

KTF11-AA

KTF11-AA DIAG
CJKDACO

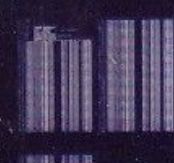
AH-F137C-MC
FICHE 1 OF 1

MAY 1980
COPYRIGHT © 1979
MADE IN USA



22

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



IDENTIFICATION

PRODUCT CODE: AC-F138C-MC
PRODUCT NAME: CJKDAC0 KTF11-AA MEMORY MANAGEMENT DIAGNOSTIC
DATE: NOV-1979
MAINTAINER: DIAGNOSTIC PROGRAMMING

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1979 BY DIGITAL EQUIPMENT CORPORATION

PROGRAM HISTORY

<u>DATE</u>	<u>REVISION</u>	<u>REASON FOR REVISION</u>
12-JAN-79	A	FIRST RELEASE
JUNE-79	B	SUBROUTINE FORMPA MODIFIED ERROR INFORMATION STORED IN R0,R2. THIS REVISION SAVES THE REGISTERS ON ENTRY TO THE ROUTINE AND RESTORES THEM ON EXIT.
NOVEMBER-79	C	CORRECTIONS WERE MADE TO THE MULTI-TESTER SUPPORT CODE. ALSO CODE WAS ADDED TO ALLOW PROGRAM OPERATION WHILE LINE CLOCK IS INTERRUPTING.

TABLE OF CONTENTS

- 1.0 PROGRAM INFORMATION
 - 1.1 ABSTRACT
 - 1.2 REQUIREMENTS
 - 1.3 RELATED DOCUMENTS AND STANDARDS
 - 1.4 PRELIMINARY PROGRAMS
- 2.0 OPERATING INSTRUCTIONS
 - 2.1 LOADING PROCEDURES
 - 2.2 STARTING PROCEDURES
 - 2.3 OPERATIONAL SWITCH SETTINGS
 - 2.4 LOADING THE SWITCH REGISTER
 - 2.5 EXECUTION TIMES
- 3.0 ERROR INFORMATION
 - 3.1 ERROR REPORTING PROCEDURES
 - 3.2 INTERPRETING ERROR REPORTS
 - 3.3 SAMPLE ERROR REPORT
- 4.0 MISCELLANEOUS INFORMATION
 - 4.1 ACT/APT/XXDP COMPATABILITY
 - 4.2 END-OF-PASS MESSAGE
 - 4.3 T-BIT TRAPPING
 - 4.4 POWER FAILURE HANDLING
 - 4.5 PHYSICAL BUS ADDRESS CONSTRUCTION
 - 4.6 RELOCATION THROUGHOUT MEMORY
- 5.0 PROGRAM DESCRIPTION
 - 5.1 SUBROUTINES USED BY THIS PROGRAM
 - 5.2 PROGRAM LISTING
 - 5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

1.0 PROGRAM INFORMATION

1.1 ABSTRACT

THIS PROGRAM WAS DESIGNED USING A 'BOTTOM UP' APPROACH STARTING WITH THE SMALLEST SEGMENT OF MEMORY MANAGEMENT LOGIC POSSIBLE AND BUILDING TO COVER ALL OF THE LOGIC. THE DIAGNOSTIC WILL PROVIDE ENOUGH INFORMATION SUCH THAT BY DEDUCTION, THE FAILURE CAN BE ISOLATED TO A SMALL SEGMENT OF THE MEMORY MANAGEMENT LOGIC.

THE PROGRAM BEGINS BY TESTING SOME OF THE INTERNAL CPU DATA AND ADDRESS PATHS AND ADDRESS DETECTION LOGIC, THEN WORKS OUTWARD THROUGH THE MEMORY MANAGEMENT REGISTERS. AFTER THE REGISTERS ARE FOUND TO BE USEABLE, RELOCATION (CONSTRUCTION OF PHYSICAL ADDRESSES FROM A VIRTUAL ADDRESS AND THE ASSOCIATED PAR/PDR INFORMATION) IS TESTED FOLLOWED BY TESTING OF THE ABORT AND STATUS SEGMENTS OF LOGIC. FINALLY, CHECKS OF SPECIAL ABORT SEQUENCES AND TESTING OF THE MFPI/MTPI INSTRUCTIONS ARE DONE.

