     4                       MP15 : INTEGER;
5416  17  13:D     5                       MP14 : INTEGER;
5417  17  13:D     6                       MP13 : INTEGER;
5418  17  13:D     7                       MP12 : INTEGER;
5419  17  13:D     8                       MP11 : INTEGER;
5420  17  13:D     9                       MP10 : INTEGER;
5421  17  13:D    10                       MP0F : INTEGER;
5422  17  13:D    11                       MP0E : INTEGER;
5423  17  13:D    12                       MP0D : INTEGER;
5424  17  13:D    13                       MP0C : INTEGER;
5425  17  13:D    14                       MP0B : INTEGER;
5426  17  13:D    15                       MP0A : INTEGER;
5427  17  13:D    16                       MP09 : INTEGER;
5428  17  13:D    17                       MP08 : INTEGER;
5429  17  13:D    18                       MP07 : INTEGER;
5430  17  13:D    19                       MP06 : INTEGER;
5431  17  13:D    20                       MP05 : INTEGER;
5432  17  13:D    21                       MP04 : INTEGER;
5433  17  13:D    22                       MP03 : INTEGER;
5434  17  13:D    23                       MP02 : INTEGER;
5435  17  13:D    24                       MP01 : INTEGER);
5436  17  13:D    25
5437  17  13:D    25
5438  17  13:D    25      VAR
5439  17  13:D    25           (* DRAWROW VARS: *)
5440  17  13:D    25          WALLTHNG : TWALL;
5441  17  13:D    26          MP1A : INTEGER;
5442  17  13:D    27          MP1B : INTEGER;
5443  17  13:D    28          BLCKFRWD : BOOLEAN;
5444  17  13:D    29
5445  17  13:D    29
5446  17  14:D     1      PROCEDURE DRAWFLOR( P0EMP04 : INTEGER;  (* P010B0E *)
5447  17  14:D     2                          MP03 : INTEGER;
5448  17  14:D     3                          MP02 : INTEGER;
5449  17  14:D     4                          MP01 : INTEGER);
5450  17  14:D     5
5451  17  14:D     5        VAR
5452  17  14:D     5             MZDIFF13 : INTEGER;
5453  17  14:D     6             SQARDESC : INTEGER;
5454  17  14:D     7             DFY      : INTEGER;
5455  17  14:D     8             DFX      : INTEGER;
5456  17  14:D     9
5457  17  14:D     9
5458  17  14:0     0        BEGIN (* DRAWFLOR *)
5459  17  14:1     0          DFX := X4DRAW;
5460  17  14:1     5          DFY := Y4DRAW;
5461  17  14:1    10          SHFTPOS( DFX, DFY, MP02, 0);
5462  17  14:1    18          SQARDESC := MAZE.SQREXTRA[ DFX][ DFY];
5463  17  14:1    32          MZDIFF13 := MAZELEV;
5464  17  14:1    35          IF MZDIFF13 > 12 THEN
```

```
5465  17   14:2    40            MZDIFF13 := 13 - MZDIFF13;
5466  17   14:2    45
5467  17   14:1    45          CASE MAZE.SQRETYPE[ SQARDESC] OF
5468  17   14:1    56
5469  17   14:1    56            STAIRS:
5470  17   14:2    56                 BEGIN
5471  17   14:3    56                   IF MZDIFF13 < MAZE.AUX0[ SQARDESC] THEN
5472  17   14:4    68                     DRMZTHNG( MP03)
5473  17   14:3    69                   ELSE
5474  17   14:4    73                     DRMZTHNG( P0EMP04)
5475  17   14:2    74                 END;
5476  17   14:2    78
5477  17   14:1    78            BUTTONZ:
5478  17   14:2    78                 BEGIN
5479  17   14:3    78                   IF MZDIFF13 < MAZE.AUX1[ SQARDESC] THEN
5480  17   14:4    90                     DRMZTHNG( MP03);
5481  17   14:3    93                   IF MZDIFF13 > MAZE.AUX2[ SQARDESC] THEN
5482  17   14:4   105                     DRMZTHNG( P0EMP04);
5483  17   14:2   108                 END;
5484  17   14:2   110
5485  17   14:1   110            SCNMSG:
5486  17   14:2   110                 DRMZTHNG( MP03);
5487  17   14:1   115          END;
5488  17   14:1   144
5489  17   14:1   144          IF ORACLEX = DFX THEN
5490  17   14:2   152            IF ORACLEY = DFY THEN
5491  17   14:3   160              BEGIN
5492  17   14:4   160                BMVORACL := TRUE;
5493  17   14:4   165                DRMZTHN2( MP04, MP01, 0)  (* CAREFUL WITH MP04 ! *)
5494  17   14:3   170              END
5495  17   14:3   172
5496  17   14:0   172        END;  (* DRAWFLOR *)
5497  17   14:0   184
5498  17   14:0   184
5499  17   15:D     1      PROCEDURE UNUSED;  (* P010B0F *)
5500  17   15:D     1
5501  17   15:0     0        BEGIN (* UNUSED *)
5502  17   15:1     0          DRAWFLOR( MP0B, MP0A, 0, 0)
5503  17   15:0     8        END;  (* UNUSED *)
5504  17   15:0    22
5505  17   15:0    22
5506  17   15:0    22
5507  17   13:0     0      BEGIN (* DRAWROW *)
5508  17   13:1     0        MP1B := LEFTCOL;
5509  17   13:1     5        MP1A := RIGHTCOL;
5510  17   13:1    10        WALLTHNG := CHKHIDEN( FRWDVIEW( 0));
5511  17   13:1    21                                    (* CONVERT HIDEDOOR TO DOOR/WALL *)
5512  17   13:1    21        BLCKFRWD := TRUE;
5513  17   13:1    24        IF WALLTHNG = WALL THEN
5514  17   13:2    30          DRMZTHNG( MP14)
5515  17   13:1    32        ELSE IF WALLTHNG = DOOR THEN
5516  17   13:3    42          DRMZTHNG( MP13)
5517  17   13:2    44        ELSE   (* WALLTHNG = OPEN *)
5518  17   13:3    48          IF ISLIGHT( 0, 1) THEN
5519  17   13:4    56            BLCKFRWD := FALSE             (* LIGHT STRAIGHT AHEAD *)
5520  17   13:3    56          ELSE
5521  17   13:4    61            DRMZTHNG( MP0D);
```

```
5522  17   13:1    64          DRAWFLOR( MP0B, MP0A, 0, 0);
5523  17   13:1    70          WALLTHNG := CHKHIDEN( LEFTVIEW( 0));
5524  17   13:1    81          IF WALLTHNG = WALL THEN
5525  17   13:2    87            BEGIN
5526  17   13:3    87              DRMZTHNG( MP18);
5527  17   13:3    91              MP1B := MP03
5528  17   13:2    91            END
5529  17   13:1    94          ELSE IF WALLTHNG = DOOR THEN
5530  17   13:3   102            BEGIN
5531  17   13:4   102              DRMZTHNG( MP17);
5532  17   13:4   106              MP1B := MP03
5533  17   13:3   106            END
5534  17   13:2   109          ELSE
5535  17   13:3   111            BEGIN
5536  17   13:4   111              IF ISLIGHT( -1, 0) THEN
5537  17   13:5   120                BEGIN                    (* LIGHT ON THE LEFT *)
5538  17   13:6   120                  WALLTHNG := CHKHIDEN( FRWDVIEW( -1));
5539  17   13:6   132                  IF WALLTHNG = WALL THEN
5540  17   13:7   138                    BEGIN
5541  17   13:8   138                      DRMZTHNG( MP12);
5542  17   13:8   142                      MP1B := MP03
5543  17   13:7   142                    END
5544  17   13:6   145                  ELSE IF WALLTHNG = DOOR THEN
5545  17   13:8   153                    BEGIN
5546  17   13:9   153                      DRMZTHNG( MP11);
5547  17   13:9   157                      MP1B := MP03
5548  17   13:8   157                    END
5549  17   13:7   160                  ELSE
5550  17   13:8   162                    BEGIN
5551  17   13:9   162                      IF NOT ISLIGHT( -1, 1) THEN
5552  17   13:0   172                        BEGIN           (* DARKNESS AHEAD ON THE LEFT *)
5553  17   13:1   172                          DRMZTHN2( MP0D, -MP01, 0);
5554  17   13:1   178                          MP1B := MP03
5555  17   13:0   178                        END
5556  17   13:8   181                    END;
5557  17   13:8   181
5558  17   13:6   181                  DRAWFLOR( MP09, MP07, -1, -MP01);
5559  17   13:6   189
5560  17   13:5   189                END
5561  17   13:4   189              ELSE
5562  17   13:5   191                BEGIN
5563  17   13:6   191                  DRMZTHNG( MP0E);
5564  17   13:6   194                  MP1B := MP03
5565  17   13:5   194                END
5566  17   13:3   197            END;
5567  17   13:3   197
5568  17   13:1   197          WALLTHNG := CHKHIDEN( RIGHVIEW( 0));
5569  17   13:1   208          IF WALLTHNG = WALL THEN
5570  17   13:2   214            BEGIN
5571  17   13:3   214              DRMZTHNG( MP16);
5572  17   13:3   218              MP1A := MP02
5573  17   13:2   218            END
5574  17   13:1   221          ELSE IF WALLTHNG = DOOR THEN
5575  17   13:3   229            BEGIN
5576  17   13:4   229              DRMZTHNG( MP15);
5577  17   13:4   233              MP1A := MP02
5578  17   13:3   233            END
```

```
5579  17   13:2   236         ELSE
5580  17   13:3   238           BEGIN
5581  17   13:4   238             IF ISLIGHT( 1, 0) THEN
5582  17   13:5   246               BEGIN                  (* LIGHT ON THE RIGHT *)
5583  17   13:6   246                 WALLTHNG := CHKHIDEN( FRWDVIEW( 1));
5584  17   13:6   257                 IF WALLTHNG = WALL THEN
5585  17   13:7   263                   BEGIN
5586  17   13:8   263                     DRMZTHNG( MP10);
5587  17   13:8   266                     MP1A := MP02
5588  17   13:7   266                   END
5589  17   13:6   269                 ELSE IF WALLTHNG = DOOR THEN
5590  17   13:8   277                   BEGIN
5591  17   13:9   277                     DRMZTHNG( MP0F);
5592  17   13:9   280                     MP1A := MP02
5593  17   13:8   280                   END
5594  17   13:7   283                 ELSE
5595  17   13:8   285                   BEGIN
5596  17   13:9   285                     IF NOT ISLIGHT( 1, 1) THEN
5597  17   13:0   294                       BEGIN          (* DARKNESS AHEAD ON THE RIGHT *)
5598  17   13:1   294                         DRMZTHN2( MP0D, MP01, 0);
5599  17   13:1   299                         MP1A := MP02
5600  17   13:0   299                       END
5601  17   13:8   302                   END;
5602  17   13:8   302
5603  17   13:6   302                 DRAWFLOR( MP08, MP06, 1, MP01)
5604  17   13:5   306               END
5605  17   13:4   308             ELSE
5606  17   13:5   310               BEGIN
5607  17   13:6   310                 DRMZTHNG( MP0C);
5608  17   13:6   313                 MP1A := MP02
5609  17   13:5   313               END;
5610  17   13:3   316           END;
5611  17   13:3   316
5612  17   13:1   316         IF BLCKFRWD THEN
5613  17   13:2   320           EXITDRMZ;
5614  17   13:1   322         LEFTCOL := MP1B;
5615  17   13:1   327         RIGHTCOL := MP1A
5616  17   13:1   327
5617  17   13:0   327       END;  (* DRAWROW *)
5618  17   13:0   344
5619  17   13:0   344
5620  17    2:0     0     BEGIN (* DRAWMAZE *)
5621  17    2:1     0       FILLCHAR( P02MP08, 14, CHR( 0));
5622  17    2:1     7       P02MP0F := P02MP08;
5623  17    2:1    13       BMVORACL := FALSE;
5624  17    2:1    18       X4DRAW := MAZEX;
5625  17    2:1    21       Y4DRAW := MAZEY;
5626  17    2:1    24       IF IDMONSTR > 0 THEN
5627  17    2:2    30         IDMONSTR := IDMONSTR - 1;
5628  17    2:1    36       IF LIGHT > 0 THEN
5629  17    2:2    42         LIGHT := LIGHT - 1;
5630  17    2:1    48       GOTLIGHT := LIGHT > 0;
5631  17    2:1    54       UNUSED := 1;
5632  17    2:1    57       FILLCHAR( MAINWIN^.DATA, 36 * 40, ' ');
5633  17    2:1    68       LEFTCOL := 0;
5634  17    2:1    71       RIGHTCOL := 35;
5635  17    2:1    74       IF NOT ISLIGHT( 0, 0) THEN
```

```
5636  17    2:2    83         BEGIN                          (* DARKNESS AT CURRENT LOCATION *)
5637  17    2:3    83           DRMZTHNG( 61);
5638  17    2:3    86           EXITDRMZ
5639  17    2:2    86         END;
5640  17    2:1    88       SHFTPOS( X4DRAW, Y4DRAW, 0, -1);
5641  17    2:1    97       STEPFRWD;
5642  17    2:1    99       DRAWROW( 6, 14, 7, 15, 18, 21, 26, 32, 27, 33, 40, 44, 41, 45, 48, 49,
5643  17    2:1   115              53, 52, 56, 58, 60, 11, 24, 14);
5644  17    2:1   125       STEPFRWD;
5645  17    2:1   127       DRAWROW( 4, 12, 5, 13, 17, 20, 24, 30, 25, 31, 38, 43, 39, 46, 47, 50,
5646  17    2:1   143              54, 51, 55, 57, 59, 15, 20, 6);
5647  17    2:1   153       IF GOTLIGHT THEN
5648  17    2:2   156         BEGIN
5649  17    2:3   156           STEPFRWD;
5650  17    2:3   158           DRAWROW( 2, 10, 3, 11, 16, 19, 22, 28, 23, 29, 36, 42, 37, -1, -1,
5651  17    2:3   175                   -1, -1, -1, -1, -1, -1, 17, 18, 2);
5652  17    2:3   192           STEPFRWD;
5653  17    2:3   194           DRAWROW( 0, 8, 1, 9, -1, -1, -1, -1, -1, -1, 34, -1, 35, -1, -1,
5654  17    2:3   218                   -1, -1, -1, -1, -1, -1, 17, 18, 2)
5655  17    2:2   233         END;
5656  17    2:2   235
5657  17    2:1   235       EXITDRMZ
5658  17    2:1   235
5659  17    2:0   235     END;  (* DRAWMAZE *)
5660  17    2:0   250
5661  17    2:0   250
5662  17   16:D     1   PROCEDURE READMAZ( LEVEL : INTEGER);  (* P010B10 *)
5663  17   16:D     2
5664  17   16:0     0     BEGIN (* READMAZ *)
5665  17   16:1     0       MOVELEFT( BASE55B[ GETREC( 1, (LEVEL - 1), SIZEOF( MAZE))],
5666  17   16:1    15                 MAZE,
5667  17   16:1    19                 SIZEOF( MAZE))
5668  17   16:0    24     END;  (* READMAZ *)
5669  17   16:0    36
5670  17   16:0    36
5671  17   17:D     1   PROCEDURE WRITEMAZ( LEVEL : INTEGER);  (* P010B11 *)
5672  17   17:D     2
5673  17   17:0     0     BEGIN (* WRITEMAZ *)
5674  17   17:1     0       MOVELEFT( MAZE,
5675  17   17:1     4                 BASE55B[ GETRECW( 1, LEVEL - 1, SIZEOF( MAZE))],
5676  17   17:1    19                 SIZEOF( MAZE))
5677  17   17:0    24     END;  (* WRITEMAZ *)
5678  17   17:0    36
5679  17   17:0    36
5680  17   18:D     1   PROCEDURE CHKWRTMZ( WRITFLAG : BOOLEAN);  (* P010B12 *)
5681  17   18:D     2
5682  17   18:0     0     BEGIN (* CHKWRTMZ *)
5683  17   18:1     0       IF WRITFLAG OR (MZLVORIG = 12) OR (MZLVORIG = 5) THEN
5684  17   18:2    17         WRITEMAZ( MZLVORIG)
5685  17   18:0    21     END;  (* CHKWRTMZ *)
5686  17   18:0    36
5687  17   18:0    36
5688  17   19:D     1   PROCEDURE SPINBLCK( P13MP02 : INTEGER;  (* P010B13 *)
5689  17   19:D     2                      P13MP01 : INTEGER);
5690  17   19:D     3
5691  17   20:D     1     PROCEDURE ROTATE1;  (* P010B14 *)
5692  17   20:D     1
```

```
5693  17   20:D     1       VAR
5694  17   20:D     1           MP01 : TWALL;
5695  17   20:D     2           MP02 : TWALL;
5696  17   20:D     3           MP03 : TWALL;
5697  17   20:D     4           MP04 : TWALL;
5698  17   20:D     5           MP05 : TWALL;
5699  17   20:D     6           MP06 : TWALL;
5700  17   20:D     7           MP07 : TWALL;
5701  17   20:D     8           MP08 : TWALL;
5702  17   20:D     9
5703  17   20:D     9
5704  17   21:D     3         FUNCTION NEXTLOC( MP03 : INTEGER) : INTEGER;  (* P010B15 *)
5705  17   21:D     4
5706  17   21:0     0           BEGIN (* NEXTLOC *)
5707  17   21:1     0             NEXTLOC := (MP03 + 20) MOD 20
5708  17   21:0     3           END;  (* NEXTLOC *)
5709  17   21:0    20
5710  17   21:0    20
5711  17   20:0     0       BEGIN (* ROTATE1 *)
5712  17   20:1     0         MP07 := MAZE.N[ P13MP02             ][ P13MP01];
5713  17   20:1    18         MP03 := MAZE.S[ P13MP02             ][ NEXTLOC( P13MP01 + 1)];
5714  17   20:1    41         MP06 := MAZE.E[ P13MP02             ][ P13MP01];
5715  17   20:1    58         MP02 := MAZE.W[ NEXTLOC( P13MP02 + 1)][ P13MP01];
5716  17   20:1    81         MP08 := MAZE.S[ P13MP02             ][ P13MP01];
5717  17   20:1    98         MP04 := MAZE.N[ P13MP02             ][ NEXTLOC( P13MP01 - 1)];
5718  17   20:1   122         MP05 := MAZE.W[ P13MP02             ][ P13MP01];
5719  17   20:1   139         MP01 := MAZE.E[ NEXTLOC( P13MP02 - 1)][ P13MP01];
5720  17   20:1   162
5721  17   20:1   162         MAZE.N[ P13MP02             ][ P13MP01]              := MP05;
5722  17   20:1   179         MAZE.S[ P13MP02             ][ NEXTLOC( P13MP01 + 1)] := MP01;
5723  17   20:1   201         MAZE.E[ P13MP02             ][ P13MP01]              := MP07;
5724  17   20:1   217         MAZE.W[ NEXTLOC( P13MP02 + 1)][ P13MP01]             := MP03;
5725  17   20:1   239         MAZE.S[ P13MP02             ][ P13MP01]              := MP06;
5726  17   20:1   255         MAZE.N[ P13MP02             ][ NEXTLOC( P13MP01 - 1)] := MP02;
5727  17   20:1   278         MAZE.W[ P13MP02             ][ P13MP01]              := MP08;
5728  17   20:1   294         MAZE.E[ NEXTLOC( P13MP02 - 1)][ P13MP01]             := MP04
5729  17   20:1   314
5730  17   20:0   314       END;  (* ROTATE1 *)
5731  17   20:0   328
5732  17   20:0   328
5733  17   20:0   328
5734  17   20:0   328
5735  17   19:0     0     BEGIN (* SPINBLCK *)
5736  17   19:1     0       IF NOT BSPIN THEN
5737  17   19:2     6         EXIT( SPINBLCK);
5738  17   19:1    10       BSPIN := FALSE;
5739  17   19:1    14       IF SPINCNT = 4 THEN
5740  17   19:2    22         SPINCNT := 1 + (RAND MOD 3);
5741  17   19:1    35       WHILE SPINCNT > 0 DO
5742  17   19:2    43         BEGIN
5743  17   19:3    43           ROTATE1;
5744  17   19:3    45           SPINCNT := SPINCNT - 1
5745  17   19:2    49         END
5746  17   19:0    55     END;  (* SPINBLCK *)
5747  17   19:0    72
5748  17   19:0    72
5749  17   22:D     1   PROCEDURE SPECMSG( MP01 : INTEGER);  (* P010B16 *)
```

```
5750  17   22:D     2
5751  17   22:D     2     VAR
5752  17   22:D     2          MP02 : TWINDOWP;
5753  17   22:D     3          MP03 : INTEGER;
5754  17   22:D     4
5755  17   22:0     0     BEGIN (* SPECMSG *)
5756  17   22:1     0       GETMSGTX( BASE47, MP01);
5757  17   22:1     6       MP03 := (MBSTRLEN( BASE47) + 1) DIV 2;
5758  17   22:1    19       MP02 := GETWIN( 19 - MP03, 10, 2 + MP03 + MP03, 3, 15, TRUE);
5759  17   22:1    38       PRPICMB( MP02, BASE47);
5760  17   22:1    44       DELWIN( MP02, FALSE);
5761  17   22:1    50       NEEDDRAW := TRUE
5762  17   22:0    50     END;  (* SPECMSG *)
5763  17   22:0    66
5764  17   22:0    66
5765  17   23:D     1   PROCEDURE DRAWSCRN( MP01 : BOOLEAN);  (* P010B17 *)
5766  17   23:D     2
5767  17   23:0     0     BEGIN (* DRAWSCRN *)
5768  17   23:1     0       DELWIN( CSIOWIN, FALSE);
5769  17   23:1     7       DELWIN( LPIWIN, FALSE);
5770  17   23:1    14       IF MP01 THEN
5771  17   23:2    17         UNPROWIN( CHARSWIN, FALSE);
5772  17   23:1    22       DRAWSCR2( WINCHAIN);
5773  17   23:1    26       CHKWRTMZ( FALSE)
5774  17   23:0    27     END;  (* DRAWSCRN *)
5775  17   23:0    42
5776  17   23:0    42
5777  17   24:D     1   PROCEDURE MVTREBOR;  (* P010B18 *)
5778  17   24:D     1
5779  17   24:D     1     VAR
5780  17   24:D     1          STEPAWAY : INTEGER;
5781  17   24:D     2          TREBWIN : TWINDOWP;
5782  17   24:D     3
5783  17   24:0     0     BEGIN (* MVTREBOR *)
5784  17   24:1     0       IF BASE7F4[ 5-1] <> 0 THEN
5785  17   24:2    13         EXIT( MVTREBOR);
5786  17   24:1    17       BMVTREB := FALSE;
5787  17   24:1    22       IF (RAND MOD 2) = 0 THEN
5788  17   24:2    33         IF TREBORX > MAZEX THEN
5789  17   24:3    40           TREBORX := TREBORX - 1
5790  17   24:2    43         ELSE
5791  17   24:3    50           TREBORX := TREBORX + 1   (* SEEMS WIERD FOR '=' TO MOVE AWAY *)
5792  17   24:1    53       ELSE
5793  17   24:2    60         IF TREBORY > MAZEY THEN
5794  17   24:3    67           TREBORY := TREBORY - 1
5795  17   24:2    70         ELSE
5796  17   24:3    77           TREBORY := TREBORY + 1;  (* SEEMS WIERD FOR '=' TO MOVE AWAY *)
5797  17   24:1    85       STEPAWAY := ABS( MAZEX - TREBORX) + ABS( MAZEY - TREBORY);
5798  17   24:1   100       IF STEPAWAY <= (CHGMSGP2 DIV 2) THEN
5799  17   24:2   109         BEGIN
5800  17   24:3   109           CHGMSGP2 := CHGMSGP2 DIV 2;
5801  17   24:3   117           TREBMSG := TREBMSG + 1;  (* W..E..R..D..N..A.., ETC. *)
5802  17   24:3   125           SPECMSG( TREBMSG);
5803  17   24:3   130           PAUSE2
5804  17   24:2   130         END
5805  17   24:1   133       ELSE
5806  17   24:2   135         BEGIN
```

```
5807  17   24:3   135          IF (RAND MOD 10) = 5 THEN
5808  17   24:4   146            BEGIN
5809  17   24:5   146              SPECMSG( TREBMSG);
5810  17   24:5   151              PAUSE2
5811  17   24:4   151            END
5812  17   24:2   154        END;
5813  17   24:2   154
5814  17   24:1   154      IF STEPAWAY = 0 THEN
5815  17   24:2   159        BEGIN
5816  17   24:3   159          PAUSE2;
5817  17   24:3   162          TREBWIN := GETWIN( 0, 9, 40, 5, 16, TRUE);
5818  17   24:3   175
5819  17   24:3   175          FOR STEPAWAY := 2506 TO 2508 DO
5820  17   24:4   190            DISP1LIN( TREBWIN, STEPAWAY);
5821  17   24:4   202
5822  17   24:4   202            (* THE GHOST OF TREBOR APPEARS  *)
5823  17   24:4   202            (* THROUGH THE WALL.  HIS TOUCH *)
5824  17   24:4   202            (* IS CHILLING UNTO DEATH!      *)
5825  17   24:4   202
5826  17   24:3   202          CHARACTR[ 6].STATUS := DEAD;
5827  17   24:3   212          BASE13 := 14;
5828  17   24:3   215          FOR STEPAWAY := 0 TO 2 DO
5829  17   24:4   226            PAUSE2;
5830  17   24:3   236          DELWIN( TREBWIN, FALSE);
5831  17   24:3   242          DRAWSCRN( FALSE);
5832  17   24:3   245          EXIT( RUNNER)
5833  17   24:2   249        END
5834  17   24:2   249
5835  17   24:0   249    END;  (* MVTREBOR *)
5836  17   24:0   266
5837  17   24:0   266
5838  17   25:D     1  PROCEDURE GETCMD;  (* P010B19 *)
5839  17   25:D     1
5840  17   25:D     1    VAR
5841  17   25:D     1         MP01 : INTEGER;
5842  17   25:D     2         MP02 : INTEGER;
5843  17   25:D     3
5844  17   25:0     0    BEGIN (* GETCMD *)
5845  17   25:1     0      MP01 := 0;
5846  17   25:1     3      REPEAT
5847  17   25:2     3        MP02 := 1;
5848  17   25:2     6        UNITWRITE( BASE12, MP02, 19, MP02);  (* TIMED WAIT LOOP FOR KEYPRESS *)
5849  17   25:2    16        IF NOT KEYPRESS THEN
5850  17   25:3    24          BEGIN
5851  17   25:4    24            MP01 := MP01 + 1;
5852  17   25:4    29            IF MP01 = 90 THEN
5853  17   25:5    34              BEGIN
5854  17   25:6    34                MVTREBOR;
5855  17   25:6    36                MP01 := 0
5856  17   25:5    36              END
5857  17   25:3    39          END
5858  17   25:1    39      UNTIL KEYPRESS;
5859  17   25:1    46      IF (RAND MOD 10) = 5 THEN
5860  17   25:2    57        BMVTREB := TRUE;
5861  17   25:1    62      GETKEY;
5862  17   25:1    65      FOR MP02 := 1 TO LENGTH( CHOICES) DO
5863  17   25:2    81        IF CHOICES[ MP02] = INCHAR THEN
```

```
5864  17   25:3    91           BEGIN
5865  17   25:4    91             COMMAND := MP02;
5866  17   25:4    96             EXIT( GETCMD)
5867  17   25:3   100           END;
5868  17   25:1   107       COMMAND := -1
5869  17   25:0   107     END;  (* GETCMD *)
5870  17   25:0   130
5871  17   25:0   130 (*$I WIZ4B:RUNNER  *)
5871  17   25:0   130 (*$I WIZ4B:RUNNER2 *)
5872  17   25:0   130
5873  17   25:0   130 (* RUNNER2 *)
5874  17   25:0   130
5875  17   26:D     1   PROCEDURE CLROOMFG( POSX : INTEGER;  (* P010B1A *)
5876  17   26:D     2                      POSY : INTEGER);
5877  17   26:D     3
5878  17   26:D     3     VAR
5879  17   26:D     3          MP03     : INTEGER;
5880  17   26:D     4          YLOOP    : INTEGER;
5881  17   26:D     5          MP05     : INTEGER;
5882  17   26:D     6          XLOOP    : INTEGER;
5883  17   26:D     7          DONECHK  : BOOLEAN;
5884  17   26:D     8
5885  17   26:D     8          FIGHTCHK : PACKED ARRAY[ 0..19] OF PACKED ARRAY[ 0..19] OF BOOLEAN;
5886  17   26:D    48
5887  17   26:D    48
5888  17   27:D     1     PROCEDURE CHK4FIGH( NEWX:     INTEGER;     (* P010B1B *)
5889  17   27:D     2                         NEWY:     INTEGER;
5890  17   27:D     3                         WALL2NEW: TWALL);
5891  17   27:D     4
5892  17   27:D     4       VAR
5893  17   27:D     4            MP04 : 0..15;
5894  17   27:D     5
5895  17   27:D     5
5896  17   27:0     0       BEGIN (* P010B1B *)
5897  17   27:1     0         IF WALL2NEW <> OPEN THEN
5898  17   27:2     5           EXIT( CHK4FIGH);
5899  17   27:1     9         NEWX := (NEWX + 20) MOD 20;
5900  17   27:1    16         NEWY := (NEWY + 20) MOD 20;
5901  17   27:1    23         MP04 := MAZE.SQREXTRA[ NEWX][ NEWY];
5902  17   27:1    37         IF (FIGHTCHK[ NEWX][ NEWY]) OR
5903  17   27:1    48            (MAZE.SQRETYPE[ MP04] = ENCOUNTE) THEN
5904  17   27:2    62           EXIT( CHK4FIGH);
5905  17   27:2    66
5906  17   27:1    66         IF (MAZE.SQRETYPE[ MP04] = SCNMSG) AND
5907  17   27:1    77            (MAZE.AUX2[ MP04] = 600) THEN
5908  17   27:2    92           EXIT( CHK4FIGH);
5909  17   27:2    96
5910  17   27:1    96         DONECHK := FALSE;
5911  17   27:1   100         FIGHTCHK[ NEWX][ NEWY] := TRUE
5912  17   27:1   110
5913  17   27:0   110       END;  (* P010B1B *)
5914  17   27:0   124
5915  17   27:0   124
5916  17   26:0     0     BEGIN (* CLROOMFG *)
5917  17   26:1     0       MVCURSOR( MAINWIN, 0, 6);
5918  17   26:1     7       FILLCHAR( FIGHTCHK, SIZEOF( FIGHTCHK), CHR( 0));
5919  17   26:1    14       CHK4FIGH( POSX, POSY, OPEN);
```

```
5920  17   26:1    19
5921  17   26:1    19       REPEAT
5922  17   26:2    19         DONECHK := TRUE;
5923  17   26:2    22         FOR MP05 := POSX - 4 TO POSX + 4 DO
5924  17   26:3    38           FOR MP03 := POSY - 4 TO POSY + 4 DO
5925  17   26:4    54             BEGIN
5926  17   26:5    54               XLOOP := (MP05 + 20) MOD 20;
5927  17   26:5    61               YLOOP := (MP03 + 20) MOD 20;
5928  17   26:5    68               IF FIGHTCHK[ XLOOP][ YLOOP] AND FIGHTMAP[ XLOOP][ YLOOP] THEN
5929  17   26:6    92                 BEGIN
5930  17   26:7    92                   CHK4FIGH( XLOOP + 1, YLOOP,     MAZE.E[ XLOOP][ YLOOP]);
5931  17   26:7   109                   CHK4FIGH( XLOOP - 1, YLOOP,     MAZE.W[ XLOOP][ YLOOP]);
5932  17   26:7   126                   CHK4FIGH( XLOOP,     YLOOP - 1, MAZE.S[ XLOOP][ YLOOP]);
5933  17   26:7   143                   CHK4FIGH( XLOOP,     YLOOP + 1, MAZE.N[ XLOOP][ YLOOP]);
5934  17   26:7   161                   FIGHTMAP[ XLOOP][ YLOOP] := FALSE
5935  17   26:6   171                 END;
5936  17   26:4   173             END
5937  17   26:1   173         UNTIL DONECHK
5938  17   26:1   187
5939  17   26:0   187     END;  (* CLROOMFG *)
5940  17   26:0   210
5941  17   26:0   210
5942  17   28:D     1   PROCEDURE DSPPARTY;  (* P010B1C *)
5943  17   28:D     1
5944  17   28:D     1     VAR
5945  17   28:D     1           MP01 : INTEGER;
5946  17   28:D     2           HEALDIFF : INTEGER;
5947  17   28:D     3           MP03 : INTEGER;
5948  17   28:D     4           MP04 : INTEGER;
5949  17   28:D     5           MP05 : ARRAY[ 0..103] OF INTEGER;
5950  17   28:D   109
5951  17   28:D   109           MP6D : BOOLEAN;
5952  17   28:D   110
5953  17   28:D   110
5954  17   29:D     1     PROCEDURE DSSTATUS;  (* P010B1D *)
5955  17   29:D     1
5956  17   29:0     0       BEGIN (* DSSTATUS *)
5957  17   29:1     0         WITH CHARACTR[ 6] DO
5958  17   29:2     8           BEGIN
5959  17   29:3     8             HEALDIFF := HEALPTS - LOSTXYL.POISNAMT[ 1];
5960  17   29:3    24             IF HEALDIFF = 0 THEN
5961  17   29:4    31               PRPICCH( CHARSWIN, ' ')
5962  17   29:3    33             ELSE IF HEALDIFF < 0 THEN
5963  17   29:5    45               PRPICCH( CHARSWIN, '-')
5964  17   29:4    47             ELSE
5965  17   29:5    52               PRPICCH( CHARSWIN, '+');
5966  17   29:5    57
5967  17   29:3    57             MP6D := MP6D OR (STATUS <= ASLEEP);
5968  17   29:3    69
5969  17   29:3    69             IF STATUS = OK THEN
5970  17   29:4    76               IF LOSTXYL.POISNAMT[ 1] = 0 THEN
5971  17   29:5    89                 PRINTNUM( CHARSWIN, HPMAX, 4)
5972  17   29:4    94               ELSE
5973  17   29:5    99                 DISP1MSG( CHARSWIN, 700)    (* POISON *)
5974  17   29:3   103             ELSE
5975  17   29:4   108               BEGIN
5976  17   29:5   108                 GETMSGTX( BASE47, 10040 + ORD( STATUS)); (* OK, AFRAID, ... *)
```

```
5977  17  29:5   120                COPYSTR( BASE47, BASE47, 1, 6);
5978  17  29:5   129                PRPICMB( CHARSWIN, BASE47)
5979  17  29:4   132              END
5980  17  29:2   135            END
5981  17  29:0   135        END;  (* DSSTATUS *)
5982  17  29:0   148
5983  17  29:0   148
5984  17  28:0     0      BEGIN (* DSPPARTY *)
5985  17  28:1     0        MP6D := FALSE;
5986  17  28:1     3        PROTWIN(  CHARSWIN, FALSE);
5987  17  28:1     8        MVCURSOR( CHARSWIN, 0, 0);
5988  17  28:1    14        DISP1MSG(  CHARSWIN, 402);  (* # CHARACTER NAME CLASS AC HITS STATUS *)
5989  17  28:1    21        FILLCHAR( CHARSWIN^.DATA[ 76], 38 * 12, ' ');
5990  17  28:1    31        MVCURSOR( CHARSWIN, 0, 1);
5991  17  28:1    37        PRINTNUM( CHARSWIN, 1, 1);
5992  17  28:1    43        CHARSWIN^.HEAD.HCURSOR := 2;
5993  17  28:1    50        PRPICMB(  CHARSWIN, CHARACTR[ 6].NAME);
5994  17  28:1    60        CHARSWIN^.HEAD.HCURSOR := 18;
5995  17  28:1    67        GETMSGTX( BASE47, 10009 + ORD( CHARACTR[ 6].ALIGN));
5996  17  28:1    84        COPYSTR( BASE47, BASE47, 1, 1);
5997  17  28:1    93        PRPICMB(  CHARSWIN, BASE47);
5998  17  28:1    99        PRPICCH(  CHARSWIN, '-');
5999  17  28:1   104        GETMSGTX( BASE47, 10030 + ORD( CHARACTR[ 6].CLASS));
6000  17  28:1   121        COPYSTR( BASE47, BASE47, 1, 3);
6001  17  28:1   130        PRPICMB(  CHARSWIN, BASE47);
6002  17  28:1   136        PRPICCH(  CHARSWIN, ' ');
6003  17  28:1   141        BASE09 := CHARACTR[ 6].ARMORCL - ACMOD2;
6004  17  28:1   154        IF BASE09 > -10 THEN
6005  17  28:2   160          PRINTNUM( CHARSWIN, BASE09, 2)
6006  17  28:1   163        ELSE
6007  17  28:2   168          DISP1MSG( CHARSWIN, 701);   (* "LO" *)
6008  17  28:1   175        PRINTNUM( CHARSWIN, CHARACTR[ 6].HPLEFT, 5);
6009  17  28:1   188
6010  17  28:1   188        DSSTATUS;
6011  17  28:1   190
6012  17  28:1   190        MVCURSOR( CHARSWIN, 0, 3);
6013  17  28:1   196        FOR MP04 := 0 TO 2 DO
6014  17  28:2   208          BEGIN
6015  17  28:3   208            IF BASE3F[ MP04] <> 0 THEN
6016  17  28:4   218              BEGIN
6017  17  28:5   218                PRPICCH( CHARSWIN, CHR( ORD( 'A') + MP04));
6018  17  28:5   225                PRPICCH( CHARSWIN, ' ');
6019  17  28:5   230                PRPICMB( CHARSWIN, BASE147[ MP04]);
6020  17  28:5   240                CHARSWIN^.HEAD.HCURSOR := 18;
6021  17  28:5   247                PRPICCH( CHARSWIN, '(');
6022  17  28:5   252                PRINTNUM( CHARSWIN, BASE3F[ MP04], 1);
6023  17  28:5   263                PRPICCH( CHARSWIN, ')');
6024  17  28:4   268              END;
6025  17  28:3   268            PRINTCR( CHARSWIN)
6026  17  28:2   269          END;
6027  17  28:2   279
6028  17  28:1   279        UNPROWIN( CHARSWIN, TRUE);
6029  17  28:1   284
6030  17  28:1   284        IF NOT MP6D THEN
6031  17  28:2   289          BEGIN
6032  17  28:2   289
6033  17  28:2   289            (* STATUS WORSE THAN 'ASLEEP' *)
```

