

DISPLAY ELEMENT LIBRARY

```

ARRAY DEL (1:NUMBER OF ELEMENT SHAPES) OF DEL-ENTRY
RECORD DEL-ENTRY [DISPLAY ELEMENT LIBRARY]
.XTOL, [X TOLERANCE]
.YTOL, [Y TOLERANCE]
.NSE, [NUMBER OF SUB-ELEMENTS]
.LENGTH: INTEGER [LENGTH OF LIGHT STRING]
ARRAY DEL.SUB (1:NSE) OF DEL.SUB-ENTRY
RECORD DEL.SUB-ENTRY [DISPLAY ELEMENT LIBRARY]
.CLASS, [SUB-ELEMENT TYPE]
.XFILL, [X FILL OFFSET]
.YFILL, [Y FILL OFFSET]
.XTEXT, [X TEXT OFFSET]
.YTEXT: INTEGER [Y TEXT OFFSET]
.TEXT: STRING(LENGTH) [LIGHT STRING]

```

TEXT ELEMENT LIBRARY

ARRAY TEL (1:NUMBER OF TEXT STRINGS) OF TEL-ENTRY
RECORD TEL-ENTRY [TEXT ELEMENT LIBRARY]
.LENGTH: INTEGER (LENGTH OF TEXT STRING)
.TEXT: STRING (LENGTH) (TEXT STRING)
.EOI: INTEGER (END OF TEXT STRING)

```

10 MESSAGE TABLES
11
12 ARRAY MESSAGE (1:16) OF MESSAGE-RECORD
13 RECORD MESSAGE-RECORD
14   .TEXT: STRING (VARIABLE)
15   .EDM: INTEGER [END OF MESSAGE]
16
17 ARRAY RGB (1:17) OF RGB-RECORD
18 RECORD RGB-RECORD
19   .ACTION, [ACTION CODE]
20   .DESTINATION, [DESTINATION APT/DHT]
21   .LENGTH, [LENGTH OF REMAINING RGB]
22   .MODIFIER, [CAMERA NUMBER, IF APPLICABLE]
23   .CODE, [RGB CODE]
24   .PARAMETERS: INTEGER [RGB PARAMETERS]
25
26 ARRAY STACK (1:16) OF STACK RECORD
27 RECORD STACK-RECORD
28   .IFLAG, [IMMEDIATE ACTION FLAG]
29   .DFLAG, [DELAYED ACTION FLAG]
30   .ELEMENT: INTEGER [ELEMENT MCT ADDRESS]
31
32 ATAD
33 PROC CTPAPT (ALT INPUT-MESSAGE)
34
35 DATA
36 RECORD INPUT-MESSAGE
37   .TEXT(24):STRING
38
39 SCALAR X,Y,MSG-ID,MSG-FLAG :INTEGER
40
41 CONSTANT MINUS1 := -1 :INTEGER
42
43 ATAD
44
45 <REENABLE INTERRUPTS AFTER DISPATCH>
46 RUN MHRMAP (USE MINUS1)
47
48 CASE
49   <TYPE OF INPUT-MESSAGE>
50
51   SELECT (FULL-INPUT-MESSAGE.HEADER.ORIG-TASK = <DHTCIP>)
52     RUN DCDCIP (ALT FULL-INPUT-MESSAGE, CTPCB)
53     RUN CNVCTP (ALT CTPCB)
54     X := CTPCB.XL
55     Y := CTPCB.YL
56     RUN CTSPT (ALT X,Y)
57
58   SELECT (INPUT-MESSAGE.TEXT(1:1) = 254)
59     MSG-FLAG := INPUT-MESSAGE.TEXT(2:1)
60     IF
61       INPUT-MESSAGE.TEXT(3:1) < '0'
62     THEN
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2
```

```

MSG-ID := INPUT-MESSAGE.TEXT(1:2)
ELSE
MSG-ID := <ADDRESS OF INPUT-MESSAGE.TEXT(4:1)>
F)
RUN MSGDSP (USE MSG-FLAG, MSG-ID)
RUN REBUF (ALT FULL-INPUT-MESSAGE)
ELSE
RUN REBUF (ALT FULL-INPUT-MESSAGE)
ESAC
CORP [ACTUALLY, BRANCH TO DISPAT]

```

A-8

```
PROC SYSREQ (USE ROB; ALT REPLY-ADDR)
DATA
RECORD INPUT-MESSAGE
.TEXT(24) :STRING
SCALAR RC,INTVL,REPLY-ADDR :INTEGER
SCALAR GOT-BUF :LOGICAL
ATAD
GOT-BUF := FALSE
WHILE
NOT GOT-BUF
DO
RUN GETBUF (ALT FULL-INPUT-MESSAGE, GOT-BUF)
IF
NOT GOT-BUF
THEN
RC := RC-TIMEOUT
WHILE
RC = RC-TIMEOUT
DO
RUN WAITAC (USE <ACTBUF>; ALT RC)
IF
RC <> RC-TIMEOUT AND RC <> RC-SUCCESS
THEN
RUN ABORT
FI
OD
FI
OD
FULL-INPUT-MESSAGE.HEADER.DEST-TASK := ROB.DST
INPUT-MESSAGE.TEXT(1:2) = ROB.CMD
N := ROB.LEN / 2 - 1
<COPY N WORDS FROM ROB.PRM TO INPUT-MESSAGE.TEXT(3:2N)>
```

A-9

```

2
3
4 CASE
5     RGB.AC
6
7     SELECT (RGB.AC = 0) [ORDINARY SEND]
8         RUN SEND (ALT FULL-INPUT-MESSAGE)
9
10    SELECT (RGB.AC > 0) [SENDAC]
11        RUN SENDAC (USE RGB.AC; ALT FULL-INPUT-MESSAGE, REPLY-ADDR, RC)
12    IF
13        RC <> RC-SUCCESS AND RC <> RC-TIMEOUT
14    THEN
15        RUN ABORT
16    FI
17
18    WHILE
19        RC = RC-TIMEOUT
20    DO
21        RUN WAITAC (USE RGB.AC; ALT REPLY-ADDR, RC)
22    IF
23        RC <> RC-SUCCESS AND RC <> RC-TIMEOUT
24    THEN
25        RUN ABORT
26    FI
27
28    RUN MRRAP (USE MINUS1)
29
30    DO
31
32    SELECT (RGB.AC < 0) [SENDTM]
33        INTVL := - RGB.AC * 6 [GIVEN IN .1 SEC INTERVALS]
34        RUN SENDTM (USE INTVL; ALT FULL-INPUT-MESSAGE, RC)
35    IF
36        RC <> RC-SUCCESS
37    THEN
38        RUN ABORT
39    FI
40
41    RUN MRRAP (USE MINUS1)
42
43    ESAC
44
45 CORP
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610

```

PROC DDCCTP (ALT FULL-INPUT-MESSAGE, CTPCB)

DATA

RECORD INPUT-MESSAGE
.TEXT(24) :STRING

SCALAR NUM-STRING(5) :STRING
SCALAR NUM,LEN :INTEGER

ATAD

LEN := 5
NUM-STRING := INPUT-MESSAGE.TEXT(1:5)
RUN ASCBIN (USE NUM-STRING,LEN; ALT NUM,RC)
CTPCB.XT := NUM

NUM-STRING := INPUT-MESSAGE.TEXT(6:5)
RUN ASCBIN (USE NUM-STRING,LEN; ALT NUM,RC)
CTPCB.YT := NUM

RUN RELBUF (ALT FULL-INPUT-MESSAGE)

CCRP

PROC CNVCCTP (ALT CTPCB)

CTPCB.XL := CTPCB.XLMN + ABS(CTPCB.XT - CTPCB.XTMN) * CTPCB.DXL / CTP
CTPCB.YL := CTPCB.YLMN + ABS(CTPCB.YT - CTPCB.YTMN) * CTPCB.DYL / CTP

CCRP

```

PROC INTMHR          (INITIATES THE MENU HANDLER MODULE)
  DATA
    SCALAR CLRCCM: STRING(V)
    SCALAR COLORS: STRING(V)
    CONSTANT FSTMEN: INTEGER          (FIRST MENU ID)
  ATAD
    ICH := <LIGHT ICB ADDRESS>
    RUN LIGHT (USE CLRCCM)          (CLEAR THE MONITOR)
    RUN LIGHT (USE COLORS)         (DEFINES THE COLORS)
    RQMEND := FSTMEN
    RUN DRMENU
  CORP

```



```

PROC MINIT                                [RESETS MENU HANDLER VALUES]
DATA
  SCALAR LOOP,
        LOOP1,
        ELEMENT,
        MENU: INTEGER
  CONSTANT NUMMEN, [NUMBER OF MENUS]
        ESTENT: INTEGER [NUMBER OF EST ENTRIES]
  ARRAY DEFEST (1:ESTENT) OF DEFEST-ENTRY
  RECORD DEFEST-ENTRY
        .FON,
        .FPRE: INTEGER

ATAD
MENU := MDT
LOOP := NUMMEN [RESET MENU DICTIONARY TABLE]
WHILE LOOP > 0
  DO
    <RESET MDT VALUES>
    ELEMENT := MDT(MENU).MCT
    LOOP1 := MDT(MENU).NEL
    WHILE LOOP1 > 0 [RESET MENU CONTROL TABLE ELEMENTS]
      DO
        <RESET MCT(ELEMENT) VALUES>
        LOOP := LOOP - 1
        ELEMENT := ELEMENT + MCTLEN
      OD
    LOOP := LOOP - 1
    MENU := MENU + MDTLEN
  OD
ESADDR := EST
LOOP := ESTENT
LOOP1 := 0
WHILE LOOP > 0 [RESET ENTITY STATUS TABLE]
  DO
    LOOP1 := LOOP1 + 1
    EST(ESADDR).CB := 0
    EST(ESADDR).FON := DEFEST(LOOP1).FON
    EST(ESADDR).FPRE := DEFEST(LOOP1).FPRE
    ESADDR := ESADDR + ESTLEN
    LOOP := LOOP - 1
  OD
  <RESET SYSTEM PARAMETERS>
  <LOAD PRESET COORDINATES FROM CONFERENCE INITIALIZATION
    TABLE TO MENU C3 ELEMENTS>
  <PFLAG(1-2,1-2) := 0>
CCRP

```

```

PROC DRAW-MENU (USE MENU)
  DATA
    SCALAR CURR-MENU: INTEGER (CURRENT MENU DISPLAYED)
    SCALAR N-ICBS: INTEGER (NUMBER OF ICBS AVAILABLE)
    SCALAR @ACT: INTEGER
    SCALAR LOOP: INTEGER
    SCALAR POS: INTEGER
    SCALAR MIN-USAGE: INTEGER
    SCALAR LOOP-MIN: INTEGER
    SCALAR DROP-MENU: INTEGER
  ATAD
  IF
    MENU = ESTMEN
  THEN
    RUN MINIT
  FI
  IF
    CURR-MENU = MENU
  THEN
    RUN REDRAW-MENU (USE MENU)
  FI
  IF
    MENU-DICT (MENU).FLAGS.ICB <> 0
  THEN
    RUN SWAP-MENU (USE MENU)
  FI
  POS := 0
  LOOP := 0
  LOOP-MIN := 1
  MIN-USAGE := 999
  DO
    LOOP := LOOP + 1
    IF
      IMAGE-ARRAY (LOOP).MENU-ID = 0 (EMPTY ICB)
    THEN
      POS := LOOP
    ELSE
      IF
        IMAGE-ARRAY (LOOP).USAGE < MIN-USAGE
      THEN
        MIN-USAGE := IMAGE-ARRAY (LOOP).USAGE
        LOOP-MIN := LOOP
      FI
    FI
  UNTIL
    POS <> 0 OR LOOP = N-ICBS
  OD
  IF
    POS = 0 (NO EMPTY SLOTS)
  THEN
    POS := LOOP-MIN
    DROP-MENU := IMAGE-ARRAY (POS).MENU-ID
    MENU-DICT (DROP-MENU).FLAGS.ICB := 0