1.2 REQUIREMENTS

A KDF11 PROCESSOR WITH A MINIMUM OF 16K OF MEMORY AND A CONSOLE TERMINAL ARE REQUIRED TO RUN THE PROGRAM UNLESS THE PROGRAM IS RUNNING UNDER APT OR ACT IN WHICH CASE THE CONSOLE TERMINAL IS NOT NECESSARY.

1.3 RELATED DOCUMENTS AND STANDARDS

1. ACT11/XXDP PROGRAMMING SPECIFICATION
2. STANDARD APT SYSTEM TO A PDP11 DIAGNOSTIC INTERFACE
3. DIAGNOSTIC ENGINEERING STANDARDS AND CONVENTIONS
4. PDP11 MAINDEC SYSMAC PACKAGE
5. XXDP USER'S MANUAL

1.4 PRELIMINARY PROGRAMS

BEFORE THIS MEMORY MANAGEMENT DIAGNOSTIC IS RUN, THE FOLLOWING CPU DIAGNOSTIC SHOULD BE RUN:

CJKDB DCF11-AA CPU TESTS

ALSO, ONE OF THE MAIN MEMORY DIAGNOSTICS SHOULD BE RUN TO SCAN AT LEAST THE FIRST 16K TO SEE THAT A PROGRAM CAN BE EXECUTED.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING PROCEDURES

THE PROGRAM IS SUPPLIED ON THE DIAGNOSTIC LOAD MEDIA. REFER TO THE XXDP USER'S MANUAL FOR FURTHER INFORMATION. FOR USE WITH ACT OR APT, REFER TO THEIR RESPECTIVE DOCUMENTS. THE PROGRAM CAN ALSO BE DIRECTLY LOADED USING THE ABSOLUTE LOADER AND THE BINARY PAPER TAPE.

2.2 STARTING PROCEDURES

THE PROGRAM IS STARTED BY LOADING ADDRESS 200. SINCE THERE IS NO HARDWARE SWITCH REGISTER, THE PROGRAM WILL USE THE SOFTWARE SWITCH REGISTER AT LOCATION 176 (LOCATION 174 WILL BE USED AS THE SOFTWARE DISPLAY REGISTER). IN THAT CASE THE PROGRAM WILL ASK FOR THE INITIAL SWITCH REGISTER VALUE BY TYPING 'SWR= XXXXXX NEW= ' AFTER TYPING THE NAME OF THE PROGRAM (XXXXXX = THE OCTAL CONTENTS OF LOCATION 176). (SEE SECTION 2.4)

2.3 CONTROL SWITCH SETTINGS

<u>SWITCH</u>	<u>OCTAL VALUE</u>	<u>USE</u>
SW15	100000	HALT ON ERROR THIS SWITCH WHEN SET WILL HALT THE PROCESSOR WHEN AN ERROR IS DETECTED AFTER THE ERROR MESSAGE HAS BEEN TYPED. PRESSING CONTINUE WILL RESUME TESTING (SEE SECTION 3.1 ABOUT LOADING THE SWITCH REG BEFORE CONTINUING).
SW14	040000	LOOP ON TEST THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE CURRENT SUBTEST.
SW13	020000	INHIBIT ERROR TYPEOUTS THIS SWITCH WHEN SET WILL INHIBIT THE TYPING OF ERROR MESSAGES.
SW12	010000	INHIBIT TRACE TRAP THIS SWITCH WHEN SET WILL INHIBIT T-BIT TRAPPING WHICH NORMALLY TAKES PLACE DURING EVERY OTHER PASS STARTING WITH THE THIRD PASS.
SW11	004000	INHIBIT SUBTEST ITERATIONS THIS SWITCH WHEN SET INHIBITS ITERATIONS OF EACH SUBTEST AFTER THE FIRST PASS. IF THIS SWITCH IS NOT SET, EACH SUBTEST IS RUN 200. TIMES.