```
6034  17  28:2   289
6035  17  28:3   289          BASE13 := 14;
6036  17  28:3   292          DRAWSCRN( FALSE);
6037  17  28:3   295          EXIT( RUNNER)
6038  17  28:2   299        END
6039  17  28:2   299
6040  17  28:0   299    END;  (* DSPPARTY *)
6041  17  28:0   314
6042  17  28:0   314
6043  17  30:D     1  PROCEDURE UPDATEHP;  (* P010B1E *)
6044  17  30:D     1
6045  17  30:D     1    VAR
6046  17  30:D     1         UNUSED1 : INTEGER;
6047  17  30:D     2         UNUSED2 : INTEGER;
6048  17  30:D     3
6049  17  30:0     0    BEGIN (* UPDATEHP *)
6050  17  30:1     0      IF (RAND MOD 8) = 1 THEN
6051  17  30:2    11        BEGIN
6052  17  30:3    11          WITH CHARACTR[ 6] DO
6053  17  30:4    19            BEGIN
6054  17  30:5    19              IF LOSTXYL.POISNAMT[ 1] > 0 THEN
6055  17  30:6    32                IF HPLEFT > 50 THEN
6056  17  30:7    39                  HPLEFT := HPLEFT - (HPLEFT DIV 10)
6057  17  30:6    50                ELSE
6058  17  30:7    54                  HPLEFT := HPLEFT - LOSTXYL.POISNAMT[ 1];
6059  17  30:5    71              HPLEFT := HPLEFT + HEALPTS;
6060  17  30:5    82              IF HPLEFT <= 0 THEN
6061  17  30:6    89                BEGIN
6062  17  30:7    89                  LOSTXYL.POISNAMT[ 1] := 0;
6063  17  30:7    99                  IF STATUS < DEAD THEN
6064  17  30:8   106                    BEGIN
6065  17  30:9   106                      HPLEFT := 0;
6066  17  30:9   111                      STATUS := DEAD;
6067  17  30:9   116                      DSPPARTY;
6068  17  30:9   118                      GETMSGTX( BASE47, 727);            (* ^ HAS DIED! *)
6069  17  30:9   126                      INSERTST( CHARACTR[ 6].NAME, BASE47);
6070  17  30:9   137                      SPECMSG( -1);    (* NOT SURE WHAT -1 DOES.  MSG # (?) *)
6071  17  30:9   141                      PAUSE2
6072  17  30:8   141                    END
6073  17  30:6   144                END
6074  17  30:5   144              ELSE
6075  17  30:6   146                IF HPLEFT > HPMAX THEN
6076  17  30:7   155                  HPLEFT := HPMAX
6077  17  30:7   158
6078  17  30:4   158            END;
6079  17  30:2   162        END;
6080  17  30:2   162
6081  17  30:0   162    END;  (* UPDATEHP *)
6082  17  30:0   176
6083  17  30:0   176
6084  17  31:D     1  PROCEDURE ENCOUNTR;    (* P010B1F *)
6085  17  31:D     1
6086  17  31:D     1    VAR
6087  17  31:D     1         ENCTYPE : INTEGER;
6088  17  31:D     2         ENEMYI  : INTEGER;
6089  17  31:D     3         ENCCALC : INTEGER;
6090  17  31:D     4
```

```
6091  17  31:D    4
6092  17  32:D    1    PROCEDURE P010B20;
6093  17  32:D    1
6094  17  32:0    0      BEGIN (* P010B20 *)
6095  17  32:1    0        ENCTYPE := ATTK012 + 1;
6096  17  32:1    7        BASE09 := 4;
6097  17  32:1   10
6098  17  32:1   10        REPEAT
6099  17  32:2   10          WITH MAZE.ENMYCALC[ ENCTYPE] DO
6100  17  32:3   23            BEGIN
6101  17  32:4   23              ENCCALC := 0;
6102  17  32:4   27              WHILE ((RAND MOD 100) < PERCWORS) AND (ENCCALC < WORSE01) DO
6103  17  32:5   46                ENCCALC := ENCCALC + 1;
6104  17  32:4   56              ENEMYI := MINENEMY + (RAND MOD RANGE0N) + (MULTWORS * ENCCALC);
6105  17  32:4   77              BASE27 := ENEMYI;
6106  17  32:4   82              BASE09 := BASE09 - 1
6107  17  32:3   83            END;
6108  17  32:1   87        UNTIL (BASE09 <= 0) OR (NOT BASE87B[ BASE27]);
6109  17  32:1  103
6110  17  32:1  103        IF BASE87B[ BASE27] THEN
6111  17  32:2  114          EXIT( P010B20);
6112  17  32:1  118        UNITCLEAR( 2);  (* SYSTERM *)
6113  17  32:1  121        SPECMSG( 702);  (* AN ENCOUNTER *)
6114  17  32:1  126        PAUSE2;
6115  17  32:1  129        MOVELEFT( MAZE.ENMYCALC, BASE75B, 3 * 10); (* SIZEOF( ENMYCALC) *)
6116  17  32:1  143        BASE13 := 6;
6117  17  32:1  146        DRAWSCRN( FALSE);
6118  17  32:1  149        EXIT( RUNNER)
6119  17  32:1  153
6120  17  32:0  153      END;  (* P010B20 *)
6121  17  32:0  170
6122  17  32:0  170
6123  17  31:0    0    BEGIN (* ENCOUNTR *)
6124  17  31:1    0      IF (MAZELEV = 1) OR (MAZELEV >= 12) THEN
6125  17  31:2    9        EXIT( ENCOUNTR)
6126  17  31:1   13      ELSE IF (MAZELEV = 5) THEN
6127  17  31:3   20        IF MAZE.FIGHTS[ MAZEX][ MAZEY] <> 1 THEN
6128  17  31:4   36          EXIT( ENCOUNTR);
6129  17  31:1   40      IF BASE7F4[ 17 - 1] <> 0 THEN
6130  17  31:2   53        EXIT( ENCOUNTR);
6131  17  31:1   57      BASE14 := 1;
6132  17  31:1   60      IF CHSTALRM = 1 THEN
6133  17  31:2   66        ATTK012 := 1
6134  17  31:1   66      ELSE
6135  17  31:2   71        BEGIN
6136  17  31:3   71          IF FIGHTMAP[ MAZEX][ MAZEY] THEN
6137  17  31:4   84            ATTK012 := 2
6138  17  31:3   84          ELSE
6139  17  31:4   89            BEGIN
6140  17  31:5   89              IF INITTURN AND (MAZE.FIGHTS[ MAZEX][ MAZEY] = 1) AND
6141  17  31:5  107                 (MAZE.FIGHTS[ SAVEX][ SAVEY] <> 1) THEN
6142  17  31:6  126                ATTK012 := 1
6143  17  31:5  126              ELSE
6144  17  31:6  131                IF (RAND MOD 10) <> 5 THEN
6145  17  31:7  142                  ATTK012 := 0
6146  17  31:6  142                ELSE
6147  17  31:7  147                  ATTK012 := 1
```

```
6148  17   31:4   147            END;
6149  17   31:2   150          END;
6150  17   31:2   150
6151  17   31:2   150       (* 5F9E *)
6152  17   31:1   150        WHILE ATTK012 >=0 DO
6153  17   31:2   156          BEGIN
6154  17   31:3   156            P010B20;
6155  17   31:3   158            ATTK012 := ATTK012 - 1
6156  17   31:2   160          END;
6157  17   31:1   166        CLROOMFG( MAZEX, MAZEY)
6158  17   31:1   168
6159  17   31:0   168      END;  (* ENCOUNTR *)
6160  17   31:0   184
6161  17   31:0   184
6162  17   33:D     1    PROCEDURE RUNMAIN;  (* P010B21 *)
6163  17   33:D     1
6164  17   33:D     1
6165  17   34:D     1      PROCEDURE P010B22( MP01 : INTEGER);
6166  17   34:D     2
6167  17   34:0     0        BEGIN (* P010B22 *)
6168  17   34:1     0          IF      MP01 =  0 THEN
6169  17   34:2     5            MP01 := 13
6170  17   34:1     5          ELSE IF MP01 = -1 THEN
6171  17   34:3    16            MP01 := 14
6172  17   34:2    16          ELSE IF MP01 = 15 THEN
6173  17   34:4    26            MP01 := 1;
6174  17   34:4    29
6175  17   34:1    29          MAZELEV := MP01;
6176  17   34:1    32          BASE13 := 7;
6177  17   34:1    35          IF MP01 = 5 THEN
6178  17   34:2    40            BEGIN
6179  17   34:3    40              READMAZ( 15);
6180  17   34:3    43              WRITEMAZ( 5)
6181  17   34:2    44            END;
6182  17   34:1    46          PROTWIN( CHARSWIN, TRUE);
6183  17   34:1    51          DRAWSCRN( FALSE);
6184  17   34:1    54
6185  17   34:1    54          EXIT( RUNNER)
6186  17   34:1    58
6187  17   34:0    58        END;  (* P010B22 *)
6188  17   34:0    70
6189  17   34:0    70
6190  17   35:D     1      PROCEDURE SPECSQAR;  (* P010B23 *)
6191  17   35:D     1
6192  17   35:D     1        VAR
6193  17   35:D     1             P23MP01 : INTEGER;
6194  17   35:D     2             P23MP02 : INTEGER;
6195  17   35:D     3             P23MP03 : INTEGER;
6196  17   35:D     4             MP04 : INTEGER;
6197  17   35:D     5             MP05 : INTEGER;
6198  17   35:D     6             MP06 : INTEGER;
6199  17   35:D     7             MP07 : 0..15;  (* SQREXTRA VALUE *)
6200  17   35:D     8
6201  17   35:D     8
6202  17   36:D     1        PROCEDURE SPINDIR;    (* P010B24 *)
6203  17   36:D     1
6204  17   36:D     1          VAR
```

```
6205  17  36:D    1          MP01 : INTEGER;
6206  17  36:D    2          MP02 : INTEGER;
6207  17  36:D    3          MP03 : INTEGER;
6208  17  36:D    4
6209  17  37:D    3      FUNCTION P010B25 : BOOLEAN;
6210  17  37:D    3
6211  17  37:0    0        BEGIN (* P010B25 *)
6212  17  37:1    0          CASE MP01 OF
6213  17  37:1    5            5:  P010B25 := MAZE.N[ MP03][ MP02] <> OPEN;
6214  17  37:1   27            6:  P010B25 := MAZE.E[ MP03][ MP02] <> OPEN;
6215  17  37:1   48            7:  P010B25 := MAZE.S[ MP03][ MP02] <> OPEN;
6216  17  37:1   69            8:  P010B25 := MAZE.W[ MP03][ MP02] <> OPEN;
6217  17  37:1   90          END;
6218  17  37:1  106
6219  17  37:0  106        END;  (* P010B25 *)
6220  17  37:0  118
6221  17  37:0  118
6222  17  36:0    0      BEGIN (* SPINDIR *)
6223  17  36:0    0
6224  17  36:1    0        CASE P23MP03 OF
6225  17  36:1    5
6226  17  36:1    5          0:
6227  17  36:2    5              BEGIN
6228  17  36:3    5                SPINCNT := P23MP02;
6229  17  36:3   12                BSPIN := TRUE
6230  17  36:2   12              END;
6231  17  36:2   18
6232  17  36:1   18          1:
6233  17  36:2   18              BEGIN
6234  17  36:3   18                SPINCNT := P23MP02;
6235  17  36:3   25                BSPIN := TRUE;
6236  17  36:3   29                SPINBLCK( MAZEX, MAZEY)
6237  17  36:2   31              END;
6238  17  36:2   35
6239  17  36:1   35          2, 3:
6240  17  36:2   35              BEGIN
6241  17  36:3   35                FOR MP03 := 0 TO 19 DO
6242  17  36:4   46                  FOR MP02 := 0 TO 19 DO
6243  17  36:5   57                    BEGIN
6244  17  36:6   57                      MP01 := MAZE.SQREXTRA[ MP03][ MP02];
6245  17  36:6   71                      IF MAZE.SQRETYPE[ MP01] = SPINNER THEN
6246  17  36:7   84                        IF P23MP03 <> 2 THEN
6247  17  36:8   91                          IF MAZE.AUX2[ MP01] <> 4 THEN
6248  17  36:9  103                            BEGIN
6249  17  36:0  103                              SPINCNT := MAZE.AUX1[ MP01];
6250  17  36:0  115                              BSPIN := TRUE;
6251  17  36:0  119                              SPINBLCK( MAZEX, MAZEY)
6252  17  36:9  121                            END;
6253  17  36:5  123                    END;
6254  17  36:2  137              END;
6255  17  36:1  139          4:
6256  17  36:2  139              BEGIN
6257  17  36:3  139                CASE P23MP02 OF
6258  17  36:3  144                  0, 1, 2:  DIRECTIO := (DIRECTIO + P23MP02 + 1) MOD 4;
6259  17  36:3  157                         3:  DIRECTIO := RAND MOD 4;
6260  17  36:3  168                END;
6261  17  36:2  184              END;
```

```
6262  17   36:2   186                                
6263  17   36:1   186            5, 6, 7, 8:
6264  17   36:2   186                BEGIN
6265  17   36:3   186                  MP01 := P23MP03;
6266  17   36:3   191                  MP03 := P23MP02;
6267  17   36:3   196                  MP02 := P23MP01;
6268  17   36:3   201                  WHILE P010B25 DO
6269  17   36:4   207                    BEGIN
6270  17   36:5   207                      BSPIN := TRUE;
6271  17   36:5   211                      SPINCNT := 1;
6272  17   36:5   216                      SPINBLCK( MP03, MP02)
6273  17   36:4   218                    END;
6274  17   36:2   222                END;
6275  17   36:1   224          END;
6276  17   36:1   250
6277  17   36:1   250          DRAWMAZE;
6278  17   36:1   252
6279  17   36:0   252        END;  (* SPINDIR *)
6280  17   36:0   274
6281  17   36:0   274
6282  17   38:D     1      PROCEDURE QUIETXFR;  (* P010B26 *)
6283  17   38:D     1
6284  17   38:0     0        BEGIN (* QUIETXFR *)
6285  17   38:1     0          SAVEX := MAZEX;
6286  17   38:1     3          SAVEY := MAZEY;
6287  17   38:1     6          SAVELEV := MAZELEV;
6288  17   38:1     9          MAZEX := P23MP03;
6289  17   38:1    14          MAZEY := P23MP02;
6290  17   38:1    19          IF MAZELEV <> P23MP01 THEN
6291  17   38:2    26            P010B22( P23MP01)
6292  17   38:0    29        END;  (* QUIETXFR *)
6293  17   38:0    44
6294  17   38:0    44
6295  17   39:D     1      PROCEDURE STAIRSYN;  (* P010B27 *)
6296  17   39:D     1
6297  17   39:D     1        VAR
6298  17   39:D     1            MP01 : INTEGER;
6299  17   39:D     2
6300  17   39:0     0        BEGIN (* STAIRSYN *)
6301  17   39:1     0          BASE10 := GETWIN( 9, 10, 22, 4, 15, TRUE);
6302  17   39:1    13          MP01 := MAZELEV;
6303  17   39:1    16          IF MP01 > 12 THEN
6304  17   39:2    21            MP01 := 13 - MP01;
6305  17   39:1    26          IF MP01 < P23MP01 THEN
6306  17   39:2    33            DISP1LIN( BASE10, 703)   (* STAIRS DOWN *)
6307  17   39:1    37          ELSE
6308  17   39:2    42            DISP1LIN( BASE10, 704);  (* STAIRS UP   *)
6309  17   39:2    49
6310  17   39:1    49          DISP1LIN( BASE10, 705);    (* TAKE THEM (Y/N)? *)
6311  17   39:1    56          GETMSGTX( BASE47, 706);    (* Y *)
6312  17   39:1    64          GETMSGTX( BASEC7, 707);    (* N *)
6313  17   39:1    73          REPEAT
6314  17   39:2    73            GETKEY
6315  17   39:1    73          UNTIL (INCHAR = BASE47[ 1]) OR (INCHAR = BASEC7[ 1]);
6316  17   39:1    92
6317  17   39:1    92          DELWIN( BASE10, TRUE);
6318  17   39:1    98          IF (INCHAR = BASE47[ 1]) THEN
```

```
6319  17  39:2  106            BEGIN
6320  17  39:3  106              IF (MAZELEV = 10) AND (P23MP01 = 9) THEN
6321  17  39:4  117                DIRECTIO := EAST;
6322  17  39:3  120              QUIETXFR
6323  17  39:2  120            END
6324  17  39:2  122
6325  17  39:0  122        END;  (* STAIRSYN *)
6326  17  39:0  136
6327  17  39:0  136
6328  17  40:D    1      PROCEDURE VERYDARK;  (* P010B28 *)
6329  17  40:D    1
6330  17  40:0    0        BEGIN (* VERYDARK *)
6331  17  40:1    0          IF P23MP02 = 0 THEN
6332  17  40:2    7            LIGHT := 0
6333  17  40:0    7        END;  (* VERYDARK *)
6334  17  40:0   22
6335  17  40:0   22
6336  17  41:D    1      PROCEDURE ROCKWATR;  (* P010B29 *)
6337  17  41:D    1
6338  17  41:0    0        BEGIN (* ROCKWATR *)
6339  17  41:1    0          CHARACTR[ 6].STATUS := LOST;
6340  17  41:1   10          SPECMSG( 709);  (* YOU ARE IN ROCK! *)
6341  17  41:1   15          DSPPARTY;
6342  17  41:1   17          MAZELEV := 0;
6343  17  41:1   20          BASE13 := 14;
6344  17  41:1   23          DRAWSCRN( FALSE);
6345  17  41:1   26          EXIT( RUNNER)
6346  17  41:0   30        END;  (* ROCKWATR *)
6347  17  41:0   42
6348  17  41:0   42
6349  17  42:D    1      PROCEDURE P010B2A;
6350  17  42:D    1
6351  17  42:D    1        VAR
6352  17  42:D    1            MP01 : INTEGER;
6353  17  42:D    2            MP02 : INTEGER;
6354  17  42:D    3            MP03 : INTEGER;
6355  17  42:D    4
6356  17  42:0    0        BEGIN (* P010B2A *)
6357  17  42:1    0          WITH CHARACTR[ 6] DO
6358  17  42:2    8            BEGIN
6359  17  42:3    8              IF STATUS < DEAD THEN
6360  17  42:4   15                BEGIN
6361  17  42:5   15                  IF P23MP02 >= 0 THEN
6362  17  42:6   22                    BEGIN
6363  17  42:7   22                      MP01 := P23MP01;
6364  17  42:7   27                      FOR MP02 := 1 TO ABS( P23MP03) DO
6365  17  42:8   41                        MP01 := MP01 + (RAND MOD ABS( P23MP02)) + 1;
6366  17  42:6   64                    END
6367  17  42:5   64                  ELSE
6368  17  42:6   66                    BEGIN
6369  17  42:7   66                      MP01 := P23MP03;
6370  17  42:7   71                      IF P23MP01 <> 0 THEN
6371  17  42:8   78                        MP01 := MP01 + (RAND MOD ABS( P23MP01))
6372  17  42:6   89                    END;
6373  17  42:5   92                  HPLEFT := HPLEFT - MP01;
6374  17  42:5  101                  IF HPLEFT <= 0 THEN
6375  17  42:6  108                    BEGIN
```

```
6376  17  42:7  108                    HPLEFT := 0;
6377  17  42:7  113                    STATUS := DEAD
6378  17  42:6  116                  END
6379  17  42:4  118                END;
6380  17  42:3  118              DSPPARTY
6381  17  42:2  118            END;
6382  17  42:2  120
6383  17  42:0  120          END;  (* P010B2A *)
6384  17  42:0  134
6385  17  42:0  134
6386  17  43:D    1        PROCEDURE CHENCOUN;  (* P010B2B *)
6387  17  43:D    1
6388  17  43:D    1          VAR
6389  17  43:D    1              MP01 : INTEGER;
6390  17  43:D    2              MP02 : INTEGER;
6391  17  43:D    3
6392  17  43:0    0          BEGIN (* CHENCOUN *)
6393  17  43:1    0            IF (P23MP01 = 0) OR (NOT FIGHTMAP[ MAZEX][ MAZEY]) THEN
6394  17  43:2   20              EXIT( CHENCOUN);
6395  17  43:1   24            IF (MAZELEV = 6) AND (BASE7F4[ 18 - 1] > 0) THEN
6396  17  43:2   41              BEGIN
6397  17  43:3   41                SPECMSG( 272);  (* THE FUMES REPEL THE INSECTS! *)
6398  17  43:3   46                BASE7F4[ 18 - 1] := BASE7F4[ 18 - 1] - 1;
6399  17  43:3   66                EXIT( CHENCOUN)
6400  17  43:2   70              END;
6401  17  43:1   70            BASE27 := P23MP03;
6402  17  43:1   75            IF P23MP02 > 1 THEN
6403  17  43:2   82              BASE27 := BASE27 + (RAND MOD P23MP02);
6404  17  43:1   96            BASE14 := 0;
6405  17  43:1   99            ATTK012 := 2;
6406  17  43:1  102            BASE13 := 6;
6407  17  43:1  105            IF BASE87B[ BASE27] THEN
6408  17  43:2  116              BEGIN
6409  17  43:3  116                CLROOMFG( MAZEX, MAZEY);
6410  17  43:3  120                EXIT( CHENCOUN)
6411  17  43:2  124              END;
6412  17  43:1  124            MAZE.SQRETYPE[ MP07] := CHUTE;
6413  17  43:1  136            CLROOMFG( MAZEX, MAZEY);
6414  17  43:1  140            MAZE.SQRETYPE[ MP07] := ENCOUNTE;
6415  17  43:1  152            IF MAZE.AUX0[ MP07] > 0 THEN
6416  17  43:2  166              BEGIN
6417  17  43:3  166                MAZE.AUX0[ MP07] := MAZE.AUX0[ MP07] - 1;
6418  17  43:3  188                IF MAZE.AUX0[ MP07] = 0 THEN
6419  17  43:4  202                  MAZE.SQRETYPE[ MP07] := NORMAL;
6420  17  43:3  214                CHKWRTMZ( TRUE)
6421  17  43:2  215              END;
6422  17  43:1  217            UNITCLEAR( 2);
6423  17  43:1  220            SPECMSG( 702);  (* AN ENCOUNTER *)
6424  17  43:1  225            PAUSE2;
6425  17  43:1  228            DRAWSCRN( FALSE);
6426  17  43:1  231
6427  17  43:1  231            EXIT( RUNNER)
6428  17  43:1  235
6429  17  43:0  235          END;  (* CHENCOUN *)
6430  17  43:0  248
6431  17  43:0  248
6432  17  44:D    1        PROCEDURE DOSCNMSG;  (* P010B2C *)
```

```
6433  17   44:D     1
6434  17   44:D     1
6435  17   45:D     1         PROCEDURE P010B2D;
6436  17   45:D     1
6437  17   45:D     1           VAR
6438  17   45:D     1                MP01 : INTEGER;
6439  17   45:D     2                MP02 : INTEGER;
6440  17   45:D     3
6441  17   45:0     0           BEGIN (* P010B2D *)
6442  17   45:1     0             IF NOT FIGHTMAP[ MAZEX][ MAZEY] THEN
6443  17   45:2    14               EXIT( DOSCNMSG);
6444  17   45:1    18             IF (MAZELEV = 6) AND (BASE7F4[ 18 - 1] > 0) THEN
6445  17   45:2    35               BEGIN
6446  17   45:3    35                 SPECMSG( 272);  (* THE FUMES REPEL THE INSECTS *)
6447  17   45:3    40                 BASE7F4[ 18 - 1] := BASE7F4[ 18 - 1] - 1;
6448  17   45:3    60                 EXIT( DOSCNMSG)
6449  17   45:2    64               END;
6450  17   45:1    64             IF BASE7F4[ 17 - 1] <> 0 THEN
6451  17   45:2    77               IF MAZELEV = 1 THEN
6452  17   45:3    82                 EXIT( DOSCNMSG);
6453  17   45:1    86             BASE27 := P23MP02;
6454  17   45:1    91             BASE14 := 0;
6455  17   45:1    94             ATTK012 := 2;
6456  17   45:1    97             IF P23MP01 = -2 THEN
6457  17   45:2   105               ATTK012 := 3;
6458  17   45:1   108             BASE13 := 6;
6459  17   45:1   111             BASE87B[ BASE27] := FALSE;
6460  17   45:1   121             MAZE.SQRETYPE[ MP07] := CHUTE;
6461  17   45:1   133             CLROOMFG( MAZEX, MAZEY);
6462  17   45:1   137             MAZE.SQRETYPE[ MP07] := SCNMSG;
6463  17   45:1   149             UNITCLEAR( 2);
6464  17   45:1   152             SPECMSG( 702);  (* AN ENCOUNTER *)
6465  17   45:1   157             PAUSE2;
6466  17   45:1   160             DRAWSCRN( FALSE);
6467  17   45:1   163
6468  17   45:1   163             EXIT( RUNNER)
6469  17   45:1   167
6470  17   45:0   167           END;  (* P010B2D *)
6471  17   45:0   180
6472  17   45:0   180
6473  17   44:0     0         BEGIN (* DOSCNMSG *)
6474  17   44:1     0           IF P23MP03 = 600 THEN
6475  17   44:2     9             P010B2D;
6476  17   44:1    11           BASE09 := MP07;
6477  17   44:1    16           BASE13 := 21;
6478  17   44:1    19           BASE2B := 5;
6479  17   44:1    22           DRAWSCRN( FALSE);
6480  17   44:1    25           EXIT( RUNNER)
6481  17   44:0    29         END;  (* DOSCNMSG *)
6482  17   44:0    42
6483  17   44:0    42
6484  17   46:D     1       PROCEDURE BUTTONS;  (* P010B2E *)
6485  17   46:D     1
6486  17   46:D     1         VAR
6487  17   46:D     1              MP01 : INTEGER;
6488  17   46:D     2              MP02 : INTEGER;
6489  17   46:D     3              MP03 : INTEGER;
```

```
6490  17  46:D    4            MP04 : INTEGER;
6491  17  46:D    5            MP05 : CHAR;
6492  17  46:D    6            MP06 : CHAR;
6493  17  46:D    7            MP07 : TLONGSTR;
6494  17  46:D  135            MP87 : STRING[ 19];
6495  17  46:D  145
6496  17  46:D  145
6497  17  46:0    0        BEGIN (* BUTTONS *)
6498  17  46:1    0          MP87 := 'CBA123456789MLBC';
6499  17  46:1   24          MP02 := P23MP03;
6500  17  46:1   29          MP01 := P23MP02;
6501  17  46:1   34          BASE10 := GETWIN( 0, 10, 40, 6, 15, TRUE);
6502  17  46:1   47          DISP1LIN( BASE10, 710);              (* THERE ARE TWO BUTTONS *)
6503  17  46:1   54          GETMSGTX( MP07, 711);   (* HERE MARKED ^ AND #. *)
6504  17  46:1   62          MP04 := POS( '^', MP07);
6505  17  46:1   72          MP06 := MP87[ MP02 + 2];
6506  17  46:1   81          MP07[ MP04] := MP06;
6507  17  46:1   86          MP04 := POS( '#', MP07);
6508  17  46:1   96          MP05 := MP87[ MP01 + 2];
6509  17  46:1  105          MP07[ MP04] := MP05;
6510  17  46:1  110          CENTSTR( BASE10, MP07);
6511  17  46:1  116          PRINTCR( BASE10);
6512  17  46:1  120          DISP1LIN( BASE10, 712);  (* PRESS ONE (~ TO LEAVE) *)
6513  17  46:1  127          DELWIN( BASE10, FALSE);
6514  17  46:1  133
6515  17  46:1  133          REPEAT
6516  17  46:2  133            GETKEY
6517  17  46:1  133          UNTIL (INCHAR = CHR( CRETURN)) OR
6518  17  46:1  139                (INCHAR = MP06) OR
6519  17  46:1  143                (INCHAR = MP05);
6520  17  46:1  149
6521  17  46:1  149          IF INCHAR = CHR( CRETURN) THEN
6522  17  46:2  154            BEGIN
6523  17  46:3  154              DRAWSCR2( WINCHAIN);
6524  17  46:3  158              EXIT( BUTTONS);
6525  17  46:2  162            END;
6526  17  46:1  162          IF P23MP01 > 0 THEN
6527  17  46:2  169            BEGIN
6528  17  46:3  169              MAZEX := RAND MOD 20;
6529  17  46:3  178              MAZEY := RAND MOD 20;
6530  17  46:3  187              SAVEX := MAZEX;
6531  17  46:3  190              SAVEY := MAZEY
6532  17  46:2  190            END;
6533  17  46:1  193          IF INCHAR = MP06 THEN
6534  17  46:2  198            P010B22( MP02)
6535  17  46:1  199          ELSE
6536  17  46:2  203            P010B22( MP01)
6537  17  46:2  204
6538  17  46:0  204        END;  (* BUTTONS *)
6539  17  46:0  220
6540  17  46:0  220
6541  17  47:D    1      PROCEDURE P010B2F;
6542  17  47:D    1
6543  17  47:D    1        VAR
6544  17  47:D    1            MP01 : INTEGER;
6545  17  47:D    2            MP02 : INTEGER;
6546  17  47:D    3            MP03 : INTEGER;
```

```
6547  17   47:D     4              MP04 : INTEGER;
6548  17   47:D     5
6549  17   47:0     0         BEGIN (* P010B2F *)
6550  17   47:1     0           MP04 := ABS( P23MP03);
6551  17   47:1     6
6552  17   47:1     6           REPEAT
6553  17   47:1     6
6554  17   47:2     6             GETMSGTX( BASE47, MP04);
6555  17   47:2    12             MP01 := 1;
6556  17   47:2    15             IF BASE47 = '**ERR**' THEN
6557  17   47:3    31               BEGIN
6558  17   47:4    31                 BASE47 := 'IND ERR';
6559  17   47:4    45                 SPECMSG( -1);
6560  17   47:4    49                 GETCR;
6561  17   47:4    52                 EXIT( P010B2F)
6562  17   47:3    56               END;
6563  17   47:2    56             MOVELEFT( BASE47[ 1], MP03, 2);
6564  17   47:2    65             MOVELEFT( BASE47[ 3], MP02, 2);
6565  17   47:2    74             IF (MAZEX = MP03) AND (MAZEY = MP02) THEN
6566  17   47:3    83               BEGIN
6567  17   47:4    83                 MOVELEFT( BASE47[ 5], P23MP03, 2);
6568  17   47:4    93                 MOVELEFT( BASE47[ 7], P23MP02, 2);
6569  17   47:4   103                 MOVELEFT( BASE47[ 9], P23MP01, 2);
6570  17   47:4   113                 EXIT( P010B2F)
6571  17   47:3   117               END;
6572  17   47:2   117             MP04 := MP04 + 1;
6573  17   47:2   122
6574  17   47:1   122           UNTIL FALSE
6575  17   47:1   122
6576  17   47:0   122         END;  (* P010B2F *)
6577  17   47:0   140
6578  17   47:0   140
6579  17   35:0     0       BEGIN (* SPECSQAR *)
6580  17   35:0     0
6581  17   35:1     0         REPEAT
6582  17   35:1     0
6583  17   35:2     0           MP07 := MAZE.SQREXTRA[ MAZEX][ MAZEY];
6584  17   35:2    14           P23MP03 := MAZE.AUX2[ MP07];
6585  17   35:2    24           P23MP02 := MAZE.AUX1[ MP07];
6586  17   35:2    34           P23MP01 := MAZE.AUX0[ MP07];
6587  17   35:2    44           IF P23MP03 < -10000 THEN
6588  17   35:3    52             P010B2F;
6589  17   35:2    54           MP06 := MAZEX;
6590  17   35:2    57           MP05 := MAZEY;
6591  17   35:2    60           MP04 := 0;
6592  17   35:2    63
6593  17   35:2    63           CASE MAZE.SQRETYPE[ MP07] OF
6594  17   35:2    74
6595  17   35:2    74               FIZZLE:  FIZZLES := 1;
6596  17   35:2    79             ROCKWATE:  ROCKWATR;
6597  17   35:2    83              BUTTONZ:  BUTTONS;
6598  17   35:2    87
6599  17   35:2    87               STAIRS:  IF INITTURN THEN
6600  17   35:4    92                          STAIRSYN;
6601  17   35:4    96
6602  17   35:2    96                  PIT:   IF INITTURN THEN   (* APIT *)
6603  17   35:4   101                           BEGIN
```

```
6604  17  35:5  101                    IF (MAZELEV > 12) OR (BASE7F4[ 1 - 1] = 0) THEN
6605  17  35:6  118                      BEGIN
6606  17  35:7  118                        IF P23MP02 < 0 THEN
6607  17  35:8  123                          BEGIN
6608  17  35:9  123                            IF MAZELEV > 12 THEN
6609  17  35:0  128                              MP04 := 13 - MAZELEV - P23MP02
6610  17  35:9  131                            ELSE
6611  17  35:0  137                              MP04 := MAZELEV - P23MP02;
6612  17  35:0  142
6613  17  35:9  142                            IF P23MP02 = -1 THEN
6614  17  35:0  148                              SPECMSG( 730)    (* YOU FELL DOWN A
6615  17  35:0  151                                                 WHOLE STORY! *)
6616  17  35:9  151                            ELSE
6617  17  35:0  155                              SPECMSG( 731);   (* YOU FELL DOWN 2
6618  17  35:0  160                                                 STORIES *)
6619  17  35:9  160                            PAUSE2;
6620  17  35:9  163                            P010B2A;
6621  17  35:9  165                            P010B22( MP04)
6622  17  35:8  166                          END;
6623  17  35:8  168                        (* 66EE *)
6624  17  35:8  168
6625  17  35:7  168                        IF MAZELEV = 9 THEN
6626  17  35:8  173                          SPECMSG( 732)   (* KABLAM! LAND MINE! *)
6627  17  35:7  176                        ELSE
6628  17  35:8  180                          SPECMSG( 723);  (* A PIT! *)
6629  17  35:8  185
6630  17  35:7  185                        PAUSE2;
6631  17  35:7  188                        P010B2A;
6632  17  35:6  190                      END
6633  17  35:5  190                    ELSE
6634  17  35:5  192
6635  17  35:5  192                      (* 6706 *)
6636  17  35:6  192                      BEGIN
6637  17  35:7  192                        IF MAZELEV = 9 THEN
6638  17  35:8  197                          MP04 := 733
6639  17  35:8  197                            (* YOU LEVITATE OVER THE MINEFIELD*)
6640  17  35:7  197                        ELSE
6641  17  35:8  204                          MP04 := 729; (* YOU LEVITATE OVER THE PIT! *)
6642  17  35:6  209                      END;
6643  17  35:6  209
6644  17  35:6  209                    (* 6717 *)
6645  17  35:4  209                  END;
6646  17  35:4  211
6647  17  35:2  211  OUCHY:  (* OUCH; *)  (* 6719 *)
6648  17  35:3  211                  BEGIN
6649  17  35:4  211                    IF MAZELEV = 8 THEN
6650  17  35:5  216                      BEGIN
6651  17  35:6  216                        IF BASE7F4[ 1 - 1] = 0 THEN
6652  17  35:7  229                          BEGIN
6653  17  35:8  229                            MP04 := 2509;
6654  17  35:8  234                              (* YOU FELL OFF THE ZIGGURAT! *)
6655  17  35:8  234                            MAZEY := MAZEY - 1;
6656  17  35:8  239                            P010B2A
6657  17  35:7  239                        END
6658  17  35:6  241                      ELSE
6659  17  35:7  243                          MP04 := 728;  (* YOU ARE FLYING! *)
6660  17  35:5  248                    END
```