```

A-14

```

10      MCT := MENU-DICT(CURR-MENU).MCT
11      LOOP := 0
12      WHILE
13          LOOP =< MENU-DICT (CURR-MENU).N-MCT-ELEMENTS
14      DO
15          LOOP := LOOP + 1
16          MCT-->MCT(LOOP).CURR-STATUS := 0
17      OD
18      FI
19      MENU-DICT(MENU).FLAGS.ICB := POS
20      MENU-DICT(MENU).PREV-MENU := CURR-MENU
21      IMAGE-ARRAY(POS).MENU-ID := MENU
22      IMAGE-ARRAY(POS).USAGE := 1
23      RUN LIGHT (< SET ICB NUMBER, IMAGE-ARRAY(POS).ICB-NU>)
24      LOOP := 0
25      WHILE
26          LOOP =< MENU-DICT(MENU).N-MCT-ELEMENTS
27      DO
28          LOOP := LOOP + 1
29          RUN DRAW-ELEMENT (USE MENU, LOOP)
30      OD
31      CURR-MENU := MENU
32      FI
33      MENU = CO3MNU AND CO3DRW = 0
34      THEN
35          RUN SIDFLI
36          CO3DRW := CO3DRW + 1
37      FI
38      RUN LIGHT (< SET ICB TO DISPLAY>)
39      CCRP

```

```
PROC GMADCR (USE MENUID)      [GET MDI ADDRESS]
  DATA
  ATAL
  MENUAL := (MENUID - 1) * MDLEN + MDI
CORP
```

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54

```
11 PROC MNEXT (ALT PDMENU)          (GET NEXT MENU ID)
12   DATA
13   ATAC
14   IF
15
16       PDI(CURRMA).NMH <> 0
17   THEN
18       PDMENU := ALT(CURRMA).NMH
19       PDI(CURRMA).NMH := 0
20   ELSE
21       PDMENU := #LT(CURRMA).NM
22   FI
23 CCRF
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
PROC ESTADD (USE MEADDR, ALT ESADDR)
DATA
  SCALAR ID: INTEGER
  ATAD
  ID := MCT(MEADDR).EID
  ESADDR := (ID - 1) * ESTLEN + ES1
CORP
```

PROC RECDRAW-MENU (USE MENU)

DATA

SCALAR CURR-MENU: INTEGER

SCALAR LOOP: INTEGER

ATAD

LOOP := 0

WHILE

LOOP <= MENU-DICT(MENU).N-ELEMENTS

DO

LOOP := LOOP + 1

RUN DRAW-ELEMENT (USE MENU, LOOP)

OD

CCRP

PROC SWAP-MENU (USE MENU)

DATA

SCALAR CURR-MENU: INTEGER
SCALAR IMAGE-LOC: INTEGER
SCALAR CURR-ICB: INTEGER (CURRENT ICB IN DISPLAY)
SCALAR LOOP: INTEGER

ATAD

MENU-DICT(MENU).PREV-MENU := CURR-MENU
IMAGE-LOC := MENU-DICT(MENU).FLAGS.ICB
IMAGE-ARRAY(IMAGE-LOC).USAGE := IMAGE-ARRAY(IMAGE-LOC).USAGE + 1
RUN LIGHT [<SET DRAW ICB TO IMAGE-LOC>]
LOOP := 0
WHILE

LOOP =< MENU-DICT(MENU).N-MCT-ELEMENTS

DO

LOOP := LOOP + 1
RUN DRAW-ELEMENT (USE MENU, LOOP)

OD

CURR-ICB := IMAGE-LOC
CURR-MENU := MENU
RUN LIGHT [<SET CURR-ICB TO DISPLAY>]

CCRP


```

PROC DRAM-ELEMENT (USE MENU, ELEM)
  DATA
    SCALAR @MCT: INTEGER
    SCALAR EST-ENTRY: INTEGER
  ATAD
    @MCT := MENU-DICT(MENU).MCT [ @MCT WILL BE USED AS MCT ]
    EST-ENTRY := @MCT-->MCT(ELEM).ENTRY.10
  IF
    EST-ENTRY <> 0
  THEN
    @MCT-->MCT(ELEM).NEXT-STATUS.BODY := EST(EST-ENTRY).FLAGS.ON
    @MCT-->MCT(ELEM).NEXT-STATUS.INHIBIT-DRAW := EST(EST-ENTRY).FLAGS.PRESENT
  FI
  IF
    @MCT-->MCT(ELEM).CURR-STATUS <> @MCT-->MCT(ELEM).NEXT-STATUS
  THEN
    IF
      @MCT-->MCT(ELEM).CURR-STATUS.BURDER <> @MCT-->MCT(ELEM).NEXT-STATUS.BURDER
    THEN
      RUN CHANGE-BURDER (USE @MCT,ELEM)
    FI
    IF
      @MCT-->MCT(ELEM).CURR-STATUS.BODY <> @MCT-->MCT(ELEM).NEXT-STATUS.BODY
    THEN
      RUN CHANGE-BODY (USE @MCT,ELEM)
    FI
    IF
      @MCT-->MCT(ELEM).CURR-STATUS.TEXT <> @MCT-->MCT(ELEM).NEXT-STATUS.TEXT
    THEN
      RUN CHANGE-TEXT (USE @MCT-->MCT(ELEM),ELEM)
    FI
    IF
      @MCT-->MCT(ELEM).CURR-STATUS.INHIBIT-DRAW <>
      @MCT-->MCT(ELEM).NEXT-STATUS.INHIBIT-DRAW
    THEN
      RUN ZAP-OUT (USE @MCT,ELEM)
    FI
  FI
  @MCT-->MCT(ELEM).CURR-STATUS := @MCT-->MCT(ELEM).NEXT-STATUS
CORP
  
```

A-21

```

PROC CHANGE-BORDER (USE @MCT,ELEM)
DATA
  SCALAR @DEL: INTEGER
  SCALAR @ELEM: INTEGER
  SCALAR COLOR: INTEGER [COLOR TO CHANGE BORDER]
  SCALAR T-COLOR: INTEGER [DEFAULT TOUCHED COLOR]
  SCALAR UT-COLOR: INTEGER [DEFAULT UNTOUCHED COLOR]
  SCALAR LOOP: INTEGER
ATAD
  COLOR := T-COLOR
IF
  @MCT-->MCT(ELEMENT).NEXT-STATUS.BORDER = 0
THEN
  COLOR := UT-COLOR
FI
@DEL := @MCT-->MCT(ELEM).DEL-ID
@ELEM := @DEL-->DEL.SUB-ELEMENT
RUN LIGHT [< SET 'HERE' COORDINATES TO @MCT.HERE>]
LOOP := 0
DO
  LOOP := LOOP + 1
  LOOP > @DEL-->DEL.NO-SUB-ELEMENTS OR @ELEM-->SUB-ELEMENT(LOOP).CLASS = 1
  LOOP > @DEL-->DEL.NO-SUB-ELEMENTS
  LOOP > @DEL-->DEL.NO-SUB-ELEMENTS
  RUN ERROR < PERFORM ERROR HANDLING >
  < CONSTRUCT LIGHT STRING USING @ELEM-->SUB-ELEMENT(LOOP).FILL.OFFSET
  AND COLOR>
  RUN LIGHT [< USING CONSTRUCTED LIGHT STRING >]
FI
CORP

```

```

PROC CHANGE-BODY (USE @MCT,ELEM)
  DATA
    SCALAR @DEL: INTEGER
    SCALAR @ELEM: INTEGER
    SCALAR COLOR: INTEGER
    SCALAR EST-ENTRY: INTEGER
    SCALAR CONTROLLER: INTEGER
  ATAD
    @DEL := @MCT-->MCT(ELEM).DEL-ID
    @ELEM := @DEL-->DEL.SUB-ELEMENT
    EST-ENTRY := @MCT-->MCT(ELEM).ENTITY.ID
    COLOR := @MCT-->MCT(ELEM).COLOR.BASE
    IF
      @MCT-->MCT(ELEM).NEXT-STATUS.BODY = 1 [ACTIVE]
    THEN
      IF
        EST(EST-ENTRY).IYPE = 0 [CONTROLLED]
      THEN
        CONTROLLER := EST(EST-ENTRY).CONTROLLED-BY
        COLOR := EST(CONTROLLER).COLOR
        EST(EST-ENTRY).COLOR := COLOR
      ELSE
        COLOR := EST(EST-ENTRY).COLOR [NOT CONTROLLED]
      FI
    FI
    RUN LIGHT [< SET 'HERE' COORDINATES TO @MCT,HERE >]
    LOOP := 0
    WHILE
      LOOP <= @DEL-->DEL.NO-SUB-ELEMENTS
    DO
      LOOP := LOOP + 1
      IF
        @ELEM-->SUB-ELEMENT(LOOP).CLASS = 2
      THEN
        < CONSTRUCT LIGHT STRING WITH COLOR AND
        @DEL.SUB-ELEMENT(LOOP).FILL-OFFSETS >
        RUN LIGHT [< USE CONSTRUCTED STRING >]
      FI
    DD
      @MCT-->MCT(ELEM).COLOR.CURRENT := COLOR

```

```

PROC CHANGE-TEXT (USE @ELEM, @ELEMENT)
DATA
  SCALAR @DEL: INTEGER
  SCALAR @TEXT: INTEGER
  SCALAR @ELEM: INTEGER
  SCALAR LOOP: INTEGER
  SCALAR CHARACTER-WIDTH: INTEGER
  SCALAR CHARACTER-HEIGHT: INTEGER

  @DEL := @MCT-->MCT(ELEMENT).DEL-ID
  @TEXT := @MCT-->MCT(ELEMENT).TEXT-ID
  @ELEM := @DEL-->DEL.SUB-ELEMENT
  LOOP := 0
DO
  LOOP := LOOP + 1
UNTIL
  @ELEM-->SUB-ELEMENT(LOOP).CLASS = 2 [BODY]
OR LOOP > @DEL-->DEL.NO-SUB-ELEMENTS
DO
  Y.POS := (@TEXT-->TEXT.LENGTH/2) * CHARACTER-WIDTH
  Y.POS := @ELEM-->SUB-ELEMENT(LOOP).TEXT-OFFSET.Y - Y.POS
  RUN LIGHT [< SET 'HERE' COORDINATES TO @MCT-->MCT(ELEMENT).HERE >]
  IF
    @MCT-->MCT(ELEMENT).NEXT-STATUS.TEXT = 0 (NO TEXT)
  THEN
    < CONSTRUCT LIGHT STRING TO ZAP TEXT AREA
    COORDINATES FOR ZAP ARE @ELEM-->SUB-ELEMENT(LOOP).TEXT-OFFSET.X,
    Y-POS, TEXT-HEIGHT AND @TEXT-->TEXT.LENGTH * CHARACTER-WIDTH.
    GET COLOR FROM @MCT-->MCT(ELEMENT).COLOR.CURRENT>
    RUN LIGHT [< USING CONSTRUCTED SENTENCE >]
  ELSE
    < CONSTRUCT LIGHT STRING FOR TEXT, COORDINATES
    ARE @ELEM-->SUB-ELEMENT(LOOP).TEXT-OFFSET.X AND Y-POS.
    COLOR WILL BE @MCT-->MCT(ELEMENT).TEXT-COLOR >
    RUN LIGHT [< USING CONSTRUCTED SENTENCE >.]
  FI
CORP