- SW10 002000 BELL ON ERROR
THIS SWITCH WHEN SET WILL RING THE CONSOLE TERMINAL BELL WHEN AN ERROR HAS BEEN DETECTED.
- SW9 001000 LOOP ON ERROR
THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE FIRST FAILURE WHICH IS ENCOUNTERED EVEN IF THE FAILURE IS INTERMITTANT
- SW8 000400 LOOP ON TEST IN SWR<7:0>
THIS SWITCH WHEN SET WILL CAUSE THE PROGRAM TO LOOP ON THE TEST WHOSE TEST NUMBER IS SET IN BITS 7-0 OF THE SWITCH REG.

2.4 LOADING THE SWITCH REGISTER

TO LOAD THE SOFTWARE SWITCH REG. WHILE THE PROGRAM IS RUNNING, A CONTROL G (^G) SHOULD BE TYPED ON THE CONSOLE TERMINAL. (THE 'SCOPE' AND 'ERROR' ROUTINES CHECK TO SEE IF A ^G HAS BEEN TYPED.) THE ORIGINAL VALUE OF THE SOFTWARE SWITCH REG. WILL BE REQUESTED AS MENTIONED IN SECTION 2.2.

IN RESPONSE TO A ^G OR AT THE BEGINNING OF THE PROGRAM, THE PROGRAM WILL TYPE:

SWR = XXXXXX NEW =

WHERE 'XXXXXX' IS THE CURRENT OCTAL CONTENTS OF LOC. 176. THE OPERATOR MAY THEN TYPE ANY ONE OF THE FOLLOWING:

- XXXXXX<CR> ONE TO SIX OCTAL DIGITS FOLLOWED BY A CARRIAGE RETURN WHICH WILL BE LOADED AS THE NEW VALUE FOR THE SWITCH REG.
- <CR> JUST A <CR>, LEAVES THE SWITCH REG. AS IT IS.
- XXX^U A CONTROL-U (^U) WILL CAUSE ALL OF THE DIGITS TYPED SO FAR TO BE IGNORED.
- ^C WILL CAUSE THE PROGRAM TO TYPE THE PRESENT TEST AND PASS NUMBERS, REQUEST A NEW VALUE FOR THE SWITCH REG., AND JUMP TO THE END-OF-PASS ROUTINE SO THE PROGRAM WILL GO DIRECTLY TO THE NEXT PASS WITH A NEW SW. REG. VALUE
- <ILL.CHAR> ANY CHARACTER TYPED WHICH IS NOT ANY OF THE ABOVE OR AN OCTAL DIGIT WILL CAUSE THE PROGRAM TO TYPE A "?<CRLF>" AND REACT AS THOUGH A ^U HAD BEEN TYPED.

NOTE: RECOGNITION OF A ^G MAY BE HAMPERED BY EXECUTION OF A COUPLE 'RESET' INSTRUCTIONS WITHIN THE PROGRAM.

2.5 EXECUTION TIMES

THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS OR TRACE TRAPPING IS APPROXIMATELY 5 SECONDS.

THE RUN TIME FOR A SINGLE PASS WITH ITERATIONS AND TRACE TRAPPING ENABLED IS APPROXIMATELY 30 SECONDS.

3.0 ERROR INFORMATION

3.1 ERROR REPORTING PROCEDURES

IF AN ERROR IS DETECTED, THE PROGRAM WILL TRAP TO THE ERROR HANDLING ROUTINE (\$ERROR). THE VALUE OF BITS 15,13,10, AND 9 IN THE SWITCH REGISTER ARE CONSIDERED IN REPORTING AN ERROR (SEE SECTION 2.3). THE ERROR INFORMATION WILL BE TYPED UNLESS SW13 = 1.