```
6661  17   35:4   248                          ELSE
6662  17   35:5   250                            BEGIN
6663  17   35:6   250                              P010B2A;
6664  17   35:6   252                              SPECMSG( 713);  (* OUCH! *)
6665  17   35:6   257                              PAUSE2
6666  17   35:5   257                            END;
6667  17   35:3   260                        END;
6668  17   35:3   262
6669  17   35:2   262                CHUTE:  BEGIN
6670  17   35:4   262                          SPECMSG( 714);      (* A CHUTE *)
6671  17   35:4   267                          PAUSE2;
6672  17   35:4   270                          QUIETXFR
6673  17   35:3   270                        END;
6674  17   35:3   274
6675  17   35:2   274              SPINNER:  IF INITTURN THEN
6676  17   35:4   279                          SPINDIR;
6677  17   35:2   283            TRANSFER:  QUIETXFR;
6678  17   35:2   287                DARK:  VERYDARK;
6679  17   35:2   291               SCNMSG:  DOSCNMSG;
6680  17   35:2   295             ENCOUNTE:  CHENCOUN;
6681  17   35:2   299           END;
6682  17   35:2   330
6683  17   35:2   330           IF (MP06 <> MAZEX) OR (MP05 <> MAZEY) THEN
6684  17   35:3   339             DRAWMAZE;
6685  17   35:2   341           IF MP04 <> 0 THEN
6686  17   35:3   346             BEGIN
6687  17   35:4   346               SPECMSG( MP04);
6688  17   35:4   349               PAUSE2;
6689  17   35:4   352               MP04 := 0
6690  17   35:3   352             END;
6691  17   35:3   355
6692  17   35:1   355         UNTIL (MP06 = MAZEX) AND (MP05 = MAZEY)
6693  17   35:1   361
6694  17   35:0   361       END;  (* SPECSQAR *)
6695  17   35:0   382
6696  17   35:0   382
6697  17   35:0   382 (*$I WIZ4B:RUNNER2 *)
6697  17   35:0   382 (*$I WIZ4B:RUNNER3 *)
6698  17   35:0   382
6699  17   35:0   382  (* RUNNER3 *)
6700  17   35:0   382
6701  17   35:0   382
6702  17   48:D     1     PROCEDURE BUMPWALL;  (* P010B30 *)
6703  17   48:D     1
6704  17   48:0     0       BEGIN (* BUMPWALL *)
6705  17   48:1     0         SPECMSG( 715);  (* YOU CAN''T WALK THROUGH WALLS! *)
6706  17   48:1     5         UNITCLEAR( 2);  (* 2 = SYSTERM;  1 = CONSOLE *)
6707  17   48:1     8         BEEPSPKR( 2)
6708  17   48:0     9       END;  (* BUMPWALL *)
6709  17   48:0    24
6710  17   48:0    24
6711  17   49:D     1     PROCEDURE DSPARROW( ARROWCH : CHAR);  (* P010B31 *)
6712  17   49:D     2
6713  17   49:D     2
6714  17   49:0     0       BEGIN (* DSPARROW *)
6715  17   49:1     0         MVCURSOR( MAINWIN, 17, 11);
6716  17   49:1     7         PRPICCH(  MAINWIN, ARROWCH);
```

```
6717  17   49:1    13          PRPICCH(  MAINWIN, ARROWCH);
6718  17   49:1    19          MVCURSOR( MAINWIN, 17, 12);
6719  17   49:1    26          PRPICCH(  MAINWIN, ARROWCH);
6720  17   49:1    32          PRPICCH(  MAINWIN, ARROWCH)
6721  17   49:0    35        END;  (* DSPARROW *)
6722  17   49:0    50
6723  17   49:0    50
6724  17   50:D     1      PROCEDURE P010B32;
6725  17   50:D     1
6726  17   50:0     0        BEGIN (* P010B32 *)
6727  17   50:1     0          IF BASE7F4[ 6 - 1] <= 0 THEN
6728  17   50:2    13            EXIT( P010B32);
6729  17   50:1    17          BASE7F4[ 6 - 1] := BASE7F4[ 6 - 1] - 1;
6730  17   50:1    37          SPECMSG( 380 + BASE7F4[ 6 - 1]);
6731  17   50:1    52          IF BASE7F4[ 6 - 1] = 0 THEN
6732  17   50:2    65            BEGIN
6733  17   50:2    65
6734  17   50:2    65              (* *** KABALAAMMMMOOOO *** *)
6735  17   50:2    65
6736  17   50:3    65              IF (BASE7F4[ 12 - 1] = 2) AND
6737  17   50:3    76                 (BASE7F4[ 10 - 1] = 15) AND
6738  17   50:3    88                 (BASE7F4[ 11 - 1] = 15) THEN
6739  17   50:4   102                BEGIN
6740  17   50:5   102                  BASE80B := 1;
6741  17   50:5   106                  MAZE.N[ 15, 15] := OPEN;
6742  17   50:5   119                  MAZE.S[ 15, 16] := OPEN;
6743  17   50:5   131                  PAUSE2;
6744  17   50:5   134                  DRAWMAZE;
6745  17   50:5   136
6746  17   50:4   136                END;
6747  17   50:4   136
6748  17   50:3   136              IF (MAZEX = BASE7F4[   10 - 1]) AND
6749  17   50:3   147                 (MAZEY = BASE7F4[   11 - 1]) AND
6750  17   50:3   159                 (MAZELEV = BASE7F4[ 12 - 1]) THEN
6751  17   50:4   173                BEGIN
6752  17   50:5   173                  CHARACTR[ 6].HPLEFT := 0;
6753  17   50:5   183                  CHARACTR[ 6].STATUS := DEAD;
6754  17   50:5   193                  DSPPARTY
6755  17   50:4   193                END
6756  17   50:4   195
6757  17   50:2   195            END;
6758  17   50:2   195
6759  17   50:0   195        END;  (* P010B32 *)
6760  17   50:0   210
6761  17   50:0   210
6762  17   51:D     1      PROCEDURE FORWRD( ARROWCH : CHAR);  (* P010B33 *)
6763  17   51:D     2
6764  17   51:D     2        VAR
6765  17   51:D     2            MP02 : INTEGER;
6766  17   51:D     3            MP03 : INTEGER;
6767  17   51:D     4            MP04 : INTEGER;
6768  17   51:D     5            MP05 : TWALL;
6769  17   51:D     6
6770  17   51:D     6
6771  17   52:D     1        PROCEDURE MOVEFRWD;  (* P010B34 *)
6772  17   52:D     1
6773  17   52:D     1          VAR
```

```
6774  17  52:D    1              SQREXTRA: 0..15;
6775  17  52:D    2
6776  17  52:D    2
6777  17  53:D    1          PROCEDURE MVORACLE;  (* P010B35 *)
6778  17  53:D    1
6779  17  53:D    1            VAR
6780  17  53:D    1                ORCLYDIR : INTEGER;
6781  17  53:D    2                ORCLXDIR : INTEGER;
6782  17  53:D    3                UNUSED   : INTEGER;
6783  17  53:D    4                NEWDIST  : INTEGER;
6784  17  53:D    5                STEPS    : INTEGER;
6785  17  53:D    6                DST2ORAC : INTEGER;
6786  17  53:D    7                ORACLDIR : INTEGER;
6787  17  53:D    8                BIZARRE  : BOOLEAN;
6788  17  53:D    9                WALLTHNG : TWALL;
6789  17  53:D   10
6790  17  53:D   10
6791  17  54:D    3            FUNCTION DISTCALC( XDIR : INTEGER;  (* P010B36 *)
6792  17  54:D    4                               YDIR : INTEGER) : INTEGER;
6793  17  54:D    5
6794  17  54:0    0              BEGIN (* DISTCALC *)
6795  17  54:1    0                ORCLXDIR := XDIR;
6796  17  54:1    4                ORCLYDIR := YDIR;
6797  17  54:1    8
6798  17  54:1    8                (* DOES NOT ACCOUNT FOR MAZE WRAP-AROUND *)
6799  17  54:1    8
6800  17  54:1    8                DISTCALC := ABS( ORACLEX + XDIR - MAZEX) +
6801  17  54:1   17                            ABS( ORACLEY + YDIR - MAZEY)
6802  17  54:0   26              END;  (* DISTCALC *)
6803  17  54:0   42
6804  17  54:0   42
6805  17  53:0    0            BEGIN (* MVORACLE *)
6806  17  53:1    0              IF (RAND MOD 100) < 10 THEN
6807  17  53:2   11                EXIT( MVORACLE);
6808  17  53:1   15              ORACLDIR := RAND MOD 4;
6809  17  53:1   24              DST2ORAC := DISTCALC( 0, 0);  (* ORIGINAL DISTANCE APART *)
6810  17  53:1   32
6811  17  53:1   32              (* BIZARRE CODE: *)
6812  17  53:1   32
6813  17  53:1   32              FOR BIZARRE := FALSE TO TRUE DO
6814  17  53:2   47                BEGIN
6815  17  53:3   47                  FOR STEPS := 0 TO 3 DO
6816  17  53:4   58                    BEGIN
6817  17  53:5   58                      CASE ORACLDIR OF
6818  17  53:5   61                        NORTH:  NEWDIST := DISTCALC(  0,  1);
6819  17  53:5   71                         EAST:  NEWDIST := DISTCALC(  1,  0);
6820  17  53:5   81                        SOUTH:  NEWDIST := DISTCALC(  0, -1);
6821  17  53:5   92                         WEST:  NEWDIST := DISTCALC( -1,  0);
6822  17  53:5  103                      END;
6823  17  53:5  118
6824  17  53:5  118                      IF NEWDIST > DST2ORAC THEN
6825  17  53:6  123                        BEGIN
6826  17  53:7  123                          CASE ORACLDIR OF
6827  17  53:7  126                            NORTH:  WALLTHNG := MAZE.N[ ORACLEX][ ORACLEY];
6828  17  53:7  148                             EAST:  WALLTHNG := MAZE.E[ ORACLEX][ ORACLEY];
6829  17  53:7  169                            SOUTH:  WALLTHNG := MAZE.S[ ORACLEX][ ORACLEY];
6830  17  53:7  190                             WEST:  WALLTHNG := MAZE.W[ ORACLEX][ ORACLEY];
```

```
6831  17   53:7   211                        END;
6832  17   53:7   226
6833  17   53:7   226                      IF (WALLTHNG = OPEN) OR
6834  17   53:7   229                           ((BIZARRE AND (WALLTHNG <> WALL)) AND
6835  17   53:7   234                            ((RAND MOD 100) < 75)) THEN
6836  17   53:8   247                        BEGIN
6837  17   53:9   247                          ORACLEX := (ORACLEX + ORCLXDIR + 20) MOD 20;
6838  17   53:9   261                          ORACLEY := (ORACLEY + ORCLYDIR + 20) MOD 20;
6839  17   53:9   275                          EXIT( MVORACLE)
6840  17   53:8   279                        END
6841  17   53:6   279                    END;
6842  17   53:6   279
6843  17   53:5   279                  ORACLDIR := (ORACLDIR + 1) MOD 4;
6844  17   53:5   286
6845  17   53:4   286                END;
6846  17   53:2   293             END;
6847  17   53:2   300
6848  17   53:0   300          END;  (* MVORACLE *)
6849  17   53:0   322
6850  17   53:0   322
6851  17   52:0     0        BEGIN (* MOVEFRWD *)
6852  17   52:1     0          SQREXTRA :=  MAZE.SQREXTRA[ MAZEX][ MAZEY];
6853  17   52:1    14          IF MAZE.SQRETYPE[ SQREXTRA] = DARK THEN
6854  17   52:2    27            IF MAZE.AUX1[ SQREXTRA] = 2 THEN
6855  17   52:3    39              IF (MP05 = DOOR) OR (MP05 = HIDEDOOR) THEN
6856  17   52:4    52                EXIT (MOVEFRWD);
6857  17   52:1    56          IF BMVORACL THEN
6858  17   52:2    62            MVORACLE;
6859  17   52:1    64          NEEDDRMZ := TRUE;
6860  17   52:1    68          INITTURN := TRUE;
6861  17   52:1    72          SAVEX   := MAZEX;
6862  17   52:1    75          SAVEY   := MAZEY;
6863  17   52:1    78          SAVELEV := MAZELEV;
6864  17   52:1    81
6865  17   52:1    81          CASE DIRECTIO OF
6866  17   52:1    84            NORTH:  MAZEY := MAZEY + 1;
6867  17   52:1    91             EAST:  MAZEX := MAZEX + 1;
6868  17   52:1    98            SOUTH:  MAZEY := MAZEY - 1;
6869  17   52:1   105             WEST:  MAZEX := MAZEX - 1;
6870  17   52:1   112          END;
6871  17   52:1   128
6872  17   52:1   128          MAZEY := (MAZEY + 20) MOD 20;
6873  17   52:1   135          MAZEX := (MAZEX + 20) MOD 20;
6874  17   52:1   142
6875  17   52:1   142          BEEPSPKR( 1)
6876  17   52:1   143
6877  17   52:0   143        END;  (* MOVEFRWD *)
6878  17   52:0   158
6879  17   52:0   158
6880  17   51:0     0      BEGIN (* FORWRD *)
6881  17   51:1     0        MP03 := MAZEX;
6882  17   51:1     3        MP02 := MAZEY;
6883  17   51:1     6        CASE DIRECTIO OF
6884  17   51:1     9          NORTH:  MP05 := MAZE.N[ MAZEX][ MAZEY];
6885  17   51:1    25           EAST:  MP05 := MAZE.E[ MAZEX][ MAZEY];
6886  17   51:1    40          SOUTH:  MP05 := MAZE.S[ MAZEX][ MAZEY];
6887  17   51:1    55           WEST:  MP05 := MAZE.W[ MAZEX][ MAZEY];
```

```
6888  17   51:1    70          END;
6889  17   51:1    86          MP04 := MAZE.SQREXTRA[ MAZEX][ MAZEY];
6890  17   51:1   100          IF (MP05 = OPEN) OR (MP05 = DOOR) OR
6891  17   51:1   107             (MP05 = HIDEDOOR) AND (LIGHT > 0) THEN
6892  17   51:2   118            MOVEFRWD
6893  17   51:1   118          ELSE
6894  17   51:2   122            IF MP05 = HIDEDOOR THEN
6895  17   51:3   127              IF MAZE.SQRETYPE[ MP04] = DARK THEN
6896  17   51:4   140                MOVEFRWD;
6897  17   51:1   142          IF NOT INITTURN THEN
6898  17   51:2   148            BUMPWALL
6899  17   51:1   148          ELSE
6900  17   51:2   152            BEGIN
6901  17   51:3   152              SPINBLCK( MP03, MP02);
6902  17   51:3   156              DSPARROW( ARROWCH)
6903  17   51:2   157            END
6904  17   51:2   159
6905  17   51:0   159        END;  (* FORWRD *)
6906  17   51:0   172
6907  17   51:0   172
6908  17   55:D     1      PROCEDURE DOTURN( MP02 : INTEGER;  (* P010B37 *)
6909  17   55:D     2                        ARROWCH : CHAR);
6910  17   55:D     3
6911  17   55:0     0        BEGIN (* DOTURN *)
6912  17   55:1     0          NEEDDRMZ := TRUE;
6913  17   55:1     4          DIRECTIO := (DIRECTIO + MP02) MOD 4;
6914  17   55:1    11          DSPARROW( ARROWCH)
6915  17   55:1    12
6916  17   55:0    12        END;  (* DOTURN *)
6917  17   55:0    26
6918  17   55:0    26
6919  17   56:D     1      PROCEDURE SETTIME;  (* P010B38 *)
6920  17   56:D     1
6921  17   56:D     1        VAR
6922  17   56:D     1            TIMEVAL : INTEGER;
6923  17   56:D     2            TIMESTR : STRING;
6924  17   56:D    43
6925  17   56:D    43
6926  17   57:D     1        PROCEDURE EXITTIME;  (* P010B39 *)
6927  17   57:D     1
6928  17   57:0     0          BEGIN (* EXITTIME *)
6929  17   57:1     0            DELWIN( BASE10, TRUE);
6930  17   57:1     6            EXIT( SETTIME)
6931  17   57:0    10          END;  (* EXITTIME *)
6932  17   57:0    22
6933  17   57:0    22
6934  17   56:0     0        BEGIN (* SETTIME *)
6935  17   56:1     0          UNITCLEAR( 1);
6936  17   56:1     3          BASE10 := GETWIN( 0, 10, 40, 3, 15, TRUE);
6937  17   56:1    16          DISP1MSG( BASE10, 716);              (* DELAY (0-99) > *)
6938  17   56:1    23          SOLICIT( BASE10, TIMESTR, 2);
6939  17   56:1    30          TIMEVAL := 0;
6940  17   56:1    33          FOR BASE09 := 1 TO LENGTH( TIMESTR) DO
6941  17   56:2    48            BEGIN
6942  17   56:3    48              IF (TIMESTR[ BASE09] >= '0') AND (TIMESTR[ BASE09] <= '9') THEN
6943  17   56:4    63                TIMEVAL := (10 * TIMEVAL) + ORD( TIMESTR[ BASE09]) - ORD( '0')
6944  17   56:3    72              ELSE
```

```
6945  17   56:4    77              EXITTIME
6946  17   56:2    77          END;
6947  17   56:2    86
6948  17   56:1    86        IF (TIMEVAL > 0) AND (TIMEVAL <= 99) THEN
6949  17   56:2    95          TIMEDLAY := TIMEVAL;
6950  17   56:2    98
6951  17   56:1    98        EXITTIME
6952  17   56:1    98
6953  17   56:0    98      END;  (* SETTIME *)
6954  17   56:0   114
6955  17   56:0   114
6956  17   58:D     1    PROCEDURE RUNINIT;  (* P010B3A *)
6957  17   58:D     1
6958  17   58:0     0      BEGIN (* RUNINIT *)
6959  17   58:1     0        CSIOWIN := GETWIN( 0, 0, 40, 3, 10, FALSE);
6960  17   58:1    14        LPIWIN := GETWIN( 14, 2, 12, 5, 11, FALSE);
6961  17   58:1    28        PROTWIN( LPIWIN, FALSE);
6962  17   58:1    35        IF INFOWOFF.B THEN
6963  17   58:2    39          PROTWIN( CSIOWIN, TRUE)
6964  17   58:1    43        ELSE
6965  17   58:2    48          UNPROWIN( CSIOWIN, TRUE);
6966  17   58:1    55        DISP1MSG( CSIOWIN, 717);    (* C)AMP S)TATUS T)IME O)FF A-W-D K *)
6967  17   58:1    64        DRAWMAZE;
6968  17   58:1    66        NEEDDRAW := INFOWOFF.B;
6969  17   58:1    71        NEEDDRMZ := FALSE;
6970  17   58:1    75        INITTURN := TRUE
6971  17   58:1    75
6972  17   58:0    75      END;  (* RUNINIT *)
6973  17   58:0    92
6974  17   58:0    92
6975  17   59:D     1    PROCEDURE DISPLPI( PROPERTY : INTEGER;  (* P010B3B *)
6976  17   59:D     2                       MSG      : INTEGER);
6977  17   59:D     3
6978  17   59:D     3      (* MSG = 719, 720, 721;  LIGHT PROTECT IDENTIFY *)
6979  17   59:D     3
6980  17   59:0     0      BEGIN (* DISPLPI *)
6981  17   59:1     0        IF PROPERTY > 0 THEN
6982  17   59:2     5          DISP1LIN( LPIWIN, MSG)
6983  17   59:1     9        ELSE
6984  17   59:2    14          DISP1LIN( LPIWIN, 718)   (* "         " *)
6985  17   59:0    20      END;  (* DISPLPI *)
6986  17   59:0    36
6987  17   59:0    36
6988  17   60:D     1    PROCEDURE QUITEXPD;  (* P010B3C *)
6989  17   60:D     1
6990  17   60:0     0      BEGIN (* QUITEXPD *)
6991  17   60:1     0        BASE10 := GETWIN( 4, 10, 32, 3, 31, TRUE);
6992  17   60:1    13        DISP1LIN( BASE10, 724);  (* QUIT THIS EXPEDITION (Y/N)? *)
6993  17   60:1    20        GETMSGTX( BASE47, 725);  (* Y *)
6994  17   60:1    28        GETMSGTX( BASEC7, 726);  (* N *)
6995  17   60:1    37
6996  17   60:1    37        REPEAT
6997  17   60:2    37          GETKEY;
6998  17   60:2    40          IF (INCHAR = CHR( CRETURN) ) OR (INCHAR = BASEC7[ 1]) THEN
6999  17   60:3    53            BEGIN
7000  17   60:4    53              DELWIN( BASE10, TRUE);
7001  17   60:4    59              EXIT( QUITEXPD)
```

```
7002  17   60:3    63            END;
7003  17   60:1    63        UNTIL INCHAR = BASE47[ 1];
7004  17   60:1    71
7005  17   60:1    71        DELWIN( CSIOWIN, FALSE);
7006  17   60:1    78        DELWIN( LPIWIN, FALSE);
7007  17   60:1    85        DELWIN( BASE10, TRUE);
7008  17   60:1    91        WITH CHARACTR[ 6] DO
7009  17   60:2    99          BEGIN
7010  17   60:3    99            INMAZE := FALSE;
7011  17   60:3   104            LOSTXYL.LOCATION[ 2] := 100 * MAZEX + MAZEY;
7012  17   60:3   118            LOSTXYL.LOCATION[ 3] := MAZELEV;
7013  17   60:3   128            BASE13 := 8;
7014  17   60:3   131            CHKWRTMZ( FALSE);
7015  17   60:3   134            READMAZ( MAZELEV);
7016  17   60:3   137            EXIT( RUNNER)
7017  17   60:2   141          END;
7018  17   60:2   141
7019  17   60:0   141      END;  (* QUITEXPD *)
7020  17   60:0   156
7021  17   60:0   156
7022  17   60:0   156
7023  17   33:0     0    BEGIN (* RUNMAIN *)
7024  17   33:1     0      RUNINIT;
7025  17   33:1     2
7026  17   33:1     2      REPEAT
7027  17   33:2     2        IF NOT INFOWOFF.B AND
7028  17   33:2     5           ((LIGHT <> 0) OR (ACMOD2 <> 0) OR (IDMONSTR <> 0)) THEN
7029  17   33:3    22          BEGIN
7030  17   33:4    22            IF LPIWIN^.HEAD.PRIORITY > 127 THEN
7031  17   33:5    34              UNPROWIN( LPIWIN, TRUE);
7032  17   33:4    41            MVCURSOR( LPIWIN, 0, 0);
7033  17   33:4    49            DISPLPI( LIGHT,    719);  (* LIGHT    *)
7034  17   33:4    56            DISPLPI( ACMOD2,   720);  (* PROTECT  *)
7035  17   33:4    63            DISPLPI( IDMONSTR, 721)   (* IDENTIFY *)
7036  17   33:3    68          END
7037  17   33:2    70        ELSE
7038  17   33:3    72          IF LPIWIN^.HEAD.PRIORITY < 128 THEN
7039  17   33:4    86            PROTWIN( LPIWIN, TRUE);
7040  17   33:4    93
7041  17   33:2    93        IF NEEDDRMZ THEN
7042  17   33:3    98          DRAWMAZE;
7043  17   33:3   100
7044  17   33:2   100        IF BMVTREB THEN
7045  17   33:3   106          MVTREBOR;
7046  17   33:3   108
7047  17   33:2   108        BASE09 := MAZE.SQREXTRA[ MAZEX][ MAZEY];
7048  17   33:2   122
7049  17   33:2   122        IF MAZE.SQRETYPE[ BASE09] <> NORMAL THEN
7050  17   33:3   135          IF INITTURN THEN
7051  17   33:4   140            IF BASE2B <> 21 THEN
7052  17   33:5   146              SPECSQAR
7053  17   33:4   146            ELSE IF MAZE.SQRETYPE[ BASE09] = STAIRS THEN
7054  17   33:6   163              SPECSQAR
7055  17   33:5   163            ELSE IF MAZE.SQRETYPE[ BASE09] = BUTTONZ THEN
7056  17   33:7   180              SPECSQAR;
7057  17   33:7   182
7058  17   33:2   182        IF MAZEX = ORACLEX THEN
```

```
7059  17   33:3   190           IF MAZEY = ORACLEY THEN
7060  17   33:4   198             BEGIN
7061  17   33:5   198               BASE13 := 15;
7062  17   33:5   201               BASE2B := 5;
7063  17   33:5   204               DRAWSCRN( FALSE);
7064  17   33:5   207               BASE09 := -1;
7065  17   33:5   211               EXIT( RUNNER)
7066  17   33:4   215             END;
7067  17   33:4   215
7068  17   33:2   215         IF CHARSWIN^.HEAD.PRIORITY < 128 THEN
7069  17   33:3   227           IF NOT BASE18.B THEN
7070  17   33:4   232             PROTWIN( CHARSWIN, FALSE);
7071  17   33:2   237         BASE2B := 5;
7072  17   33:2   240         NEEDDRMZ := FALSE;
7073  17   33:2   244         IF (MAZELEV < 12) OR (MAZELEV > 14) THEN
7074  17   33:3   253           BEGIN
7075  17   33:4   253             IF
7076  17   33:4   253               (RAND MOD 99 = 35)                  OR
7077  17   33:4   262               ((RAND MOD 50 = 35) AND (MAZELEV = 11)) OR
7078  17   33:4   276               (CHSTALRM = 1)                        OR
7079  17   33:4   281               FIGHTMAP[ MAZEX][ MAZEY]
7080  17   33:4   291
7081  17   33:4   291                OR
7082  17   33:4   293
7083  17   33:4   293               (INITTURN AND
7084  17   33:4   296                 ((COMMAND = 23) OR ((COMMAND >= 1) AND (COMMAND <= 4))) AND
7085  17   33:4   317                 (MAZE.FIGHTS[ MAZEX][ MAZEY] = 1) AND
7086  17   33:4   332                 (RAND MOD 8 = 3))
7087  17   33:4   342
7088  17   33:4   342                 (* 23 = ???; 1..4 = FORWARD *)
7089  17   33:4   342                 (* THERE ARE ONLY 22 COMMANDS *)
7090  17   33:4   342
7091  17   33:4   342             THEN
7092  17   33:4   345
7093  17   33:5   345               ENCOUNTR
7094  17   33:5   345
7095  17   33:3   345           END;
7096  17   33:3   347
7097  17   33:3   347         (* 6EAF *)
7098  17   33:3   347
7099  17   33:2   347         IF INITTURN THEN
7100  17   33:3   352           UPDATEHP;
7101  17   33:2   354         INITTURN := FALSE;
7102  17   33:2   358         P010B32;
7103  17   33:2   360         GETCMD;
7104  17   33:2   362         IF NEEDDRAW THEN
7105  17   33:3   367           BEGIN
7106  17   33:4   367             DRAWSCR2( WINCHAIN);
7107  17   33:4   371             NEEDDRAW := FALSE
7108  17   33:3   371           END;
7109  17   33:3   375                                              (* WK 8AL 4DR 6XB 2OSTCIQ *)
7110  17   33:2   375         CASE COMMAND OF                      (* 123456789012345678901 2 *)
7111  17   33:2   381           1, 2, 3, 4, 23: FORWRD(    CHR( 13)); (* WK 8                Q *)
7112  17   33:2   386               5, 6, 7, 8: DOTURN( 3, CHR( 12)); (*     AL 4              *)
7113  17   33:2   392            9, 10, 11, 12: DOTURN( 1, CHR( 14)); (*         DR 6          *)
7114  17   33:2   398           13, 14, 15, 16: DOTURN( 2, CHR( 15)); (*             XB 2      *)
7115  17   33:2   404
```

```
7116  17   33:2   404                          (* 13 == FORWARD ARROW         3 == $0B == CTRL-K *)
7117  17   33:2   404                          (* 12 == LEFT    ARROW         7 == $09 == CTRL-I *)
7118  17   33:2   404                          (* 14 == RIGHT   ARROW        11 == $0C == CTRL-L *)
7119  17   33:2   404                          (* 15 == BACK    ARROW        15 == $0A == CTRL-J *)
7120  17   33:2   404
7121  17   33:2   404
7122  17   33:2   404          17: (* 'O' OFF/ON ;DISPLAY COMMAND LINE AT TOP AND
7123  17   33:2   404                              LIGHT/PROTECTION/IDENTIFY (?) AND
7124  17   33:2   404                              CHARACTER STATUS AT BOTTOM     *)
7125  17   33:3   404              BEGIN
7126  17   33:4   404                IF INFOWOFF.B THEN
7127  17   33:5   408                  BEGIN
7128  17   33:6   408                    UNPROWIN( CSIOWIN, FALSE);
7129  17   33:6   415                    IF (ABS( LIGHT) + ABS( ACMOD2) + ABS( IDMONSTR)) > 0 THEN
7130  17   33:7   430                      UNPROWIN( LPIWIN, FALSE);
7131  17   33:6   437                    DSPPARTY;
7132  17   33:6   439                    INFOWOFF.B := FALSE;
7133  17   33:6   442                    BASE18.B := TRUE
7134  17   33:5   442                  END
7135  17   33:4   445                ELSE
7136  17   33:5   447                  BEGIN
7137  17   33:6   447                    PROTWIN( CSIOWIN, FALSE);
7138  17   33:6   454                    PROTWIN( LPIWIN, FALSE);
7139  17   33:6   461                    PROTWIN( CHARSWIN, TRUE);
7140  17   33:6   466                    INFOWOFF.B := TRUE;
7141  17   33:6   469                    BASE18.B := FALSE
7142  17   33:5   469                  END
7143  17   33:3   472              END;
7144  17   33:3   474
7145  17   33:2   474          18: (* 'S' STATUS; TOGGLE DISPLAY OF CHARACTERS AT BOTTOM *)
7146  17   33:3   474              BEGIN
7147  17   33:4   474                IF CHARSWIN^.HEAD.PRIORITY > 127 THEN
7148  17   33:5   484                  DSPPARTY
7149  17   33:4   484                ELSE
7150  17   33:5   488                  PROTWIN( CHARSWIN, TRUE);
7151  17   33:4   493                BASE18.B := CHARSWIN^.HEAD.PRIORITY < 128
7152  17   33:3   499              END;
7153  17   33:3   507
7154  17   33:2   507          19: (* 'T' TIME *)
7155  17   33:3   507              SETTIME;
7156  17   33:3   511
7157  17   33:2   511          20: (* 'C' CAMP *)
7158  17   33:3   511              BEGIN
7159  17   33:4   511                DRAWSCRN( FALSE);
7160  17   33:4   514                BASE13 := 10;
7161  17   33:4   517                EXIT( RUNNER)
7162  17   33:3   521              END;
7163  17   33:3   523
7164  17   33:3   523        (* 21:    'I' INSPECT;  IS IGNORED *)
7165  17   33:3   523
7166  17   33:2   523          22: (* 'Q' QUIT;  QUIT THE EXPEDITION *)
7167  17   33:3   523              QUITEXPD;
7168  17   33:3   527
7169  17   33:2   527        END;
7170  17   33:2   580
7171  17   33:1   580      UNTIL FALSE;
7172  17   33:1   583
```

```
7173  17   33:0   583    END;  (* RUNMAIN *)
7174  17   33:0   602
7175  17   33:0   602
7176  17   61:D     1  PROCEDURE RDMZINFO( VAR MP01 : TMAZEDAT);  (* P010B3D *)
7177  17   61:D     2
7178  17   61:D     2    (* READ MAZE.INFO *)
7179  17   61:D     2
7180  17   61:D     2    VAR
7181  17   61:D     2         MP02 : PACKED ARRAY[ 0..0] OF INTEGER;
7182  17   61:D     3
7183  17   61:0     0    BEGIN (* RDMZINFO *)
7184  17   61:1     0      MAZDAT00 := MP02[ -1];
7185  17   61:1    11      UNITREAD( DRIVE1, MAZEBLK0, 248,  MAZEBLCK);
7186  17   61:1    26      UNITREAD( DRIVE1, MAZEDATA, 3004, MAZEBLCK + 1)
7187  17   61:0    43    END;  (* RDMZINFO *)
7188  17   61:0    56
7189  17   61:0    56
7190  17    1:0     0    BEGIN (* RUNNER MAIN LINE P010B01 *)
7191  17    1:0     0
7192  17    1:1     0      IF CHARSWIN = NIL THEN
7193  17    1:2     5        IF BLINES24 THEN
7194  17    1:3    10          CHARSWIN := GETWIN( 0, 15, 40, 9, 3, FALSE)
7195  17    1:2    16        ELSE
7196  17    1:3    25          CHARSWIN := GETWIN( 0, 16, 40, 9, 3, FALSE);
7197  17    1:1    38      IF BASE18.B THEN
7198  17    1:2    42        DSPPARTY
7199  17    1:1    42      ELSE
7200  17    1:2    46        BEGIN
7201  17    1:3    46          PROTWIN( CHARSWIN, TRUE);
7202  17    1:3    51          FILLCHAR( CHARSWIN^.DATA[ 76], 38 * 12, ' ')
7203  17    1:2    61        END;
7204  17    1:1    61      MZLVORIG := MAZELEV;
7205  17    1:1    65      READMAZ( MAZELEV);
7206  17    1:1    68      RDMZINFO( MAZEDATA);
7207  17    1:1    73      IF BASE80B = 1 THEN
7208  17    1:2    80        IF MAZELEV = 2 THEN
7209  17    1:3    85          BEGIN
7210  17    1:4    85            MAZE.N[ 15][ 15] := OPEN;
7211  17    1:4    97            MAZE.S[ 15][ 16] := OPEN
7212  17    1:3   106          END;
7213  17    1:1   108      IF BASE7F4[ 8 - 1] = 1 THEN
7214  17    1:2   121        BEGIN
7215  17    1:3   121          BASE7F4[ 8 - 1] := 0;
7216  17    1:3   131          WHILE (MAZE.N[ MAZEX][ MAZEY] = WALL) AND
7217  17    1:3   144                (MAZE.S[ MAZEX][ MAZEY] = WALL) AND
7218  17    1:3   157                (MAZE.E[ MAZEX][ MAZEY] = WALL) AND
7219  17    1:3   170                (MAZE.W[ MAZEX][ MAZEY] = WALL)    DO
7220  17    1:4   185            BEGIN
7221  17    1:5   185              CASE RAND MOD 4 OF
7222  17    1:5   194                0:  MAZEX := MAZEX + 1;
7223  17    1:5   201                1:  MAZEX := MAZEX - 1;
7224  17    1:5   208                2:  MAZEY := MAZEY + 1;
7225  17    1:5   215                3:  MAZEY := MAZEY - 1;
7226  17    1:5   222              END;
7227  17    1:5   238
7228  17    1:5   238              MAZEX := (MAZEX + 20) MOD 20;
7229  17    1:5   245              MAZEY := (MAZEY + 20) MOD 20
```

```
7230  17   1:4   248            END;
7231  17   1:2   254          END;
7232  17   1:2   254
7233  17   1:2   254      (* 70E4 *)
7234  17   1:2   254
7235  17   1:1   254        IF BASE37 THEN
7236  17   1:2   258          BEGIN
7237  17   1:3   258            WHILE MAZE.SQRETYPE[ MAZE.SQREXTRA[ MAZEX][ MAZEY]] <> NORMAL DO
7238  17   1:4   280              BEGIN
7239  17   1:5   280                MAZEX := RAND MOD (19 + 1);
7240  17   1:5   291                MAZEY := RAND MOD (19 + 1)
7241  17   1:4   299              END;
7242  17   1:2   304          END;
7243  17   1:2   304
7244  17   1:2   304      (* 7116 *)
7245  17   1:2   304
7246  17   1:1   304        ORACLEX := (15 + MAZEX + (RAND MOD 10)) MOD 20;
7247  17   1:1   320        ORACLEY := (15 + MAZEY + (RAND MOD 10)) MOD 20;
7248  17   1:1   336        BASE37 := FALSE;
7249  17   1:1   339        CLROOMFG( MAZEX, MAZEY);
7250  17   1:1   343                                  (* 1234567890123456789012 *)
7251  17   1:1   343        GETMSGTX( CHOICES, 722);  (* WK 8AL 4DR 6XB 2OSTCIQ *)
7252  17   1:1   352
7253  17   1:1   352                                  (* C)AMP S)TATUS T)IME O)FF A-W-D K, ETC. *)
7254  17   1:1   352
7255  17   1:1   352                                  (*  3 == <0B> UP ARROW   OR CTRL-K *)
7256  17   1:1   352                                  (*  7 == <09> TAB        OR CTRL-I *)
7257  17   1:1   352                                  (* 11 == <0C>               CTRL-L *)
7258  17   1:1   352                                  (* 15 == <0A> DOWN ARROW OR CTRL-J *)
7259  17   1:1   352        COMMAND := -1;
7260  17   1:1   357        BSPIN := FALSE;
7261  17   1:1   360        SPINCNT := 0;
7262  17   1:1   364
7263  17   1:1   364        REPEAT
7264  17   1:2   364          RUNMAIN
7265  17   1:1   364        UNTIL FALSE
7266  17   1:1   366
7267  17   1:0   366      END;  (* P010B01 *)
7268  17   1:0   390
7269  17   1:0   390 (*$I WIZ4B:RUNNER3 *)
7270  17   1:0   390
7270  17   1:0   390 (*$I WIZ4B:DOCOPY  *)
7271  17   1:0   390
7272  18   1:D     1 SEGMENT PROCEDURE DOCOPY;   (* P010C01 *)
7273  18   1:D     1
7274  18   1:D     1   VAR
7275  18   1:D     1         DCMP01 : INTEGER;
7276  18   1:D     2         MP02 : INTEGER;
7277  18   1:D     3         DCTEMP   : INTEGER;
7278  18   1:D     4         CHUNKCNT : INTEGER;
7279  18   1:D     5         IOCHUNKS : INTEGER;
7280  18   1:D     6         RETRYCNT : INTEGER;
7281  18   1:D     7         CURRBLCK : INTEGER;
7282  18   1:D     8         UNUSED1  : INTEGER;
7283  18   1:D     9         UNUSED2  : INTEGER;
7284  18   1:D    10         AUXBUFN  : INTEGER;
7285  18   1:D    11         CACHBUFN : INTEGER;
```