```

```
1  
2  
3 PROC ZAP-OUT (USE #MCT,ELEM)  
4 DATA  
5     SCALAR @DEL: INTEGER  
6 ATAD  
7 RUN LIGHT [< SET 'HERE' TO @MCT-->MCT(ELEM).HERE >]  
8 @DEL := @MCT-->MCT(ELEM).DEL-ID  
9 <CONSTRUCT LIGHT STRING TO ZAP AREA,  
10 COORDINATES ONE 0,0,@DEL.TOLERANCE.X AND  
11 @DEL-->DEL.TOLERANCE.Y. COLOR WILL BE #MCT-->MCT(ELEM).COLOR.BASE >  
12 RUN LIGHT [< USE CONSTRUCTED SENTENCE >]  
13 CCRP  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200
```

```

PROC TOUCH (USE_ELEMENT; OUT:RC)
  DATA
    SCALAR RC, ELEMENT: INTEGER
    SCALAR @MCT: INTEGER
  ATAD
    @MCT := MENU-DICT(CURR-MENU),MCT
    RC := @MCT-->MCT(ELEMENT).CURR-STATUS.BORDER
  IF
    RC = 0
  THEN
    @MCT-->MCT(ELEMENT).NEXT-STATUS.BORDER := 1
  ELSE
    @MCT-->MCT(ELEMENT).NEXT-STATUS.BORDER := 0
  FI
  RUN_DRAW-ELEMENT (USE @MCT,ELEMENT)
CCRP

```

A-26

```

PROC CCISFI          [TOUCH INTERPRETER]
  DATA
    SCALAR ON: INTEGER
    MORE: LOGICAL
  ATAD
  ON := 1
  RUN FINDEL (ALT MEADDR) [LOCATE TOUCHED ELEMENT]
  IF
    <ELEMENT FOUND>
  THEN
    RUN COMPRS
  FI
  <STACK ELEMENT>
  STACK(TOP).IFLAG := ON
  MORE := TRUE
  WHILE
    MORE
  DO
    IF
      <STACK EMPTY>
    THEN
      MORE := FALSE
    ELSE
      IF
        STACK(TOP).IFLAG = ON
      THEN
        RUN <IMMEDIATE ACTION ROUTINE>
      ELSE
        IF
          STACK(TOP).DFLAG = ON
        THEN
          RUN <DELAYED ACTION ROUTINE>
        ELSE
          MORE := FALSE
        FI
      FI
    FI
  FI
  IF
    NOT MORE
  THEN
    RUN MNEXT (ALT PMENU)
    ROMENU := PMENU
    RUN DRMENU (USE RGMENU)
  FI
  OD
CORP

```

A-27

```

PROC GACTAD (USE ACTION.RTN, ALT ACTION.ADDR)  (MAP ACTION ROUTINES)
  DATA
    SCALAR  ORIGIN,
            MMS.NUMBER,
            MMR.NUMBER: INTEGER
  ATAD
  IF
    ACTION.RTN <> 0
  THEN
    ORIGIN := MMSTAB(ACTION.RTN)
    MMS.NUMBER := MMSTAB(ACTION.RTN + 1)
    BLOCKS := MMSTAB(ACTION.RTN + 2)
    MMR.NUMBER := ORIGIN/512 - 64
    WHILE
      BLOCKS > 0
    DO
      RUN MRHMAP (USE MMS.NUMBER, MMR.NUMBER)
      MMS.NUMBER := MMS.NUMBER + 1
      MMR.NUMBER := MMR.NUMBER + 1
      BLOCKS := BLOCKS - 1
    DU
  FI
CORP
PROC FINDEL (USE X.TOUCH, Y.TOUCH)  (FIND THE TOUCHED ELEMENT)
  DATA
    SCALAR  LOOP,
            X,
            Y,
            X.TOUCH,
            Y.TOUCH: INTEGER
    SCALAR  FOUND: LOGICAL
  ATAD
  LOOP := MDT(CURPMA).NEL
  FOUND := FALSE
  MEADDR := MDT(CURPMA).MCT - MCTLEN
  WHILE
    LOOP > 0 AND NOT FOUND
  DO
    MEADDR := MEADDR + MCTLEN
    IF
      MCT(MEADDR).DEL <> 0  (HAS A DEL ENTRY)
    THEN
      IF
        MCT(MEADDR).CURRENT.INSENSITIVE = <OFF> AND
        MCT(MEADDR).CURRENT.INHIBIT = <OFF>
      THEN
        X := X.TOUCH - MCT(MEADDR).XHERE
        Y := Y.TOUCH - MCT(MEADDR).YHERE
        IF
          X >= 0 AND Y >= 0
        THEN
          RUN GLELAD (USE MEADDR, ALT MEADDR)
          IF

```



```

PROC LEXSCN (LEXICAL SCANNER ROUTINE)
DATA
    SCALAR LEXRC,
    ELEMENT: INTEGER
    SCALAR FOUND: LOGICAL
ATAD
LEXRC := 0
RUN COMPRS
IF
    <STACK EMPTY>
THEN
    LEXRC := 1
ELSE
    RUN SRIES
    ELEMENT := STACK(TOP).ELEMENT
    IF
        MCT(ELEMENT).SC = 2 AND <NUMBER STACKED = 1>
    THEN
        LEXRC := 2
        ERRMSG := 7
        RUN GRNST
    ELSE
        IF
            MCT(ELEMENT).SC <> 0
        THEN
            RUN ESTADD (USE ELEMENT, ALT ESADDR)
            IF
                EST(ESADDR).CB <> 0
            THEN
                FOUND := FALSE
                MEADDR := MDT(CURRMA).MCT
                WHILE
                    (NOT FOUND) AND (<MORE ELEMENTS>)
                DO
                    IF
                        MCT(MEADDR).ESID = EST(ESADDR).ID
                    THEN
                        <STACK ELEMENT>
                        FOUND := TRUE
                    ELSE
                        MEADDR := MEADDR + MCTLEN
                    FI
                DD
            ELSE
                LEXRC := 2
                ERRMSG := 8
                RUN GRNST
            FI
        FI
    FI
    FI
    CDRP
    A-30
    
```

```

PROC COMPRS          (COMPRESS SENTENCE STACK)
DATA
    SCALAR ELEMENT,
    TOP,
    PTR,
    PTR2: INTEGER
    SCALAR MORE,
    MATCH: LOGICAL
ATAD
TOP := MDT(CURRMA).ELST
MORE := TRUE
WHILE
    MORE
DO
    ELEMENT := STACK(TOP).ELEMENT
    PTR := TOP
    MATCH := FALSE
    WHILE
        PTR > MDT(CURRMA).ESTK OR NOT MATCH
    DO
        PTR := PTR - 1
        IF
            ELEMENT = STACK(PTR).ELEMENT
        THEN
            STACK(PTR).IFLAG = <DELETE>
            STACK(TOP).IFLAG = <DELETE>
            MATCH := TRUE
        FI
    OD
    TOP := TOP - 1
    WHILE
        STACK(TOP).IFLAG = <DELETE> AND
        MORE
    DO
        TOP := TOP - 1
        IF
            TOP = MDT(CURRMA).ESTK
        THEN
            MORE := FALSE
        FI
    OD
    PTR := MDT(CURRMA).ESTK
    PTR2 := 0
    TOP := MDT(CURRMA).ELST
    WHILE
        PTR =< TOP
    DO
        IF
            STACK(PTR).IFLAG <> <DELETE>
        THEN
            PTR2 := PTR2 + 1
            STACK(PTR2).IFLAG := STACK(PTR).IFLAG

```

A-31

```
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000
```

```

PROC SRTES          (SORT SENTENCE STACK)
  DATA
    ARRAY TEMP,STACK(1:10)
    SCALAR I,SC,PTR: INTEGER
  ATAD
    I := 0
    SC := 0
  WHILE
    SC < 5
  DO
    PTR := STACK.BOTIOM
    WHILE
      PTR < STACK.TOP
    DO
      MEADDR := STACK(TOP).ELEMENT
      IF
        MCT(MEADDR).SC = SC
      THEN
        I := I + 1
        TEMP.STACK(I) := STACK(PTR)
      FI
      PTR := PTR + 1
    OD
    SC := SC + 1
  OD
  PTR := 0
  WHILE
    PTR <= I
  DO
    STACK(PTR) := TEMP.STACK(PTR)
    PTR := PTR + 1
  OD
CORP

```

A-33

```

1  PROC GDELAD (USE MEADDR)      (GET DISPLAY ELEMENT ADDRESS AND MAP DEL)
2  DATA
3      SCALAR  DEL.ID,
4              DEL.INDEX,
5              MMS.NUMBER,
6              DEL.ORIGIN,
7              MAP.ORIGIN,
8              SDEL,      (LOCATION OF DEX AND DEL)
9              MMR.NUMBER: INTEGER
10     ARRAY  DEX (1:NUMBER OF DEL ENTRIES) OF DEX-RECORD
11     RECORD DEX-RECORD
12         .ADDRESS: INTEGER
13
14     ATAD
15     DEL.ID = MCL(MEADDR).DEL
16     IF
17         DEL.ID = 1
18     THEN
19         DEADDR = LOGO.ADDR      (FROM CONF INIT TABLE)
20     ELSE
21         DEADDR := 0
22         MMS.NUMBER := SDEL(2)
23         MMR.NUMBER := MAPDEL/512 - 64
24         RUN MHRMAP (USE MMS.NUMBER, MMR.NUMBER)
25         MAP.ORIGIN := MOD(DEX,512) + *MAPDEL
26         DEL.INDEX := DEL.ID * 2 + MAP.ORIGIN
27         DEADDR := DEX(DEL.INDEX).ADDRESS
28         DEL.ORIGIN := MMSTAB(SDEL + 1)
29         MMS.NUMBER := (DEADDR - DEL.ORIGIN)/512 + MMS.NUMBER
30         MMR.NUMBER := MAPDEL/512 - 64
31         RUN MHRMAP (USE MMS.NUMBER, MMR.NUMBER)
32         MMS.NUMBER := MMS.NUMBER + 1
33         MMR.NUMBER := MMR.NUMBER + 1
34         RUN MHRMAP (USE MMS.NUMBER, MMR.NUMBER)
35         DEADDR := MOD(DEADDR,512) + MAPCEL
36     FI
37
38     CGRP
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

PROC GTELAD (USE MEADDR) (GET TEXT ELEMENT ADDRESS AND MAP TEL)

DATA

SCALAR TEL.ID,
 TEL.INDEX,
 MMS.NUMBER,
 TEL.ORIGIN,
 MAP.ORIGIN,
 STEL, (LOCATION OF TEX AND TEL)
 MMR.NUMBER: INTEGER
 ARRAY TEX (1:NUMBER OF TEL ENTRIES) OF TEX-RECORD
 RECORD TEX-RECORD
 .ADDRESS: INTEGER

ATAC

TLADDR := 0
 MMS.NUMBER := STEL(2)
 MMR.NUMBER := MAPTEL/512 - 64
 RUN MHRMAP (USE MMS.NUMBER, MMR.NUMBER)
 MAP.ORIGIN := MOD(TEX,512) + MAPTEL
 TEL.ID := NCT(MEADDR).TEL
 TEL.INDEX := TEL.ID * 2 + MAP.ORIGIN
 TLADDR := TEX(TEL.INDEX).ADDRESS
 TEL.ORIGIN := MMSTAB(STEL + 1)
 MMS.NUMBER := (TLADDR - TEL.ORIGIN)/512 + MMS.NUMBER
 MMR.NUMBER := MAPTEL/512 - 64
 RUN MHRMAP (USE MMS.NUMBER, MMR.NUMBER)
 MMS.NUMBER := MMS.NUMBER + 1
 MMR.NUMBER := MMR.NUMBER + 1
 RUN MHRMAP (USE MMS.NUMBER, MMR.NUMBER)
 TLADDR := MOD(TLADDR,512) + MAPTEL

CORD