IF SW15 = 1, THE PROCESSOR WILL HALT AFTER THE ERROR IS REPORTED. IF THE CONTENTS OF THE SOFTWARE SWITCH REGISTER ARE TO BE CHANGED, A ^G SHOULD BE TYPED BEFORE PRESSING 'CONTINUE' TO RESUME TESTING.

IF SW9 = 1 (LOOP ON ERROR), THE PROGRAM WILL GO TO THE ADDRESS CONTAINED IN LOCATION '\$LPERR'. AFTER REPORTING THE ERROR, '\$LPERR' IS SET BY EACH 'SCOPE' CALL AND IS SET DIRECTLY DURING SOME SUBTESTS TO PROVIDE THE SMALLEST LOOP FOR LOOPING ON ERROR. IF SW9 = 0, THE PROGRAM WILL RETURN TO THE INSTRUCTION FOLLOWING THE ERROR CALL. (SEE SECTION 5.3 FOR MORE ON 'LOOP ON ERROR').

3.2 INTERPRETING ERROR REPORTS

EVERY ERROR REPORT TYPES THE NUMBER OF THE TEST IN WHICH THE ERROR TOOK PLACE (TESTNO) AND THE LOCATION OF THE ERROR CALL (ERRORPC). THESE TWO VALUES PINPOINT THE PLACE IN THE CODE THAT THE ERROR OCCURRED. BY REFERRING TO THE PROGRAM LISTING, THE OPERATOR CAN THEN READ THE COMMENTS ASSOCIATED WITH THAT PARTICULAR ERROR AND SUBTEST. A DESCRIPTION OF THE TEST FOUND IN THE PROGRAM LISTING WILL ALSO PROVIDE THE OPERATOR WITH INFORMATION ON THE LOGIC AND FUNCTIONS BEING TESTED.

EVERY ERROR REPORT ALSO TYPES AN ERROR MESSAGE GIVING A VERBAL DESCRIPTION OF THE ERROR THAT HAS BEEN DETECTED.

BY USING THE COMMENTS AND TEST DESCRIPTION FOUND IN THE PROGRAM LISTING TO DETERMINE WHAT FUNCTION OR LOGIC WAS BEING TESTED, THE OPERATOR CAN THEN REFER TO THE ENGINEERING DRAWINGS TO ISOLATE THE PROBABLE CAUSE FOR THE FAILURE.

3.3 SAMPLE ERROR REPORT

I 1

SEQ 0008

BELOW IS AN EXAMPLE OF AN ERROR WHICH COULD HAVE OCCURRED DURING EXECUTION OF THE PROGRAM:

```
MEM. MGMT. REG. BITS NOT SET CORRECTLY
REGISTR WROTE  READ  READ-(BINARY)
ADDRESS (OCTAL) (OCTAL) 5432109876543210  TESTNO  ERRORPC
177572  040000  060000  0110000000000000  000012  022060
```

WE SEE THAT THE ERROR OCCURRED IN TEST 12 AT LOACTION 022060. THE 'REGISTR ADDRESS' TELLS US THAT WE WERE TESTING MEMORY MANAGEMENT'S STATUS REGISTER 0 (SRO). IN THE LISTING, THE TEST DESCRIPTION SAYS THAT THE ERROR BITS (BITS <15:13>) OF SRO WERE BEING SET AND CLEARED INDIVIDUALLY. THE ERROR REPORT SAYS WE TRIED TO SET BIT 14 BY WRITING '040000' TO SRO BUT WHEN WE READ IT BACK WE READ '060000'. IT APPEARS THAT BIT 13 IS STUCK AT '1' OR IT IS GETTING SET WHEN BIT 14 IS SET TO '1'. ERROR REPORTS BEFORE AND AFTER THIS ONE COULD TELL US WHICH IS THE CASE.

4.0 MISCELLANEOUS INFORMATION

4.1 ACT/APT/XXDP COMPATABILITY

THE PROGRAM IS FULLY ACT AND APT COMPATABLE AND IS SUPPORTED UNDER THE XXDP PACKAGE.