```
7286  18   1:D   12       MEMBLCKS : INTEGER;
7287  18   1:D   13       TOTCACHE : INTEGER;
7288  18   1:D   14       DSKBLCKS : INTEGER;
7289  18   1:D   15       DRIVECNT : INTEGER;
7290  18   1:D   16       MEMBUFFX : PINT;
7291  18   1:D   17       MEMBUFF  : PINT;
7292  18   1:D   18       CACHEPTR : TCACHEP;
7293  18   1:D   19       MESSAGES : ARRAY[ 0..17] OF STRING;
7294  18   1:D  757
7295  18   1:D  757       INSERTA  : STRING;
7296  18   1:D  798       CYCLEMSG : STRING;
7297  18   1:D  839       FORMTERR : STRING;
7298  18   1:D  880       NOTBLANK : STRING;
7299  18   1:D  921       NOTMASTR : STRING;
7300  18   1:D  962       UWINOUT  : INTEGER;
7301  18   1:D  963       UWINOUT2 : INTEGER;
7302  18   1:D  964
7303  18   1:D  964
7304  18   2:D    1   PROCEDURE CLRRECTA;         (* P010C02 *)
7305  18   2:D    1
7306  18   2:0    0     BEGIN
7307  18   2:1    0       CLRRECT( MAINWIN, 0, 10, 36, 4);
7308  18   2:1    9       MVCURSOR( MAINWIN, 0, 10)
7309  18   2:0   13     END;
7310  18   2:0   28
7311  18   2:0   28
7312  18   3:D    1   PROCEDURE INSERTDK( MSGNUM   : INTEGER;  (* P010C03 *)
7313  18   3:D    2                        DRIVE12  : INTEGER;
7314  18   3:D    3                        DEVICE45 : INTEGER;
7315  18   3:D    4                        CHECK012 : INTEGER);
7316  18   3:D    5
7317  18   3:D    5     (* MSGNUM IS 2004:  INSERT MASTER DISKETTE INTO DRIVE ^.
7318  18   3:D    5        MSGNUM IS 2005:  INSERT BLANK DISKETTE INTO DRIVE ^.
7319  18   3:D    5
7320  18   3:D    5        CHECK012:  0 = DON'T CHECK DISKETTE IN DRIVE
7321  18   3:D    5                   1 = CHECK FOR "BLANK" IN DRIVE
7322  18   3:D    5                   2 = CHECK FOR "MASTER" IN DRIVE          *)
7323  18   3:D    5
7324  18   4:D    3     FUNCTION CHKDSKID : BOOLEAN;
7325  18   4:D    3
7326  18   4:D    3       (* RETURNS "TRUE" IF DISK # IS 1 TO 11 *)
7327  18   4:D    3
7328  18   4:0    0       BEGIN (* CHKDSKID *)
7329  18   4:1    0         UNITREAD( DEVICE45, BASE55B, 512, 1, 0); (* GET DISK # FROM BLOCK #1 *)
7330  18   4:1   14         MOVELEFT( BASE55B[ 510], DCTEMP, 2);  (* DISK # *)
7331  18   4:1   27         CHKDSKID := (DCTEMP > 0) AND (DCTEMP < 12)
7332  18   4:0   37       END;  (* CHKDSKID *)
7333  18   4:0   52
7334  18   4:0   52
7335  18   3:0    0     BEGIN (* INSERTDK *)
7336  18   3:0    0
7337  18   3:1    0       REPEAT
7338  18   3:1    0
7339  18   3:2    0         CLRRECTA;
7340  18   3:2    2         BASE47 := MESSAGES[ MSGNUM - 2000];
7341  18   3:2   16         COPYSTR( BASEC7, MESSAGES[ 2003 - 2000], DRIVE12, 1);
7342  18   3:2   36                                    (* DEST, SRC, F, CNT *)
```

```
7343  18   3:2   36         (* 2003:  12 *)
7344  18   3:2   36         (* IF DRIVE12 = 1 THEN BASEC7 := '1'; *)
7345  18   3:2   36         (* IF DRIVE12 = 2 THEN BASEC7 := '2'; *)
7346  18   3:2   36
7347  18   3:2   36         INSERTST( BASEC7, BASE47);
7348  18   3:2   44         CENTSTR( MAINWIN, BASE47);
7349  18   3:2   51         CENTSTR( MAINWIN, MESSAGES[ 2017 - 2000]); (* THEN PRESS ~ *)
7350  18   3:2   68         GETCR;
7351  18   3:2   71         CLRRECTA;
7352  18   3:2   73         IF CHECK012 = 0 THEN
7353  18   3:3   78           EXIT( INSERTDK)
7354  18   3:2   82         ELSE
7355  18   3:3   84           BEGIN
7356  18   3:4   84             IF CHECK012 = 2 THEN
7357  18   3:5   89               BEGIN
7358  18   3:6   89                 IF CHKDSKID THEN
7359  18   3:7   95                   EXIT( INSERTDK)
7360  18   3:6   99                 ELSE
7361  18   3:7  101                   CENTSTR( MAINWIN, NOTMASTR)
7362  18   3:5  107               END
7363  18   3:4  110             ELSE
7364  18   3:5  112               BEGIN
7365  18   3:6  112                 IF CHKDSKID THEN
7366  18   3:7  118                   CENTSTR( MAINWIN, NOTBLANK)
7367  18   3:6  124                 ELSE
7368  18   3:7  129                   EXIT( INSERTDK)
7369  18   3:5  133               END
7370  18   3:3  133           END;
7371  18   3:3  133
7372  18   3:2  133         GETCR;
7373  18   3:2  136
7374  18   3:1  136       UNTIL FALSE;
7375  18   3:1  139
7376  18   3:0  139     END;  (* INSERTDK *)
7377  18   3:0  154
7378  18   3:0  154
7379  18   5:D    1   PROCEDURE CENTMSG( MSGI : INTEGER);    (* P010C05 *)
7380  18   5:D    2
7381  18   5:0    0     BEGIN
7382  18   5:1    0       IF MSGI = 2030 THEN       (* FORMAT ERROR! *)
7383  18   5:2    7         BEGIN
7384  18   5:3    7           CLRRECTA;
7385  18   5:3    9           CENTSTR( MAINWIN, FORMTERR)
7386  18   5:2   15         END
7387  18   5:1   18       ELSE IF MSGI = 2032 THEN  (* INSERT SCENARIO DISKETTE "A" *)
7388  18   5:3   27         BEGIN
7389  18   5:4   27           CLRRECTA;
7390  18   5:4   29           CENTSTR( MAINWIN, INSERTA)
7391  18   5:3   35         END
7392  18   5:2   38       ELSE                       (* ALL OTHER MESSAGES *)
7393  18   5:3   40         BEGIN
7394  18   5:4   40           CLRRECTA;
7395  18   5:4   42           CENTSTR( MAINWIN, MESSAGES[ MSGI - 2000])
7396  18   5:3   54         END;
7397  18   5:0   57     END;
7398  18   5:0   70
7399  18   5:0   70
```

```
7400  18   6:D    1  PROCEDURE EXITDC;  (* P010C06 *)
7401  18   6:D    1
7402  18   6:0    0    BEGIN
7403  18   6:1    0      IF DRIVECNT = 1 THEN
7404  18   6:2    7        INSERTDK( 2004, 1, 4, 2);  (* "INSERT MASTER DISKETTE INTO DRIVE ^" *)
7405  18   6:2   15                                   (* DRIVE 1, DEVICE 4, CHECK FOR MASTER   *)
7406  18   6:1   15      RELEASE( MEMBUFF);
7407  18   6:1   20      DCTEMP := 1;
7408  18   6:1   24      UNITWRITE( BASE12, DCTEMP, 22, DCTEMP);
7409  18   6:1   37      IF BRELOC THEN
7410  18   6:2   42        BEGIN
7411  18   6:3   42          BASE09 := 1;
7412  18   6:3   45          UNITWRITE( BASE12, BASE09, 31, BASE09)
7413  18   6:2   55        END;
7414  18   6:1   55      BASE09 := 1;
7415  18   6:1   58      EXIT( DOCOPY)
7416  18   6:0   62    END;
7417  18   6:0   74
7418  18   6:0   74
7419  18   7:D    3  FUNCTION NOERROR( MSGNUM : INTEGER;
7420  18   7:D    4                       IOSTAT : INTEGER) : BOOLEAN; (* P010C07 *)
7421  18   7:D    5
7422  18   7:0    0    BEGIN
7423  18   7:1    0      NOERROR := TRUE;
7424  18   7:1    3      IF IOSTAT = 0 THEN
7425  18   7:2    8        BEGIN
7426  18   7:3    8          RETRYCNT := 10;  (* RESET RETRYCNT ON A GOOD I/O *)
7427  18   7:3   12          EXIT( NOERROR)
7428  18   7:2   16        END;
7429  18   7:1   16      NOERROR := FALSE;
7430  18   7:1   19      RETRYCNT := RETRYCNT - 1;
7431  18   7:1   27      IF RETRYCNT = 0 THEN
7432  18   7:2   34        BEGIN
7433  18   7:3   34          CENTMSG( MSGNUM);
7434  18   7:3   37          CENTSTR( MAINWIN, MESSAGES[ 2011 - 2000]); (* MAKE HAS FAILED *)
7435  18   7:3   54          GETCR;
7436  18   7:3   57          EXITDC
7437  18   7:2   57        END
7438  18   7:0   59    END;
7439  18   7:0   72
7440  18   7:0   72
7441  18   8:D    1  PROCEDURE DSPCYCLE;  (* P010C08 *)
7442  18   8:D    1
7443  18   8:D    1   (* # OF ^ CYCLES COMPLETED *)
7444  18   8:D    1   (* <CHUNKCNT> OF <IOCHUNKS> CYCLES COMPLETED *)
7445  18   8:D    1
7446  18   8:0    0    BEGIN
7447  18   8:1    0      BASE47 := CYCLEMSG;
7448  18   8:1    8      INT2STR( CHUNKCNT, BASEC7);
7449  18   8:1   17      INSERT2( BASEC7, BASE47, '#');
7450  18   8:1   26      INT2STR( IOCHUNKS, BASEC7);
7451  18   8:1   35      INSERT2( BASEC7, BASE47, '^');
7452  18   8:1   44      MVCURSOR( MAINWIN, 0, 2);
7453  18   8:1   51      CENTSTR(  MAINWIN, BASE47);
7454  18   8:1   58      CHUNKCNT := CHUNKCNT + 1
7455  18   8:0   61    END;
7456  18   8:0   78
```

```
7457  18    8:0    78
7458  18    9:D     1   PROCEDURE CACHEMAG;     (* P010C09 *)
7459  18    9:D     1
7460  18    9:D     1     (* CACHE MAGIC *)
7461  18    9:D     1
7462  18    9:0     0     BEGIN
7463  18    9:1     0       IF BCACHE THEN
7464  18    9:2     5         BEGIN
7465  18    9:3     5           UWINOUT := -1;
7466  18    9:3    11           UNITWRITE( BASE12, UWINOUT, 25, 0);
7467  18    9:3    23           UWINOUT := -3;
7468  18    9:3    29           UNITWRITE( BASE12, UWINOUT, 25, 0)
7469  18    9:2    41         END;
7470  18    9:0    41     END;
7471  18    9:0    54
7472  18    9:0    54
7473  18   10:D     1   PROCEDURE P010C0A;
7474  18   10:D     1
7475  18   10:0     0     BEGIN
7476  18   10:1     0       CLEARWIN( MAINWIN, TRUE);
7477  18   10:1     6       FOR DCTEMP := 2000 TO 2017 DO
7478  18   10:2    24         GETMSGTX( MESSAGES[ DCTEMP - 2000], DCTEMP);
7479  18   10:2    52
7480  18   10:2    52       (*
7481  18   10:2    52           2000: MAKE SCENARIO DISKETTE:
7482  18   10:2    52           2001: DO YOU HAVE 1) OR 2) DISK DRIVES ?
7483  18   10:2    52           2002: (PRESS (ESC) TO EXIT)
7484  18   10:2    52           2003: 12
7485  18   10:2    52           2004: INSERT MASTER DISKETTE INTO DRIVE ^.
7486  18   10:2    52           2005: INSERT BLANK DISKETTE INTO DRIVE ^.
7487  18   10:2    52           2006: READING MASTER DISKETTE...
7488  18   10:2    52           2007: WRITING TO BLANK DISKETTE...
7489  18   10:2    52           2008: FORMATTING BLANK DISKETTE...
7490  18   10:2    52           2009: READ ERROR!
7491  18   10:2    52           2010: WRITE ERROR!
7492  18   10:2    52           2011: "MAKE" HAS FAILED - PRESS ~
7493  18   10:2    52           2012: SCENARIO DISKETTE MADE - PRESS ~
7494  18   10:2    52           2013: THE DISKETTE YOU INSERTE HAS DATA
7495  18   10:2    52           2014: ON IT.  IF YOU USE IT, THE DATA
7496  18   10:2    52           2015: WILL BE ERASED! USE IT (Y/N)?
7497  18   10:2    52           2016: YN
7498  18   10:2    52           2017: THEN PRESS ~
7499  18   10:2    52       *)
7500  18   10:2    52
7501  18   10:1    52       GETMSGTX( NOTMASTR, 2028);  (* THAT IS NOT A MASTER DISKETTE-PRESS ~ *)
7502  18   10:1    62       GETMSGTX( NOTBLANK, 2029);  (* THAT IS NOT A BLANK DISKETTE-PRESS ~ *)
7503  18   10:1    72       GETMSGTX( FORMTERR, 2030);  (* FORMAT ERROR! *)
7504  18   10:1    82       GETMSGTX( CYCLEMSG, 2031);  (* # OF ^ CYCLES COMPLETED *)
7505  18   10:1    92       GETMSGTX( INSERTA,  2032);  (* INSERT SCENARIO DISKETTE "A"-PRESS ~ *)
7506  18   10:1   102
7507  18   10:1   102       IF BRELOC THEN
7508  18   10:2   107         BEGIN
7509  18   10:2   107
7510  18   10:2   107           (* RESET TO 'NORMAL' DISK I/O AND NOT 'VIRTUAL' DISK I/O *)
7511  18   10:2   107
7512  18   10:3   107           BASE09 := 2;
7513  18   10:3   110           UNITWRITE( BASE12, BASE09, 31, BASE09)
```

```
7514  18   10:2   120          END;
7515  18   10:1   120        CENTSTR( MAINWIN, MESSAGES[ 2000-2000]); (* MAKE SCENARIO DISK *)
7516  18   10:1   137
7517  18   10:1   137        (* RESERVE A LARGE CHUNK OF MEMORY *)
7518  18   10:1   137        TOTCACHE := (MEMAVAIL - 2000) DIV 256; (* MEMAVAIL IS IN 'WORDS'  *)
7519  18   10:1   150        IF TOTCACHE > 63 THEN                  (* TOTCACHE IS IN 'BLOCKS' *)
7520  18   10:2   157          TOTCACHE := 63;
7521  18   10:1   161        MARK( MEMBUFF);
7522  18   10:1   166        MOVELEFT( MEMBUFF, BASE09, 2);
7523  18   10:1   176        BASE09 := BASE09 + 512 * TOTCACHE;     (* BASE09 IS IN 'BYTES' *)
7524  18   10:1   187        MOVELEFT( BASE09, MEMBUFFX, 2);
7525  18   10:1   197        RELEASE( MEMBUFFX);
7526  18   10:1   202
7527  18   10:1   202        (* CLEAR THE DISK CACHE BUFFERS *)
7528  18   10:1   202        CACHBUFN := 0;
7529  18   10:1   206        CACHEPTR := DSKCACHF;
7530  18   10:1   211        WHILE CACHEPTR <> NIL DO
7531  18   10:2   218          BEGIN
7532  18   10:3   218            CACHBUFN := CACHBUFN + 1;
7533  18   10:3   226            CACHEPTR^.KEY := -32767;
7534  18   10:3   234            CACHEPTR := CACHEPTR^.FORWRD
7535  18   10:2   237          END;
7536  18   10:2   243
7537  18   10:1   243        CACHEMAG;  (* CACHE MAGIC *)
7538  18   10:1   245
7539  18   10:1   245        IF BCACHE THEN
7540  18   10:2   250          BEGIN
7541  18   10:2   250
7542  18   10:2   250            (* MORE CACHE MAGIC *)
7543  18   10:2   250
7544  18   10:3   250            UWINOUT  := 0;
7545  18   10:3   255            UWINOUT2 := 0;
7546  18   10:3   260            UNITWRITE( BASE12, UWINOUT, 25, 0); (* MEM AVAIL IN AUX MEMORY *)
7547  18   10:3   272            AUXBUFN := UWINOUT
7548  18   10:2   272          END
7549  18   10:1   279        ELSE
7550  18   10:2   281          AUXBUFN := 0;
7551  18   10:2   285
7552  18   10:1   285        MEMBLCKS := TOTCACHE;
7553  18   10:1   291        TOTCACHE := TOTCACHE + CACHBUFN + AUXBUFN;
7554  18   10:1   305        MVCURSOR( MAINWIN, 0, 10);
7555  18   10:1   312        CENTSTR( MAINWIN, MESSAGES[ 2001 - 2000]);  (* '1' OR '2'  *)
7556  18   10:1   329        CENTSTR( MAINWIN, MESSAGES[ 2002 - 2000]);  (* PRESS (ESC) *)
7557  18   10:1   346
7558  18   10:1   346        REPEAT
7559  18   10:2   346          GETKEY;
7560  18   10:2   349          IF INCHAR = CHR( 27) THEN
7561  18   10:3   354            EXITDC;
7562  18   10:1   356        UNTIL (INCHAR = '1') OR (INCHAR = '2');
7563  18   10:1   365
7564  18   10:1   365        DRIVECNT := ORD( INCHAR) - ORD( '0');
7565  18   10:1   371        DCTEMP := 0;
7566  18   10:1   375        UNITWRITE( BASE12, DCTEMP, 22, DCTEMP);
7567  18   10:1   388
7568  18   10:1   388        IF DRIVECNT = 2 THEN
7569  18   10:2   395          BEGIN
7570  18   10:3   395            INSERTDK( 2004, 1, 4, 2);   (* "INSERT MASTER IN DRIVE ^" *)
```

```
7571  18   10:3   403                                    (* DRIVE 1, DEVICE 4, CHECK 4 MASTER *)
7572  18   10:3   403          INSERTDK( 2005, 2, 5, 0)    (* "INSERT BLANK IN DRIVE ^" *)
7573  18   10:3   409                                    (* DRIVE 2, DEVICE 5, NO DISK CHECK  *)
7574  18   10:2   409        END
7575  18   10:1   411      ELSE
7576  18   10:2   413        INSERTDK( 2005, 1, 4, 0);     (* "INSERT BLANK IN DRIVE ^" *)
7577  18   10:2   421                                    (* DRIVE 1, DEVICE 4, NO DISK CHECK  *)
7578  18   10:2   421
7579  18   10:2   421      (* IF DRIVECNT = 2, THEN DISK IN DRIVE 2 HAS NOT BEEN CHECKED *)
7580  18   10:2   421
7581  18   10:1   421      UNITREAD( 3 + DRIVECNT, MEMBUFF^, 512, 0);
7582  18   10:1   437      IF IORESULT = 0 THEN
7583  18   10:2   443        BEGIN
7584  18   10:3   443          CLRRECTA;
7585  18   10:3   445          CENTSTR( MAINWIN, MESSAGES[ 2013 - 2000]); (* DATA ON DISK *)
7586  18   10:3   462          CENTSTR( MAINWIN, MESSAGES[ 2014 - 2000]); (* USE IT?      *)
7587  18   10:3   479          CENTSTR( MAINWIN, MESSAGES[ 2015 - 2000]); (* Y/N          *)
7588  18   10:3   496
7589  18   10:3   496          REPEAT
7590  18   10:4   496            GETKEY;
7591  18   10:4   499            IF INCHAR = MESSAGES[ 2016 - 2000][ 2] THEN   (* N *)
7592  18   10:5   517              EXITDC
7593  18   10:3   517          UNTIL INCHAR = MESSAGES[ 2016 - 2000][ 1]       (* Y *)
7594  18   10:3   533
7595  18   10:2   533        END;
7596  18   10:1   537      CENTMSG( 2008)   (* FORMATTING BLANK DISK *)
7597  18   10:0   540    END;
7598  18   10:0   562
7599  18   10:0   562
7600  18   11:D     1  PROCEDURE DO1CYCLE( MP02 : INTEGER; MP01 : INTEGER); (* P010C0B *)
7601  18   11:D     3
7602  18   11:D     3    (* MP02 = CURRBLCK;   MP01 := TOTCACHE *)
7603  18   11:D     3
7604  18   11:D     3
7605  18   11:D     3
7606  18   12:D     1    PROCEDURE READMAST( MP02 : INTEGER;  (* P010C0C *)
7607  18   12:D     2                        MP01 : INTEGER);
7608  18   12:D     3
7609  18   12:D     3      (* MP02 = CURRBLCK;   MP01 = TOTCACHE                    *)
7610  18   12:D     3      (*                         = AMOUNT OF I/O NEEDED       *)
7611  18   12:D     3      (*                           (SOMETIMES LESS THAN TOTCACHE *)
7612  18   12:D     3
7613  18   12:D     3      VAR
7614  18   12:D     3           MP03 : INTEGER;
7615  18   12:D     4           MP04 : INTEGER;  (* AMOUNT OF AUX MEMORY NEEDED FOR I/O *)
7616  18   12:D     5
7617  18   12:0     0      BEGIN (* READMAST *)
7618  18   12:1     0        IF BCACHE THEN
7619  18   12:2     5          BEGIN
7620  18   12:3     5            CACHEMAG;
7621  18   12:3     7            IF AUXBUFN > MP01 THEN
7622  18   12:3    14
7623  18   12:3    14              (* IF AMOUNT OF I/O IS LESS THAN AUXBUFN SET MP04
7624  18   12:3    14                   TO THE SMALLER AMOUNT TO INDICATE TOTAL
7625  18   12:3    14                   AMOUNT OF AUX MEMORY NEEDED FOR I/O         *)
7626  18   12:3    14
7627  18   12:4    14              MP04 := MP01
```

```
7628  18   12:3    14             ELSE
7629  18   12:4    19               MP04 := AUXBUFN;   (* USE ALL OF AUX MEMORY *)
7630  18   12:3    24             UWINOUT  := MP02;
7631  18   12:3    29             UWINOUT2 := MP04;
7632  18   12:3    34             UNITWRITE( BASE12, UWINOUT, 25, 0);
7633  18   12:3    46             MP02 := MP02 + MP04;
7634  18   12:3    51             MP01 := MP01 - MP04
7635  18   12:2    52           END;
7636  18   12:1    56         IF MP01 = 0 THEN
7637  18   12:2    61           EXIT( READMAST);
7638  18   12:1    65         IF MEMBLCKS > MP01 THEN
7639  18   12:2    72           MP04 := MP01
7640  18   12:1    72         ELSE
7641  18   12:2    77           MP04 := MEMBLCKS;
7642  18   12:2    82
7643  18   12:1    82         REPEAT
7644  18   12:2    82           UNITREAD( 4, MEMBUFF^, 512 * MP04, MP02)
7645  18   12:1    96         UNTIL NOERROR( 2009, IORESULT);  (* READ ERROR! *)
7646  18   12:1   107
7647  18   12:1   107         MP02 := MP02 + MP04;
7648  18   12:1   112         MP01 := MP01 - MP04;
7649  18   12:1   117         IF MP01 = 0 THEN
7650  18   12:2   122           EXIT( READMAST);
7651  18   12:1   126         IF CACHBUFN > MP01 THEN
7652  18   12:2   133           MP04 := MP01
7653  18   12:1   133         ELSE
7654  18   12:2   138           MP04 := CACHBUFN;
7655  18   12:1   143         CACHEPTR := DSKCACHF;
7656  18   12:1   148
7657  18   12:1   148         FOR MP03 := 0 TO MP04 - 1 DO
7658  18   12:2   161           BEGIN
7659  18   12:3   161             REPEAT
7660  18   12:4   161               UNITREAD( 4, CACHEPTR^.DATA, 512, MP02 + MP03);
7661  18   12:3   177             UNTIL NOERROR( 2009, IORESULT);   (* READ ERROR! *)
7662  18   12:3   188             CACHEPTR := CACHEPTR^.FORWRD
7663  18   12:2   191           END;
7664  18   12:2   202
7665  18   12:0   202       END;  (* READMAST *)
7666  18   12:0   220
7667  18   12:0   220
7668  18   13:D     1     PROCEDURE WRITBLNK( MP02 : INTEGER;
7669  18   13:D     2                         MP01 : INTEGER);
7670  18   13:D     3
7671  18   13:D     3       VAR
7672  18   13:D     3            MP03 : INTEGER;
7673  18   13:D     4            MP04 : INTEGER;
7674  18   13:D     5
7675  18   13:0     0       BEGIN (* WRITBLNK *)
7676  18   13:1     0         IF BCACHE THEN
7677  18   13:2     5           BEGIN
7678  18   13:3     5             IF AUXBUFN > MP01 THEN
7679  18   13:4    12               MP04 := MP01
7680  18   13:3    12             ELSE
7681  18   13:4    17               MP04 := AUXBUFN;
7682  18   13:3    22             UWINOUT := -2;
7683  18   13:3    28             UNITWRITE( BASE12, UWINOUT, 25, 0);
7684  18   13:3    40             FOR MP03 := 0 TO MP04 - 1 DO
```

```
7685  18   13:4    53              BEGIN
7686  18   13:5    53                REPEAT
7687  18   13:6    53                  UNITREAD( 4, BASE55B, 512, MP02 + MP03);
7688  18   13:5    67                UNTIL NOERROR( 2009, IORESULT);  (* READ ERROR *)
7689  18   13:5    78
7690  18   13:5    78                REPEAT
7691  18   13:6    78                  UNITWRITE( 3 + DRIVECNT, BASE55B, 512, MP02 + MP03);
7692  18   13:5    96                UNTIL NOERROR( 2009, IORESULT)   (* READ ERROR *)
7693  18   13:4   101              END;
7694  18   13:4   114
7695  18   13:3   114            CACHEMAG;
7696  18   13:3   116            MP02 := MP02 + MP04;
7697  18   13:3   121            MP01 := MP01 - MP04
7698  18   13:2   122          END; (* BCACHE *)
7699  18   13:2   126
7700  18   13:2   126        (* 55D2 *)
7701  18   13:2   126
7702  18   13:1   126        IF MP01 = 0 THEN
7703  18   13:2   131          EXIT( WRITBLNK);
7704  18   13:1   135        IF MEMBLCKS > MP01 THEN
7705  18   13:2   142          MP04 := MP01
7706  18   13:1   142        ELSE
7707  18   13:2   147          MP04 := MEMBLCKS;
7708  18   13:2   152
7709  18   13:1   152        REPEAT
7710  18   13:2   152          UNITWRITE( 3 + DRIVECNT, MEMBUFF^, 512 * MP04, MP02)
7711  18   13:1   170        UNTIL NOERROR( 2009, IORESULT);  (* READ ERROR *)
7712  18   13:1   181
7713  18   13:1   181        MP02 := MP02 + MP04;
7714  18   13:1   186        MP01 := MP01 - MP04;
7715  18   13:1   191        IF MP01 = 0 THEN
7716  18   13:2   196          EXIT( WRITBLNK);
7717  18   13:1   200        IF CACHBUFN > MP01 THEN
7718  18   13:2   207          MP04 := MP01
7719  18   13:1   207        ELSE
7720  18   13:2   212          MP04 := CACHBUFN;
7721  18   13:1   217        CACHEPTR := DSKCACHF;
7722  18   13:1   222        FOR MP03 := 0 TO MP04 - 1 DO
7723  18   13:2   235          BEGIN
7724  18   13:3   235            REPEAT
7725  18   13:4   235              UNITWRITE( 3 + DRIVECNT, CACHEPTR^.DATA, 512, MP02 + MP03);
7726  18   13:3   255            UNTIL NOERROR( 2009, IORESULT);  (* READ ERROR *)
7727  18   13:3   266            CACHEPTR := CACHEPTR^.FORWRD
7728  18   13:2   269          END
7729  18   13:0   273      END;  (* WRITBLNK *)
7730  18   13:0   304
7731  18   13:0   304
7732  18   11:0     0    BEGIN (* DO1CYCLE *)
7733  18   11:1     0      IF DRIVECNT = 1 THEN
7734  18   11:2     7        INSERTDK( 2004, 1, 4, 2); (* "INSERT MASTER INTO DRIVE ^"      *)
7735  18   11:2    15                                  (* DRIVE 1, DEVICE 4, CHECK 4 MASTER *)
7736  18   11:1    15      CENTMSG( 2006);             (* READING MASTER *)
7737  18   11:1    20      READMAST( MP02, MP01);
7738  18   11:1    24      IF DRIVECNT = 1 THEN
7739  18   11:2    31        INSERTDK( 2005, 1, 4, 1); (* "INSERT BLANK INTO DRIVE ^"      *)
7740  18   11:2    39                                  (* DRIVE 1, DEVICE 4, CHECK 4 BLANK *)
7741  18   11:1    39      CENTMSG( 2007);             (* WRITING BLANK *)
```

```
7742  18   11:1    44       WRITBLNK( MP02, MP01)
7743  18   11:0    46     END;  (* DO1CYCLE *)
7744  18   11:0    60
7745  18   11:0    60
7746  18    1:0     0   BEGIN (* DOCOPY, P010C01 *)
7747  18    1:0     0
7748  18    1:1     0     CLEARWIN( MAINWIN, TRUE);
7749  18    1:1     6     DCTEMP := 2050;                  (* MAKING A SCENARIO DISK...*)
7750  18    1:1    11                                      (* BEFORE YOU PLAY THE GAME....*)
7751  18    1:1    11     GETMSGTX( BASE47, DCTEMP);
7752  18    1:1    17     WHILE BASE47 <> '**ERR**' DO   (* DO 16 LINES OF MESSAGES *)
7753  18    1:2    33       BEGIN
7754  18    1:3    33         CENTSTR( MAINWIN, BASE47);
7755  18    1:3    40         DCTEMP := DCTEMP + 1;
7756  18    1:3    45         GETMSGTX( BASE47, DCTEMP)
7757  18    1:2    48       END;
7758  18    1:2    53
7759  18    1:1    53     GETCR;
7760  18    1:1    56     P010C0A;
7761  18    1:1    58     DCTEMP := 3 + DRIVECNT;
7762  18    1:1    63     RETRYCNT := 1;
7763  18    1:1    66
7764  18    1:1    66     REPEAT
7765  18    1:2    66       UNITWRITE( BASE12, MEMBUFF^, 20, DCTEMP)
7766  18    1:1    76     UNTIL NOERROR( 2030, IORESULT);       (* FORMAT ERROR!  (??) *)
7767  18    1:1    87
7768  18    1:1    87     UNITREAD( BASE12, DSKBLCKS, 21, 0);  (* GET # OF BLOCKS PER DISK 280 *)
7769  18    1:1    97     CURRBLCK := 6;  (* START WITH BLOCK #6 *)
7770  18    1:1   100     CHUNKCNT := 0;
7771  18    1:1   103     IOCHUNKS := 1 +  (DSKBLCKS - 6) DIV TOTCACHE;
7772  18    1:1   112     IF ((DSKBLCKS - 6) MOD TOTCACHE) <> 0 THEN
7773  18    1:2   121       IOCHUNKS := IOCHUNKS + 1;
7774  18    1:2   126
7775  18    1:1   126     WHILE CURRBLCK < DSKBLCKS DO
7776  18    1:2   131       BEGIN
7777  18    1:3   131         DSPCYCLE;
7778  18    1:3   133         IF CURRBLCK + TOTCACHE <= DSKBLCKS THEN
7779  18    1:4   140           DO1CYCLE( CURRBLCK, TOTCACHE)          (* DO AS MUCH AS POSSIBLE *)
7780  18    1:3   142         ELSE
7781  18    1:4   146           DO1CYCLE( CURRBLCK, DSKBLCKS - CURRBLCK);  (* FINISH LAST PARTIAL *)
7782  18    1:3   152         CURRBLCK := CURRBLCK + TOTCACHE
7783  18    1:2   153       END;
7784  18    1:2   159
7785  18    1:1   159     DSPCYCLE;
7786  18    1:1   161     DO1CYCLE( 0, 6);
7787  18    1:1   165     DSPCYCLE;
7788  18    1:1   167     CENTMSG( 2032);  (* INSERT SCENARIO DISKETTE "A" *)
7789  18    1:1   172     GETCR;
7790  18    1:1   175     EXITDC
7791  18    1:0   175   END;  (* DOCOPY, P010C01 *)
7792  18    1:0   196
7793  18    1:0   196 (*$I WIZ4B:DOCOPY  *)
7793  18    1:0   196 (*$I WIZ4B:DOCACHE *)
7794  18    1:0   196
7795  19    1:D     1 SEGMENT PROCEDURE DOCACHE;  (* P010D01 *)
7796  19    1:D     1
7797  19    2:D     1   PROCEDURE DOCACHE2;  (* P010D02 *)
```

```
7798  19   2:D    1
7799  19   2:D    1    TYPE
7800  19   2:D    1        TSEGDICT = RECORD
7801  19   2:D    1            DISKINFO: ARRAY[ 0..15] OF
7802  19   2:D    1              RECORD
7803  19   2:D    1                CODEADDR : INTEGER;
7804  19   2:D    1                CODELENG : INTEGER;
7805  19   2:D    1              END;
7806  19   2:D    1            SEGNAME  : ARRAY[ 0..15] OF PACKED ARRAY[ 0..7] OF CHAR;
7807  19   2:D    1            SEGKIND  : ARRAY[ 0..15] OF INTEGER; (* MORE STUFF *)
7808  19   2:D    1            TEXTADDR : ARRAY[ 0..15] OF INTEGER;
7809  19   2:D    1            SEGINFO  : PACKED ARRAY[ 0..15] OF INTEGER; (* MORE STUFF *)
7810  19   2:D    1            INTRINS  : SET OF 0..31;
7811  19   2:D    1            (* UNDEFINED FORMAT STUFF *)
7812  19   2:D    1            COMMENT  : STRING;
7813  19   2:D    1          END;
7814  19   2:D    1
7815  19   2:D    1
7816  19   2:D    1    VAR
7817  19   2:D    1        SYSPASBL : INTEGER;
7818  19   2:D    2        NAMEX    : INTEGER;
7819  19   2:D    3        SEGI : INTEGER;
7820  19   2:D    4        MP04 : INTEGER;
7821  19   2:D    5
7822  19   2:D    5        SC   : RECORD CASE INTEGER OF
7823  19   2:D    5                   1: (SEG : RECORD
7824  19   2:D    5                                ADDR : INTEGER;
7825  19   2:D    5                                LENG : INTEGER;
7826  19   2:D    5                              END);
7827  19   2:D    5
7828  19   2:D    5                   2: (CACHE : RECORD
7829  19   2:D    5                                 ZERO : INTEGER;
7830  19   2:D    5                                 ONE : INTEGER;
7831  19   2:D    5                               END)
7832  19   2:D    5                 END;
7833  19   2:D    7
7834  19   2:D    7        SEGNAMES : PACKED ARRAY[ 0..15] OF STRING[ 8];
7835  19   2:D   87        DIR      : ARRAY[ 0..77] OF DIRENTRY;  (* 78 ENTRIES *)
7836  19   2:D 1101                                                (* 77 IN FINDFILE() *)
7837  19   2:D 1101        SEGDICT  : TSEGDICT;
7838  19   2:D 1288        BMSTCACH : PACKED ARRAY[ 0..1023] OF BOOLEAN;
7839  19   2:D 1352        BTSTCACH : PACKED ARRAY[ 0..1023] OF BOOLEAN;
7840  19   2:D 1416        CACHSIZ  : INTEGER;
7841  19   2:D 1417
7842  19   2:D 1417
7843  19   3:D    1    PROCEDURE P010D03( MP07 : STRING;
7844  19   3:D    2                       MP06 : INTEGER;
7845  19   3:D    3                       MP05 : INTEGER;
7846  19   3:D    4                       MP04 : INTEGER;
7847  19   3:D    5                       MP03 : INTEGER;
7848  19   3:D    6                       MP02 : INTEGER;
7849  19   3:D    7                       MP01 : INTEGER);
7850  19   3:D   49
7851  19   3:D   49       (* UNREACHABLE CODE !? *)
7852  19   3:D   49
7853  19   3:D   49       VAR
7854  19   3:D   49           MP31 : TWINDOWP;
```