```

10  PROC GMCIAD (USE MCADDR)      [DO MCT MAPPING AND COMPUTE ADDRESSES]
11  DATA
12  SCALAR MCT.ORIGIN,
13         MMS.NUMBER,
14         MMR.NUMBER,
15         S*CTF,      [MCT MAPPED LOCATION FROM LINKER]
16         LOOP: INTEGER
17  ATAD
18  MCT.ORIGIN := MMSTAB(S*CTF)
19  MMS.NUMBER := MMSTAB(S*CTF+1)
20  MMS.NUMBER := (MCADDR - MCT.ORIGIN)/512 + MMS.NUMBER
21  MMR.NUMBER := MAPMCT/512 - 64
22  LOOP := 4
23  WHILE
24      LOOP > 0
25  DO
26      RUN MHRMAP (USE MMS.NUMBER, MMR.NUMBER)
27      MMS.NUMBER := MMS.NUMBER + 1
28      MMR.NUMBER := MMR.NUMBER + 1
29      LOOP := LOOP - 1
30  OD
31  MCADDR := MOD(MCADDR,512) + MAPMCT
32  CORP
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```



```

3
5
PROC MHRMAP (USE MMS, MMR) (PERFORM MEMORY MAPPING)
DATA
    SCALAR LOOP: INTEGER
    ARRAY  MAPTAB (1:32) OF MAPTAB-ENTRY
    RECORD MAPTAB-ENTRY
        .MMS
ATAD
IF
    MMS <> 0
THEN
    MMS := (MMS/8)*16 + MOD(MMS,8)
    MMS := INVERT(MMS) [1'S COMPLIMENT]
    >FOO2 := MMR
    >FOO4 := MMS
    MAPTAB(MMR).MMS := MMS
ELSE
    LOOP := 0
    WHILE LOOP < 32
    DO
        IF
            MAPTAB(LOOP).MMS <> 0
        THEN
            >FOO2 := LOOP
            >FOO4 := MAPTAB(LOOP).MMS
        FI
        LOOP := LOOP + 1
    OD
FI
CCRP

```

```

1  PROC GRMST (USE FLAG, ERR.NO)
2  DATA
3      SCALAR ERR.NO,
4      SAVEGO: INTEGER
5  ATAD
6  IF
7      ERRMSG <> 0 OR ERR.NO <> 0
8  THEN
9      <DUPLICATE GO KEY ELEMENT IN MENU>
10     SAVEGO := <MCT ADDRESS OF GO KEY>
11     IF
12         FLAG = <RESET>
13     THEN
14         IF
15             MCT(SAVEGO).NEXT.BODY <> <OFF>
16         THEN
17             MCT(SAVEGO).NEXT.BODY = <OFF>
18             RUN CBODY (USE SAVEGO)
19             MCT(SAVEGO).CURRENT := MCT(SAVEGO).NEXT
20         ELSE
21             RUN ERRERS (ERASE ERROR MSG)
22             ERRMSG := 0
23         FI
24     ELSE
25         ERRMSG := MSG.NO
26         IF
27             <GO INDICATOR IS OFF>
28         THEN
29             <SET GO INDICATOR>
30         FI
31         IF
32             <A MSG ALREADY DISPLAYED>
33         THEN
34             RUN ERRERS
35         FI
36     FI
37     FI
38     CORP
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

1 PROC ERRUSP (DISPLAY A MESSAGE)
2 DATA
3 SCALAR CBOX: INTEGER
4 ATAC
5 <FIND CONTROL KEY BOX IN MENU>
6 CBOX := <CONTROL BOX MCT ADDRESS>
7 RUN LIGHT (USE RECTAN) [ZAP AREA FOR MESSAGE]
8 <CONVERT ERRMSG NUMBER TO ERROR TEXT ADDRESS>
9 RUN LTEXT (USE <TEXT ADDRESS>)
10 ERRELG := ERRMSG [INDICATE THAT MESSAGE IS DISPLAYED]
11 <SET INSENSITIVITY BIT FOR EACH MENU ELEMENT WHERE
12 MESSAGE IS DISPLAYED>
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

31 PROC ERRERS          [ERASE A DISPLAYED MESSAGE]
32 DATA
33     SCALAR LOOP: INTEGER
34     FOUND: LOGICAL
35
36 ATAD
37 IF
38     ERRFLG <> 0
39 THEN
40     RUN LIGHT (USE RECTAN)          [ZAP MESSAGE AREA]
41     LOOP := 3
42     ELEMENT := CBOX
43     FOUND := FALSE
44     WHILE
45         LOOP > 0 OR NOT FOUND
46     DO
47         ELEMENT := ELEMENT + MCLEN
48     IF
49         IF
50             MCT(ELEMENT).DEL = GUNY
51         THEN
52             FOUND := TRUE
53         ELSE
54             MCT(ELEMENT).CURRENT := 0
55             MCT(ELEMENT).NEXT := MCT(ELEMENT).DEFAULT
56             RUN DREL (USE ELEMENT) [REDRAW ELEMENT]
57         FI
58     LOOP := LOOP - 1
59     OD
60     ERRFLG := 0
61 FI
62 CORP
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```
PROC GMSGAD (USE RGBID, ALT RGB)      GET RGB MESSAGE ADDRESS
DATA
  SCALAR INDEX: INTEGER
ATAC
INDEX := RGBID * 2 + RGBLKS
RGB := RGBREC(INDEX)
CORP
```

A-41

```

3
5
PROC MSGDSP (USE FLAG, MSG.NO)
DATA
  SCALAR MSG.NO,
  MSG.ADDR,
  POS: INTEGER
  SCALAR MSG024: STRING(66)
  SCALAR FINISHED: LOGICAL
ATAD
IF
  MSG.NO > 4000
THEN
  MSG.ADDR := MSG.NO [MSG.NO IS AN ADDRESS]
  MSG.NO := 024
  POS := 0
  NUM.CHAR := 0
  FINISHED := FALSE
  WHILE
    NOT FINISHED
  DO
    MSG024(POS) := MSG.ADDR(POS)
    IF
      MSG024(POS) = 0 OR
      POS > 66
    THEN
      FINISHED := TRUE
    ELSE
      POS := POS + 1
    FI
  FI
FI
IF
  FLAG <> 0
THEN
  RUN GRMST (USE 1)
ELSE
  RUN ERRDSP
FI
CCRP

```

A-42

```

PRUC RESET          (GENERAL CANCEL FUNCTION)
DATA
  SCALAR PTR: INTEGER
  ATAD
  PTR := MUT(CURRMA).ELST          (TOP ELEMENT ON STACK)
  WHILE
    PTR <> MUT(CURRMA).LSTK
  DO
    MEADDR := STACK(PTR).ELEMENT  (GET STACKED ELEMENT)
    IF
      MCT(MEADDR).CURRENT.BORDER = <ON>
    THEN
      RUN TOUCH (USE MEADDR)
    FI
    PTR := PTR - 1
  OD
  IF
    CAMTCH = 0                    (CAMERA IN LAST SENTENCE?)
  THEN
    MEADDR = CAMTCH + MCTLEN      (POINT TO FIRST PRESET)
    WHILE
      CAMLOP > 0
    DO
      IF
        MCT(MEADDR).CURRENT.BODY = <OFF>
      THEN
        RUN ESTADD (USE MEADDR, ALT ESADDR)
        EST(ESADDR).ON := <OFF>
        EST(ESADDR).FFRE := <OFF>
        EST(ESADDR).CE := 0
        RUN GDECAD (USE MEADDR, ALT DEADDR)
        RUN ZAPOUT (USE DEADDR)
        MCT(MEADDR).NEXT := MCT(MEADDR).DEFAULT
        MCT(MEADDR).CURRENT := MCT(MEADDR).DEFAULT
      FI
      CAMLOP := CAMLOP - 1
    OD
  FI
  CAMTCH := 0
  CURPRE := 0
  STACK.TOP := STACK.BOTTOM
  RUN GRMST (USE 0)              [ERASE MESSAGES]
CDRP

```

A-43

```

1  PROC PREVIEW
2  DATA
3      SCALAR PTR, PREVIEW.ID: INTEGER
4
5  ATAC
6      PTR := STACK.TOP
7      PREVIEW.ID := 0
8  WHILE
9      (PTR > STACK.BOTTOM) AND (PREVIEW.ID = 0)
10
11      DO
12          MEADDR := STACK(PTR).ELEMENT
13          RUN ESTADD (USE MEADDR, ALT ESADDR)
14          IF
15              ESADDR <> 0
16          THEN
17              IF
18                  EST(ESADDR).TYPE = 1
19              THEN
20                  PREVIEW.ID := EST(ESADDR).ID
21              ELSE
22                  IF
23                      EST(ESADDR).CB <> 0
24                  THEN
25                      PREVIEW.ID := EST(ESADDR).CB
26                  FI
27              FI
28          FI
29      OD
30  IF
31      PREVIEW.ID <> 0
32  THEN
33      <SEND PREVIEW MESSAGE>
34      PREVIEW.FLAG := OFF
35  FI
36  CORP
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60

```



```
PROC SLCTIME (USE HEADR) (SELECT MUTUALLY EXCLUSIVE)
DATA
  SCALAR CONFLICT, PTR: INTEGER
ATAD
  RUN ESTADD (USE HEADR, ALT ESADDR)
  CONFLICT := EST(ESADDR).CONF
  IF
    CONFLICT <> 0
  THEN
    PTR := STACK.BOTTOM
    WHILE
      PTR <= STACK.TOP
    DO
      ELEMENT := STACK(PTR).ELEMENT
      RUN ESTADD (USE ELEMENT, ALT ESADDR)
      IF
        CONFLICT <> EST(ESADDR).CONF
      THEN
        RUN TOUCH (USE ELEMENT) (RETOUCH KEY IN CONFLICT)
        STACK(PTR).IFLAG := >FF
      FI
      PTR := PTR + 1
    OD
  FI
  IF
    PREVIEW.FLAG = ON
  THEN
    RUN PREVUE
  FI
CORP
```

```

1  PROC STEFLT          [SET MASTER MENU DEFAULTS]
2  DATA
3      SCALAR SENTENCE, I, J, PTR: INTEGER
4      SCALAR FOUND, MORE: LOGICAL
5
6  ATAD
7      SENTENCE := 0
8      MORE := TRUE
9      WHILE
10         MORE
11     DO
12         SENTENCE := SENTENCE + 10
13         I := SENTENCE
14         J := 0
15         FOUND := FALSE
16         WHILE
17             (J <= 9) OR (NOT FOUND)
18         DO
19             I = I + J
20             PTR := 0
21             WHILE
22                 (PTR <= ESTENT) AND (NOT FOUND)
23             DO
24                 IF
25                     EST(PTR).INIT = I
26                 THEN
27                     RUN FINDIT (USE EST(PTR).ID)
28                     IF
29                         FOUND
30                     THEN
31                         <STACK ELEMENT>
32                         STACK(TOP).IFLAG := ON
33                         RUN EXECIT
34                     FI
35                 ELSE
36                     PTR := PTR + 1
37                 FI
38             OD
39             J := J + 1
40         OD
41         IF
42             J > 1
43         THEN
44             RUN FINDIT (USE 0)
45             RUN EXECIT
46         ELSE
47             MORE := FALSE
48         FI
49     OD
50  CCRP
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```
1  PROC LSENS      [DISCONNECT LOOSE ENDS]
2  DATA
3      SCALAR SOURCE: INTEGER
4  ATAD
5      ESADDR := 0
6  WHILE
7      ESADDR <= ESTENT [THIS LOOP DISCONNECTS ALL SINKS]
8          [WHOSE SOURCE HAS BEEN DISCONNECTED, AND]
9          [DISCONNECTS ALL SOURCES WHICH HAVE NO SINKS]
10     DO
11         IF
12             EST(ESADDR).TYPE = 1
13         THEN
14             SOURCE := EST(ESADDR).ID
15             IF
16                 EST(ESADDR).FCN = <ONS>
17             THEN
18                 <LOOP THROUGH THE EST LOOKING FOR A SINK
19                 WHICH IS CONTROLLED BY THIS SOURCE. IF NO
20                 SINKS ARE FOUND, THEN SET EST(ESADDR).FCN
21                 TO <OFF>>
22             FI
23             IF
24                 EST(ESADDR).FCN = <OFF>
25             THEN
26                 <LOOP THROUGH THE EST LOOKING FOR ANY SINK
27                 WHICH IS CONTROLLED BY THIS SOURCE. WHEN
28                 FOUND, SET EST FIELDS FCN, CB, COLOR TO OFF
29                 AND SEND THE DISCONNECT MESSAGE>
30             FI
31         FI
32         ESADDR := ESADDR + 1
33     OD
34 CORP
```

```

PROC BACK          [BACK KEY, ALL MENUS]
DATA
  SCALAR SAVE: INTEGER
ATAD
RUN RESET
RQMENU := MDT(CURRMA).PM          [RQMENU = PREVIOUS MENU]
RUN GMADDR (USE RQMENU, ALT MENUAD) [GET NEW MENU MDT ADDRESS]
SAVE := MDT(MENUAD).PM          [SAVE NEW MENU PREVIOUS MENU]
<TO PRE   DRMENU ALWAYS SETS THE NEW MENU "PREVIOUS MENU" FIELD TO
<BE THE MENU ID OF THE LAST MENU DRAWN. TO AVOID GETTING>
<INTO A MENU LOOP WHEN THE BACK KEY IS USED, WE LET THE   >
<MENU WHICH WE ARE GOING BACK TO KEEP ITS EXISTING PREVIOUS>
<MENU FIELD.>
RUN DRMENU
MDT(MENUAD).PM := SAVE
CORP