4.2 END-OF-PASS MESSAGE

AT THE END OF EACH PASS OF THE PROGRAM THE PASS NUMBER AND TOTAL NUMBER OF ERRORS SINCE THE LAST END-OF-PASS ARE REPORTED IN THE END-OF-PASS MESSAGE. FOR EXAMPLE:

```
END OF PASS #2  TOTAL ERRORS SINCE LAST REPORT 0
```

THAT WOULD INDICATE THAT PASS TWO WAS JUST COMPLETED AND NO ERRORS WERE DETECTED DURING THAT PASS. BOTH THE PASS NUMBER AND NUMBER OF ERRORS ARE DECIMAL NUMBERS.

4.3 T-BIT TRAPPING

THE 'T-BIT' (BIT 4) IN THE PROCESSOR STATUS WORD IS SET BY AN 'RTI' IN THE END-OF-PASS ROUTINE FOR EVERY OTHER PASS BEGINNING WITH THE THIRD PASS (PASSES 3,5,7,9...). T-BIT TRAPPING CAN BE INHIBITED BY SETTING BIT 12 = 1 IN THE SWITCH REGISTER (SEE SECTION 2.4).

4.4 POWER FAILURE HANDLING

IF A POWER FAIL OCCURS (FOLLOWED BY A POWER UP), THE MESSAGE 'POWER FAILURE-RESTARTING' IS TYPED OUT AND THE PROGRAM WILL RESTART EXECUTION AT 'RESTRT:'

5.0 PROGRAM DESCRIPTION

5.1 SUBROUTINES USED BY THIS PROGRAM

FOLLOWING IS A LIST OF THE SUBROUTINES AND HANDLERS USED BY THIS PROGRAM THAT ARE NOT PROVIDED BY THE 'SYSMAC PACKAGE'. DETAILS OF THE SUBROUTINES UNIQUE TO THIS PROGRAM MAY BE FOUND IN THE PROGRAM LISTING. REFER TO THE 'SYSMAC' DOCUMENT AND PROGRAM LISTING FOR THE OTHER ROUTINES.

1. TURN OFF T-BIT AND SAVE CURRENT PSW
2. TURN ON T-BIT AND RESTORE PREVIOUS PSW
3. SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
4. READ AND COMPARE KERNEL AND USER PAR/PDR'S
5. CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

5.2 PROGRAM LISTING

A TABLE OF CONTENTS APPEARS AT THE BEGINNING OF THE LISTING WHICH CONTAINS THE NAMES OF EACH SECTION, SUBTEST, AND ROUTINE AND THE LINE NUMBERS CORRESPONDING TO THE START OF EACH.

FOLLOWING THIS SECTION OF DOCUMENTATION IS THE ACTUAL PROGRAM LISTING COMPLETE WITH SUBTEST DESCRIPTIONS AND 'CODING COMMENTS'.

5.3 USING THE PROGRAM TO DIAGNOSE A FAULT

WHEN AN ERROR OCCURS, ONE OF THE THINGS THAT'S IMPORTANT TO NOTE IS WHAT PASS THE ERROR OCCURRED ON. IF THE PASS NUMBER IS ODD AND IS THREE OR GREATER, THE ERROR MIGHT BE T-BIT SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 12 OF THE SWITCH REG. EQUAL TO '1' TO INHIBIT T-BIT TRAPPING. IF THE PASS NUMBER IS GREATER THAN ONE, THE ERROR MAY BE ITERATION SENSITIVE. TRY RUNNING THE PROGRAM AGAIN WITH BIT 11 OF THE SWITCH REG. EQUAL TO '1' TO INHIBIT ITERATIONS. THESE HINTS SHOULD HELP YOU DETERMINE WHAT MAKES THE MACHINE FAIL AND WHEN.