```
7855  19    3:D    50
7856  19    3:0     0       BEGIN (* P010D03 *)
7857  19    3:1     0         MP31 := GETWIN( 0, 10, 40, 5, 31, TRUE);
7858  19    3:1    18         CENTSTR( MP31, MP07);
7859  19    3:1    25         PRINTNUM( MP31, MP06, 5);
7860  19    3:1    32         PRINTNUM( MP31, MP05, 5);
7861  19    3:1    39         PRINTNUM( MP31, MP04, 5);
7862  19    3:1    46         PRINTNUM( MP31, MP03, 5);
7863  19    3:1    53         PRINTNUM( MP31, MP02, 5);
7864  19    3:1    60         PRINTNUM( MP31, MP01, 5);
7865  19    3:1    67         PRINTCR( MP31);
7866  19    3:1    72         CENTSTR( MP31, 'PRESS RETURN');
7867  19    3:1    92         GETCR;
7868  19    3:1    95         DELWIN(  MP31, TRUE)
7869  19    3:0    98       END;  (* P010D03 *)
7870  19    3:0   114
7871  19    3:0   114
7872  19    4:D     1     PROCEDURE CACHOFF;  (* P010D04 *)
7873  19    4:D     1
7874  19    4:D     1       VAR
7875  19    4:D     1           UWZERO : INTEGER;
7876  19    4:D     2
7877  19    4:0     0       BEGIN (* CACHOFF *)
7878  19    4:1     0         IF BCACHE THEN
7879  19    4:2     5           BEGIN
7880  19    4:3     5             UWZERO := -1;   (* TURN OFF LOOKING FOR CACHE *)
7881  19    4:3     9             UNITWRITE( BASE12, UWZERO, 25, 0)
7882  19    4:2    19           END
7883  19    4:0    19       END;  (* CACHOFF *)
7884  19    4:0    32
7885  19    4:0    32
7886  19    5:D     1     PROCEDURE CACHON;  (* P010D05 *)
7887  19    5:D     1
7888  19    5:D     1       VAR
7889  19    5:D     1           UWZERO : INTEGER;
7890  19    5:D     2
7891  19    5:0     0       BEGIN (* CACHON *)
7892  19    5:1     0         IF BCACHE THEN
7893  19    5:2     5           BEGIN
7894  19    5:3     5             UWZERO := -2;
7895  19    5:3     9             UNITWRITE( BASE12, UWZERO, 25, 0)  (* $EE := -1 *)
7896  19    5:2    19           END
7897  19    5:0    19       END;  (* CACHON *)
7898  19    5:0    32
7899  19    5:0    32
7900  19    6:D     1     PROCEDURE INITCACH;  (* P010D06 *)
7901  19    6:D     1
7902  19    6:D     1       VAR
7903  19    6:D     1           UWZERO : INTEGER;
7904  19    6:D     2
7905  19    6:0     0       BEGIN (* SETE6 *)
7906  19    6:1     0         CACHON;
7907  19    6:1     2         UWZERO := 1;
7908  19    6:1     5         UNITWRITE( BASE12, UWZERO, 22, UWZERO);  (* $E6 := #$80 *)
7909  19    6:1    15         EXIT( DOCACHE2)
7910  19    6:0    19       END;  (* INITCACH *)
7911  19    6:0    32
```

```
7912  19    6:0    32
7913  19    7:D     3     FUNCTION GETFILEI( UNUSED : INTEGER;  (* P010D07 *)
7914  19    7:D     4                        FILENM : STRING) : INTEGER;
7915  19    7:D    46
7916  19    7:D    46       VAR
7917  19    7:D    46            FILEI : INTEGER;
7918  19    7:D    47
7919  19    7:0     0       BEGIN (* GETFILEI *)
7920  19    7:1     0         FOR FILEI := 1 TO DIR[ 0].FILECNT DO
7921  19    7:2    25           BEGIN
7922  19    7:3    25             IF (DIR[ FILEI].FILEKIND.FT >= BADBLK) AND
7923  19    7:3    39                (DIR[ FILEI].FILEKIND.FT <= FOTOFILE) THEN
7924  19    7:3    56
7925  19    7:4    56                 IF DIR[ FILEI].FILENAME = FILENM THEN
7926  19    7:5    71                   BEGIN
7927  19    7:6    71                     GETFILEI := FILEI;
7928  19    7:6    75                     EXIT( GETFILEI)
7929  19    7:5    79                   END;
7930  19    7:2    79           END;
7931  19    7:1    87         GETFILEI := -9
7932  19    7:0    87       END;  (* GETFILEI *)
7933  19    7:0   106
7934  19    7:0   106
7935  19    8:D     1     PROCEDURE SETBMST;  (* P010D08 *)
7936  19    8:D     1
7937  19    8:D     1       VAR
7938  19    8:D     1            X     : INTEGER;
7939  19    8:D     2
7940  19    8:0     0       BEGIN (* SETBMST *)
7941  19    8:1     0         FOR X := SC.SEG.ADDR TO SC.SEG.ADDR + SC.SEG.LENG - 1 DO
7942  19    8:2    21           IF CACHSIZ > 0 THEN
7943  19    8:3    29             BEGIN
7944  19    8:4    29               BMSTCACH[ X] := TRUE;
7945  19    8:4    39               CACHSIZ := CACHSIZ - 1
7946  19    8:3    43             END;
7947  19    8:0    56       END;  (* SETBMST *)
7948  19    8:0    70
7949  19    8:0    70
7950  19    9:D     1     PROCEDURE STBCACH3( MP01 : INTEGER);  (* P010D09 *)
7951  19    9:D     2
7952  19    9:D     2       VAR
7953  19    9:D     2            P09MP02 : INTEGER;
7954  19    9:D     3            P09MP03 : INTEGER;
7955  19    9:D     4
7956  19    9:D     4
7957  19   10:D     1       PROCEDURE P010D0A( MP03 : INTEGER;
7958  19   10:D     2                          MP02 : INTEGER;
7959  19   10:D     3                          MP01 : INTEGER);
7960  19   10:D     4
7961  19   10:D     4         (* SIMILAR TO GETREC() *)
7962  19   10:D     4
7963  19   10:D     4         VAR
7964  19   10:D     4              MP04 : INTEGER;
7965  19   10:D     5              MP05 : INTEGER;
7966  19   10:D     6              MP06 : INTEGER;
7967  19   10:D     7              MP07 : INTEGER;
7968  19   10:D     8
```

```
7969  19   10:0     0         BEGIN (* P010D0A *)
7970  19   10:1     0           P09MP03 := MP03 + SCENBLK;
7971  19   10:1     7           P09MP02 := 0;
7972  19   10:1    11           MP04 := 30000 DIV MP01;
7973  19   10:1    18           WHILE MP02 > 0 DO
7974  19   10:2    23             BEGIN
7975  19   10:3    23               IF MP02 > MP04 THEN
7976  19   10:4    28                 MP07 := MP04
7977  19   10:3    28               ELSE
7978  19   10:4    33                 MP07 := MP02;
7979  19   10:3    36               MP02 := MP02 - MP07;
7980  19   10:3    41               P09MP02 := P09MP02 + MP07 * MP01;
7981  19   10:3    51               P09MP03 := P09MP03 + (P09MP02 DIV 512);
7982  19   10:3    65               P09MP02 := P09MP02 MOD 512
7983  19   10:2    68             END
7984  19   10:0    75         END;  (* P010D0A *)
7985  19   10:0    92
7986  19   10:0    92
7987  19    9:0     0       BEGIN (* STBCACH3 *)
7988  19    9:1     0         IF CACHSIZ = 0 THEN
7989  19    9:2     8           EXIT( STBCACH3);
7990  19    9:1    12         P010D0A( SCENARIO.BLOFF[    MP01],
7991  19    9:1    19                  SCENARIO.RECPERDK[ MP01],
7992  19    9:1    26                  SCENARIO.RECSIZE[  MP01]);
7993  19    9:1    35         SC.SEG.ADDR := SCENBLK + SCENARIO.BLOFF[ MP01];
7994  19    9:1    48         SC.SEG.LENG := 1 + P09MP03 - SC.SEG.ADDR;
7995  19    9:1    58         IF P09MP02 = 0 THEN
7996  19    9:2    63           SC.SEG.LENG := SC.SEG.LENG - 1;
7997  19    9:1    71         SETBMST
7998  19    9:0    71       END;  (* STBCACH3 *)
7999  19    9:0    86
8000  19    9:0    86
8001  19   11:D     1     PROCEDURE STBCACH2( MP01 : STRING);  (* P010D0B *)
8002  19   11:D    43
8003  19   11:D    43       VAR
8004  19   11:D    43            FILEI  : INTEGER;
8005  19   11:D    44            UNUSED : STRING;
8006  19   11:D    85
8007  19   11:D    85
8008  19   11:0     0       BEGIN (* STBCACH2 *)
8009  19   11:1     0         IF CACHSIZ = 0 THEN
8010  19   11:2    13           EXIT( STBCACH2);
8011  19   11:1    17         FILEI := GETFILEI( 4, MP01);
8012  19   11:1    26         IF FILEI <= 0 THEN
8013  19   11:2    32           EXIT( STBCACH2);
8014  19   11:1    36         SC.SEG.ADDR := DIR[ FILEI].FIRSTBLK;
8015  19   11:1    47         SC.SEG.LENG := DIR[ FILEI].LASTBLK - DIR[ FILEI].FIRSTBLK;
8016  19   11:1    67         SETBMST
8017  19   11:0    67       END;  (* STBCACH2 *)
8018  19   11:0    82
8019  19   11:0    82
8020  19   12:D     1     PROCEDURE STBCACH( SEGNAM : STRING);   (* P010D0C *)
8021  19   12:D    43
8022  19   12:D    43       (* SET BMSTCACH[] ELEMENTS TRUE TO INDICATE THE BLOCK SHOULD BE *)
8023  19   12:D    43       (* READ INTO CACHE FOR THE 'SEGNAM'. *)
8024  19   12:D    43
8025  19   12:D    43       VAR
```

```
8026  19   12:D    43            UNUSED  : INTEGER;
8027  19   12:D    44            SEGI    : INTEGER;
8028  19   12:D    45            UNUSED2 : ARRAY[ 0..40] OF INTEGER;
8029  19   12:D    86
8030  19   12:0     0       BEGIN (* STBCACH *)
8031  19   12:1     0         IF CACHSIZ = 0 THEN
8032  19   12:2    13           EXIT( STBCACH);
8033  19   12:1    17         FOR SEGI := 0 TO 15 DO
8034  19   12:2    30           BEGIN
8035  19   12:3    30             IF SEGNAM = SEGNAMES[ SEGI] THEN
8036  19   12:4    43               BEGIN
8037  19   12:5    43                 SC.SEG.ADDR := DIR[ SYSPASBL].FIRSTBLK +
8038  19   12:5    52                                SEGDICT.DISKINFO[ SEGI].CODEADDR;
8039  19   12:5    65                 SC.SEG.LENG :=
8040  19   12:5    65                                  (SEGDICT.DISKINFO[ SEGI].CODELENG + 511) DIV 512;
8041  19   12:5    85                 SETBMST;
8042  19   12:5    87                 EXIT( STBCACH)
8043  19   12:4    91               END;
8044  19   12:2    91           END
8045  19   12:0    91       END;  (* STBCACH *)
8046  19   12:0   114
8047  19   12:0   114
8048  19   13:D     1     PROCEDURE P010D0D;
8049  19   13:D     1
8050  19   13:D     1       (* UNREACHABLE CODE (!?) *)
8051  19   13:D     1
8052  19   13:D     1       VAR
8053  19   13:D     1            MP01  : INTEGER;
8054  19   13:D     2            MP02  : PACKED ARRAY[ 0..511] OF CHAR;
8055  19   13:D   258            MP102 : PACKED ARRAY[ 0..511] OF CHAR;
8056  19   13:D   514
8057  19   13:0     0       BEGIN (* P010D0D *)
8058  19   13:1     0         P010D03( 'Testing Cache against Disk',
8059  19   13:1    29                  0, 0, 0, 0, 0, 0);
8060  19   13:1    37         FOR MP01 := 0 TO 1023 DO
8061  19   13:2    53           BEGIN
8062  19   13:3    53             IF BTSTCACH[ MP01] THEN
8063  19   13:4    64               BEGIN
8064  19   13:5    64                 CACHOFF;
8065  19   13:5    66                 UNITREAD( 4, MP102, 512, MP01);
8066  19   13:5    78                 CACHON;
8067  19   13:5    80                 UNITREAD( 4, MP02, 512, MP01);
8068  19   13:5    91                 IF MP102 <> MP02 THEN
8069  19   13:6   102                   P010D03( 'Cache <> Disk at block',
8070  19   13:6   127                            MP01, 0, 0, 0, 0, 0)
8071  19   13:4   133               END;
8072  19   13:2   135           END;
8073  19   13:2   142
8074  19   13:0   142       END;  (* P010D0D *)
8075  19   13:0   156
8076  19   13:0   156
8077  19   14:D     1     PROCEDURE FILLCACH;  (* P010D0E *)
8078  19   14:D     1
8079  19   14:D     1       VAR
8080  19   14:D     1            UNUSED   : INTEGER;
8081  19   14:D     2            UNUSED2  : INTEGER;
8082  19   14:D     3            VBLOCK   : INTEGER;
```

```
8083  19  14:D   4            THISDISK : INTEGER;
8084  19  14:D   5            MORE2DO  : BOOLEAN;
8085  19  14:D   6
8086  19  14:D   6
8087  19  15:D   3       FUNCTION GETDSKID( VBLOCK : INTEGER) : INTEGER;  (* P010D0F *)
8088  19  15:D   4
8089  19  15:D   4         (*
8090  19  15:D   4            THE UNITREAD() MAY CAUSE:
8091  19  15:D   4
8092  19  15:D   4              PLEASE INSERT SCENARIO DISKETTE %N
8093  19  15:D   4
8094  19  15:D   4         *)
8095  19  15:D   4
8096  19  15:D   4         VAR
8097  19  15:D   4              DISKID : INTEGER;  (* 1, 2, 3, 4, 5 *)
8098  19  15:D   5
8099  19  15:0   0         BEGIN (* GETDSKID *)
8100  19  15:1   0           IF NOT BRELOC THEN
8101  19  15:2   6             GETDSKID := 1
8102  19  15:1   6           ELSE
8103  19  15:2  11             BEGIN
8104  19  15:3  11               DISKID := 4;
8105  19  15:3  14               UNITREAD( BASE12, DISKID, 31, VBLOCK);
8106  19  15:3  24               GETDSKID := DISKID
8107  19  15:2  24             END
8108  19  15:0  27         END;  (* GETDSKID *)
8109  19  15:0  40
8110  19  15:0  40
8111  19  14:0   0       BEGIN (* FILLCACH *)
8112  19  14:1   0         FOR VBLOCK := 0 TO 1023 DO
8113  19  14:2  13           IF BMSTCACH[ VBLOCK] THEN
8114  19  14:3  24             IF GETDSKID( VBLOCK) = 0 THEN
8115  19  14:3  33
8116  19  14:3  33               (* NOT A VALID VIRTUAL DISK ADDRESS !? *)
8117  19  14:3  33
8118  19  14:4  33               BMSTCACH[ VBLOCK] := FALSE;
8119  19  14:4  50
8120  19  14:1  50         BTSTCACH := BMSTCACH;
8121  19  14:1  60
8122  19  14:1  60         THISDISK := 1;
8123  19  14:1  63
8124  19  14:1  63         REPEAT
8125  19  14:1  63
8126  19  14:2  63           MORE2DO := FALSE;
8127  19  14:2  66           VBLOCK := 0;
8128  19  14:2  69           WHILE VBLOCK < 1024 DO
8129  19  14:3  76             BEGIN
8130  19  14:4  76               IF BMSTCACH[ VBLOCK] THEN
8131  19  14:5  87                 BEGIN
8132  19  14:6  87                   IF GETDSKID( VBLOCK) = THISDISK THEN
8133  19  14:7  96                     BEGIN
8134  19  14:8  96                       SC.SEG.ADDR := VBLOCK;
8135  19  14:8 100                       BMSTCACH[ VBLOCK] := FALSE;
8136  19  14:8 110                       VBLOCK := VBLOCK + 1;
8137  19  14:8 115                       WHILE (VBLOCK < 1024) AND
8138  19  14:8 120                             (BMSTCACH[ VBLOCK]) AND
8139  19  14:8 130                             (GETDSKID( VBLOCK) = THISDISK) DO
```

```
8140  19   14:9   140                        BEGIN
8141  19   14:0   140                          BMSTCACH[ VBLOCK] := FALSE;
8142  19   14:0   150                          VBLOCK := VBLOCK + 1
8143  19   14:9   151                        END;
8144  19   14:9   157
8145  19   14:8   157                      VBLOCK := VBLOCK - 1;
8146  19   14:8   162                      SC.SEG.LENG := 1 + VBLOCK - SC.SEG.ADDR;
8147  19   14:8   172                      UNITWRITE( BASE12, SC.SEG.ADDR, 25, 0)
8148  19   14:7   183                    END
8149  19   14:6   183                  ELSE
8150  19   14:7   185                    BEGIN
8151  19   14:8   185                      MORE2DO := TRUE
8152  19   14:7   185                    END
8153  19   14:5   188                END;  (* IF BMSTCACH[ VBLOCK] *)
8154  19   14:5   188
8155  19   14:4   188              VBLOCK := VBLOCK + 1
8156  19   14:3   189            END; (* END WHILE *)
8157  19   14:3   195
8158  19   14:2   195          THISDISK := THISDISK + 1
8159  19   14:2   196
8160  19   14:1   196        UNTIL NOT MORE2DO
8161  19   14:1   200
8162  19   14:0   200      END;  (* FILLCACH *)
8163  19   14:0   224
8164  19   14:0   224
8165  19    2:0     0    BEGIN  (* DOCACHE2 *)
8166  19    2:1     0      BASE13 := 1;  (* SHOPS *)
8167  19    2:1     3      IF NOT BCACHE THEN
8168  19    2:2     9        EXIT( DOCACHE2);
8169  19    2:2    13
8170  19    2:1    13      CACHOFF;      (* TURN OFF LOOKING FOR CACHE, I.E, $E8 := 0 *)
8171  19    2:1    15
8172  19    2:1    15      SC.CACHE.ZERO := -3;  (* INIT $E9, $EA, $EB TO 0; $E8 := 1 *)
8173  19    2:1    19      UNITWRITE( BASE12, SC.CACHE.ZERO, 25, 0);
8174  19    2:1    29
8175  19    2:1    29      SC.CACHE.ZERO := 0;   (* GET AVAILABLE CACHE SIZE *)
8176  19    2:1    32      SC.CACHE.ONE  := 0;
8177  19    2:1    35      UNITWRITE( BASE12, SC.CACHE.ZERO, 25, 0);
8178  19    2:1    45      CACHSIZ := SC.CACHE.ZERO;
8179  19    2:1    49      IF CACHSIZ = 0 THEN
8180  19    2:2    56        EXIT( DOCACHE2);
8181  19    2:2    60
8182  19    2:1    60      CLEARWIN( MAINWIN, TRUE);
8183  19    2:1    66      MP04 := 60;
8184  19    2:1    69      GETMSGTX( BASE47, MP04);         (* NOW LOADING PROGRAM SEGMENTS *)
8185  19    2:1    75      WHILE  BASE47 <> '**ERR**' DO    (* INTO EXTRA MEMORY...         *)
8186  19    2:2    91        BEGIN
8187  19    2:3    91          CENTSTR( MAINWIN, BASE47);
8188  19    2:3    98          MP04 := MP04 + 1;
8189  19    2:3   103          GETMSGTX( BASE47, MP04)
8190  19    2:2   106        END;
8191  19    2:2   111
8192  19    2:1   111      FILLCHAR( DIR, 2028, CHR( 0));
8193  19    2:1   120      UNITREAD( DRIVE1, DIR, 2028, 2);
8194  19    2:1   131      SYSPASBL := GETFILEI( 4, 'SYSTEM.PASCAL');
8195  19    2:1   154      UNITREAD( DRIVE1, SEGDICT, SIZEOF( SEGDICT),
8196  19    2:1   162                DIR[ SYSPASBL].FIRSTBLK );  (* SYSTEM.PASCAL SEG DICTIONARY *)
```

```
8197  19   2:1   171
8198  19   2:1   171       (*  GET "SEGNAMES".                  *)
8199  19   2:1   171       (*  SET SEGNAME LENGTHS TO 8.        *)
8200  19   2:1   171       (*  CHANGE NULL OR ' ' TO BLANK CHAR *)
8201  19   2:1   171
8202  19   2:1   171       FILLCHAR( SEGNAMES, 160, CHR( 0));
8203  19   2:1   180
8204  19   2:1   180       FOR SEGI := 0 TO 15 DO
8205  19   2:2   194         BEGIN
8206  19   2:3   194           FOR NAMEX := 0 TO 7 DO
8207  19   2:4   208             SEGNAMES[ SEGI][ NAMEX + 1] :=  SEGDICT.SEGNAME[ SEGI][ NAMEX];
8208  19   2:3   232           SEGNAMES[ SEGI][ 0] := CHR( 8);
8209  19   2:3   240           FOR NAMEX := 1 TO 8 DO
8210  19   2:4   254             IF (SEGNAMES[ SEGI][ NAMEX] = CHR( 0) ) OR
8211  19   2:4   263                (SEGNAMES[ SEGI][ NAMEX] = ' ') THEN
8212  19   2:5   275               SEGNAMES[ SEGI][ NAMEX] := ' '
8213  19   2:2   281         END;
8214  19   2:2   297
8215  19   2:2   297       (* NOTE THE FOLLOWING ARE IN ORDER OF PRECEDENCE.          *)
8216  19   2:2   297       (* ONLY WHEN CACHE IS AVAILABLE ARE THE CACHE REQUESTS SET. *)
8217  19   2:2   297
8218  19   2:2   297       (* EACH CALL WILL "SET MASTER BCACHE" FOR EACH VIRTUAL BLOCK REQUESTED *)
8219  19   2:2   297
8220  19   2:1   297       FILLCHAR( BMSTCACH, SIZEOF( BMSTCACH), CHR( 0) );
8221  19   2:1   307       STBCACH( 'CASTASPE');
8222  19   2:1   320       STBCACH( 'SWINGASW');
8223  19   2:1   333       STBCACH( 'CINIT   ');
8224  19   2:1   346       STBCACH( 'CUTIL   ');
8225  19   2:1   359       STBCACH( 'COMBAT  ');
8226  19   2:1   372       STBCACH( 'RUNNER  ');
8227  19   2:1   385       STBCACH( 'CAMP    ');
8228  19   2:1   398
8229  19   2:1   398       CASE ORD( BASE75B[ 0]) OF
8230  19   2:1   405         0:  STBCACH2( 'ASCII.KRN');
8231  19   2:1   421         1:  STBCACH2( 'KANA.KRN' );
8232  19   2:1   436         2:  STBCACH2( 'KANJI.KRN' );
8233  19   2:1   452       END;
8234  19   2:1   466
8235  19   2:1   466       STBCACH( 'UTILITIE');
8236  19   2:1   479       STBCACH( 'SPECIALS');
8237  19   2:1   492
8238  19   2:1   492       STBCACH3( 4); (* SCENARIO.DATA RECORDS (?) *)
8239  19   2:1   495       STBCACH3( 2); (* SCENARIO.DATA RECORDS (?) *)
8240  19   2:1   498
8241  19   2:1   498       STBCACH( 'SHOPS   ' );
8242  19   2:1   511       STBCACH( 'KANJIREA' );
8243  19   2:1   524
8244  19   2:1   524       CASE ORD( BASE75B[ 1]) OF
8245  19   2:1   531         0:  STBCACH2( '200.MONSTERS');
8246  19   2:1   550         1:  STBCACH2( '400.MONSTERS');
8247  19   2:1   569       END;
8248  19   2:1   580
8249  19   2:1   580       STBCACH3( 5); (* SCENARIO.DATA RECORDS (?) *)
8250  19   2:1   583       FILLCACH;
8251  19   2:1   585       INITCACH;
8252  19   2:1   587
8253  19   2:0   587     END;   (* DOCACHE2 *)
```

```
8254  19    2:0   608
8255  19    2:0   608
8256  19    1:0     0   BEGIN  (* DOCACHE P010D01 *)
8257  19    1:0     0
8258  19    1:1     0     DOCACHE2;
8259  19    1:1     2     BASE09 := 0
8260  19    1:1     2
8261  19    1:0     2   END;
8262  19    1:0    18 (*$I WIZ4B:DOCACHE *)
8263  19    1:0    18
8264  19    1:0    18 (* ============ END  SEGMENTS ================= *)
8265  19    1:0    18
8266  19    1:0    18
8267   1    3:D     3 FUNCTION GETADDR;  (* ( VAR A:STRING) : INTEGER; *)  (* P010003 *)
8268   1    3:D     4
8269   1    3:D     4   VAR
8270   1    3:D     4         MP04 : ARRAY[ 0..0] OF INTEGER;
8271   1    3:D     5
8272   1    3:0     0   BEGIN
8273   1    3:1     0     GETADDR := MP04[ -1]
8274   1    3:0     6   END;
8275   1    3:0    22
8276   1    3:0    22
8277   1   56:D     1 PROCEDURE BEEPSPKR;  (* MP01 : INTEGER *)   (* P010038 *)
8278   1   56:D     2
8279   1   56:0     0   BEGIN (* P010038 *)
8280   1   56:1     0     UNITWRITE( BASE12, MP01, 24, MP01)
8281   1   56:0    10   END;  (* P010038 *)
8282   1   56:0    22
8283   1   56:0    22
8284   1   47:D     3 FUNCTION P01002F;  (* MP04 : TLONGSTR;
8285   1   47:D   261                       MP03 : TLONGSTR) : INTEGER   *)
8286   1   47:D   261
8287   1   47:D   261   VAR
8288   1   47:D   261        MP105 : INTEGER;
8289   1   47:D   262
8290   1   47:D   262
8291   1   60:D     1   PROCEDURE P01003C( MP01 : INTEGER);
8292   1   60:D     2
8293   1   60:0     0     BEGIN
8294   1   60:1     0       P01002F := MP01;
8295   1   60:1     4       EXIT( P01002F)
8296   1   60:0     8     END;
8297   1   60:0    20
8298   1   60:0    20
8299   1   61:D     1   PROCEDURE P01003D;
8300   1   61:D     1
8301   1   61:D     1     VAR
8302   1   61:D     1          MP01 : INTEGER;
8303   1   61:D     2          MP02 : INTEGER;
8304   1   61:D     3
8305   1   61:0     0     BEGIN
8306   1   61:1     0       MP01 := MP105 + 1;
8307   1   61:1     8       FOR MP02 := 2 TO LENGTH( MP04) DO
8308   1   61:2    24         BEGIN
8309   1   61:3    24           IF MP04[ MP02] <> MP03[ MP01] THEN
8310   1   61:4    38             EXIT( P01003D)
```

```
8311  1  61:3   42          ELSE
8312  1  61:4   44            MP01 := MP01 + 1
8313  1  61:2   45        END;
8314  1  61:1   56      P01003C( MP105)
8315  1  61:0   60    END;
8316  1  61:0   76
8317  1  61:0   76
8318  1  47:0    0  BEGIN (* P01002F *)
8319  1  47:1    0    IF BSTROPS THEN
8320  1  47:2   16      BEGIN
8321  1  47:3   16        UNITWRITE( BASE12, MP105, 29, GETADDR( MP04), GETADDR( MP03));
8322  1  47:3   38        P01002F := MP105;
8323  1  47:3   43        EXIT( P01002F)
8324  1  47:2   47      END;
8325  1  47:2   47
8326  1  47:2   47    (* NEVER EXECUTED *)
8327  1  47:2   47
8328  1  47:1   47    IF (LENGTH( MP04) = 0) OR (LENGTH( MP03) = 0) THEN
8329  1  47:2   63      P01003C( 0);
8330  1  47:1   66      FOR MP105 := 1 TO LENGTH( MP03) - LENGTH( MP04) + 1 DO
8331  1  47:2   94        IF MP03[ MP105] = MP04[ 1] THEN
8332  1  47:3  108          P01003D;
8333  1  47:1  120      P01003C( 0)
8334  1  47:1  121
8335  1  47:0  121  END;  (* P01002F *)
8336  1  47:0  138
8337  1  47:0  138
8338  1  48:D    3  FUNCTION POS;  (* SUBSTR MP04 : TLONGSTR;         P010030
8339  1  48:D  261                    BIGSTR MP03 : TLONGSTR) : INTEGER *)
8340  1  48:D  261
8341  1  48:D  261           (* POS() STRING FUNCTION *)
8342  1  48:D  261
8343  1  48:D  261    VAR
8344  1  48:D  261         MP105 : INTEGER;
8345  1  48:D  262         POSFND : INTEGER;
8346  1  48:D  263
8347  1  48:D  263         MPXXXX : ARRAY[ 0..3] OF INTEGER;
8348  1  48:D  267
8349  1  48:0    0    BEGIN
8350  1  48:1    0      IF BSTROPS THEN
8351  1  48:2   16        BEGIN
8352  1  48:3   16          UNITWRITE( BASE12, POSFND, 29,
8353  1  48:3   23                     GETADDR( SUBSTR), GETADDR( BIGSTR));
8354  1  48:3   38          POS := POSFND;
8355  1  48:3   43          EXIT( POS)
8356  1  48:2   47        END;
8357  1  48:2   47
8358  1  48:2   47      (* CODE BELOW HERE NEVER EXECUTED! *)
8359  1  48:2   47
8360  1  48:1   47      MP105 := 0;
8361  1  48:1   51
8362  1  48:1   51      REPEAT
8363  1  48:2   51        POSFND := P01002F( SUBSTR, BIGSTR);
8364  1  48:2   63        IF POSFND <> 0 THEN
8365  1  48:3   70          BEGIN
8366  1  48:4   70            IF POSFND > 1 THEN
8367  1  48:5   77              BEGIN
```

```
8368  1  48:6   77                IF (BIGSTR[ POSFND - 1] >= CHR( 129)) AND
8369  1  48:6   89                   (BIGSTR[ POSFND - 1] <= CHR( 159))   THEN
8370  1  48:7  104                     BEGIN
8371  1  48:8  104                       COPYSTR( BIGSTR, BIGSTR, (POSFND + 1),
8372  1  48:8  113                                LENGTH( BIGSTR) - POSFND);
8373  1  48:8  123                       MP105 := MP105 + POSFND;
8374  1  48:8  133                       POSFND := -1;
8375  1  48:7  138                     END;
8376  1  48:5  138                END;
8377  1  48:5  138
8378  1  48:4  138             IF POSFND > 0 THEN
8379  1  48:5  145               BEGIN
8380  1  48:6  145                 POS := POSFND + MP105;
8381  1  48:6  154                 EXIT( POS)
8382  1  48:5  158               END;
8383  1  48:3  158             END;
8384  1  48:3  158
8385  1  48:1  158         UNTIL POSFND = 0;
8386  1  48:1  165
8387  1  48:1  165         POS := 0;
8388  1  48:1  168
8389  1  48:0  168       END;
8390  1  48:0  182
8391  1  48:0  182
8392  1  48:0  182
8392  1  48:0  182 (*$I WIZ4A: WIZ2.TEXT *)
8393  1  48:0  182
8394  1  48:0  182
8395  1  48:0  182 (* WIZ2 *)
8396  1  48:0  182
8397  1  48:0  182
8398  1  46:D    1   PROCEDURE COPYSTR;  (*  VAR DEST:  TLONGSTR;    P01002E
8399  1  46:D  133                               SRC:   TLONGSTR;
8400  1  46:D  133                               FIRST: INTEGER;
8401  1  46:D  133                               COUNT: INTEGER *)
8402  1  46:D  133
8403  1  46:0    0     BEGIN
8404  1  46:1    0       IF (FIRST + COUNT) - 1 > LENGTH( SRC) THEN
8405  1  46:2   17         COUNT := LENGTH( SRC) - FIRST + 1;
8406  1  46:1   27       IF COUNT < 0 THEN
8407  1  46:2   32         COUNT := 0;
8408  1  46:1   35       DEST[ 0] := CHR( COUNT);
8409  1  46:1   39       IF COUNT > 0 THEN
8410  1  46:2   44         MOVELEFT( SRC[  FIRST], DEST[ 1], COUNT)
8411  1  46:0   52     END;
8412  1  46:0   64
8413  1  46:0   64
8414  1  43:D    1   PROCEDURE PCONCAT;  (*  (VAR A: STRING;  B: STRING)  *)
8415  1  43:D  131
8416  1  43:0    0     BEGIN
8417  1  43:1    0       IF BSTROPS THEN
8418  1  43:2   10         BEGIN
8419  1  43:3   10           UNITWRITE( BASE12, A, 26, GETADDR( B) );
8420  1  43:3   24           EXIT( PCONCAT)
8421  1  43:2   28         END;
8422  1  43:1   28       MOVELEFT( B[1], A[ LENGTH( A) + 1], LENGTH( B));
8423  1  43:1   43       A[ 0] := CHR( LENGTH( A) + LENGTH( B))
```

```
8424  1  43:0   53    END;
8425  1  43:0   66
8426  1  43:0   66
8427  1  44:D    1  PROCEDURE PCONCAT2;  (* VAR MP03 : TLONGSTR;      P01002C
8428  1  44:D  260                             MP02 : TLONGSTR;
8429  1  44:D  260                             MP01 : TLONGSTR)  *)
8430  1  44:D  260
8431  1  44:0    0    BEGIN
8432  1  44:1    0      IF BSTROPS THEN
8433  1  44:2   16        BEGIN
8434  1  44:3   16          UNITWRITE( BASE12, MP03, 26, GETADDR( MP02), GETADDR( MP01));
8435  1  44:3   36          EXIT( PCONCAT2)
8436  1  44:2   40        END;
8437  1  44:2   40
8438  1  44:2   40      (* CODE BELOW HERE NEVER EXECUTED *)
8439  1  44:2   40
8440  1  44:1   40      PCONCAT( MP03, MP02);
8441  1  44:1   46      PCONCAT( MP03, MP01)
8442  1  44:0   49    END;
8443  1  44:0   64
8444  1  44:0   64
8445  1  45:D    1  PROCEDURE PCONCAT3;  (* VAR MP04 : TLONGSTR;          P01002D
8446  1  45:D  389                             MP03 : TLONGSTR;
8447  1  45:D  389                             MP02 : TLONGSTR;
8448  1  45:D  389                             MP01 : TLONGSTR)  *)
8449  1  45:D  389
8450  1  45:0    0    BEGIN  (* PCONCAT3 *)
8451  1  45:1    0      IF BSTROPS THEN
8452  1  45:2   22        BEGIN
8453  1  45:3   22          UNITWRITE( BASE12, MP04, 26, GETADDR( MP03), GETADDR( MP02));
8454  1  45:3   43          UNITWRITE( BASE12, MP04, 26, GETADDR( MP01));
8455  1  45:3   57          EXIT( PCONCAT3)
8456  1  45:2   61        END;
8457  1  45:2   61
8458  1  45:2   61      (* NEVER EXECUTED *)
8459  1  45:2   61
8460  1  45:1   61      PCONCAT( MP04, MP03);
8461  1  45:1   67      PCONCAT( MP04, MP02);
8462  1  45:1   73      PCONCAT( MP04, MP01)
8463  1  45:1   76
8464  1  45:0   76    END;  (* PCONCAT3 *)
8465  1  45:0   90
8466  1  45:0   90
8467  1  49:D    1  PROCEDURE INT2STR;  (* (     ININT  : INTEGER;       P010031
8468  1  49:D    3                           VAR OUTSTR : TLONGSTR); *)
8469  1  49:D    3
8470  1  49:D    3    VAR
8471  1  49:D    3        UNUSED : INTEGER;
8472  1  49:D    4        MP04 : INTEGER;
8473  1  49:D    5        OUTI : INTEGER;
8474  1  49:D    6        NEGATIVE : BOOLEAN;
8475  1  49:D    7        MP07 : INTEGER;
8476  1  49:D    8
8477  1  49:0    0    BEGIN (* P010031 *)
8478  1  49:1    0      NEGATIVE := ININT < 0;
8479  1  49:1    5      ININT := ABS( ININT);
8480  1  49:1    9      OUTI := 1;
```