```

```
PROC HELP [HELP KEY, ALL MENUS]
DATA
ATAC
<POP THE STACK>
IF
ERRMSG <> 0 AND
ERRMSG <> ERRELG
THEN
RUN ERRDSP
ELSE
IF
<STACK IS EMPTY>
THEN
PDMENU := MDT(CURRMA).HELP
ELSE
PDMENU := MCT(MEADDR).HELP
FI
IF
PDMENU = 0
THEN
<DISPLAY NO HELP AVAILABLE MSG>
ELSE
RUN GRMST (USE 0)
FI
FI
CORP
```

```

9
10
11
12
13 PROC FINDIT (USE ID) (FIND NCT ELEMENT WITH ESID OF ID)
14 DATA
15     SCALAR FOUND: LOGICAL
16     SCALAR ID: INTEGER
17
18     ATAD
19     MEADDR := MDI(CO3MNU).NCT
20     FOUND := FALSE
21     WHILE
22         <MORE ELEMENTS> AND (NOT FOUND)
23     DO
24         IF
25             ID = 0
26         THEN
27             IF
28                 MCI(MEADDR).DEL = <GO KEY>
29             THEN
30                 FOUND := TRUE
31             ELSE
32                 MEADDR := MEADDR + MCTLEN
33             FI
34         ELSE
35             IF
36                 MCI(MEADDR).ESID = ID
37             THEN
38                 FOUND := TRUE
39             FI
40         FI
41     OD
42
43 CORP
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

```

```

3
5
PROC EXECIT [EXECUTE DEFAULT STACK]
DATA
  SCALAR MORE: LOGICAL
  ATAC
  MORE := TRUE
  WHILE
    MORE
  DO
    MORE := FALSE
    IF
      THEN
        STACK(TOP).IFLAG = <ON>
        RUN <IMMEDIATE ACTION ROUTINE>
        MORE := TRUE
      ELSE
        IF
          THEN
            STACK(TOP).DFLAG = <ON>
            RUN <DELAYED ACTION ROUTINE>
            MORE := TRUE
          FI
    FI
  DD
CORP

```

A-51

```

17 PROC ZAPCAM (USE FLAG) [ERASE A TOUCHED OR ACTIVE CAMERA]
18 DATA
19 SCALAR FLAG: LOGICAL
20 SCALAR LOOP: INTEGER
21 SCALAR ELEM: INTEGER
22
23 ATAD
24 IF
25     CAMTCH = 0
26 THEN
27     <RETURN>
28
29 CORPRE := 0 [SET CURRENT PRESET OFF]
30 IF
31     CAMTCH < 5000 [CAMERA IS NOT AN ADDRESS]
32 THEN
33     RUN PNCIT (USE CAMTCH, MEADDR)
34     <POP STACK>
35
36 FI
37 RUN COMPRS
38 RUN ESTADD [USE MEADDR, ALT ESADDR]
39 IF
40     FLAG = 1
41 THEN
42     EST(ESADDR).FON := 0
43
44 FI
45 LOOP := EST(ESADDR).FLD2 [GET NUMBER OF CAMERA PRESETS]
46 ELEM := CAMTCH
47 WHILE
48     LOOP > 0
49 DO
50     IF
51         MCT(ELEM).CURRENT.BORDER = <ON>
52     THEN
53         MCT(ELEM).CURRENT.BORDER := <OFF>
54         RUN REFSK (USE ELEM)
55
56 FI
57 ELEM := ELEM + MCTLEN
58 IF
59     LOOP <> 0
60 THEN
61     RUN ESTADD (USE ELEM, ALT ESADDR)
62     IF
63         FLAG = 1 OR EST(ESADDR).FON = 0
64     THEN
65         EST.(ESADDR).FON := 0
66         EST.(ESADDR).FPRE := 0
67         EST.(ESADDR).CB := 0
68         EST.(ESADDR).COLOR := 0
69         MCT(ELEM).NEXT.INSENSITIVE := <ON>
70
71 FI
72 LOOP := LOOP - 1
73
74 OD
75 RUN CAMHIN

```

A-52

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

CARICH := 0
CARLUP := 0
CORP

A-53

```

10  PROC CAMRTN [RETURN CAMERA TO LAST ACTIVE PRESET]
11  DATA
12  SCALAR LOOP: INTEGER
13  SCALAR ELEM: INTEGER
14  ATAD
15  IF CAMTCH <> 0
16  THEN
17  LOOP := CAMLUP
18  ELEM := CAMTCH
19  WHILE LOOP > 0
20  DO
21  ELEM := ELEM + MCILEN
22  IF MCT(ELEM).CURRENT.BODY = <OFF>
23  THEN LOOP := LOOP - 1
24  ELSE
25  RUN ESTADD (USE ELEM, ALT ESADDR)
26  RGBND := 5
27  RUN GMSGAD (USE RGBND, ALT RGB)
28  RGB(7) := EST(ESADDR).PLD1
29  RUN ESTADD (USE CAMTCH, ALT ESADDR)
30  RGB(4) := EST(ESADDR).PLD1
31  RUN SYSREQ (USE RGB)
32  FI
33  DO
34  FI
35  CCRP

```

A-54

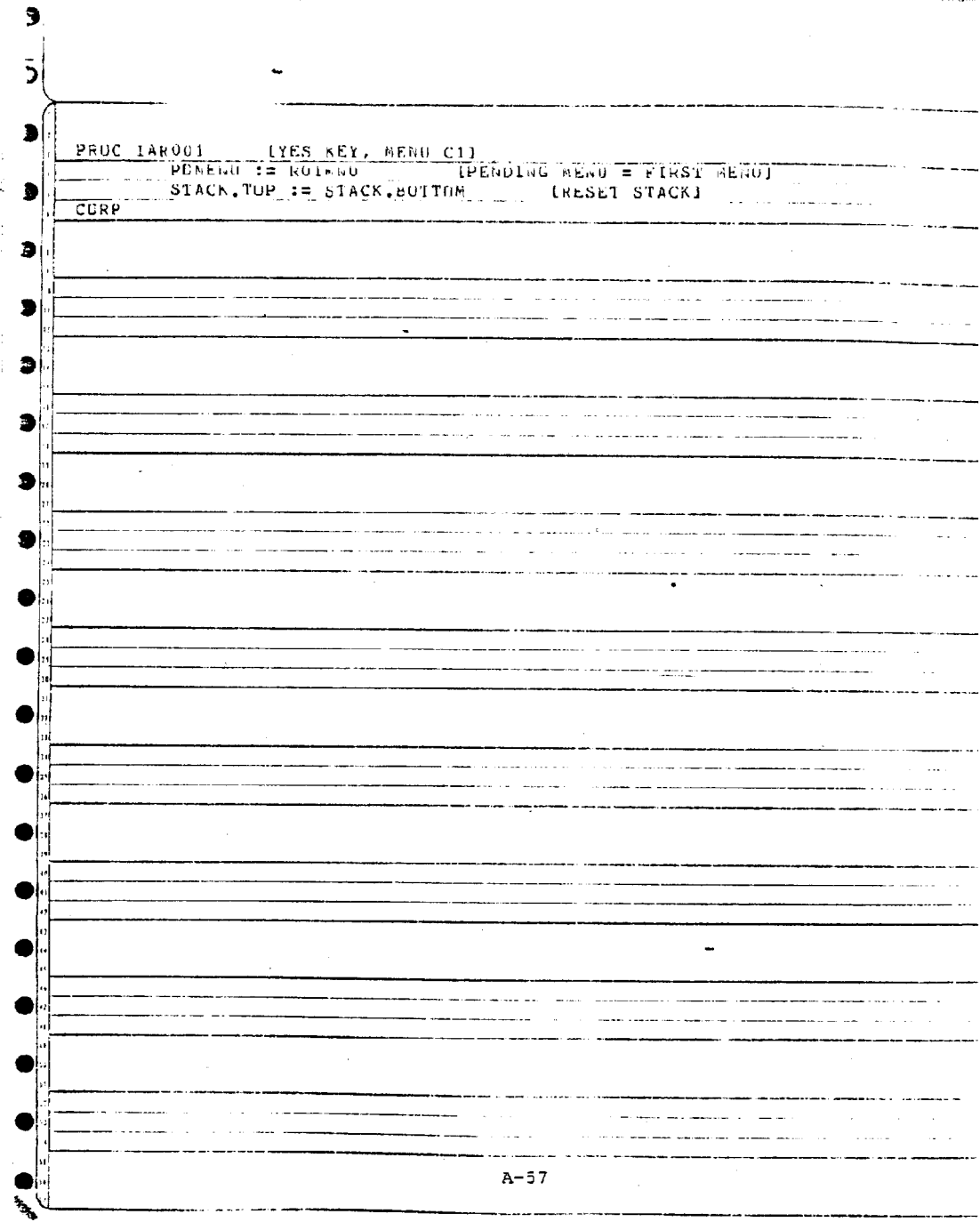
```
3  
5  
7  
9  
11  
13  
15  
17  
19  
21  
23  
25  
27  
29  
31  
33  
35  
37  
39  
41  
43  
45  
47  
49  
51  
53  
55  
57  
59  
61  
63  
65  
67  
69  
71  
73  
75  
77  
79  
81  
83  
85  
87  
89  
91  
93  
95  
97  
99  
101  
103  
105  
107  
109  
111  
113  
115  
117  
119  
121  
123  
125  
127  
129  
131  
133  
135  
137  
139  
141  
143  
145  
147  
149  
151  
153  
155  
157  
159  
161  
163  
165  
167  
169  
171  
173  
175  
177  
179  
181  
183  
185  
187  
189  
191  
193  
195  
197  
199  
201  
203  
205  
207  
209  
211  
213  
215  
217  
219  
221  
223  
225  
227  
229  
231  
233  
235  
237  
239  
241  
243  
245  
247  
249  
251  
253  
255  
257  
259  
261  
263  
265  
267  
269  
271  
273  
275  
277  
279  
281  
283  
285  
287  
289  
291  
293  
295  
297  
299  
301  
303  
305  
307  
309  
311  
313  
315  
317  
319  
321  
323  
325  
327  
329  
331  
333  
335  
337  
339  
341  
343  
345  
347  
349  
351  
353  
355  
357  
359  
361  
363  
365  
367  
369  
371  
373  
375  
377  
379  
381  
383  
385  
387  
389  
391  
393  
395  
397  
399  
401  
403  
405  
407  
409  
411  
413  
415  
417  
419  
421  
423  
425  
427  
429  
431  
433  
435  
437  
439  
441  
443  
445  
447  
449  
451  
453  
455  
457  
459  
461  
463  
465  
467  
469  
471  
473  
475  
477  
479  
481  
483  
485  
487  
489  
491  
493  
495  
497  
499  
501  
503  
505  
507  
509  
511  
513  
515  
517  
519  
521  
523  
525  
527  
529  
531  
533  
535  
537  
539  
541  
543  
545  
547  
549  
551  
553  
555  
557  
559  
561  
563  
565  
567  
569  
571  
573  
575  
577  
579  
581  
583  
585  
587  
589  
591  
593  
595  
597  
599  
601  
603  
605  
607  
609  
611  
613  
615  
617  
619  
621  
623  
625  
627  
629  
631  
633  
635  
637  
639  
641  
643  
645  
647  
649  
651  
653  
655  
657  
659  
661  
663  
665  
667  
669  
671  
673  
675  
677  
679  
681  
683  
685  
687  
689  
691  
693  
695  
697  
699  
701  
703  
705  
707  
709  
711  
713  
715  
717  
719  
721  
723  
725  
727  
729  
731  
733  
735  
737  
739  
741  
743  
745  
747  
749  
751  
753  
755  
757  
759  
761  
763  
765  
767  
769  
771  
773  
775  
777  
779  
781  
783  
785  
787  
789  
791  
793  
795  
797  
799  
801  
803  
805  
807  
809  
811  
813  
815  
817  
819  
821  
823  
825  
827  
829  
831  
833  
835  
837  
839  
841  
843  
845  
847  
849  
851  
853  
855  
857  
859  
861  
863  
865  
867  
869  
871  
873  
875  
877  
879  
881  
883  
885  
887  
889  
891  
893  
895  
897  
899  
901  
903  
905  
907  
909  
911  
913  
915  
917  
919  
921  
923  
925  
927  
929  
931  
933  
935  
937  
939  
941  
943  
945  
947  
949  
951  
953  
955  
957  
959  
961  
963  
965  
967  
969  
971  
973  
975  
977  
979  
981  
983  
985  
987  
989  
991  
993  
995  
997  
999  
A-55
```