```
8481  1  49:1   12      OUTSTR[ 1] := '0';
8482  1  49:1   16
8483  1  49:1   16      (* FIRST BUILD THE STRING IN REVERSE ORDER *)
8484  1  49:1   16
8485  1  49:1   16      WHILE ININT <> 0 DO
8486  1  49:2   21        BEGIN
8487  1  49:3   21          OUTSTR[  OUTI] :=  CHR( ORD( '0') + (ININT MOD 10));
8488  1  49:3   29          ININT := ININT DIV 10;
8489  1  49:3   34          OUTI := OUTI + 1
8490  1  49:2   35        END;
8491  1  49:1   41      IF OUTI = 1 THEN
8492  1  49:2   46        OUTI := 2;
8493  1  49:1   49      IF NEGATIVE THEN
8494  1  49:2   52        BEGIN
8495  1  49:3   52          OUTSTR[ OUTI] := '-';
8496  1  49:3   56          OUTI := OUTI + 1
8497  1  49:2   57        END;
8498  1  49:2   61
8499  1  49:2   61      (* NOW REVERSE THE ORDER TO NORMAL *)
8500  1  49:2   61
8501  1  49:1   61      MP04 := 1;
8502  1  49:1   64      MP07 := OUTI DIV 2;
8503  1  49:1   69      WHILE MP04 <= MP07 DO
8504  1  49:2   74        BEGIN
8505  1  49:3   74          OUTSTR[ 0] := OUTSTR[ MP04];
8506  1  49:3   80          OUTSTR[ MP04] := OUTSTR[ OUTI - MP04];
8507  1  49:3   88          OUTSTR[ OUTI - MP04] := OUTSTR[ 0];
8508  1  49:3   96          MP04 := MP04 + 1
8509  1  49:2   97        END;
8510  1  49:1  103      OUTSTR[ 0] := CHR( OUTI - 1)  (* STRING LENGTH *)
8511  1  49:0  108    END;  (* P010031 *)
8512  1  49:0  126
8513  1  49:0  126
8514  1  50:D    1  PROCEDURE P010032;  (*     MP02 : TWIZLONG;
8515  1  50:D    6                           VAR MP01 : TLONGSTR)  *)
8516  1  50:D    6
8517  1  50:D    6    VAR
8518  1  50:D    6      (* MP03 == MP02.LOW
8519  1  50:D    6         MP04 == MP02.MID
8520  1  50:D    6         MP05 == MP02.HIGH *)
8521  1  50:D    6
8522  1  50:D    6         MP06 : STRING[ 10];
8523  1  50:D   12         MP0C : STRING[ 10];
8524  1  50:D   18
8525  1  50:D   18
8526  1  62:D    1    PROCEDURE  P01003E(      ININT : INTEGER;
8527  1  62:D    2                         VAR OUTSTR : TLONGSTR);
8528  1  62:D    3
8529  1  62:D    3      VAR
8530  1  62:D    3           MP03 : STRING[ 10];
8531  1  62:D    9
8532  1  62:0    0      BEGIN (* P01003E *)
8533  1  62:1    0        INT2STR( ININT, OUTSTR);
8534  1  62:1    4        WHILE LENGTH( OUTSTR) < 4 DO
8535  1  62:2   11          BEGIN
8536  1  62:3   11            MP03 := '0';
8537  1  62:3   16            PCONCAT( MP03, OUTSTR);
```

```
8538  1  62:3   21            OUTSTR := MP03
8539  1  62:2   22          END
8540  1  62:0   26      END;  (* P01003E *)
8541  1  62:0   42
8542  1  62:0   42
8543  1  50:0    0  BEGIN (* P010032 *)
8544  1  50:1    0    P01003E( MP02.HIGH, MP01);
8545  1  50:1    9    P01003E( MP02.MID,  MP0C);
8546  1  50:1   14    P01003E( MP02.LOW,  MP06);
8547  1  50:1   19    PCONCAT2( MP01, MP0C, MP06);
8548  1  50:1   26    WHILE (LENGTH( MP01) > 1) AND (MP01[ 1] = '0') DO
8549  1  50:2   39      COPYSTR( MP01, MP01, 2, LENGTH( MP01) - 1)
8550  1  50:2   47
8551  1  50:0   47  END;  (* P010032 *)
8552  1  50:0   66
8553  1  50:0   66
8554  1  63:D    3  FUNCTION SEARCHBF( VAR MP05 : TCACHEP;              (* P01003F *)
8555  1  63:D    4                    VAR MP04 : TCACHEP;
8556  1  63:D    5                        MP03 : INTEGER) : TCACHEP; FORWARD;
8557  1  63:D    6
8558  1  63:D    6
8559  1  64:D    1  PROCEDURE READCAC( VAR MP03 : TBUFFER;              (* P010040 *)
8560  1  64:D    2                        MP02 : INTEGER;
8561  1  64:D    3                        MP01 : BOOLEAN); FORWARD;
8562  1  64:D    4
8563  1  65:D    1  PROCEDURE WRITECAC( VAR MP03 : TBUFFER;              (* P010041 *)
8564  1  65:D    2                         MP02 : INTEGER;
8565  1  65:D    3                         MP01 : BOOLEAN); FORWARD;
8566  1  65:D    4
8567  1  66:D    1  PROCEDURE DRAWSCR( SHOW: BOOLEAN); FORWARD;          (* P010042 *)
8568  1  66:D    2
8569  1  66:D    2
8570  1  66:D    2
8571  1  66:D    2
8572  1  66:D    2
8573  1  51:D    3  FUNCTION GTLNGSUB;  (* MENUSTR : TLONGSTR) : INTEGER      (* P010033 *)
8574  1  51:D  132
8575  1  51:D  132    (* RETURN LENGTH OF LONGEST SUBSTRING BETWEEN '/' IN MENU STRING.  *)
8576  1  51:D  132    (* COULD BE FIRST PART, LAST PART, OR WHOLE MENU STRING.           *)
8577  1  51:D  132
8578  1  51:D  132    VAR
8579  1  51:D  132         SLASHPOS : INTEGER;
8580  1  51:D  133         LONGEST  : INTEGER;
8581  1  51:D  134         SUBSTRNG : TLONGSTR;
8582  1  51:D  262
8583  1  51:0    0    BEGIN
8584  1  51:1    0      LONGEST := 0;
8585  1  51:1    9      SLASHPOS := POS( '/', MENUSTR);
8586  1  51:1   19      WHILE SLASHPOS <> 0 DO
8587  1  51:2   26        BEGIN
8588  1  51:3   26          COPYSTR( SUBSTRNG, MENUSTR, 1, (SLASHPOS - 1));
8589  1  51:3   39          LONGEST := GREATER( LONGEST, MBSTRLEN( SUBSTRNG));
8590  1  51:3   56          COPYSTR( MENUSTR, MENUSTR, (SLASHPOS + 1),
8591  1  51:3   65                   (LENGTH( MENUSTR) - SLASHPOS));
8592  1  51:3   75          SLASHPOS := POS( '/', MENUSTR)
8593  1  51:2   78        END;
8594  1  51:1   87      LONGEST := GREATER( LONGEST, MBSTRLEN( MENUSTR));
```

```
8595  1  51:1  103      GTLNGSUB := LONGEST
8596  1  51:0  103    END;
8597  1  51:0  122
8598  1  51:0  122
8599  1  52:D    3  FUNCTION GREATER; (* NUM1 : INTEGER;               P010034
8600  1  52:D    5                       NUM2 : INTEGER) : INTEGER; *)
8601  1  52:D    5
8602  1  52:0    0    BEGIN
8603  1  52:1    0      IF NUM1 > NUM2 THEN
8604  1  52:2    5        GREATER := NUM1
8605  1  52:1    5      ELSE
8606  1  52:2   10        GREATER := NUM2
8607  1  52:0   10    END;
8608  1  52:0   26
8609  1  52:0   26
8610  1  53:D    3  FUNCTION GETWIN2;  (* MBHORSIZ : INTEGER;       P010035
8611  1  53:D    8                        VPOS     : INTEGER;
8612  1  53:D    8                        VSIZE    : INTEGER;
8613  1  53:D    8                        PRIORITY : INTEGER;
8614  1  53:D    8                        SHOW     : BOOLEAN) : TWINDOWP;  *)
8615  1  53:D    8
8616  1  53:0    0    BEGIN
8617  1  53:1    0      MBHORSIZ := (MBHORSIZ + 2) DIV 2;
8618  1  53:1    7      GETWIN2 := GETWIN( (19 - MBHORSIZ),   (* CENTERED *)
8619  1  53:1   10                         VPOS,
8620  1  53:1   11                         (MBHORSIZ + MBHORSIZ + 2),
8621  1  53:1   16                         VSIZE,
8622  1  53:1   17                         PRIORITY,
8623  1  53:1   18                         SHOW)
8624  1  53:0   19    END;
8625  1  53:0   38
8626  1  53:0   38
8627  1  54:D    1  PROCEDURE INSERT2; (* VAR INSERTTX : TLONGSTR;   P010036
8628  1  54:D    5                        VAR WHOLESTR : TLONGSTR;
8629  1  54:D    5                            INSERTCH : TCHARSTR)  *)
8630  1  54:D    5
8631  1  54:D    5    VAR
8632  1  54:D    5         UNUSED   : INTEGER;
8633  1  54:D    6         FNDPOS   : INTEGER;
8634  1  54:D    7         BUILDSTR : TLONGSTR;
8635  1  54:D  135
8636  1  54:0    0    BEGIN
8637  1  54:1    0      FNDPOS := POS( INSERTCH, WHOLESTR);  (* FIND INSERTCH IN WHOLESTR *)
8638  1  54:1   14      IF FNDPOS > 0 THEN
8639  1  54:2   19        BEGIN
8640  1  54:2   19          (* COPY FIRST PART OF WHOLESTR TO BUILDSTR *)
8641  1  54:3   19          COPYSTR( BUILDSTR, WHOLESTR, 1, FNDPOS - 1);
8642  1  54:3   28
8643  1  54:3   28          (* COPY LAST PART OF WHOLESTR TO FIRST PART OF WHOLESTR *)
8644  1  54:3   28          COPYSTR( WHOLESTR, WHOLESTR, (FNDPOS + 1), LENGTH( WHOLESTR) - 1);
8645  1  54:3   40
8646  1  54:3   40          PCONCAT2( BUILDSTR, INSERTTX, WHOLESTR);
8647  1  54:3   46          WHOLESTR := BUILDSTR;
8648  1  54:2   51        END;
8649  1  54:0   51    END;
8650  1  54:0   64
8651  1  54:0   64
```

```
8652  1  55:D   1  PROCEDURE INSERTST; (* VAR INSERTTX : TLONGSTR;  P010037
8653  1  55:D   3                        VAR WHOLESTR : TLONGSTR  *)
8654  1  55:D   3
8655  1  55:0   0    BEGIN
8656  1  55:1   0      INSERT2( INSERTTX, WHOLESTR, '^')
8657  1  55:0   3    END;
8658  1  55:0  18
8659  1  55:0  18
8660  1  63:D   3  FUNCTION SEARCHBF; (* ( VAR MP05 : TCACHEP;         P01003F
8661  1  63:D   6                      VAR MP04 : TCACHEP;
8662  1  63:D   6                          MP03 : INTEGER) : TCACHEP ;         *)
8663  1  63:D   6
8664  1  63:D   6    VAR
8665  1  63:D   6        MP06 : ARRAY[ 0..1] OF TCACHEP;
8666  1  63:D   8
8667  1  63:D   8
8668  1  63:0   0    BEGIN
8669  1  63:1   0      MP06[ 0] := MP05;
8670  1  63:1   8      MP06[ 1] := MP04;
8671  1  63:1  16      UNITREAD( BASE12, MP06, 16, MP03);
8672  1  63:1  26      MP05 := MP06[ 0];
8673  1  63:1  34      MP04 := MP06[ 1];
8674  1  63:1  42      SEARCHBF := MP05
8675  1  63:0  42    END;
8676  1  63:0  58
8677  1  63:0  58  (* WIZ1.TEXT *)
8678  1  63:0  58 (*$I WIZ4A: WIZ2.TEXT *)
8679  1  63:0  58
8679  1  63:0  58 (*$I WIZ4A: WIZ3.TEXT *)
8680  1  63:0  58 (* WIZ3.TEXT *)
8681  1  63:0  58
8682  1  64:D   1  PROCEDURE READCAC; (* ( VAR MP03 : TBUFFER;     P010040
8683  1  64:D   4                        MP02 : INTEGER;
8684  1  64:D   4                        MP01 : BOOLEAN);             *)
8685  1  64:D   4
8686  1  64:D   4    VAR
8687  1  64:D   4        MP04 : TCACHEP;
8688  1  64:D   5
8689  1  64:0   0    BEGIN
8690  1  64:1   0      MP04 := SEARCHBF( DSKCACHF, DSKCACHL, MP02);
8691  1  64:1  11      IF MP04^.KEY <> MP02 THEN
8692  1  64:2  17        BEGIN
8693  1  64:3  17          UNITREAD( DRIVE1, MP04^.DATA, 512, MP02);
8694  1  64:3  29          MP04^.KEY := MP02
8695  1  64:2  30        END;
8696  1  64:1  32      MOVELEFT( MP04^.DATA, MP03, 512);
8697  1  64:1  43      IF MP01 THEN
8698  1  64:2  46        EXIT( READCAC);
8699  1  64:1  50      MP02 := MP02 + 1;
8700  1  64:1  55      MP04 := SEARCHBF( DSKCACHF, DSKCACHL, MP02);
8701  1  64:1  66      IF MP04^.KEY <> MP02 THEN
8702  1  64:2  72        BEGIN
8703  1  64:3  72          UNITREAD( DRIVE1, MP04^.DATA, 512, MP02);
8704  1  64:3  84          MP04^.KEY := MP02
8705  1  64:2  85        END;
8706  1  64:1  87      MOVELEFT( MP04^.DATA, MP03[ 512], 512)
8707  1  64:0 100    END;
```

```
8708  1  64:0  112
8709  1  64:0  112
8710  1  65:D    1  PROCEDURE WRITECAC; (* ( VAR MP03 : TBUFFER;   P010041
8711  1  65:D    4                         MP02 : INTEGER;
8712  1  65:D    4                         MP01 : BOOLEAN);       *)
8713  1  65:D    4
8714  1  65:D    4    VAR
8715  1  65:D    4         MP04 : TCACHEP;
8716  1  65:D    5
8717  1  65:D    5    (* SEE READCAC ALSO *)
8718  1  65:D    5
8719  1  65:0    0    BEGIN
8720  1  65:1    0      IF MP01 THEN
8721  1  65:2    3        UNITWRITE( DRIVE1, MP03, 512, MP02)
8722  1  65:1   13      ELSE
8723  1  65:2   15        UNITWRITE( DRIVE1, MP03, 1024, MP02);
8724  1  65:2   25
8725  1  65:1   25      MP04 := SEARCHBF( DSKCACHF, DSKCACHL, MP02);
8726  1  65:1   36      MP04^.KEY := MP02;
8727  1  65:1   39      MOVELEFT( MP03, MP04^.DATA, 512);
8728  1  65:1   50      IF MP01 THEN
8729  1  65:2   53        EXIT( WRITECAC);
8730  1  65:1   57      MP02 := MP02 + 1;
8731  1  65:1   62      MP04 := SEARCHBF( DSKCACHF, DSKCACHL, MP02);
8732  1  65:1   73      MP04^.KEY := MP02;
8733  1  65:1   76      MOVELEFT( MP03[ 512], MP04^.DATA, 512)
8734  1  65:0   89    END;
8735  1  65:0  102
8736  1  65:0  102
8737  1  32:D    3  FUNCTION RAND;  (* P010020; *)
8738  1  32:D    3
8739  1  32:D    3    VAR
8740  1  32:D    3         MP03 : INTEGER;
8741  1  32:D    4
8742  1  32:0    0    BEGIN
8743  1  32:1    0      UNITREAD( BASE12, MP03, 10, 0);
8744  1  32:1   10      RAND := MP03
8745  1  32:0   10    END;
8746  1  32:0   26
8747  1  32:0   26
8748  1  31:D    3  FUNCTION KEYPRESS;  (* : BOOLEAN *)          (* P01001F *)
8749  1  31:D    3
8750  1  31:D    3    (* CALLED CHKKEYBD IN WIZARDRY III *)
8751  1  31:D    3
8752  1  31:D    3    VAR
8753  1  31:D    3         PRESSED : INTEGER;
8754  1  31:D    4
8755  1  31:0    0    BEGIN
8756  1  31:1    0      UNITREAD( BASE12, PRESSED, 11, 0);
8757  1  31:1   10      KEYPRESS := (PRESSED = 1)
8758  1  31:0   13    END;
8759  1  31:0   28
8760  1  31:0   28
8761  1   2:D    1  PROCEDURE PRINTBEL;  (* P010002 *)
8762  1   2:D    1
8763  1   2:0    0    BEGIN
8764  1   2:1    0      BEEPSPKR( 0)
```

```
8765  1   2:0    1    END;
8766  1   2:0   16
8767  1   2:0   16
8768  1  35:D    1  PROCEDURE DRAWSCR2;  (* **** UNUSED **** : TWINDOWP *)     (* P010023 *)
8769  1  35:D    2
8770  1  35:D    2    (* NOTE THE SMALL "BUG" WITH THE UNUSED MP01 PARAMETER.
8771  1  35:D    2       THE CODE ACTUALLY DOES PASS WINCHAIN SO IT EXECUTES PROPERLY. *)
8772  1  35:D    2
8773  1  35:0    0    BEGIN
8774  1  35:1    0      UNITWRITE( BASE12, WINCHAIN^, 2, 0)
8775  1  35:0    9    END;
8776  1  35:0   22
8777  1  35:0   22
8778  1  66:D    1  PROCEDURE DRAWSCR;  (* SHOW: BOOLEAN; *)                   (* P010042 *)
8779  1  66:D    2
8780  1  66:0    0    BEGIN
8781  1  66:1    0      IF SHOW THEN
8782  1  66:2    3        UNITWRITE( BASE12, WINCHAIN^, 2, 0)
8783  1  66:0   12    END;
8784  1  66:0   24
8785  1  66:0   24
8786  1  67:D    1  PROCEDURE PRIORDER;              (* P010043 *)
8787  1  67:D    1
8788  1  67:D    1    (* ORDER WINDOWS FROM HIGHER PRIORITY # TO LOWER PRIORITY # *)
8789  1  67:D    1
8790  1  67:D    1    VAR
8791  1  67:D    1         NEXT    : TWINDOWP;
8792  1  67:D    2         CURRENT : TWINDOWP;
8793  1  67:D    3         PREV    : TWINDOWP;
8794  1  67:D    4
8795  1  67:0    0    BEGIN
8796  1  67:1    0      PREV := NIL;
8797  1  67:1    3      CURRENT := WINCHAIN;
8798  1  67:1    6      WHILE CURRENT <> NIL DO
8799  1  67:2   11        BEGIN
8800  1  67:3   11          NEXT := CURRENT^.HEAD.NEXTW;
8801  1  67:3   15          IF NEXT <> NIL THEN
8802  1  67:4   20            BEGIN
8803  1  67:5   20              IF (CURRENT^.HEAD.PRIORITY MOD 128) <
8804  1  67:5   30                 (NEXT^.HEAD.PRIORITY MOD 128) THEN
8805  1  67:6   43                BEGIN
8806  1  67:7   43                  IF PREV = NIL THEN
8807  1  67:8   48                    WINCHAIN := NEXT
8808  1  67:7   48                  ELSE
8809  1  67:8   53                    PREV^.HEAD.NEXTW := NEXT;
8810  1  67:7   56                  CURRENT^.HEAD.NEXTW := NEXT^.HEAD.NEXTW;
8811  1  67:7   60                  NEXT^.HEAD.NEXTW := CURRENT;
8812  1  67:7   63                  CURRENT := NEXT
8813  1  67:6   63                END;
8814  1  67:4   66            END;
8815  1  67:3   66          PREV := CURRENT;
8816  1  67:3   69          CURRENT := CURRENT^.HEAD.NEXTW
8817  1  67:2   70        END
8818  1  67:0   73    END;
8819  1  67:0   90
8820  1  67:0   90
8821  1  30:D    3  FUNCTION P01001E;  (* MP03 : INTEGER) : INTEGER) *)
```

```
8822  1  30:D   4
8823  1  30:D   4    VAR
8824  1  30:D   4         MP04 : INTEGER;
8825  1  30:D   5
8826  1  30:0   0    BEGIN
8827  1  30:1   0      REPEAT
8828  1  30:2   0        GETKEY;
8829  1  30:2   2        IF INCHAR = CHR( CRETURN) THEN
8830  1  30:3   7          INCHAR := '0';
8831  1  30:2  10        MP04 := ORD( INCHAR) - ORD( '0');
8832  1  30:2  15        IF (MP04 >= 0) AND (MP04 <= MP03) THEN
8833  1  30:3  24          BEGIN
8834  1  30:4  24            P01001E := MP04;
8835  1  30:4  27            EXIT( P01001E)
8836  1  30:3  31          END
8837  1  30:2  31        ELSE
8838  1  30:3  33          PRINTBEL
8839  1  30:1  33      UNTIL FALSE
8840  1  30:1  35
8841  1  30:0  35    END;
8842  1  30:0  52
8843  1  30:0  52
8844  1   4:D   3  FUNCTION GETREC;   (* DATATYPE : INTEGER;       P010004
8845  1   4:D   6                       DATAINDX : INTEGER;
8846  1   4:D   6                       DATASIZE : INTEGER : INTEGER *)
8847  1   4:D   6
8848  1   4:D   6    VAR
8849  1   4:D   6        MP06 : INTEGER;
8850  1   4:D   7        RECSIZE : INTEGER;
8851  1   4:D   8        MP08 : INTEGER;
8852  1   4:D   9        BUFFADDR : INTEGER;
8853  1   4:D  10        DSKBLOCK : INTEGER;
8854  1   4:D  11        NEED1BLK : BOOLEAN;  (* RECORD CROSS BLOCK BOUNDARY? *)
8855  1   4:D  12
8856  1   4:0   0    BEGIN
8857  1   4:1   0      DSKBLOCK := SCENBLK + SCENARIO.BLOFF[ DATATYPE];
8858  1   4:1  12      BUFFADDR := 0;
8859  1   4:1  15      RECSIZE := SCENARIO.RECSIZE[ DATATYPE];
8860  1   4:1  24      MP06 := 30000 DIV RECSIZE;
8861  1   4:1  31      WHILE DATAINDX > 0 DO
8862  1   4:2  36        BEGIN
8863  1   4:3  36          IF DATAINDX > MP06 THEN
8864  1   4:4  41            MP08 := MP06
8865  1   4:3  41          ELSE
8866  1   4:4  46            MP08 := DATAINDX;
8867  1   4:3  49          DATAINDX := DATAINDX - MP08;
8868  1   4:3  54          BUFFADDR := BUFFADDR + MP08 * RECSIZE;
8869  1   4:3  61          DSKBLOCK := DSKBLOCK + BUFFADDR DIV 512;
8870  1   4:3  70          BUFFADDR := BUFFADDR MOD 512
8871  1   4:2  71        END;
8872  1   4:2  79
8873  1   4:1  79      NEED1BLK := (BUFFADDR + RECSIZE) <= 512;
8874  1   4:1  88      IF (CACHEBL <> DSKBLOCK) OR (ONECACBL AND NOT NEED1BLK) THEN
8875  1   4:2 100        BEGIN
8876  1   4:3 100          IF CACHEWRI THEN
8877  1   4:4 104            WRITECAC( BASE55B, CACHEBL, ONECACBL);
8878  1   4:3 113          CACHEWRI := FALSE;
```

```
8879  1   4:3  116          CACHEBL := DSKBLOCK;
8880  1   4:3  119          ONECACBL := NEED1BLK;
8881  1   4:3  122          READCAC( BASE55B, CACHEBL, ONECACBL)
8882  1   4:2  129        END;
8883  1   4:2  131
8884  1   4:1  131      GETREC := BUFFADDR
8885  1   4:0  131    END;
8886  1   4:0  148
8887  1   4:0  148
8888  1   5:D    3  FUNCTION GETRECW;  (* DATATYPE : INTEGER;
8889  1   5:D    6                        DATAINDX : INTEGER;
8890  1   5:D    6                        DATASIZE : INTEGER) : INTEGER *)
8891  1   5:D    6
8892  1   5:0    0    BEGIN
8893  1   5:1    0      GETRECW := GETREC( DATATYPE, DATAINDX, DATASIZE);
8894  1   5:1    9      CACHEWRI := TRUE
8895  1   5:0    9    END;
8896  1   5:0   24
8897  1   5:0   24
8898  1  68:D    1  PROCEDURE OVERUNDR( VAR FIRST  : INTEGER;    (* P010044 *)
8899  1  68:D    2                      VAR SECOND : INTEGER);
8900  1  68:D    3
8901  1  68:0    0    BEGIN
8902  1  68:1    0      IF FIRST < 0 THEN
8903  1  68:2    6        BEGIN
8904  1  68:3    6          FIRST := FIRST + 10000;
8905  1  68:3   14          SECOND := SECOND - 1
8906  1  68:2   17        END
8907  1  68:1   20      ELSE
8908  1  68:2   22        IF FIRST >= 10000 THEN
8909  1  68:3   30          BEGIN
8910  1  68:4   30            FIRST := FIRST - 10000;
8911  1  68:4   38            SECOND := SECOND + 1
8912  1  68:3   41          END
8913  1  68:0   44    END;
8914  1  68:0   56
8915  1  68:0   56
8916  1   6:D    1  PROCEDURE ADDLONGS;  (*  FIRST  : TWIZLONG;     P010006
8917  1   6:D    3                          SECOND : TWIZLONG);  *)
8918  1   6:D    3
8919  1   6:0    0    BEGIN
8920  1   6:1    0      FIRST.LOW := FIRST.LOW + SECOND.LOW;
8921  1   6:1    7      OVERUNDR( FIRST.LOW, FIRST.MID);
8922  1   6:1   13
8923  1   6:1   13      FIRST.MID := FIRST.MID + SECOND.MID;
8924  1   6:1   22      OVERUNDR( FIRST.MID, FIRST.HIGH);
8925  1   6:1   30
8926  1   6:1   30      FIRST.HIGH := FIRST.HIGH + SECOND.HIGH
8927  1   6:0   36    END;
8928  1   6:0   52
8929  1   6:0   52
8930  1   7:D    1  PROCEDURE SUBLONGS;   (*  FIRST  : TWIZLONG           P010007;
8931  1   7:D    3                           SECOND : TWIZLONG  )     *)
8932  1   7:D    3
8933  1   7:0    0    BEGIN
8934  1   7:1    0      FIRST.LOW := FIRST.LOW - SECOND.LOW;
8935  1   7:1    7      OVERUNDR( FIRST.LOW, FIRST.MID);
```

```
8936  1   7:1   13
8937  1   7:1   13      FIRST.MID := FIRST.MID - SECOND.MID;
8938  1   7:1   22      OVERUNDR( FIRST.MID, FIRST.HIGH);
8939  1   7:1   30
8940  1   7:1   30      FIRST.HIGH := FIRST.HIGH - SECOND.HIGH;
8941  1   7:1   39
8942  1   7:1   39      IF FIRST.HIGH < 0 THEN
8943  1   7:2   45        FIRST.HIGH := 0
8944  1   7:0   48    END;
8945  1   7:0   62
8946  1   7:0   62
8947  1  69:D    1  PROCEDURE LONG2BCD( VAR LONGNUM : TWIZLONG;     (* P010045 *)
8948  1  69:D    2                      VAR BCDNUM  : TBCD);
8949  1  69:D    3
8950  1  69:D    3    VAR
8951  1  69:D    3         DIGITX : INTEGER;
8952  1  69:D    4
8953  1  69:D    4
8954  1  70:D    1    PROCEDURE INT2BCD( PARTLONG : INTEGER);  (* P010046 *)
8955  1  70:D    2
8956  1  71:D    1      PROCEDURE PUTDIGIT( POWOF10 : INTEGER);  (* P010047 *)
8957  1  71:D    2
8958  1  71:0    0        BEGIN (* PUTDIGIT *)
8959  1  71:1    0          BCDNUM[ DIGITX] := PARTLONG DIV POWOF10;
8960  1  71:1   14          DIGITX := DIGITX + 1;
8961  1  71:1   22          PARTLONG := PARTLONG MOD POWOF10
8962  1  71:0   25        END;  (* PUTDIGIT *)
8963  1  71:0   42
8964  1  71:0   42
8965  1  70:0    0      BEGIN (* INT2BCD *)
8966  1  70:1    0        PUTDIGIT( 1000);
8967  1  70:1    5        PUTDIGIT(  100);
8968  1  70:1    8        PUTDIGIT(   10);
8969  1  70:1   11        PUTDIGIT(    1);
8970  1  70:0   14      END;  (* INT2BCD *)
8971  1  70:0   26
8972  1  70:0   26
8973  1  69:0    0    BEGIN (* LONG2BCD *)
8974  1  69:1    0      BCDNUM[ 0] := 0;
8975  1  69:1    6      DIGITX := 1;
8976  1  69:1    9      INT2BCD( LONGNUM.HIGH);
8977  1  69:1   13      INT2BCD( LONGNUM.MID);
8978  1  69:1   17      INT2BCD( LONGNUM.LOW)
8979  1  69:0   19    END;  (* LONG2BCD *)
8980  1  69:0   34
8981  1  69:0   34
8982  1  72:D    1  PROCEDURE BCD2LONG( VAR LONGNUM : TWIZLONG;    (* P010048 *)
8983  1  72:D    2                      VAR BCDNUM  : TBCD);
8984  1  72:D    3
8985  1  72:D    3    VAR
8986  1  72:D    3         DIGITX : INTEGER;
8987  1  72:D    4
8988  1  72:D    4
8989  1  73:D    1    PROCEDURE BCD2INT( VAR LONGPART : INTEGER);  (* P010049 *)
8990  1  73:D    2
8991  1  74:D    1      PROCEDURE GETDIGIT;  (* P01004A *)
8992  1  74:D    1
```

```
8993  1  74:0   0        BEGIN (* GETDIGIT *)
8994  1  74:1   0          LONGPART := (10 * LONGPART) + BCDNUM[ DIGITX];
8995  1  74:1  20          DIGITX := DIGITX + 1
8996  1  74:0  23        END;  (* GETDIGIT *)
8997  1  74:0  40
8998  1  74:0  40
8999  1  73:0   0      BEGIN (* BCD2INT *)
9000  1  73:1   0        LONGPART := 0;
9001  1  73:1   3        GETDIGIT;
9002  1  73:1   5        GETDIGIT;
9003  1  73:1   7        GETDIGIT;
9004  1  73:1   9        GETDIGIT
9005  1  73:0   9      END;  (* BCD2INT *)
9006  1  73:0  24
9007  1  73:0  24
9008  1  72:0   0    BEGIN (* BCD2LONG *)
9009  1  72:1   0      FILLCHAR( LONGNUM, 6, 0);
9010  1  72:1   6      DIGITX := 1;
9011  1  72:1   9      BCD2INT( LONGNUM.HIGH);
9012  1  72:1  14      BCD2INT( LONGNUM.MID);
9013  1  72:1  19      BCD2INT( LONGNUM.LOW)
9014  1  72:0  20    END;  (* BCD2LONG *)
9015  1  72:0  34
9016  1  72:0  34
9017  1   8:D   1  PROCEDURE MULTLONG;  (* P010008 *)
9018  1   8:D   3
9019  1   8:D   3    VAR
9020  1   8:D   3         UNUSEDXX : INTEGER;
9021  1   8:D   4         UNUSEDYY : INTEGER;
9022  1   8:D   5         DIGITX   : INTEGER;
9023  1   8:D   6         BCDNUM   : TBCD;
9024  1   8:D  20
9025  1   8:0   0    BEGIN
9026  1   8:1   0      LONG2BCD( LONGNUM, BCDNUM);
9027  1   8:1   5      FOR DIGITX := 12 DOWNTO 1 DO
9028  1   8:2  17        BCDNUM[ DIGITX] := BCDNUM[ DIGITX] * INTNUM;
9029  1   8:1  39      FOR DIGITX := 12 DOWNTO 1 DO
9030  1   8:2  51        IF BCDNUM[ DIGITX] > 9 THEN
9031  1   8:3  61          BEGIN
9032  1   8:4  61            BCDNUM[ DIGITX - 1] := BCDNUM[ DIGITX - 1] +
9033  1   8:4  76                                   BCDNUM[ DIGITX] DIV 10;
9034  1   8:4  86            BCDNUM[ DIGITX] := BCDNUM[ DIGITX] MOD 10
9035  1   8:3  97          END;
9036  1   8:1 107      BCD2LONG( LONGNUM, BCDNUM)
9037  1   8:0 110    END;
9038  1   8:0 128
9039  1   9:D   1  PROCEDURE DIVLONG;  (* P010009 *)
9040  1   9:D   3
9041  1   9:D   3    VAR
9042  1   9:D   3         NXTDIGIT : INTEGER;
9043  1   9:D   4         DIGITX   : INTEGER;
9044  1   9:D   5         BCDNUM   : TBCD;
9045  1   9:D  19
9046  1   9:0   0    BEGIN  (* DIVLONG *)
9047  1   9:1   0      LONG2BCD( LONGNUM, BCDNUM);
9048  1   9:1   5      FOR DIGITX := 1 TO 12 DO
9049  1   9:2  17        BEGIN
```

```
9050  1   9:3   17          NXTDIGIT := BCDNUM[ DIGITX] DIV INTNUM;
9051  1   9:3   28          BCDNUM[ DIGITX + 1] := BCDNUM[ DIGITX + 1] +
9052  1   9:3   43                                (10 * (BCDNUM[ DIGITX] - NXTDIGIT * INTNUM));
9053  1   9:3   58          BCDNUM[ DIGITX] := NXTDIGIT
9054  1   9:2   63        END;
9055  1   9:1   72      BCD2LONG( LONGNUM, BCDNUM)
9056  1   9:0   75    END;  (* DIVLONG *)
9057  1   9:0   92
9058  1   9:0   92
9059  1  10:D    3  FUNCTION TESTLONG;  (* P01000A *)
9060  1  10:D   11
9061  1  75:D    1    PROCEDURE LTEQGT( FIRSTX  : INTEGER;  (* P01004B *)
9062  1  75:D    2                      SECONDX : INTEGER);
9063  1  75:D    3
9064  1  75:0    0      BEGIN  (* LTEQGT *)
9065  1  75:1    0        IF FIRSTX = SECONDX THEN
9066  1  75:2    5          EXIT( LTEQGT)
9067  1  75:1    9        ELSE
9068  1  75:2   11          BEGIN
9069  1  75:3   11            IF FIRSTX > SECONDX THEN
9070  1  75:4   16              TESTLONG := 1
9071  1  75:3   16            ELSE
9072  1  75:4   22              TESTLONG := -1
9073  1  75:2   22          END;
9074  1  75:1   27        EXIT( TESTLONG)
9075  1  75:0   31      END;  (* LTEQGT *)
9076  1  75:0   44
9077  1  75:0   44
9078  1  10:0    0    BEGIN  (* TESTLONG *)
9079  1  10:1    0      LTEQGT( FIRST.HIGH, SECOND.HIGH);
9080  1  10:1   14      LTEQGT( FIRST.MID,  SECOND.MID);
9081  1  10:1   18      LTEQGT( FIRST.LOW,  SECOND.LOW);
9082  1  10:1   22      TESTLONG := 0
9083  1  10:0   22    END;  (* TESTLONG *)
9084  1  10:0   38
9085  1  10:0   38
9086  1  11:D    1  PROCEDURE PRNTLONG;  (* PRNTWIN: TWINDOWP;      P01000B
9087  1  11:D    6                          LONGNUM: TWIZLONG P01000B *)
9088  1  11:D    6
9089  1  11:D    6    VAR
9090  1  11:D    6         MP06 : INTEGER;
9091  1  11:D    7         MP07 : STRING;
9092  1  11:D   48
9093  1  11:D   48
9094  1  11:0    0    BEGIN
9095  1  11:1    0      P010032( LONGNUM, MP07);
9096  1  11:1   11      FOR MP06 := LENGTH( MP07) TO 11 DO
9097  1  11:2   26        PRPICCH( PRNTWIN, ' ');
9098  1  11:1   37      PRPICMB( PRNTWIN, MP07)
9099  1  11:0   40    END;
9100  1  11:0   56
9101  1  11:0   56
9102  1  12:D    1  PROCEDURE GETKEY;                (* P01000C *)
9103  1  12:D    1
9104  1  12:D    1    (* THIS IS A MODIFIED GETKEY.  SEE WIZ III.  *)
9105  1  12:D    1
9106  1  12:D    1    VAR
```

```
9107  1  12:D   1          MP01 : PACKED RECORD CASE INTEGER OF
9108  1  12:D   1              1: (I: INTEGER);
9109  1  12:D   1              2: (S: STRING[1]);
9110  1  12:D   1            END;
9111  1  12:D   2
9112  1  12:D   2          MP02 : INTEGER;
9113  1  12:D   3          MP03 : TWIZLONG;
9114  1  12:D   6
9115  1  12:0   0      BEGIN
9116  1  12:1   0        REPEAT
9117  1  12:2   0          WHILE NOT KEYPRESS DO
9118  1  12:3   7            MP02 := RAND;
9119  1  12:2  15          UNITREAD( 2, MP01, 1, 0);
9120  1  12:2  24          INCHAR := MP01.S[0];
9121  1  12:2  30          IF (INCHAR > '_') AND (ORD( INCHAR) < 127) THEN
9122  1  12:3  39            INCHAR :=  CHR( ORD( INCHAR) - 32);
9123  1  12:1  44        UNTIL (ORD( INCHAR) <  91  ) OR
9124  1  12:1  47              (ORD( INCHAR) >  127 );
9125  1  12:1  53
9126  1  12:1  53        IF CHARACTR[ 6].EXP.LOW +
9127  1  12:1  61           CHARACTR[ 6].EXP.MID +
9128  1  12:1  70           CHARACTR[ 6].EXP.HIGH > 0 THEN
9129  1  12:2  83          BEGIN
9130  1  12:3  83            FILLCHAR( MP03, 6, 0);
9131  1  12:3  90            MP03.LOW := 1;
9132  1  12:3  93            SUBLONGS( CHARACTR[ 6].EXP, MP03)
9133  1  12:2 103          END;
9134  1  12:0 105      END;
9135  1  12:0 122
9136  1  12:0 122
9137  1  13:D   1    PROCEDURE GETCR;       (* P01000D *)
9138  1  13:D   1
9139  1  13:0   0      BEGIN
9140  1  13:1   0        REPEAT
9141  1  13:2   0          GETKEY;
9142  1  13:1   2        UNTIL INCHAR = CHR( CRETURN)
9143  1  13:0   4      END;
9144  1  13:0  22
9145  1  13:0  22
9146  1  15:D   1    PROCEDURE PAUSE1;  (* P01000F *)
9147  1  15:D   1
9148  1  15:0   0      BEGIN  (* PAUSE1 *)
9149  1  15:1   0        IF TIMEDLAY > 0 THEN
9150  1  15:2   6          UNITWRITE( BASE12, TIMEDLAY, 19, TIMEDLAY)
9151  1  15:0  17      END;   (* PAUSE1 *)
9152  1  15:0  30
9153  1  15:0  30
9154  1  16:D   1    PROCEDURE PAUSE2;  (* P010010 *)
9155  1  16:D   1
9156  1  16:D   1      VAR
9157  1  16:D   1           MP01 : INTEGER;
9158  1  16:D   2
9159  1  16:0   0      BEGIN  (* PAUSE2 *)
9160  1  16:1   0        MP01 := 20;
9161  1  16:1   3
9162  1  16:1   3        (* UPDATE RNG AND LOOP AWHILE WAITING FOR KEYBOARD INPUT, THEN RTS *)
9163  1  16:1   3        UNITWRITE( BASE12, MP01, 19, MP01)
```

```
9164  1  16:0   13    END;   (* PAUSE2 *)
9165  1  16:0   26
9166  1  16:0   26
9167  1  14:D    1  PROCEDURE CENTSTR;  (* MP02: TWINDOWP; MP01: TLONGSTR *)  (* P01000E *)
9168  1  14:D  131
9169  1  14:D  131    VAR
9170  1  14:D  131        MP83 : INTEGER;
9171  1  14:D  132
9172  1  14:0    0    BEGIN
9173  1  14:1    0      MP83 := MP02^.HEAD.VCURSOR;
9174  1  14:1   14      MVCURSOR( MP02,
9175  1  14:1   15                ((MP02^.HEAD.HSIZE - 2) DIV 2) - (1 + MBSTRLEN( MP01)) DIV 2,
9176  1  14:1   36                MP02^.HEAD.VCURSOR);
9177  1  14:1   44      PRPICMB( MP02, MP01);
9178  1  14:1   49      MVCURSOR( MP02, 0, MP83 + 1)
9179  1  14:0   56    END;
9180  1  14:0   70
9181  1  14:0   70
9182  1  14:0   70 (*$I WIZ4A: WIZ3.TEXT *)
9183  1  14:0   70
9183  1  14:0   70 (*$I WIZ4A: WIZ4.TEXT *)
9184  1  14:0   70
9185  1  14:0   70 (* WIZ4 *)
9186  1  14:0   70
9187  1  76:D    1  PROCEDURE GARBMEM;   (* P01004C *)
9188  1  76:D    1
9189  1  76:D    1    VAR
9190  1  76:D    1        CHAIN2   : ^TMEMORY;
9191  1  76:D    2        B4NOTINU : ^TMEMORY;
9192  1  76:D    3        CHAINPTR : ^TMEMORY;
9193  1  76:D    4
9194  1  76:D    4
9195  1  76:0    0    BEGIN
9196  1  76:1    0      DOGARBMM := FALSE;
9197  1  76:1    3      CHAINPTR := MEMHEAD;
9198  1  76:1    7      B4NOTINU := NIL;
9199  1  76:1   10      WHILE CHAINPTR <> NIL DO
9200  1  76:2   15        BEGIN
9201  1  76:3   15          IF NOT CHAINPTR^.INUSE THEN
9202  1  76:4   24            BEGIN
9203  1  76:5   24              IF CHAINPTR^.NEXT <> NIL THEN
9204  1  76:6   30                BEGIN
9205  1  76:7   30                  CHAIN2 := CHAINPTR^.NEXT;
9206  1  76:7   34                  WHILE (CHAIN2 <> NIL) AND (NOT CHAIN2^.INUSE) DO
9207  1  76:8   47                    BEGIN
9208  1  76:9   47                      CHAINPTR^.NEXT := CHAIN2^.NEXT;
9209  1  76:9   51                      CHAINPTR^.SIZE := CHAINPTR^.SIZE + CHAIN2^.SIZE;
9210  1  76:9   70                      CHAIN2 := CHAIN2^.NEXT
9211  1  76:8   71                    END
9212  1  76:6   74                END;
9213  1  76:5   76              IF CHAINPTR^.NEXT = NIL THEN
9214  1  76:6   82                BEGIN
9215  1  76:7   82                  IF B4NOTINU = NIL THEN
9216  1  76:8   87                    MEMHEAD := NIL
9217  1  76:7   87                  ELSE
9218  1  76:8   92                    B4NOTINU^.NEXT := NIL;
9219  1  76:7   95                  RELEASE( CHAINPTR)
```

```
9220  1  76:6   97              END
9221  1  76:4   99            END;
9222  1  76:3   99          B4NOTINU := CHAINPTR;
9223  1  76:3  102          CHAINPTR := CHAINPTR^.NEXT
9224  1  76:2  103        END;
9225  1  76:0  108    END;
9226  1  76:0  124
9227  1  76:0  124
9228  1  77:D    3  FUNCTION GETMEM( MEMSIZE: INTEGER) : TWINDOWP;  (* P01004D *)
9229  1  77:D    4
9230  1  77:D    4    VAR
9231  1  77:D    4         SPLITBUF : ^TMEMORY;
9232  1  77:D    5         MEMCHAIN : ^TMEMORY;
9233  1  77:D    6         MP06     : RECORD CASE INTEGER OF
9234  1  77:D    6                      1: ( PWIN:TWINDOWP);
9235  1  77:D    6                      2: ( I:   INTEGER);
9236  1  77:D    6                    END;
9237  1  77:D    7         PTR2INT  : TMEMOVER;
9238  1  77:D    8
9239  1  77:0    0    BEGIN
9240  1  77:1    0      IF DOGARBMM THEN
9241  1  77:2    4        GARBMEM;
9242  1  77:1    6      MEMSIZE := MEMSIZE + SIZEOF( TMEMORY);
9243  1  77:1   11      MEMCHAIN :=MEMHEAD;
9244  1  77:1   15      WHILE MEMCHAIN <> NIL DO
9245  1  77:2   20        BEGIN
9246  1  77:3   20          IF NOT MEMCHAIN^.INUSE THEN
9247  1  77:4   29            IF MEMCHAIN^.SIZE >= MEMSIZE THEN
9248  1  77:5   39              BEGIN
9249  1  77:6   39                IF (MEMCHAIN^.SIZE - MEMSIZE) > SIZEOF( TMEMORY) THEN
9250  1  77:7   51                  BEGIN
9251  1  77:8   51                    PTR2INT.PMEM := MEMCHAIN;
9252  1  77:8   54                    PTR2INT.I := PTR2INT.I + MEMSIZE;
9253  1  77:8   59                    SPLITBUF := PTR2INT.PMEM;
9254  1  77:8   62                    SPLITBUF^.NEXT := MEMCHAIN^.NEXT;
9255  1  77:8   66                    SPLITBUF^.SIZE := MEMCHAIN^.SIZE - MEMSIZE;
9256  1  77:8   80                    SPLITBUF^.INUSE := FALSE;
9257  1  77:8   87                    MEMCHAIN^.SIZE := MEMSIZE;
9258  1  77:8   94                    MEMCHAIN^.NEXT := SPLITBUF
9259  1  77:7   95                  END;
9260  1  77:6   97                MEMCHAIN^.INUSE := TRUE;
9261  1  77:6  104                PTR2INT.PMEM := MEMCHAIN;
9262  1  77:6  107                MP06.I := PTR2INT.I + SIZEOF( TMEMORY);
9263  1  77:6  112                GETMEM := MP06.PWIN;
9264  1  77:6  115                EXIT( GETMEM)
9265  1  77:5  119              END;
9266  1  77:3  119            SPLITBUF := MEMCHAIN;
9267  1  77:3  122            MEMCHAIN := MEMCHAIN^.NEXT
9268  1  77:2  123          END;
9269  1  77:1  128        MARK( MEMCHAIN);
9270  1  77:1  132        IF MEMHEAD = NIL THEN
9271  1  77:2  138          MEMHEAD := MEMCHAIN
9272  1  77:1  138        ELSE
9273  1  77:2  143          SPLITBUF^.NEXT := MEMCHAIN;
9274  1  77:1  146        MEMCHAIN^.NEXT := NIL;
9275  1  77:1  149        MEMCHAIN^.SIZE := MEMSIZE;
9276  1  77:1  156        MEMCHAIN^.INUSE := TRUE;
```

```
9277  1  77:1  163       PTR2INT.PMEM := MEMCHAIN;
9278  1  77:1  166       MP06.I := PTR2INT.I + SIZEOF( TMEMORY);
9279  1  77:1  171       GETMEM := MP06.PWIN;
9280  1  77:1  174       PTR2INT.I := PTR2INT.I + MEMSIZE;
9281  1  77:1  179       RELEASE( PTR2INT.PMEM)
9282  1  77:0  181     END;
9283  1  77:0  198
9284  1  77:0  198
9285  1  78:D    1   PROCEDURE RELMEM( RELWIN : TWINDOWP);  (* P01004E *)
9286  1  78:D    2
9287  1  78:D    2     VAR
9288  1  78:D    2         BUFHEAD  : TMEMORYP;
9289  1  78:D    3         CHAINPTR : TMEMORYP;
9290  1  78:D    4         PTR2INT  : TMEMOVER;
9291  1  78:D    5         MP05     : TMEMOVER;
9292  1  78:D    6
9293  1  78:0    0     BEGIN
9294  1  78:1    0       IF MEMHEAD = NIL THEN
9295  1  78:2    6         EXIT( RELMEM);
9296  1  78:1   10       CHAINPTR := MEMHEAD;
9297  1  78:1   14       PTR2INT.PWIN := RELWIN;
9298  1  78:1   17       MP05.I := PTR2INT.I - SIZEOF( TMEMORY);
9299  1  78:1   22       BUFHEAD := MP05.PMEM;
9300  1  78:1   25       WHILE CHAINPTR <> BUFHEAD DO
9301  1  78:2   30         BEGIN
9302  1  78:3   30           CHAINPTR := CHAINPTR^.NEXT;
9303  1  78:3   34           IF CHAINPTR = NIL THEN
9304  1  78:4   39             EXIT( RELMEM)
9305  1  78:2   43         END;
9306  1  78:1   45       CHAINPTR^.INUSE := FALSE;
9307  1  78:1   52       DOGARBMM := TRUE
9308  1  78:0   52     END;
9309  1  78:0   70
9310  1  78:0   70
9311  1  17:D    3   FUNCTION GETWIN;       (* P010011 *)
9312  1  17:D    9                     (* HPOS:     INTEGER;
9313  1  17:D    9                        VPOS:     INTEGER;
9314  1  17:D    9                        HSIZE:    INTEGER:
9315  1  17:D    9                        VSIZE:    INTEGER;
9316  1  17:D    9                        PRIORITY: INTEGER;
9317  1  17:D    9                        SHOW:     BOOLEAN) : TWINDOWP;  *)
9318  1  17:D    9
9319  1  17:D    9     VAR
9320  1  17:D    9         NEWWIN  : TWINDOWP;
9321  1  17:D   10         BUFSIZE : INTEGER;
9322  1  17:D   11
9323  1  17:0    0     BEGIN
9324  1  17:1    0       BUFSIZE := SIZEOF( TWINHEAD) + 2 * (HSIZE - 2) * (VSIZE - 2);
9325  1  17:1   13       NEWWIN := GETMEM( BUFSIZE);
9326  1  17:1   20       IF NEWWIN <> NIL THEN
9327  1  17:2   25         BEGIN
9328  1  17:3   25           NEWWIN^.HEAD.NEXTW := WINCHAIN;
9329  1  17:3   28           WINCHAIN := NEWWIN;
9330  1  17:3   31           NEWWIN^.HEAD.HPOS := HPOS;
9331  1  17:3   38           NEWWIN^.HEAD.VPOS := VPOS;
9332  1  17:3   45           NEWWIN^.HEAD.HSIZE := HSIZE;
9333  1  17:3   52           NEWWIN^.HEAD.VSIZE := VSIZE;
```

```
9334  1  17:3   59          NEWWIN^.HEAD.HCURSOR := 0;
9335  1  17:3   66          NEWWIN^.HEAD.VCURSOR := 0;
9336  1  17:3   73          NEWWIN^.HEAD.PRIORITY := PRIORITY + 1;
9337  1  17:3   82          FILLCHAR( NEWWIN^.DATA, BUFSIZE - SIZEOF( TWINHEAD), ' ');
9338  1  17:3   92          PRIORDER;
9339  1  17:3   94          DRAWSCR( SHOW)
9340  1  17:2   95        END
9341  1  17:1   97      ELSE
9342  1  17:2   99          NEWWIN := NIL;
9343  1  17:1  102      GETWIN := NEWWIN
9344  1  17:0  102    END;
9345  1  17:0  118
9346  1  17:0  118
9347  1  18:D    1  PROCEDURE DELWIN;                              (* P010012 *)
9348  1  18:D    3                  (* VAR WIN2DEL : TWINDOWP;
9349  1  18:D    3                         REDRAW  : BOOLEAN *)
9350  1  18:D    3
9351  1  18:D    3    VAR
9352  1  18:D    3        CHAINPTR : TWINDOWP;
9353  1  18:D    4
9354  1  18:0    0    BEGIN
9355  1  18:1    0      IF WINCHAIN = NIL THEN
9356  1  18:2    5        EXIT( DELWIN)
9357  1  18:1    9      ELSE
9358  1  18:2   11        IF WIN2DEL = WINCHAIN THEN
9359  1  18:3   17          WINCHAIN := WINCHAIN^.HEAD.NEXTW
9360  1  18:2   18        ELSE
9361  1  18:3   23          BEGIN
9362  1  18:4   23            CHAINPTR := WINCHAIN;
9363  1  18:4   26            WHILE CHAINPTR^.HEAD.NEXTW <> WIN2DEL DO
9364  1  18:5   33              CHAINPTR := CHAINPTR^.HEAD.NEXTW;
9365  1  18:4   39            CHAINPTR^.HEAD.NEXTW := WIN2DEL^.HEAD.NEXTW;
9366  1  18:3   44          END;
9367  1  18:1   44      RELMEM( WIN2DEL);
9368  1  18:1   48      WIN2DEL := NIL;
9369  1  18:1   51      DRAWSCR( REDRAW)
9370  1  18:0   52    END;
9371  1  18:0   68
9372  1  18:0   68
9373  1  19:D    1  PROCEDURE SETWNPRI;  (* PRIWIN   : TWINDOWP;       P010013
9374  1  19:D    4                          PRIORITY : INTEGER;
9375  1  19:D    4                          REDRAW   : BOOLEAN);    *)
9376  1  19:D    4
9377  1  19:D    4    VAR
9378  1  19:D    4         UNUSED : ARRAY[ 0..2] OF INTEGER;
9379  1  19:D    7
9380  1  19:0    0    BEGIN
9381  1  19:1    0      PRIWIN^.HEAD.PRIORITY := PRIORITY + 1;
9382  1  19:1    9      PRIORDER;
9383  1  19:1   11      DRAWSCR( REDRAW)
9384  1  19:0   12    END;
9385  1  19:0   26
9386  1  19:0   26
9387  1  23:D    1  PROCEDURE CLEARWIN; (* MP02 : TWINDOWP; *)      (* P010017 *)
9388  1  23:D    3                      (* MP01 : BOOLEAN   *)
9389  1  23:D    3
9390  1  23:0    0    BEGIN
```

```
9391  1  23:0   0
9392  1  23:1   0       FILLCHAR( MP02^.DATA[ 0],
9393  1  23:1   4                 (2 * ( MP02^.HEAD.HSIZE - 2)) *
9394  1  23:1  14                      ( MP02^.HEAD.VSIZE - 2),
9395  1  23:1  23                    ' ');
9396  1  23:1  26       MP02^.HEAD.HCURSOR := 0;
9397  1  23:1  33       MP02^.HEAD.VCURSOR := 0;
9398  1  23:1  40       IF MP01 THEN
9399  1  23:2  43         DRAWSCR2( WINCHAIN)
9400  1  23:1  44       ELSE
9401  1  23:2  48         UNITWRITE( BASE12, MP02^, 3, 0)
9402  1  23:0  57     END;
9403  1  23:0  70
9404  1  23:0  70
9405  1  24:D   1   PROCEDURE CLRRECT;  (* CLRWIN : TWINDOWP;      P010018
9406  1  24:D   6                         HOR1   : INTEGER;
9407  1  24:D   6                         VER1   : INTEGER;
9408  1  24:D   6                         HOR2   : INTEGER;
9409  1  24:D   6                         VER2   : INTEGER)   *)
9410  1  24:D   6
9411  1  24:D   6     VAR
9412  1  24:D   6          BYTS2CLR : INTEGER;
9413  1  24:D   7          HPOS     : INTEGER;
9414  1  24:D   8          HORBYTES : INTEGER;
9415  1  24:D   9          LINE     : INTEGER;
9416  1  24:D  10          UNUSED   : INTEGER;
9417  1  24:D  11
9418  1  79:D   1     PROCEDURE ADJUSTHV(    ADJWIN: TWINDOWP;  (* P01004F *)
9419  1  79:D   2                         VAR HPOS :  INTEGER;
9420  1  79:D   3                         VAR VPOS :  INTEGER);
9421  1  79:D   4
9422  1  79:0   0       BEGIN (* ADJUSTHV *)
9423  1  79:1   0         IF HPOS < 0 THEN
9424  1  79:2   6           HPOS := 0
9425  1  79:1   7         ELSE
9426  1  79:2  11           IF HPOS > (ADJWIN^.HEAD.HSIZE - 3) THEN
9427  1  79:3  24             HPOS := ADJWIN^.HEAD.HSIZE - 3;
9428  1  79:3  34
9429  1  79:1  34         IF VPOS < 0 THEN
9430  1  79:2  40           VPOS := 0
9431  1  79:1  41         ELSE
9432  1  79:2  45           IF VPOS > (ADJWIN^.HEAD.VSIZE - 3) THEN
9433  1  79:3  58             VPOS := ADJWIN^.HEAD.VSIZE - 3
9434  1  79:0  65       END;  (* ADJUSTHV *)
9435  1  79:0  80
9436  1  79:0  80
9437  1  24:0   0     BEGIN (* CLRRECT *)
9438  1  24:1   0       ADJUSTHV( CLRWIN, HOR1, VER1);
9439  1  24:1   7       HOR2 := HOR1 + HOR2 - 1;
9440  1  24:1  14       VER2 := VER1 + VER2 - 1;
9441  1  24:1  21       ADJUSTHV( CLRWIN, HOR2, VER2);
9442  1  24:1  28       HORBYTES := 2 * ( CLRWIN^.HEAD.HSIZE - 2); (* BYTES PER ENTIRE ROW *)
9443  1  24:1  40       HPOS := HORBYTES * VER1 + HOR1 + HOR1;     (* HORIZONTAL POSITION *)
9444  1  24:1  49       BYTS2CLR := 2 * (HOR2 - HOR1 + 1);         (* # BYTES TO CLEAR PER ROW *)
9445  1  24:1  58       FOR LINE := VER1 TO VER2 DO
9446  1  24:2  69         BEGIN
9447  1  24:3  69           FILLCHAR( CLRWIN^.DATA[ HPOS], BYTS2CLR, ' ');
```

```
9448  1  24:3   77          HPOS := HPOS + HORBYTES
9449  1  24:2   78        END;
9450  1  24:1   89      UNITWRITE( BASE12, CLRWIN^, 3, 0)
9451  1  24:0   98    END;  (* CLRRECT *)
9452  1  24:0  112
9453  1  24:0  112
9454  1  25:D    1  PROCEDURE PROTWIN;  (* PROTWIN : TWINDOWP;    P010019
9455  1  25:D    3                         REDRAW  : BOOLEAN);  *)
9456  1  25:D    3
9457  1  25:0    0    BEGIN
9458  1  25:1    0      PROTWIN^.HEAD.PRIORITY := 128 + (PROTWIN^.HEAD.PRIORITY MOD 128);
9459  1  25:1   20      DRAWSCR( REDRAW)
9460  1  25:0   21    END;
9461  1  25:0   36
9462  1  25:0   36
9463  1  26:D    1  PROCEDURE UNPROWIN;  (* MP02 : TWINDOWP;           P01001A
9464  1  26:D    3                          MP01 : BOOLEAN   *)
9465  1  26:D    3
9466  1  26:0    0    BEGIN
9467  1  26:1    0      MP02^.HEAD.PRIORITY := MP02^.HEAD.PRIORITY MOD 128;
9468  1  26:1   16      DRAWSCR( MP01)
9469  1  26:0   17    END;
9470  1  26:0   32
9471  1  26:0   32
9472  1  20:D    1  PROCEDURE MVCURSOR;   (* MVWIN : TWINDOWP         P010014
9473  1  20:D    4                           HPOS  : INTEGER;
9474  1  20:D    4                           VPOS  : INTEGER  *)
9475  1  20:D    4
9476  1  20:0    0    BEGIN
9477  1  20:1    0      IF HPOS < 0 THEN
9478  1  20:2    5        HPOS := 0
9479  1  20:1    5      ELSE
9480  1  20:2   10        BEGIN
9481  1  20:3   10          IF HPOS >= MVWIN^.HEAD.HSIZE - 2 THEN
9482  1  20:4   22            BEGIN
9483  1  20:5   22              HPOS := 0;
9484  1  20:5   25              VPOS := VPOS + 1
9485  1  20:4   26            END;
9486  1  20:2   30        END;
9487  1  20:2   30
9488  1  20:1   30      IF VPOS < 0 THEN
9489  1  20:2   35        VPOS := 0
9490  1  20:1   35      ELSE
9491  1  20:2   40        IF VPOS >= MVWIN^.HEAD.VSIZE - 2 THEN
9492  1  20:3   52          VPOS := 0;
9493  1  20:3   55
9494  1  20:1   55      MVWIN^.HEAD.HCURSOR := HPOS;
9495  1  20:1   62      MVWIN^.HEAD.VCURSOR := VPOS
9496  1  20:0   67    END;
9497  1  20:0   82
9498  1  20:0   82
9499  1  33:D    1  PROCEDURE PRPICCH; (* MYWIN  : TWINDOWP;     P010021 *)
9500  1  33:D    3                        (* VAR (?) MYCHAR : STRING); CHAR(?) *)
9501  1  33:D    3
9502  1  33:D    3    (* PRINT PICTURE CHAR *)
9503  1  33:D    3
9504  1  33:0    0    BEGIN
```

```
9505  1  33:1    0      UNITWRITE( BASE12, MYWIN^, 4, ORD( MYCHAR))
9506  1  33:0    9    END;
9507  1  33:0   22
9508  1  33:0   22
9509  1  34:D    1  PROCEDURE PRPICMB; (* MYWIN  : TWINDOWP;     P010022 *)
9510  1  34:D  131                    (* MYMBCH : TLONGSTR; *)
9511  1  34:D  131
9512  1  34:D  131    (* PRINT PICTURE MULTI BYTE CHAR *)
9513  1  34:D  131
9514  1  34:D  131    VAR
9515  1  34:D  131         MP83 : INTEGER;
9516  1  34:D  132         MP84 : INTEGER;
9517  1  34:D  133         MP85 : CHAR;
9518  1  34:D  134         MP86 : RECORD CASE INTEGER OF
9519  1  34:D  134                  1: ( I: INTEGER);
9520  1  34:D  134                  2: ( C: PACKED ARRAY[ 0..1] OF CHAR);
9521  1  34:D  134                END;
9522  1  34:D  135
9523  1  34:0    0    BEGIN
9524  1  34:1    0      IF BSTROPS THEN
9525  1  34:2   10        BEGIN
9526  1  34:3   10          IF LENGTH( MYMBCH) <> 0 THEN
9527  1  34:4   18            UNITWRITE( BASE12, MYWIN^, 30, GETADDR( MYMBCH));
9528  1  34:3   32          EXIT( PRPICMB)
9529  1  34:2   36        END;
9530  1  34:2   36
9531  1  34:1   36      MP83 := 1;
9532  1  34:1   40      MP84 := LENGTH( MYMBCH);
9533  1  34:1   47      WHILE MP83 <= MP84 DO
9534  1  34:2   56        BEGIN
9535  1  34:3   56          MP85 := MYMBCH[ MP83];
9536  1  34:3   65          IF (ORD( MP85) >= 129) AND (ORD( MP85) <= 159) THEN
9537  1  34:4   82            BEGIN
9538  1  34:5   82              MP86.C[ 1] := MP85;
9539  1  34:5   90              MP83 := MP83 + 1;
9540  1  34:5   98              MP86.C[ 0] := MYMBCH[ MP83];
9541  1  34:5  109              UNITWRITE( BASE12, MYWIN^, 4, MP86.I)
9542  1  34:4  120            END
9543  1  34:3  120          ELSE
9544  1  34:4  122            BEGIN
9545  1  34:5  122              UNITWRITE( BASE12, MYWIN^, 4, ORD( MP85))
9546  1  34:4  133            END;
9547  1  34:3  133          MP83 := MP83 + 1
9548  1  34:2  136        END;
9549  1  34:0  143    END;
9550  1  34:0  158
9551  1  34:0  158
9552  1  21:D    1  PROCEDURE PRINTNUM; (* PRWIN   : TWINDOWP;      P010015
9553  1  21:D    4                         NUM     : INTEGER;
9554  1  21:D    4                         FIELDSZ : INTEGER);  *)
9555  1  21:D    4
9556  1  21:D    4    VAR
9557  1  21:D    4         NUMSTR : STRING;
9558  1  21:D   45
9559  1  21:D   45
9560  1  80:D    1    PROCEDURE PRCHAR( ACHAR : CHAR);  (* P010050 *)
9561  1  80:D    2
```

```
9562  1  80:0   0      BEGIN (* PRCHAR *)
9563  1  80:1   0        PRPICCH( PRWIN, ACHAR);
9564  1  80:1   6        FIELDSZ := FIELDSZ - 1
9565  1  80:0   9      END;  (* PRCHAR *)
9566  1  80:0  26
9567  1  80:0  26
9568  1  21:0   0    BEGIN (* PRINTNUM *)
9569  1  21:1   0      INT2STR( NUM, NUMSTR);
9570  1  21:1   5      IF LENGTH( NUMSTR) > FIELDSZ THEN
9571  1  21:2  13        BEGIN
9572  1  21:3  13          IF NUM < 0 THEN
9573  1  21:4  18            PRCHAR( '-');
9574  1  21:3  21          WHILE FIELDSZ > 0 DO
9575  1  21:4  26            PRCHAR( '*')
9576  1  21:2  27        END
9577  1  21:1  31      ELSE
9578  1  21:2  33        BEGIN
9579  1  21:3  33          WHILE FIELDSZ > LENGTH( NUMSTR) DO
9580  1  21:4  41            PRCHAR( ' ');
9581  1  21:3  46          PRPICMB( PRWIN, NUMSTR)
9582  1  21:2  49        END
9583  1  21:0  51    END;  (* PRINTNUM *)
9584  1  21:0  68
9585  1  21:0  68
9586  1  22:D   1  PROCEDURE PRINTCR;  (* MP01 : TWINDOWP *)    (* P010016 *)
9587  1  22:D   2
9588  1  22:0   0    BEGIN
9589  1  22:1   0      MP01^.HEAD.VCURSOR := MP01^.HEAD.VCURSOR + 1;
9590  1  22:1  14      MP01^.HEAD.HCURSOR := 0;
9591  1  22:1  21      IF MP01^.HEAD.VCURSOR >= (MP01^.HEAD.VSIZE - 2) THEN
9592  1  22:2  38        MP01^.HEAD.VCURSOR := 0
9593  1  22:0  43    END;
9594  1  22:0  58
9595  1  22:0  58
9596  1  22:0  58
9597  1  22:0  58
9598  1  27:D   1  PROCEDURE SOLICIT;   (*        PRWIN : TWINDOWP;    P01001B
9599  1  27:D   4                           VAR REPLYSTR : TLONGSTR;
9600  1  27:D   4                               MAXSTR : INTEGER);  *)
9601  1  27:D   4
9602  1  27:D   4            (* IN WIZ III THIS IS CALLED GETSTR() *)
9603  1  27:D   4
9604  1  27:D   4    VAR
9605  1  27:D   4         STRPOS : INTEGER;
9606  1  27:D   5         VPOS   : INTEGER;
9607  1  27:D   6         HPOS   : INTEGER;
9608  1  27:D   7         MB2    : STRING[ 2];
9609  1  27:D   9
9610  1  27:D   9
9611  1  81:D   1    PROCEDURE BACKSPAC;  (* P010051 *)
9612  1  81:D   1
9613  1  81:0   0      BEGIN  (* BACKSPAC *)
9614  1  81:1   0        STRPOS := STRPOS - 1;
9615  1  81:1   8        IF (REPLYSTR[ STRPOS] >= CHR( 129)) AND
9616  1  81:1  19           (REPLYSTR[ STRPOS] <= CHR( 159)) THEN
9617  1  81:2  33          STRPOS := STRPOS - 1;
9618  1  81:1  41        HPOS := HPOS - 1;
```

```
9619  1  81:1   49          MVCURSOR( PRWIN, HPOS, VPOS);
9620  1  81:1   60          PRPICCH( PRWIN, ' ');
9621  1  81:1   66          MVCURSOR( PRWIN, HPOS, VPOS)
9622  1  81:0   75        END;   (* BACKSPAC *)
9623  1  81:0   90
9624  1  81:0   90
9625  1  82:D    1      PROCEDURE P010052( VAR MYCHAR : CHAR);
9626  1  82:D    2
9627  1  82:D    2        VAR
9628  1  82:D    2             MYCHARIN : PACKED RECORD CASE INTEGER OF
9629  1  82:D    2                1: (I: INTEGER);
9630  1  82:D    2                2: (S: STRING[1]);
9631  1  82:D    2              END;
9632  1  82:D    3
9633  1  82:0    0        BEGIN  (* P010052 *)
9634  1  82:1    0          UNITREAD( 2, MYCHARIN, 1, 0);
9635  1  82:1    9          MYCHAR := MYCHARIN.S[ 0]
9636  1  82:0   13        END;   (* P010052 *)
9637  1  82:0   28
9638  1  82:0   28
9639  1  27:0    0      BEGIN (* SOLICIT *)
9640  1  27:1    0        FILLCHAR( REPLYSTR, MAXSTR + 1, 0);
9641  1  27:1    8        STRPOS := 0;
9642  1  27:1   11        MB2 := '  ';
9643  1  27:1   20
9644  1  27:1   20        REPEAT
9645  1  27:2   20          HPOS := PRWIN^.HEAD.HCURSOR;
9646  1  27:2   28          VPOS := PRWIN^.HEAD.VCURSOR;
9647  1  27:2   36          MVCURSOR( PRWIN, HPOS, VPOS);
9648  1  27:2   41          IF BAPPLE THEN
9649  1  27:3   46            PRPICCH( PRWIN, CHR( 17))  (* SPECIAL "<" FILLED TRIANGLE CURSOR *)
9650  1  27:2   48          ELSE
9651  1  27:3   52            PRPICCH( PRWIN, CHR( 95)); (* "^" CIRCUMFLEX *)
9652  1  27:2   56          GETKEY;
9653  1  27:2   58          MVCURSOR( PRWIN, HPOS, VPOS);
9654  1  27:2   63          PRPICCH( PRWIN, ' ');
9655  1  27:2   67          MVCURSOR( PRWIN, HPOS, VPOS);
9656  1  27:2   72          IF (INCHAR = CHR( 8) ) AND (STRPOS > 0) THEN
9657  1  27:3   81            BACKSPAC
9658  1  27:2   81          ELSE
9659  1  27:3   85            BEGIN
9660  1  27:4   85              IF (INCHAR = CHR( CRETURN)) OR (INCHAR = CHR( ESCAPE)) THEN
9661  1  27:5   94                BEGIN
9662  1  27:6   94                  PRPICCH( PRWIN, ' ');
9663  1  27:6   98                  IF INCHAR = CHR( ESCAPE) THEN
9664  1  27:7  103                    WHILE STRPOS > 0 DO
9665  1  27:8  108                      BACKSPAC;
9666  1  27:6  112                  REPLYSTR[ 0] := CHR( STRPOS);
9667  1  27:6  116                  EXIT( SOLICIT)
9668  1  27:5  120                END
9669  1  27:4  120              ELSE
9670  1  27:5  122                BEGIN
9671  1  27:6  122                  IF (INCHAR > CHR( 31)) AND
9672  1  27:6  125                     (INCHAR <= CHR( 127)) AND
9673  1  27:6  129                     (HPOS < PRWIN^.HEAD.HSIZE - 3) AND
9674  1  27:6  140                     (STRPOS < MAXSTR) THEN
9675  1  27:7  146                    BEGIN
```

```
9676  1  27:8  146                    STRPOS := STRPOS + 1;
9677  1  27:8  151                    REPLYSTR[ STRPOS] := INCHAR;
9678  1  27:8  155                    PRPICCH( PRWIN, INCHAR)
9679  1  27:7  157                  END
9680  1  27:6  159                ELSE
9681  1  27:7  161                  BEGIN
9682  1  27:8  161                    IF (INCHAR > CHR( 129)) AND
9683  1  27:8  166                       (INCHAR <= CHR( 159)) AND
9684  1  27:8  172                       (HPOS < PRWIN^.HEAD.HSIZE - 3) AND
9685  1  27:8  183                       (STRPOS < MAXSTR - 1) THEN
9686  1  27:9  191                      BEGIN
9687  1  27:0  191                        STRPOS := STRPOS + 1;
9688  1  27:0  196                        REPLYSTR[ STRPOS] := INCHAR;
9689  1  27:0  200                        MB2[ 1] := INCHAR;
9690  1  27:0  205                        P010052( INCHAR);
9691  1  27:0  209                        STRPOS := STRPOS + 1;
9692  1  27:0  214                        REPLYSTR[ STRPOS] := INCHAR;
9693  1  27:0  218                        MB2[ 2] := INCHAR;
9694  1  27:0  223                        PRPICMB( PRWIN, MB2)
9695  1  27:9  226                      END
9696  1  27:8  228                    ELSE
9697  1  27:9  230                      BEGIN
9698  1  27:0  230                        PRINTBEL
9699  1  27:9  230                      END
9700  1  27:7  232                  END
9701  1  27:5  232              END;
9702  1  27:3  232          END;
9703  1  27:3  232
9704  1  27:1  232        UNTIL FALSE
9705  1  27:1  232
9706  1  27:0  232    END;  (* SOLICIT *)
9707  1  27:0  254
9708  1  27:0  254
9709  1  27:0  254 (*$I WIZ4A: WIZ4.TEXT *)
9710  1  27:0  254
9710  1  27:0  254 (*$I WIZ4A: WIZ5.TEXT *)
9711  1  27:0  254
9712  1  27:0  254   (* WIZ5.TEXT *)
9713  1  27:0  254
9714  1  28:D    3   FUNCTION MENU;  (* MENUWIN : TWINDOWP;     P01001C
9715  1  28:D  133                      MENUSTR : TLONGSTR) : INTEGER;  *)
9716  1  28:D  133
9717  1  28:D  133    VAR
9718  1  28:D  133         VPOSMENU : INTEGER;
9719  1  28:D  134         INDX     : INTEGER;
9720  1  28:D  135         MXCHOICE : INTEGER;
9721  1  28:D  136         HPOS     : INTEGER;
9722  1  28:D  137         HPOSMENU : INTEGER;
9723  1  28:D  138         CHOICEN  : INTEGER;
9724  1  28:D  139         DEFAULTX : INTEGER;
9725  1  28:D  140         RTNSYMBL : INTEGER;
9726  1  28:D  141         GOTSLASH : BOOLEAN;
9727  1  28:D  142         CHOICES  : ARRAY[ 0..24] OF CHAR;
9728  1  28:D  167         MBCHAR   : STRING[ 2];
9729  1  28:D  169         CH       : CHAR;
9730  1  28:D  170
9731  1  28:D  170
```