5

PRDC CHELEM [CHANGE SINGLE MENU ELEMENT]
RUN DRED2 (USE CHELM)

CDRP

A-56



PROC IAH002 (NO KEY, MENU C1)
PDMENU := COZANO (PENDING MENU = SECOND CONTROL MENU)
STACK.TOP := STACK.BOTTOM (RESET STACK)
CCRP

A-58

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

```
PROC 1AR003 (BACK KEY, GENERAL MENUS)  
  <POP STACK>  
  RUN BACK  
CCRP
```

A-59

```
1
2
3
4 PROC IAK004      [GU KEY, MENUS C2,R1,R2 AND R3]
5   <POP STACK>
6   IF
7     ERRMSG = <ON> [ERROR PENDING]
8   THEN
9     RUN GRMST      [RESET ERROR]
10  FI
11  RUN NNEXT       [GET NEXT MENU NUMBER]
12  IF
13    C03DRW <> 0
14  THEN
15    IF
16      PDMENU = R03MNU
17    THEN
18      PDMENU := C03MNU [SET TO MASTER MENU]
19    FI
20  FI
21  CGRP
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
```

A-60


```
PROC LAH005 (TOUCH ELEMENT KEY)
  HEADLR := STACK(STACK.TOP).ELEMENT
  RUN SECIME (DO MUTUAL EXCLUSIVE TEST)
  RUN TOUCH (USE HEADLR) (TOUCH ELEMENT)
  STACK(STACK.TOP).TFLAG := <OFF>
CCRP
```

A-61

```
PROC 1AR006 IX (CANCEL) KEY, MENU C21
  IF
    ERRMSG = <ON> (ERROR PENDING)
  THEN
    <POP STACK>
  ELSE
    PDMENU := C06MMU (GO TO MENU C6)
    STACK.TOP := STACK.BOTTOM
  F1
CORP
```

1
2
3 PROC 1AR007 [HELP KEY, GENERAL MENUS]
4 <POP STACK>
5 RUN HELP
6 CCRP
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200

A-63

```
PROC 1AR008 [GJ KEY, MENUS C3 AND C4]
  <POP STACK>
  IF
  THEN ERRMSG = <ON>
  THEN RUN GRMST (USE 0) [RESET ERROR]
  FI
  IF
  THEN <STACK IS NOT EMPTY>
  THEN
    RUN DEXSCN
    IF
    THEN LEXRC = 0
    THEN STACK(STACK.TOP).DFLAG := <ON>
    FI
  ELSE
    IF
    THEN <MENU = C03MNU>
    THEN
      R1 := 6 [SET ERROR NUMBER]
      R0 := 1
      RUN GRMST (USE R0,R1) [INDICATE ERROR]
    ELSE
      R1 := 10 [SET ERROR NUMBER]
      R0 := 1
      RUN GRMST (USE R0,R1) [INDICATE ERROR]
    FI
  FI
CORP
```

```
PROC IAR00Y (X KEY, MENUS R1, R2 AND R3)
  <POP STACK>
  IF
    THEN ERRMSG = <OW> [ERROR PENDING]
  ELSE
    RUN GRMST (USE 0) [RESET ERROR]
  STACK.TOP := STACK.BOTTOM [RESET STACK]
  IF
    THEN CO3DRW = <OFF> [MENU C3 HAS NOT BEEN DISPLAYED]
  ELSE
    PDMENU := CO2MNU
  PDMENU := CO3MNU
  FI
CORP
```

A-65

```

3
5
PROC IAR010      (X KEY, MENUS C4, C5, C6 AND ALL HELPS)
  <POP STACK>
  IF              ERRMSG = <ON>      [ERROR IS PENDING]
  THEN
    RUN GRMST (USE 0)      [RESET ERROR]
  FI
CCRP

```

A-66

```

31 PROC IAR011 (FRONT, LEFT AND RIGHT CAMERAS)
32 DATA
33     SCALAR LOOP,
34     PRESET,
35     CONTROL: INTEGER
36
37 ATAD
38 RUN SLCTIME (ELIMINATE MUTUALLY EXCLUSIVE TOUCHES)
39 MEADDR := STACK(STACK.TOP).ELEMENT
40 IF
41     MEADDR = CAPTCH (A RE-TOUCH)
42 THEN
43     RUN ZAPCAM (ERASE THIS CAMERA)
44     RUN RDMENU (REDRAW THE MENU)
45     <RETURN>
46 FI
47 RUN ESTADD (USE MEADDR, ALT ESADDR)
48 LOOP := EST(ESADDR).FLD1 (GET NUMBER OF CAMERA PRESETS)
49 CAMLOOP := LOOP (SAVE NUMBER OF PRESETS)
50 CONTROL := EST(ESADDR).ID
51 RUN TOUCH (USE MEADDR)
52 WHILE
53     LOOP > 0 (MORE PRESETS)
54 DO
55     MEADDR := MEADDR + NCTLEN (POINT TO NEXT PRESET)
56     RUN ESTADD (USE MEADDR, ALT ESADDR)
57     IF
58         EST(ESADDR).FLD2 <> 0 (PRESET HAS BEEN SET)
59     THEN
60         EST(ESADDR).PRESENT := <ON>
61         EST(ESADDR).CB := CONTROL
62     FI
63     LOOP := LOOP - 1
64 OD
65 RUN RDMENU (REDRAW THE MENU)
66 CORR

```

```
PROC IAR013 (CCN KEY(PREVIEW), MENU C03)
STACK(STACK.TOP).IFLG := <OFF>
MEADDR := STACK(STACK.TOP).ELEMENT
RUN TOUCH (USE MEADDR)
IF
    PRVFLG = <ON> [PREVIEW IS ALREADY ON]
THEN
    PRVFLG := <OFF>
ELSE
    PRVFLG := MEADDR
    RUN PREVUE
FI
CCRP
```

A-68


```

3
5
PROC IAK014          [CAMERA PRESETS, MENU C03]
  RUN SLCR6         [CHECK FOR MUTUAL EXCLUSIVITY]
  MEADDR := STACK(STACK.TOP).ELEMENT
  STACK(STACK.TOP).IFLAG := <OFF>
  RUN TOUCH (USE MEADDR)
  FOUND := FALSE
  IF
  THEN
    ACT(MEADDR).CURRENT.BORDER = <OFF>
  THEN
    CURPRE := 0
    RUN CAMRTN      [RETURN CAMERA TO OLD PRESET]
  ELSE
    IF
    THEN
      CURPRE <> 0      [THERE IS A CURRENT PRESET]
    THEN
      RUN TOUCH (USE CURPRE) [RE-TOUCH OLD PRESET]
      RUN REFSTK (USE CURPRE) [REMOVE CURPRE FROM STACK]
    FI
    CURPRE := MEADDR
    IF
    THEN
      CMTCH <> 0      [A CAMERA HAS BEEN TOUCHED]
    THEN
      RUN ESTADD (USE CMTCH, ALT ESADDR)
      ROBID := 5      [CAMERA GO TO PRESET MESSAGE]
      RUN GMSGAD (USE ROBID, ALT ROB)
      RGB(4) := EST(ESADDR).FLD1 [CAM TO MSG]
      RUN ESTADD (USE CURPRE, ALT ESADDR)
      ROB(7) := EST(ESADDR).FLD1 [PRESET TO MSG]
      RUN SYSREQ (USE RGB)
    FI
    IF
    THEN
      PRVFLG = <ON> [PREVIEW FLAG ON]
    THEN
      RUN PREVUE
    FI
  FI
CCRP

```

```

PHUC 1AR015 [OVERHEAD TV MONITORS, MENU C03]
IF
  <STACK IS FULL>
THEN
  RUN CGMPRS
FI
MEADDR := STACK(STACK.TOP).ELEMENT
STACK(STACK.TOP).IFLAG := <OFF>
RUN TOUCH (USE MEADDR)
IF
  MCT(MEADDR).ID <> 27 [LEFT MONITOR IS 27]
THEN
  MEADDR := MEADDR - MCTLEN [POINT TO LEFT TV MONITOR]
ELSE
  MEADDR := MEADDR + MCTLEN [POINT TO RIGHT TV MONITOR]
FI
RUN TOUCH (USE MEADDR)
STACK.TOP := STACK.TOP - 4
STACK(STACK.TOP).ELEMENT := MEADDR [STACK OTHER MONITOR]
STACK(STACK.TOP).IFLAG := <OFF>
STACK(STACK.TOP).DFLAG := <OFF>
CORP

```

A-70

```

PROC 1AR016 [AUDIO KEY]
DATA
  SCALAR VALUE: INTEGER
  ATAL
  MEADDR := STACK(STACK.TOP).ELEMENT
  RGBID := 7 [AUDIO MESSAGE ID]
  RUN GMSGAD (USE RGBID, ALT RGB) [GET AUDIO MESSAGE]
  IF
    ACT(MEADDR).CURRENT.BODY = <UN> [AUDIO UN?]
  THEN
    VALUE := 0
    RGB(RGBID).PARAMETERS := -1
  ELSE
    VALUE := 1
    RGB(RGBID).PARAMETERS := 1
  FI
  RUN SYSREQ [SEND MESSAGE]
  RUN ESTADD (USE MEADDR, ALT ESADDR)
  EST(ESADDR).UN := VALUE
  RUN RCMENU [RE-DRAW THE MENU]
CORP

```

A-71

```

PROC IAK018      [X (CANCEL) KEY, MENU C03]
DATA
    SCALAR PTR: INTEGER
    SCALAR ELEM: INTEGER
ATAD
STACK.TOP := STACK.TOP + 4      [POP X KEY FROM STACK]
IF
    ERRMSG = <UN>                [ERROR PENDING]
THEN
    RUN GRMST (USE 0)           [RESET ERROR]
ELSE
    RUN COMPRS                 [COMPRESS STACK]
    IF
        <STACK IS EMPTY>
    THEN
        ERRMSG := 4
        RUN GRMST (USE ERRMSG)
    ELSE
        RUN SRTES              [SORT STACK]
        IF
            <HI-RES ELEMENT IS ON STACK>
        THEN
            ERRMSG := 2
            RUN GRMST (USE ERRMSG)
        ELSE
            ERRMSG := 0
            PTR := STACK.BOTTOM
            WHILE
                PTR <= STACK.TOP AND ERRMSG = 0
            DO
                ELEM := STACK(PTR).ELEMENT
                RUN ESTADD (USE ELEM, ALL ESACDR)
                IF
                    EST(ESACDR).ON = <OFF>
                THEN
                    ERRMSG := 5
                    RUN GRMST (USE ERRMSG)
                FI
                PTR := PTR - 4 [POINT TO NEXT ELEMENT]
            DD
        IF
            ERRMSG = 0
        THEN
            PTR := STACK.BOTTOM
            WHILE
                PTR <= STACK.TOP
            DO
                ELEM := STACK(PTR).ELEMENT
                RUN ESTADD (USE ELEM, ALL ESACDR)
                RUN TOUCH (USE ELEM)
                EST(ESACDR).ON := <OFF>
                <SEND DISCONNECT MESSAGE>
                PTR := PTR - 4
    
```

A-72

The table consists of approximately 30 rows and several columns. The text is sparse and appears to be a mix of labels and data. Some of the text visible includes:

- UD
- RUN LSERDS
- RUN RCMENU
- FI
- FI
- FI
- CCRP

The table is mostly empty with many horizontal lines, suggesting it might be a placeholder or a table with very little data.

A-73

- 1
- 2
- 3
- 4
- 5
- 6
- 7
- 8
- 9
- 10
- 11
- 12
- 13
- 14
- 15
- 16
- 17
- 18
- 19
- 20
- 21
- 22
- 23
- 24
- 25
- 26
- 27
- 28
- 29
- 30
- 31
- 32
- 33
- 34
- 35
- 36
- 37
- 38
- 39
- 40
- 41
- 42
- 43
- 44
- 45
- 46
- 47
- 48
- 49
- 50
- 51
- 52
- 53
- 54
- 55
- 56
- 57
- 58
- 59
- 60
- 61
- 62
- 63
- 64
- 65
- 66
- 67
- 68
- 69
- 70
- 71
- 72
- 73
- 74
- 75
- 76
- 77
- 78
- 79
- 80
- 81
- 82
- 83
- 84
- 85
- 86
- 87
- 88
- 89
- 90
- 91
- 92
- 93
- 94
- 95
- 96
- 97
- 98
- 99
- 100

```
PROC 1A017 [OTHER KEY, MENU C03]
PUMENU := C04440 [PENDING MENU = MENU OF MENUS]
RUN RESET [ELIMINATE ALL MENU TOUCHES]
STACK.TOP := STACK.BOTTOM [RESET STACK]
CGRP
```

```
PROC IAR019 (CAMERA KEY, MENU C05)
DATA
    SCALAR CAMERA,
    CAM.ID,
    CAM.NO: INTEGER
ATAD
STACK.TOP := STACK.BOTTOM (RESET STACK)
MEADDR := STACK(STACK.TOP).ELEMENT
STACK.TOP := STACK.BOTTOM (RESET STACK)
IF
    <CAMERA IS UN>
THEN
    RUN IAR027 (TURN OFF CAMERA MOTION)
    RUN IAR19A (DISCONNECT THIS CAMERA)
FI
RUN ESTADD (USE MEADDR, ALT ESADDR)
IF
    MEADDR = CAMERA (A RE-TOUCH)
THEN
    CAMERA := 0
ELSE
    EST(ESADDR).ON := <ON>
    CAMERA := MEADDR
    CAM.NO := EST(ESADDR).FLD1
    CAM.ID := EST(ESADDR).FLD2
    RGBID := 0
    RUN GMSGAD (USE RGBID, ALT RGB)
    RGB(4) := CAM.NO
    RUN SYSREQ (USE RGB)
    RUN IAR19A (CONNECT CAMERA)
FI
RUN RLMENU
```

CORP

A-75

```
10 PROC IARIYA (USE SWITCH) [CONNECT/DISCONNECT CAMERA TO CCM]
11   IF
12     <CAMERA IS ON>
13   THEN
14     IF
15       <SWITCH = DISCONNECT>
16     THEN
17       RGBID := 8 [DISCONNECT MESSAGE]
18       RUN GMSGAD (USE RGBID, ALT RGB)
19       RGB(6) := <EST ID OF CCM>
20       RUN ESTADD (USE CAMERA, ALT ESADDR)
21       EST(ESADDR).ON := <OFF>
22       RUN SYSREQ (USE RGB)
23     ELSE
24       RGBID := 9 [CONNECT MESSAGE]
25       RUN GMSGAD (USE RGBID, ALT RGB)
26       RGB(2) := 6 [SET LENGTH OF MESSAGE]
27       RGB(6) := CAM.ID
28       RGB(8) := <EST ID OF CCM>
29       RUN SYSREQ (USE RGB)
30     FI
31   FI
32 CCRP
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```



```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200
```

PROC IAR020 (BACK KEY, MENU C05)
STACK.TOP := STACK.BOTTOM (RESET STACK)
RUN IAR027 (STOP CAMERA MOTION)
RUN IAR19A (USE DISCONNECT) [DISCONNECT CAMERA]
CAMERA := 0
RUN BACK

CGRP

```
PROC IAR021 (HELP KEY, MENU C05)
  IF
    <ERROR PENDING AND ERROR NOT DISPLAYED>
  THEN
    STACK.TOP := STACK.TOP + 4 (POP STACK)
    RUN ERRDSP (DISPLAY ERROR)
  ELSE
    RUN IAR027 (STOP CAMERA MOTION)
    RUN IAR19A (USE DISCONNECT) (DISCONNECT CAMERA)
    CAMERA := 0
    RUN HELF
  FI
CERP
```

A-78

PROC IAR022 [CAMERA PRESET KEYS, CAMERA POSITIONING MENU]

DATA

SCALAR ELEM: INTEGER

ATAD

ELEM := STACK(STACK.TOP).ELEMENT

STACK.TOP := STACK.BOTTOM [RESET STACK]

IF

<CAMERA NOT ON>

THEN

ERRMSG := 12

RUN GRMST (USE ERRMSG)

ELSE

PRESET := ELEM

RUN IAR027 [STOP CAMERA MOTION]

RUN ESTADD (USE ELEM, ALT ESADDR)

EST(ESADDR).ON := 1

RUN RCMENU

RQBID := 4

RUN MSGAD (USE RQBID, ALT RQB)

RQB(4) := CAM.NU

RQB(6) := EST(ESADDR).FLD1

RUN SYSREQ (USE RQB)

EST(ESADDR).ON := <OFF>

RUN LIGHT (USE WAIT)

RUN RCMENU

<DETERMINE EST ID OF TOUCHED PRESET KEYS>

EST(ESADDR).FLD2 := -1 [INDICATE THAT PRESET IS SET]

FI

CCRP

```

PROC IAR023          [ZOOM KEYS, CAMERA POSITIONING MENU]
  DATA
    SCALAR  INDEX,
            CAMERA: INTEGER
    ARRAY   ZOOM(1:4) OF ZOOM-ENTRY
    RECORD  ZOOM-ENTRY
            .STEP,
            .DIRECTION: INTEGER
  ATAG
  MEADDR := STACK(STACK.TOP).ELEMENT
  STACK.TOP := STACK.BOTTOM
  IF
    CAMERA = 0
  THEN
    ERRMSG := 12
    RUN GRMST (USE ERRMSG)
  ELSE
    RGBID := 2          [ZOOM MESSAGE]
    RUN MSGAD (USE RGBID, ALT RGB)
    RGB(RGBID).MODIFIER := CAMERA.NUMBER
    MCT(HEADDR).NEXT.BODY := <ON>
    RUN SYSREQ (USE RGB)
    RUN RDMENU
    INDEX := (MCT(HEADDR).ID - 21)
    RGB(RGBID).PARAMETERS := ZOOM(INDEX).STEP
    RGB(RGBID).PARAMETERS(2) := ZOOM(INDEX).DIRECTION
    MCT(HEADDR).NEXT.BODY := <OFF>
    RUN RDMENU
  FI
CORP
  
```

A-80

```
PRUC IAK024          (FOCUS KEYS, CAMERA POSITIONING MENU)
```

```
DATA
```

```
SCALAR  INDEX,  
        CAMERA: INTEGER  
ARRAY   FOCUS (1:4) OF FOCUS-ENTRY  
RECORD  FOCUS-ENTRY  
        .STEP,  
        .DIRECTION: INTEGER
```

```
ATAD
```

```
MEADDR := STACK(STACK.TOP).ELEMENT
```

```
STACK.TOP := STACK.BOTTOM
```

```
IF
```

```
    CAMERA = 0
```

```
THEN
```

```
    ERRMSG := 12
```

```
    RUN GRMST (USE ERRMSG)
```

```
ELSE
```

```
    ROBJID := 3          (FOCUS MESSAGE)
```

```
    RUN MSGAD (USE ROBJID, ALT RQB)
```

```
    RQB(ROBJID).MODIFIER := CAMERA.NUMBER
```

```
    MCT(MEADDR).NEXT.BODY := <ON>
```

```
    RUN SYSREQ (USE RQB)
```

```
    RUN RDMENU
```

```
    INDEX := (MCT(MEADDR).ID - 25)
```

```
    RQB(ROBJID).PARAMETERS := FOCUS(INDEX).STEP
```

```
    RQB(ROBJID).PARAMETERS(2) := FOCUS(INDEX).DIRECTION
```

```
    MCT(MEADDR).NEXT.BODY := <OFF>
```

```
    RUN RDMENU
```

```
FI
```

```
CCRP
```

```
A-81
```

```

1  PROC JAR025          (PAN/TILT, CAMERA POSITIONING MENU)
2  DATA
3      SCALAR CAMERA,
4          SLEW, TOUCH,
5          INDEX: INTEGER
6      ARRAY SLEW (1:8) OF SLEW-ENTRY
7      RECORD SLEW-ENTRY
8          .PAN,
9          .TILT: INTEGER
10
11  ATAD
12  MEADDR := STACK(STACK.TOP).ELEMENT
13  STACK.TOP := STACK.BUITEM
14  IF
15      CAMERA = 0
16  THEN
17      ERRMSG := 12
18      RUN GRMST (USE ERRMSG)
19
20  ELSE
21      ROBJID := 1          (SLEW MESSAGE)
22      RUN GMSGAD (USE ROBJID, ALT RNE)
23      ROB(ROBJID).MODIFIER := CAMERA.NUMBER
24      IF
25          MEADDR = SLEW.TOUCH      (A RE-TOUCH?)
26      THEN
27          ROB(ROBJID).PARAMETERS := <OFF>          (PAN)
28          ROB(ROBJID).PARAMETERS(2) := <OFF>      (TILT)
29          RUN SYSREQ (USE ROB)
30      ELSE
31          IF
32              SLEW.TOUCH <> 0      (ANOTHER SLEW ON?)
33          THEN
34              MCT(SLEW.TOUCH).NEXT.BODY := <OFF>
35              ROB(ROBJID).PARAMETERS := <OFF>
36              ROB(ROBJID).PARAMETERS(2) := <OFF>
37              RUN SYSREQ (USE ROB)
38          FI
39          MCT(MEADDR).NEXT.BODY := <ON>
40          INDEX := (MCT(MEADDR).IE - 29)
41          ROB(ROBJID).PARAMETERS := SLEW(INDEX).PAN
42          ROB(ROBJID).PARAMETERS(2) := SLEW(INDEX).TILT
43          RUN SYSREQ (USE ROB)
44      FI
45  FI
46  RUN RCMENU
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80

```

1	
2	
3	PRUC 1AR026 [GO KEY, MENU C05]
4	RUN 1AR027 [STOP CAMERA MOTION]
5	RUN 1AR19A (USE DISCONNECT)
6	CAMERA := 0
7	RUN MNEXT [GET NEXT MENU NUMBER]
8	RUN GRMST (USE 0) [RESET ANY PENDING ERRORS]
9	CORP
10	
11	
12	
13	
14	
15	
16	
17	
18	
19	
20	
21	
22	
23	
24	
25	
26	
27	
28	
29	
30	
31	
32	
33	
34	
35	
36	
37	
38	
39	
40	
41	
42	
43	
44	
45	
46	
47	
48	
49	
50	
51	
52	
53	
54	
55	
56	
57	
58	
59	
60	
61	
62	
63	
64	
65	
66	
67	
68	
69	
70	
71	
72	
73	
74	
75	
76	
77	
78	
79	
80	
81	
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	
96	
97	
98	
99	
100	

A-83