```
9732  1  83:D   1    PROCEDURE EXITMENU( MP01 : INTEGER);  (* P010053 *)
9733  1  83:D   2
9734  1  83:0   0      BEGIN (* EXITMENU *)
9735  1  83:1   0        MENU := MP01;
9736  1  83:1   4        EXIT( MENU)
9737  1  83:0   8      END;  (* EXITMENU *)
9738  1  83:0  20
9739  1  28:0   0    BEGIN (* MENU *)
9740  1  28:1   0      DEFAULTX := -1;
9741  1  28:1  10      RTNSYMBL := -1;
9742  1  28:1  15      MXCHOICE := 0;
9743  1  28:1  19      HPOSMENU := MENUWIN^.HEAD.HCURSOR;
9744  1  28:1  28      HPOS := -1;
9745  1  28:1  33      GOTSLASH := TRUE;
9746  1  28:1  37      VPOSMENU := MENUWIN^.HEAD.VCURSOR;
9747  1  28:1  46      INDX := 1;
9748  1  28:1  50      MBCHAR[ 0] := CHR( 2);
9749  1  28:1  56      WHILE INDX <= LENGTH( MENUSTR) DO
9750  1  28:2  66        BEGIN
9751  1  28:3  66          CH := MENUSTR[ INDX];
9752  1  28:3  75          IF CH =  '/' THEN
9753  1  28:4  82            BEGIN
9754  1  28:5  82              IF MENUWIN^.HEAD.HCURSOR > HPOS THEN
9755  1  28:6  94                HPOS := MENUWIN^.HEAD.HCURSOR;
9756  1  28:5 103              PRINTCR( MENUWIN);
9757  1  28:5 106              GOTSLASH := TRUE
9758  1  28:4 106            END
9759  1  28:3 110          ELSE
9760  1  28:4 112            BEGIN
9761  1  28:5 112              IF GOTSLASH THEN
9762  1  28:6 117                BEGIN
9763  1  28:7 117                  IF MENUWIN^.HEAD.VCURSOR = 0 THEN
9764  1  28:8 127                    BEGIN
9765  1  28:9 127                      MENUWIN^.HEAD.VCURSOR := VPOSMENU;
9766  1  28:9 136                      HPOSMENU := HPOS + 1
9767  1  28:8 139                    END;
9768  1  28:7 144                  MVCURSOR( MENUWIN, HPOSMENU, MENUWIN^.HEAD.VCURSOR);
9769  1  28:7 156                  IF CH >= 'a' THEN
9770  1  28:8 163                    BEGIN
9771  1  28:9 163                      IF DEFAULTX < 0 THEN
9772  1  28:0 170                        BEGIN
9773  1  28:1 170                          RTNSYMBL := 1;
9774  1  28:1 174                          DEFAULTX := MXCHOICE
9775  1  28:0 174                        END;
9776  1  28:9 180                      CH := CHR( ORD( CH) - 32)  (* CONVERT TO UPPER CASE *)
9777  1  28:8 185                    END;
9778  1  28:7 188                  CHOICES[ MXCHOICE] := CH;
9779  1  28:7 200                  GOTSLASH := FALSE;
9780  1  28:7 204                  MXCHOICE := MXCHOICE + 1
9781  1  28:6 207                END;
9782  1  28:6 212
9783  1  28:5 212              IF (CH >= CHR( 129)) AND (CH <= CHR( 159)) THEN
9784  1  28:6 229                BEGIN
9785  1  28:7 229                  MBCHAR[ 1] := CH;
9786  1  28:7 237                  INDX := INDX + 1;
9787  1  28:7 245                  MBCHAR[ 2] := MENUSTR[ INDX];
9788  1  28:7 256                  PRPICMB( MENUWIN, MBCHAR)
```

```
9789  1  28:6  260                END
9790  1  28:5  262              ELSE
9791  1  28:6  264                BEGIN
9792  1  28:7  264                  IF RTNSYMBL = 0 THEN
9793  1  28:8  271                    PRPICCH( MENUWIN, '~')
9794  1  28:7  273                  ELSE
9795  1  28:8  277                    PRPICCH( MENUWIN, CH);
9796  1  28:6  283                END;
9797  1  28:6  283
9798  1  28:5  283              IF RTNSYMBL >= 0 THEN
9799  1  28:6  290                RTNSYMBL := RTNSYMBL - 1
9800  1  28:4  293            END;
9801  1  28:4  298
9802  1  28:3  298          INDX := INDX + 1
9803  1  28:2  301        END;  (* WHILE *)
9804  1  28:2  308
9805  1  28:1  308      REPEAT
9806  1  28:2  308        GETKEY;
9807  1  28:2  310        CHOICEN := 0;
9808  1  28:2  314        FOR INDX := 0 TO MXCHOICE - 1 DO
9809  1  28:3  335          BEGIN
9810  1  28:4  335            IF (INCHAR = CHR( 13)) AND (DEFAULTX = INDX) THEN
9811  1  28:5  348              BEGIN
9812  1  28:6  348                EXITMENU( CHOICEN)
9813  1  28:5  351              END
9814  1  28:4  353            ELSE
9815  1  28:5  355              BEGIN
9816  1  28:6  355                IF (CHOICES[ INDX] = INCHAR) AND
9817  1  28:6  366                   (INCHAR <> '#') THEN
9818  1  28:7  372                  BEGIN
9819  1  28:8  372                    EXITMENU( CHOICEN)
9820  1  28:7  375                  END
9821  1  28:6  377                ELSE
9822  1  28:7  379                  BEGIN
9823  1  28:8  379                    IF CHOICES[ INDX] = '#' THEN
9824  1  28:9  392                      BEGIN
9825  1  28:0  392                        IF (INCHAR > CHR( 48)) AND
9826  1  28:0  395                           (INCHAR <= CHR( (48 + BASE04))) THEN
9827  1  28:1  403                          EXITMENU( CHOICEN + ORD( INCHAR) - 49);
9828  1  28:0  412                        CHOICEN := CHOICEN + 6
9829  1  28:9  415                      END
9830  1  28:8  420                    ELSE
9831  1  28:9  422                      CHOICEN := CHOICEN + 1
9832  1  28:7  425                  END
9833  1  28:5  430              END;
9834  1  28:3  430          END;
9835  1  28:3  440
9836  1  28:2  440        PRINTBEL (* UNITWRITE( XXX, 0, 24, XXX) *)
9837  1  28:1  440      UNTIL FALSE;
9838  1  28:1  445
9839  1  28:0  445    END;  (* MENU *)
9840  1  28:0  468
9841  1  28:0  468
9842  1  29:D    1  PROCEDURE P01001D;  (* VAR MP02 : TLONGSTR;
9843  1  29:D    3                           MP01 : INTEGER)  *)
9844  1  29:D    3
9845  1  29:D    3    VAR
```

```
9846  1  29:D    3
9847  1  29:D    3          MP03 : CHAR;
9848  1  29:D    4          MP04 : TLONGSTR;
9849  1  29:D  132
9850  1  29:D  132
9851  1  84:D    1     PROCEDURE P010054( P54MP01 : INTEGER);
9852  1  84:D    2
9853  1  84:0    0       BEGIN (* P010054 *)
9854  1  84:1    0         MP01 := MP01 + P54MP01;
9855  1  84:1    8         COPYSTR( MP02, MP02, 2, LENGTH( MP02) - 1);
9856  1  84:1   24         MP03 := MP02[ 1]
9857  1  84:0   28       END;  (* P010054 *)
9858  1  84:0   44
9859  1  84:0   44
9860  1  29:0    0     BEGIN (* P01001D *)
9861  1  29:1    0       MP03 := MP02[ 1];
9862  1  29:1    5       MP04 := '';
9863  1  29:1   12       IF MP03 = '*' THEN
9864  1  29:2   17         MP04 := MP02
9865  1  29:1   19       ELSE
9866  1  29:2   24         BEGIN
9867  1  29:3   24           IF MP03 = '+' THEN
9868  1  29:4   29             P010054( 2)
9869  1  29:3   30           ELSE IF MP03 = '-' THEN
9870  1  29:5   39             P010054( 4);
9871  1  29:5   42
9872  1  29:3   42           IF (MP03 = 'A') OR (MP03 = 'E') OR (MP03 = 'I') OR
9873  1  29:3   53              (MP03 = 'O') OR (MP03 = 'U')  THEN
9874  1  29:4   63             GETMSGTX( MP04, MP01)
9875  1  29:3   66           ELSE
9876  1  29:4   70             GETMSGTX( MP04, MP01 + 1);
9877  1  29:4   77
9878  1  29:3   77           INSERTST( MP02, MP04)
9879  1  29:2   80         END;
9880  1  29:2   82
9881  1  29:1   82       MP02 := MP04
9882  1  29:1   83
9883  1  29:0   83     END;  (* P01001D *)
9884  1  29:0  100
9885  1  29:0  100
9886  1  36:D    1   PROCEDURE GETMSGTX;  (* VAR MSGSTR : TLONGSTR; *)       (* P010024 *)
9887  1  36:D    3                        (*     MSGNUM : INTEGER   *)
9888  1  36:D    3
9889  1  36:D    3     VAR
9890  1  36:D    3          HIGHINDX : INTEGER;
9891  1  36:D    4          LOWINDX  : INTEGER;
9892  1  36:D    5
9893  1  36:D    5          UNUSEDXX : INTEGER;
9894  1  36:D    6          UNUSEDYY : INTEGER;
9895  1  36:D    7          UNUSEDZZ : INTEGER;
9896  1  36:D    8
9897  1  36:D    8          CACHEBUF : TCACHEP;
9898  1  36:D    9          MSGBUF  : TCACHEP;
9899  1  36:D   10
9900  1  36:D   10
9901  1  85:D    1     PROCEDURE RDHUFF( HUFFXXXX : INTEGER);   (* P010055 *)
9902  1  85:D    2
```

```
9903  1  85:D   2      (* HUFFXXXX IS INDEX INTO  ASCIIHSH  *)
9904  1  85:D   2
9905  1  85:D   2      VAR
9906  1  85:D   2          BOTPTR   : INTEGER;
9907  1  85:D   3          CONSECXX : INTEGER;
9908  1  85:D   4          GROUPCNT : INTEGER;
9909  1  85:D   5          FIRSTMSG : INTEGER;
9910  1  85:D   6          UNUSEDXX : INTEGER;
9911  1  85:D   7          MSGI     : INTEGER;
9912  1  85:D   8          MAGICHSH : TMEMOVER;
9913  1  85:D   9          ADDXXLEN : INTEGER;
9914  1  85:D  10          OFFSET   : INTEGER;
9915  1  85:D  11
9916  1  85:D  11
9917  1  86:D   1      PROCEDURE MSGERR;             (* P010056 *)
9918  1  86:D   1
9919  1  86:0   0        BEGIN (* MSGERR *)
9920  1  86:1   0          MSGSTR := '**ERR**';
9921  1  86:1  15          EXIT( GETMSGTX)
9922  1  86:0  19        END;  (* MSGERR *)
9923  1  86:0  32
9924  1  86:0  32
9925  1  87:D   1      PROCEDURE RDHUFFTX;         (* P010057 *)
9926  1  87:D   1
9927  1  87:0   0        BEGIN (* RDHUFFTX *)
9928  1  87:1   0          IF BASCIIHU THEN     (* ASCII.HUFF EXISTS, USE IT *)
9929  1  87:2   5            BEGIN
9930  1  87:2   5
9931  1  87:2   5              (* CONVERT HUFFMAN BITS INTO ASCII STRING *)
9932  1  87:2   5
9933  1  87:3   5              UNITWRITE( BASE12, CACHEBUF^.DATA[ OFFSET],
9934  1  87:3  15                         33, GETADDR( MSGSTR));
9935  1  87:3  26              ADDXXLEN := 1 + LENGTH( MSGSTR);   (* STRLEN *)
9936  1  87:2  36            END
9937  1  87:1  36          ELSE
9938  1  87:1  38
9939  1  87:1  38            (* V---- NOT EXECUTED ----V *)
9940  1  87:2  38              BEGIN
9941  1  87:3  38                ADDXXLEN := 1 + ORD( CACHEBUF^.DATA[ OFFSET]);
9942  1  87:3  52                MOVELEFT( CACHEBUF^.DATA[ OFFSET], MSGSTR, ADDXXLEN);
9943  1  87:3  69                MAGICHSH.I := 67 * (MSGNUM MOD 51);
9944  1  87:3  79                FOR MSGI := 1 TO LENGTH( MSGSTR) DO
9945  1  87:4  97                  BEGIN
9946  1  87:5  97                    MSGSTR[ MSGI] :=
9947  1  87:5 103                     CHR(( ( ORD( MSGSTR[ MSGI]) + 16384 - MAGICHSH.I) -
9948  1  87:5 118                       (23 * MSGI)) MOD 256);
9949  1  87:4 129                  END;
9950  1  87:2 139              END;
9951  1  87:2 139            (* ^---- NOT EXECUTED ----^ *)
9952  1  87:2 139
9953  1  87:1 139          IF ADDXXLEN <= 20 THEN  (* STRLEN *)
9954  1  87:2 146            BEGIN
9955  1  87:3 146              MOVELEFT( MSGSTR, MSGBUF^.DATA[ 0], ADDXXLEN);
9956  1  87:3 161              MSGBUF^.KEY := MSGNUM;
9957  1  87:2 168            END;
9958  1  87:2 168
9959  1  87:1 168          EXIT( GETMSGTX)
```

```
 9960   1   87:1   172
 9961   1   87:0   172        END;  (* RDHUFFTX *)
 9962   1   87:0   186
 9963   1   87:0   186
 9964   1   85:0     0      BEGIN (* RDHUFF *)
 9965   1   85:1     0        ADDXXLEN := HUFFXXXX + ASCIIKBL;  (* ADDR *)
 9966   1   85:1     6        CACHEBUF := SEARCHBF( DSKCACHF, DSKCACHL, ADDXXLEN);  (* ADDR *)
 9967   1   85:1    18        IF CACHEBUF^.KEY <> ADDXXLEN THEN  (* ADDR *)
 9968   1   85:2    26          BEGIN
 9969   1   85:3    26            UNITREAD( DRIVE1, CACHEBUF^.DATA, 512, ADDXXLEN);  (* ADDR *)
 9970   1   85:3    40            CACHEBUF^.KEY := ADDXXLEN  (* ADDR *)
 9971   1   85:2    43          END;
 9972   1   85:1    45        FIRSTMSG := ASCIIHSH.PARR^[ HUFFXXXX];
 9973   1   85:1    53        OFFSET := 0;
 9974   1   85:1    56        IF HUFFXXXX = 0 THEN  (* ACCOUNT FOR 2 PTRS AT BEGIN OF HUFF *)
 9975   1   85:2    61          OFFSET := 4;        (*   BLOCK #0                          *)
 9976   1   85:1    64        IF BKRNSRCH THEN
 9977   1   85:2    69          BEGIN
 9978   1   85:2    69
 9979   1   85:2    69            (* CALCULATE ADDRESS TO MSG IN HUFF BUFFER *)
 9980   1   85:3    69            UNITWRITE( BASE12, CACHEBUF^.KEY, 27, OFFSET, MSGNUM -
 9981   1   85:3    80                                                              FIRSTMSG);
 9982   1   85:3    84            OFFSET := CACHEBUF^.KEY;    (* ADDR *)
 9983   1   85:3    90            CACHEBUF^.KEY := ADDXXLEN;  (* ADDR *)
 9984   1   85:3    95            IF OFFSET >= 0 THEN                (* ADDR *)
 9985   1   85:4   100              RDHUFFTX            (* GET MSG TEXT *)
 9986   1   85:3   100            ELSE
 9987   1   85:4   104              MSGERR
 9988   1   85:2   104          END;
 9989   1   85:2   106
 9990   1   85:2   106      (* CODE BELOW HERE IS NEVER EXECUTED (?) *)
 9991   1   85:2   106
 9992   1   85:1   106        GROUPCNT := ORD( CACHEBUF^.DATA[ 511]);
 9993   1   85:1   117        BOTPTR := 510;
 9994   1   85:1   122        WHILE GROUPCNT > 0 DO
 9995   1   85:2   127          BEGIN
 9996   1   85:3   127            IF FIRSTMSG > MSGNUM THEN
 9997   1   85:4   134              MSGERR;
 9998   1   85:3   136            CONSECXX := ORD( CACHEBUF^.DATA[ BOTPTR]);
 9999   1   85:3   145            BOTPTR := BOTPTR - 1;
10000   1   85:3   150
10001   1   85:3   150            IF MSGNUM >= FIRSTMSG + CONSECXX THEN
10002   1   85:4   159              BEGIN
10003   1   85:5   159                FOR ADDXXLEN := 1 TO CONSECXX DO   (* ADDR *)
10004   1   85:6   170                  BEGIN
10005   1   85:7   170                    OFFSET := OFFSET + ORD( CACHEBUF^.DATA[ BOTPTR]);
10006   1   85:7   181                    BOTPTR := BOTPTR - 1
10007   1   85:6   182                  END;
10008   1   85:5   193                FIRSTMSG := FIRSTMSG + CONSECXX - 1
10009   1   85:4   196              END
10010   1   85:3   200            ELSE
10011   1   85:4   202              BEGIN
10012   1   85:5   202                FOR ADDXXLEN := FIRSTMSG + 1 TO MSGNUM DO  (* ADDR *)
10013   1   85:6   217                  BEGIN
10014   1   85:7   217                    OFFSET := OFFSET + ORD( CACHEBUF^.DATA[ BOTPTR]);
10015   1   85:7   228                    BOTPTR := BOTPTR - 1
10016   1   85:6   229                  END;
```

```
10017   1   85:5   240                RDHUFFTX;
10018   1   85:4   242              END;
10019   1   85:3   242            ADDXXLEN := ORD( CACHEBUF^.DATA[ BOTPTR]);  (* ADDR *)
10020   1   85:3   251            IF ADDXXLEN < 128 THEN   (* ADDR *)
10021   1   85:4   258              BEGIN
10022   1   85:5   258                FIRSTMSG := FIRSTMSG + ADDXXLEN;  (* ADDR *)
10023   1   85:5   263                BOTPTR := BOTPTR - 1
10024   1   85:4   264              END
10025   1   85:3   268            ELSE
10026   1   85:4   270              BEGIN
10027   1   85:5   270                FIRSTMSG := FIRSTMSG + (256 * (ADDXXLEN - 128)) +  (* ADDR *)
10028   1   85:5   281                        ORD( CACHEBUF^.DATA[ BOTPTR - 1]);
10029   1   85:5   293                BOTPTR := BOTPTR - 2
10030   1   85:4   294              END;
10031   1   85:3   298            GROUPCNT := GROUPCNT - 1
10032   1   85:2   299          END;  (* WHILE *)
10033   1   85:2   305
10034   1   85:1   305        MSGERR
10035   1   85:1   305
10036   1   85:0   305      END;  (* RDHUFF *)
10037   1   85:0   328
10038   1   85:0   328
10039   1   36:0     0    BEGIN (* GETMSGTX *)
10040   1   36:1     0      IF MSGNUM = -1 THEN
10041   1   36:2     6        BEGIN
10042   1   36:3     6          MSGSTR := BASE47;
10043   1   36:3    11          EXIT( GETMSGTX)
10044   1   36:2    15        END;
10045   1   36:1    15      MSGBUF := SEARCHBF( MSGCACHF, MSGCACHL, MSGNUM);
10046   1   36:1    26      IF MSGBUF^.KEY = MSGNUM THEN
10047   1   36:2    32        BEGIN
10048   1   36:3    32          MOVELEFT( MSGBUF^.DATA, MSGSTR, 1 + LENGTH( MSGBUF^.DATA) );
10049   1   36:3    47          EXIT( GETMSGTX)
10050   1   36:2    51        END;
10051   1   36:2    51
10052   1   36:1    51      IF MSGNUM >= ASCIIHSH.PARR^[ HUFFBLK] THEN
10053   1   36:2    62        IF MSGNUM < ASCIIHSH.PARR^[ HUFFBLK + 1] THEN
10054   1   36:3    75          RDHUFF( HUFFBLK);
10055   1   36:3    79
10056   1   36:3    79      (* BINARY SEARCH ASCIIHSH TO FIND THE BLOCK WITH THE
10057   1   36:3    79         MESSAGE WE WANT.  THE BLOCK IS IN HUFF FORMAT.
10058   1   36:3    79
10059   1   36:3    79         PASS THE BLOCK ADDRESS TO RDHUFF()     *)
10060   1   36:3    79
10061   1   36:1    79      LOWINDX := 0;
10062   1   36:1    82      HIGHINDX := HFBLKCNT;
10063   1   36:1    86      WHILE HIGHINDX >= LOWINDX DO
10064   1   36:2    91        BEGIN
10065   1   36:3    91          HUFFBLK := (LOWINDX + HIGHINDX) DIV 2;
10066   1   36:3    98          IF MSGNUM < ASCIIHSH.PARR^[ HUFFBLK] THEN
10067   1   36:4   109            HIGHINDX := HUFFBLK - 1
10068   1   36:3   111          ELSE
10069   1   36:4   117            IF MSGNUM >= ASCIIHSH.PARR^[ HUFFBLK + 1] THEN
10070   1   36:5   130              LOWINDX := HUFFBLK + 1
10071   1   36:4   132            ELSE
10072   1   36:5   138              RDHUFF( HUFFBLK)
10073   1   36:2   140        END;
```

```
10074   1   36:2   144
10075   1   36:0   144     END;  (* GETMSGTX *)
10076   1   36:0   158
10077   1   36:0   158
10078   1   37:D     1   PROCEDURE DISP1MSG;  (* MSGWIN : TWINDOWP;   P010025
10079   1   37:D     3                           MSGNUM : INTEGER  *)
10080   1   37:D     3
10081   1   37:D     3     VAR
10082   1   37:D     3          MESSAGE : TLONGSTR;
10083   1   37:D   131
10084   1   37:0     0     BEGIN
10085   1   37:1     0       GETMSGTX( MESSAGE, MSGNUM);
10086   1   37:1     5       PRPICMB( MSGWIN, MESSAGE)
10087   1   37:0     8     END;
10088   1   37:0    22
10089   1   37:0    22
10090   1   38:D     1   PROCEDURE DISP1LIN;  (* MSGWIN: TWINDOWP;   MSGNUM: INTEGER  *) (* P010026 *)
10091   1   38:D     3
10092   1   38:D     3     VAR
10093   1   38:D     3          MESSAGE : TLONGSTR;
10094   1   38:D   131
10095   1   38:0     0     BEGIN
10096   1   38:1     0       GETMSGTX( MESSAGE, MSGNUM);
10097   1   38:1     5       CENTSTR( MSGWIN, MESSAGE)
10098   1   38:0     8     END;
10099   1   38:0    22
10100   1   38:0    22
10101   1   40:D     3   FUNCTION MBSTRLEN;  (*    (MP01 : TLONGSTR) : INTEGER;   P010028  *)
10102   1   40:D   132
10103   1   40:D   132     VAR
10104   1   40:D   132          MP84 : INTEGER;
10105   1   40:D   133          MP85 : INTEGER;
10106   1   40:D   134          MP86 : INTEGER;
10107   1   40:D   135
10108   1   40:0     0     BEGIN
10109   1   40:1     0       MP84 := 0;
10110   1   40:1     9       MP85 := LENGTH( MP01);
10111   1   40:1    16       MP86 := 1;
10112   1   40:1    20       WHILE  MP86 <= MP85 DO
10113   1   40:2    29         BEGIN
10114   1   40:2    29
10115   1   40:2    29           (* CHECK FOR FIRST BYTE OF MULTI-BYTE CHARACTER *)
10116   1   40:2    29
10117   1   40:3    29           IF (ORD( MP01[ MP86]) >= 129) AND (ORD( MP01[ MP86]) <= 159) THEN
10118   1   40:4    52             MP86 := MP86 + 1;
10119   1   40:3    60           MP84 := MP84 + 1;
10120   1   40:3    68           MP86 := MP86 + 1
10121   1   40:2    71         END;
10122   1   40:1    78       MBSTRLEN := MP84
10123   1   40:0    78     END;
10124   1   40:0    98
10125   1   40:0    98
10126   1   39:D     3   FUNCTION MENUINDX;  (* MP04 : TWINDOWP            P010027
10127   1   39:D     5                          MP03 : INTEGER) INTEGER;    *)
10128   1   39:D     5
10129   1   39:D     5     VAR
10130   1   39:D     5          MP05 : TLONGSTR;
```

```
10131   1   39:D   133
10132   1   39:0     0     BEGIN  (* MENUINDX *)
10133   1   39:1     0       GETMSGTX( MP05, MP03);
10134   1   39:1     5       MENUINDX := MENU( MP04, MP05)
10135   1   39:0     8     END;   (* MENUINDX *)
10136   1   39:0    26
10137   1   39:0    26
10138   1   41:D     3   FUNCTION GTTITWIN; (* ( MSGINDX  : INTEGER;     P010029
10139   1   41:D     7                         VPOS     : INTEGER;
10140   1   41:D     7                         VSIZE    : INTEGER;
10141   1   41:D     7                         PRIORITY : INTEGER) : TWINDOWP;  *)
10142   1   41:D     7
10143   1   41:D     7     VAR
10144   1   41:D     7         MSG    : TLONGSTR;
10145   1   41:D   135         MSGW   : INTEGER;
10146   1   41:D   136         TITWIN : TWINDOWP;
10147   1   41:D   137
10148   1   41:0     0     BEGIN (* GTTITWIN *)
10149   1   41:1     0       GETMSGTX( MSG, MSGINDX);
10150   1   41:1     5       MSGW := (1 + MBSTRLEN( MSG)) DIV 2;
10151   1   41:1    18       TITWIN := GETWIN( 19 - MSGW, VPOS,
10152   1   41:1    24                         MSGW + MSGW + 2, VSIZE + 2,
10153   1   41:1    36                         PRIORITY, TRUE);
10154   1   41:1    45       PRPICMB( TITWIN, MSG);
10155   1   41:1    52       GTTITWIN := TITWIN
10156   1   41:0    52     END;  (* GTTITWIN *)
10157   1   41:0    70
10158   1   41:0    70
10159   1   42:D     1   PROCEDURE POOLGOLD;  (* ( CHARI : INTEGER);         P01002A *)
10160   1   42:D     2
10161   1   42:D     2     VAR
10162   1   42:D     2         CHARX    : INTEGER;
10163   1   42:D     3         ZEROGOLD : TWIZLONG;
10164   1   42:D     6         TOTLGOLD : TWIZLONG;
10165   1   42:D     9
10166   1   42:0     0     BEGIN  (* POOLGOLD *)
10167   1   42:1     0       ZEROGOLD.LOW  := 0;
10168   1   42:1     3       ZEROGOLD.MID  := 0;
10169   1   42:1     6       ZEROGOLD.HIGH := 0;
10170   1   42:1     9       TOTLGOLD := ZEROGOLD;
10171   1   42:1    15       FOR CHARX := 0 TO BASE04 - 1 DO
10172   1   42:2    28         BEGIN
10173   1   42:3    28           ADDLONGS( TOTLGOLD, CHARACTR[ CHARX].GOLD);
10174   1   42:3    40           CHARACTR[ CHARX].GOLD := ZEROGOLD
10175   1   42:2    48         END;
10176   1   42:1    59       CHARACTR[ CHARI].GOLD := TOTLGOLD
10177   1   42:0    67     END;  (* POOLGOLD *)
10178   1   42:0    86
10179   1   42:0    86
10180   1   57:D     1   PROCEDURE P010039;  (* MP06 : TLONGSTR;
10181   1   57:D   135                         MP05 : INTEGER;
10182   1   57:D   135                         MP04 : INTEGER;
10183   1   57:D   135                         MP03 : INTEGER;
10184   1   57:D   135                         MP02 : INTEGER;
10185   1   57:D   135                         MP01 : INTEGER *)
10186   1   57:D   135
10187   1   57:D   135     VAR
```

```
10188   1   57:D   135          MP87 : TWINDOWP;
10189   1   57:D   136          MP88 : CHAR;
10190   1   57:D   137
10191   1   57:D   137
10192   1   88:D     1     PROCEDURE P010058( MP01 : INTEGER);
10193   1   88:D     2
10194   1   88:0     0       BEGIN  (* P010058 *)
10195   1   88:1     0         MP87^.HEAD.VCURSOR := 15;
10196   1   88:1    10         PRINTNUM( MP87, MP01, 6);
10197   1   88:1    18         PRINTCR(  MP87)
10198   1   88:0    22       END;   (* P010058 *)
10199   1   88:0    36
10200   1   88:0    36
10201   1   57:0     0     BEGIN (* P010039 *)
10202   1   57:1     0       MP88 := INCHAR;
10203   1   57:1     9       MP87 := GETWIN( 0, 10 - (MP05 DIV 2), 40, (4 + MP05), 30, TRUE);
10204   1   57:1    28       CENTSTR( MP87, MP06);
10205   1   57:1    35       IF MP05 >= 1 THEN
10206   1   57:2    40         P010058( MP04);
10207   1   57:1    43       IF MP05 >= 2 THEN
10208   1   57:2    48         P010058( MP03);
10209   1   57:1    51       IF MP05 >= 3 THEN
10210   1   57:2    56         P010058( MP02);
10211   1   57:1    59       IF MP05 >= 4 THEN
10212   1   57:2    64         P010058( MP01);
10213   1   57:1    67       CENTSTR( MP87, 'PRESS RETURN');
10214   1   57:1    87       GETCR;
10215   1   57:1    89       DELWIN( MP87, TRUE);
10216   1   57:1    95       INCHAR := MP88
10217   1   57:0    95     END;  (* P010039 *)
10218   1   57:0   112
10219   1   57:0   112
10220   1   58:D     1   PROCEDURE P01003A;  (* ( MP01 : TLONGSTR)  *)
10221   1   58:D   130
10222   1   58:D   130     VAR
10223   1   58:D   130          MP82 : TWINDOWP;
10224   1   58:D   131
10225   1   58:0     0     BEGIN (* P01003A *)
10226   1   58:1     0       MP82 := GETWIN( 0, 0, 40, 3, 30, TRUE);
10227   1   58:1    18       CENTSTR( MP82, MP01);
10228   1   58:1    25       PAUSE2;
10229   1   58:1    27       DELWIN( MP82, FALSE)
10230   1   58:0    31     END;  (* P01003A *)
10231   1   58:0    46
10232   1   58:0    46
10233   1   59:D     3   FUNCTION FINDFILE;  (* ( DRIVE  : INTEGER;            P01003B *)
10234   1   59:D    46                       (*   FILENM : STRING) : INTEGER;          *)
10235   1   59:D    46
10236   1   59:D    46     VAR
10237   1   59:D    46          DIR: ARRAY[ 0..76] OF DIRENTRY;  (* ONLY 76 FILE ENTRIES (?) *)
10238   1   59:D  1047          FILEI : INTEGER;
10239   1   59:D  1048          FILEX : INTEGER;
10240   1   59:D  1049
10241   1   59:0     0     BEGIN
10242   1   59:1     0       FILLCHAR( DIR, SIZEOF( DIR), 0);
10243   1   59:1    14       UNITREAD( DRIVE, DIR, SIZEOF( DIR), 2);
10244   1   59:1    25                                        (* 2 = VOLUME HEADER +
```

```
10245   1   59:1    25                                          UP TO 77 FILE ENTRIES *)
10246   1   59:1    25       IF IORESULT <> 0 THEN
10247   1   59:2    31         FINDFILE := -ABS( IORESULT)   (* THIS IS A BUG *)
10248   1   59:1    34       ELSE
10249   1   59:2    39         BEGIN
10250   1   59:3    39           FILEI := 0;
10251   1   59:3    43           FOR FILEX := 1 TO DIR[ 0].FILECNT DO
10252   1   59:4    66             BEGIN
10253   1   59:5    66               WITH DIR[ FILEX] DO
10254   1   59:6    76                 BEGIN
10255   1   59:7    76                   IF (FILEKIND.FT >= BADBLK) AND
10256   1   59:7    86                      (FILEKIND.FT <= FOTOFILE) THEN
10257   1   59:8    99                      IF FILENAME = FILENM THEN
10258   1   59:9   110                        FILEI := FILEX;
10259   1   59:6   116                 END;
10260   1   59:4   116             END;
10261   1   59:3   126           IF FILEI = 0 THEN
10262   1   59:4   133             FINDFILE := -9
10263   1   59:3   133           ELSE
10264   1   59:4   139             FINDFILE := DIR[ FILEI].FIRSTBLK
10265   1   59:2   146         END
10266   1   59:0   149     END;
10267   1   59:0   164
10268   1   59:0   164
10269   1   59:0   164 (*$I WIZ4A: WIZ5.TEXT *)
10270   1   59:0   164
10271   1    1:0     0 BEGIN  (* WIZARDRY MAIN LINE *)
10272   1    1:0     0
10273   1    1:1     0   BASE12 := 13;
10274   1    1:1     5   BAPPLE   := FINDFILE( DRIVE1, 'IS.APPLE') > 0;
10275   1    1:1    26   BLINES24 := FINDFILE( DRIVE1, 'LINES.24') > 0;
10276   1    1:1    47   BCACHE   := FINDFILE( DRIVE1, 'HAS.CACHE') > 0;
10277   1    1:1    69   BSTROPS  := FINDFILE( DRIVE1, 'HAS.STROPS') > 0;
10278   1    1:1    92   BRELOC   := FINDFILE( DRIVE1, 'SYSTEM.RELOC') > 0;
10279   1    1:1   117   BKRNSRCH := FINDFILE( DRIVE1, 'HAS.KRNSRCH') > 0;
10280   1    1:1   141
10281   1    1:1   141   MAZEBLCK := FINDFILE( DRIVE1, 'MAZE.INFO');
10282   1    1:1   161   IF BRELOC THEN
10283   1    1:2   166     BEGIN
10284   1    1:3   166       BASE09 := 0;
10285   1    1:3   169       UNITWRITE( BASE12, BASE09, 31, FINDFILE(4, 'SYSTEM.RELOC'));
10286   1    1:3   198
10287   1    1:3   198       BASE09 := 1;
10288   1    1:3   201       UNITWRITE( BASE12, BASE09, 31, BASE09)
10289   1    1:3   211         (* DISK TABLE INIT:  $5D..$65 *)
10290   1    1:3   211         (*             $F2:  #$80     *)
10291   1    1:3   211         (*             $2F:  #$80     *)
10292   1    1:2   211     END;
10293   1    1:2   211
10294   1    1:1   211   BASE09 := 0;
10295   1    1:1   214
10296   1    1:1   214   REPEAT     (* SEEMS LIKE THIS ONE IS A "DEAD" REPEAT LOOP *)
10297   1    1:1   214
10298   1    1:2   214     REPEAT
10299   1    1:3   214       KANJIREA( NIL, BASE47, BASE09);  (* P010601 *)
10300   1    1:3   221       IF BASE09 = 1 THEN
10301   1    1:4   226         DOCOPY                         (* DOCOPY   *)
```

```
10302   1    1:3   226       ELSE IF BASE09 = 2 THEN
10303   1    1:5   236         DOCACHE                          (* DOCACHE  *)
10304   1    1:5   236
10305   1    1:5   236                                          (* DOCACHE SETS:   *)
10306   1    1:5   236                                          (*   BASE09 = 0    *)
10307   1    1:5   236                                          (*   BASE13 = 1    *)
10308   1    1:2   236     UNTIL BASE09 = 0;
10309   1    1:2   244
10310   1    1:2   244     REPEAT
10311   1    1:3   244       CASE BASE13 OF
10312   1    1:3   248                              22:  CASE BASE26 OF
10313   1    1:4   252                                0, 1, 2:  CAMP;
10314   1    1:4   257                                       4:  UTILITIE;
10315   1    1:4   262                                       3:  P010901;
10316   1    1:4   267                                   END;
10317   1    1:3   286                     15, 21, 24:  P010901;
10318   1    1:3   291            1, 3, 4, 8, 14, 23:  SHOPS;  (* P010801; *)
10319   1    1:3   296              7, 11, 12, 13, 18:  UTILITIE;
10320   1    1:3   301                               5:  RUNNER;
10321   1    1:3   306                           6, 19:  P010101;
10322   1    1:3   311          2, 9, 10, 16, 20, 25:  CAMP;
10323   1    1:3   316       END
10324   1    1:3   374
10325   1    1:2   374     UNTIL BASE13 = 0;
10326   1    1:2   380
10327   1    1:2   380     EXIT( WIZARDRY);
10328   1    1:2   384
10329   1    1:2   384     BASE09 := 5
10330   1    1:2   384
10331   1    1:1   384   UNTIL FALSE
10332   1    1:0   387 END.
```