```

1  PROC 1AR027    (STOP SLEW CAMERA ACTION)
2  DATA
3           SCALAR  CAMERA,
4           SLEW.TOUCH,
5           PRESET: INTEGER
6
7  ATAD
8  IF
9           <CAMERA ON AND SLEW ON>
10
11    THEN
12           MCI(SLEW.TOUCH).NEXT.BODY := <OFF>
13           RQBID := 1
14           RUN GMSGAD (USE RQBID, ALT RQB)
15           RQB(6) := <OFF>
16           RUN SYSREQ (USE RQB)
17           SLEW.TOUCH := 0
18           RQBID := 9
19           RUN GMSGAD (USE RQBID, ALT RQB)
20           RUN SYSREQ (USE RQB)    (DO A WAIT TIME)
21           RUN RCMENU
22    FI
23
24  CCRP
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90

```



```
PROC RSTPST (RESET MASTER MENU ACTIVE PRESETS)
DATA
  SCALAR LEFT,
  PTR,
  TYPE: INTEGER
ATAD
IF
  <CAMERA TOUCHED>
THEN
  RUN ESTADD (USE CAMERA, ALT ESADDR)
  IF
    EST(ESADDR) = <LEFT CAMERA>
  THEN
    PTR := <LEFT CAMERA PRESET 1 EST ID>
    TYPE := >0400
  ELSE
    PTR := <RIGHT CAMERA PRESET 1 EST ID>
    TYPE := >0500
  FI
  FOUND := FALSE
  WHILE
    EST(PTR).TYPE = TYPE AND NOT FOUND
  DO
    PTR := PTR + ESTLEN
    IF
      EST(PTR).ON = 1 (PRESET IS ON)
    THEN
      FOUND := TRUE
      EST(PTR).ON := <OFF>
      EST(PTR).COLOR := 0
      EST(PTR).CB := 0
      EST(PTR).PRESENT := 0
    FI
  OD
  FI
CORP
```

A-85

```

PROC DAH001 [SOURCE = SINK CONNECTION]
DATA
    SCALAR SOURCE,
    CONTROLLER,
    COLOR,
    PIR,
    LOOP: INTEGER
    SCALAR KEEP,
    PRESET: LOGICAL
ATAD
    RQBID := 8 [CONNECT MESSAGE]
    RUN GMSGAD (USE RQBID, ALT RQB)
    MEADDR := STACK(STACK.TOP).ELEMENT
    SOURCE := MEADDR
    MCT(MEADDR).NEXT.BORDER := <OFF>
    RUN ESTADD (USE MEADDR, ALT ESADDR)
    RQB(RQBID).PARAMETERS := SOURCE
    EST(ESADDR).FON := <ON>
    CONTROLLER := EST(ESADDR).ID
    COLOR := EST(ESADDR).COLOR
    WHILE
        <STACK NOT EMPTY>
    DO
        RUN POP-ELEM (ALT MEADDR)
        MCT(MEADDR).NEXT.BORDER := <OFF>
        RUN ESTADD (USE MEADDR, ALT ESADDR)
        IF
            MEADDR = <STILL FRAMED> OR
            MEADDR = <AUTO TRANSMIT>
        THEN
            <DETERMINE WHICH KEY WILL GO
            ON AND WHICH WILL GO OFF, BASED
            ON PREVIOUS STATE>
        FI
        RQB(RQBID).PARAMETERS(2) := EST(ESADDR).ID
        IF
            EST(ESADDR).COLOR <> COLOR OR
            EST(ESADDR).FON = <OFF>
        THEN
            IF
                EST(ESADDR).TYPE = < 1
            THEN
                IF
                    EST(ESADDR).FON <> 0
                THEN
                    RQB(RQBID).PARAMETERS := EST(ESADDR).CB
                    RQB(RQBID).PARAMETERS(2) := EST(ESADDR).II
                    RUN SYSREQ (USE RQB)
                    RQB(RQBID).PARAMETERS := SOURCE
                    RQB(RQBID).PARAMETERS(2) := EST(ESADDR).II
                FI
                RUN SYSREQ (USE RQB)
    
```

```

                                FI
                                EST(ESADDR).CB := CONTROL
                                EST(ESADDR).COLOR := COLOR
                                EST(ESADDR).FUN := <ON>
                                MCT(MEADDR).NEXT.BODY := <OFF>
                                MCT(MEADDR).CURRENT.BGDY := <OFF>
                                FI
                                OD
                                IF
                                SOURCE = CANTCH           [SOURCE WAS A TOUCHED CAMERA]
                                THEN
                                PRESET := <TRUE IF A PRESET WAS TOUCHED,
                                FALSE IF A PRESET WAS NOT TOUCHED>
                                PTR := CANTCH
                                WHILE
                                CARLUP > 0
                                DO
                                PTR := PTR + MCILEN
                                KEEP := FALSE
                                IF
                                (PRESET AND MCT(PTR).CURRENT.BORDER = <OFF>)
                                OR
                                (NOT PRESET AND MCT(PTR).CURRENT.BODY = <OFF>)
                                THEN
                                RUN ESTADD (USE PTR, ALT ESADDR)
                                EST(ESADDR).CB := 0
                                EST(ESADDR).FCN := <OFF>
                                EST(ESADDR).FPRE := <OFF>
                                RUN ZAPOUT (USE PTR) [ERASE PRESET]
                                MCT(PTR).CURRENT := MCT(PTR).NEXT
                                FI
                                CARLUP := CARLUP - 1
                                OD
                                CANTCH := 0
                                FI
                                RUN LSEND5
CORP
```

PRUC DAROUZ (SELECI OTHER FUNCTION MENU, KEYSJ HEADDR := STACK(STACK.TOP), ELEMENT RUN ESTADD (USE HEADR
ALT ESALDK) PUMENU := EST(FSADR), FLDI KUN RESET CURP

A-88

```
3  
5  
7  
8 PRJC DAP003 (HI-RES SCREEN)  
9 STACK(STACK.TOP),DELAG := <OFF>  
10 ERRMSG := 2  
11 RUN GMSI (USE ERRMSG)  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200
```

A-89

```
1  
2  
3  
4 PROC DAK004 (PREVIEW WITH NO SOURCE)  
5 STACK(STACK.TOP),DELAG := <DEF>  
6 ERRMSG := 3  
7 RUN GRMST (USE ERRMSG)  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200
```

A-90

```

=====
*PROC = CCTSPI
*
*   PROCEDURE NAME: CCTSPI
*
*   CCTSPI RECEIVES X AND Y TOUCH COORDINATES FROM THE
*   TOUCH SENSITIVE PANEL CONTROLLER. IT LOCATES THE
*   ELEMENT TOUCHED (IF ANY), STACKS IT AND SETS ITS
*   IMMEDIATE FLAG ON. THE NEXT PHASE OF THIS PROCEDURE
*   CONTINUALLY CHECKS THE TOP ELEMENT ON THE STACK TO
*   DETERMINE IF AN IMMEDIATE OR DELAYED ACTION SHOULD BE
*   CALLED. AFTER EXECUTION OF AN ACTION ROUTINE, THE
*   TOP OF THE STACK IS AGAIN EXAMINED. IF THE STACK IS
*   EMPTY OR BOTH FLAGS ARE OFF, THE PROCEDURE RETURNS TO
*   THE CALLER.
*
*   INPUT: R0 CONTAINS THE X COORDINATE
*          R1 CONTAINS THE Y COORDINATE
*   OUTPUT: NONE
*   RESULT: VARIABLE BASED ON ELEMENT STACKED
*   TO CALL: BL @CCTSPI
=====

```

```
*****
*PROC = GACTAD
*
*  PROCEDURE NAME: GACTAD
*
*  GACTAD WILL DERIVE THE ADDRESS OF A ACTION ROUTINE.
*
*  INPUT: R3 CONTAINS THE ACTION ROUTINE ADDRESS
*  OUTPUT: R3 CONTAINS THE MAPPED ADDRESS
*  RESULT: NONE
*  TO CALL: BL @GACTAD
*
*****
```



```

=====
*PROC = FINDEL
*
*   PROCEDURE NAME: FINDEL
*
*   FINDEL LOOPS THROUGH THE CURRENT MENU'S MCT TABLE TO
*   DETERMINE WHICH MENU ELEMENT WAS TOUCHED. FIRST, THE
*   HERE COORDINATES IN THE MCT ARE SUBTRACTED FROM THE
*   COORDINATES SUPPLIED BY THE TOUCH SENSITIVE PANEL. IF
*   EITHER X OR Y VALUE IS MADE NEGATIVE BY THE SUBTRACTION,
*   THEN THIS WAS NOT THE ELEMENT TOUCHED. NEXT, THE
*   ELEMENT'S DISPLAY ELEMENT LIBRARY (DEL) ENTRY IS
*   FOUND. IF THE COORDINATES ARE BOTH LESS THAN THE
*   X AND Y TOLERANCES FOUND IN THE DEL, THEN THIS IS THE
*   TOUCHED ELEMENT. IF NOT, THE SEARCH CONTINUES UNTIL
*   ALL ELEMENTS ARE EXAMINED. IF AN ELEMENT DOES NOT
*   HAVE A DEL ENTRY, OR ITS INSENSITIVE OR INHIBIT MCT
*   BITS ARE ON, THEN DO NOT TEST THIS ELEMENT.
*
*   INPUT: R0 CONTAINS X COORDINATE
*           R1 CONTAINS Y COORDINATE
*   OUTPUT: R2 CONTAINS THE TOUCHED ELEMENTS MCT ADDRESS,
*           OR ZERO IF NO ELEMENT IS FOUND.
*   RESULT: TOUCHED ELEMENT IS LOCATED IN THE MCT.
*   TO CALL: BL @FINDEL
*
=====

```

```
*****
*PROC = LEXSCN
*
*  PROCEDURE NAME: LEXSCN
*
*  LEXSCN EXAMINES THE ELEMENT STACK TO DETERMINE IF ENOUGH
*  INFORMATION HAS BEEN GIVEN IN A CORRECT MANNER SO
*  THAT SUBSEQUENT ROUTINE ACTIONS WILL BE SUCCESSFUL.
*
*  BASIC REQUIRMENTS ARE SIMPLE.  EACH SENTENCE MUST
*  INCLUDE A SOURCE (CAMERA, SLIDE CAMERA, RECEIVE) AND
*  ONE OR MORE DESTINATIONS (TV SCREEN, OVERHEAD MONITORS,
*  SEND).  IF THE ELEMENT STACK DOES NOT INCLUDE A SOURCE
*  BUT DOES HAVE AN ACTIVE DESTINATION (A DESTINATION
*  WHICH IS CURRENTLY LINKED TO A SOURCE), THE SOURCE FOR
*  THAT DESTINATION IS PLACED ON THE STACK.
*
*  IF LEXSCN DETECTS AN ERROR, IT SETS LEXRC TO TWO (2)
*  AND INDICATES THE APPROPRIATE ERROR.
*
*  INPUT: ELEMENT STACK
*  OUTPUT: NONE
*  RESULT: ELEMENTS EITHER PASS OR FAIL SCAN
*  TO CALL: BL @LEXSCN
*****

```