IF YOU HAVE BEEN RUNNING WITH BIT 15 OF THE SWITCH REG. EQUAL TO '0', THEN YOU ARE ABLE TO LOOK AT ALL THE ERRORS THAT MAY BE RELATED TO THE FAULT YOU ARE DIAGNOSING. A FAULT IN AN EARLIER TEST MAY RESULT IN ERRORS DURING LATER TESTS WHICH MAY GIVE YOU MORE CLUES ABOUT THE NATURE OF THE FAULT. NOW USE THE METHOD OUTLINED IN SECTION 3.2 FOR EACH ERROR TO GATHER AS MUCH INFORMATION AS POSSIBLE.

NOW TO TEST YOUR IDEAS ON THE CAUSE OF THE FAILURE,
YOU MAY WANT TO SCOPE THIS ERROR CONDITION. SET BIT 09
OF THE SWITCH REG. EQUAL TO '1' TO LOOP ON THE ERROR.
FOR AN EVEN TIGHTER SCOPE LOOP THE ERROR CALL CAN BE
REPLACED WITH A BRANCH (REFER TO COMMENTS BY ERROR CALLS
IN THE PROGRAM LISTING).

OR YOU COULD LOOP ON THE TEST BY EITHER SETTING BIT 14
OF THE SWITCH REG. EQUAL TO '1' OF BY SETTING BIT 08 OF THE
SWITCH REG. EQUAL TO '1' AND THEN SETTING THE TEST NUMBER
IN BITS 07-00 OF THE SWITCH REG. YOU WILL PROBABLY WANT TO
INHIBIT ERROR TYPEOUTS BY SETTING BIT 13 OF THE SWITCH REG.
EQUAL TO '1'.

20	OPERATIONAL SWITCH SETTINGS
21	BASIC DEFINITIONS
22	MEMORY MANAGEMENT DEFINITIONS
258	TRAP CATCHER
(1)	STARTING ADDRESS(ES)
259	ACT11 HOOKS
260	APT PARAMETER BLOCK
261	COMMON TAGS
(2)	APT MAILBOX-ETABLE
(1)	ERROR POINTER TABLE
560	***** TRAP HANDLING ROUTINES *****
562	CPU TRAP HANDLER ROUTINE
590	MEMORY MANAGEMENT TRAP HANDLER ROUTINE
621	
622	***** STARTING POINT OF TEST *****
623	***** STARTING ADDRESS OF 200 *****
626	INITIALIZE THE COMMON TAGS
629	TYPE PROGRAM NAME
(2)	GET VALUE FOR SOFTWARE SWITCH REGISTER
652	T1 PSW PRIORITY BIT TEST
674	T2 PSW MODE BIT TEST
699	T3 BYTE ADDRESSING TEST FOR PSW
737	T4 TEST AND SETUP OF STACK POINTERS
775	T5 SRO,SR1,SR2,SR3 TIMEOUT TEST
812	T6 KERNEL PAR'S TIMEOUT TEST
818	T7 KERNEL PDR'S TIMEOUT TEST
824	T10 USER PAR'S TIMEOUT TEST
830	T11 USER PDR'S TIMEOUT TEST
843	T12 SRO(15:13) BIT TEST & SR2 TEST
898	T13 SRO & PSW DUAL ADDRESSING TEST
919	T14 TEST THAT SR1 READS ALL ZEROS
945	T15 BIT TEST OF KERNEL & USER PAR'S
988	T16 BIT TEST OF KERNEL & USER PDR'S
1033	T17 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
1077	T20 TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
1124	T21 PAR-PDR DUAL ADDRESSING TEST
1170	T22 TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
1236	T23 RELOCATION & ADDER TEST (NO CARRIES)
1315	T24 RELOCATION & ADDER TEST (WITH CARRIES)
1352	T25 READ AND WRITE WHILE IN RELOCATE MODE
1457	T26 W-BIT LOGIC TEST, KERNEL PDR'S
1464	T27 W-BIT LOGIC TEST, USER PDR'S
1478	T30 TEST 'W-BIT' SPECIAL CASES
1533	T31 NON-RESIDENT ABORT TEST (ACF=0&4)
1605	T32 READ-ONLY ABORT TEST (ACF=2)
1697	T33 PAGE LENGTH FAULTS-UPWARD EXPANSION
1811	T34 PAGE LENGTH FAULTS-DOWNWARD EXPANSION
1926	T35 SR2 BIT TEST
1975	T36 MORE CHECKS OF SRO & SR2
2064	T37 USER ABORT PICKS UP KERNEL SPACE VECTOR
2102	T40 RTI IN USER MODE DOES NOT CHANGE PSW
2134	T41 KT ERROR SERVICED BEFORE TIMEOUT ERROR
2183	T42 PC & PSW SAVED FOR KT ERROR DURING SERVICE OF TIMEOUT ERROR
2260	T43 MOVE FROM PREVIOUS (USER) I-SPACE
2433	T44 MOVE TO PREVIOUS (USER) I-SPACE
2604	T45 MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE

2755	T46	MOVE FROM/TO D-SPACE = MOVE FROM/TO I-SPACE
2797	T47	MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
2820		END OF PASS ROUTINE
2896		SCOPE HANDLER ROUTINE
2897		ERROR HANDLER ROUTINE
2898		ERROR MESSAGE TIMEOUT ROUTINE
2987	*****	SUBROUTINES USED BY THIS PROGRAM *****
2989		TURN OFF T-BIT AND SAVE CURRENT PSW
3008		TURN ON T-BIT AND RESTORE PREVIOUS PSW
3024		SET ALL WRITEABLE BITS IN ALL PAR/PDR'S
3053		READ & COMPARE KERNEL & USER PAR/PDR'S
3122		CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS
3165		TTY INPUT ROUTINE
3167		CONTROL-C SERVICING ROUTINE
3192		TYPE ROUTINE
3193		APT COMMUNICATIONS ROUTINE
3194		BINARY TO ASCII AND TYPE ROUTINE
3195		BINARY TO OCTAL (ASCII) AND TYPE
3196		CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3197		SAVE AND RESTORE R0-R5 ROUTINES
3198		DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
3200		TRAP DECODER
(3)		TRAP TABLE
3201		POWER DOWN AND UP ROUTINES
3207		ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

(1)	000100	PR2=	100	::PRIORITY LEVEL 2
(1)	000140	PR3=	140	::PRIORITY LEVEL 3
(1)	000200	PR4=	200	::PRIORITY LEVEL 4
(1)	000240	PR5=	240	::PRIORITY LEVEL 5
(1)	000300	PR6=	300	::PRIORITY LEVEL 6
(1)	000340	PR7=	340	::PRIORITY LEVEL 7

(1) ;*'SWITCH REGISTER' SWITCH DEFINITIONS

(1)	100000	SW15=	100000
(1)	040000	SW14=	40000
(1)	020000	SW13=	20000
(1)	010000	SW12=	10000
(1)	004000	SW11=	4000
(1)	002000	SW10=	2000
(1)	001000	SW09=	1000
(1)	000400	SW08=	400
(1)	000200	SW07=	200
(1)	000100	SW06=	100
(1)	000040	SW05=	40
(1)	000020	SW04=	20
(1)	000010	SW03=	10
(1)	000004	SW02=	4
(1)	000002	SW01=	2
(1)	000001	SW00=	1

(1)		.EQUIV	SW09,SW9
(1)		.EQUIV	SW08,SW8
(1)		.EQUIV	SW07,SW7
(1)		.EQUIV	SW06,SW6
(1)		.EQUIV	SW05,SW5
(1)		.EQUIV	SW04,SW4
(1)		.EQUIV	SW03,SW3
(1)		.EQUIV	SW02,SW2
(1)		.EQUIV	SW01,SW1
(1)		.EQUIV	SW00,SW0

(1) ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

(1)	100000	BIT15=	100000
(1)	040000	BIT14=	40000
(1)	020000	BIT13=	20000
(1)	010000	BIT12=	10000
(1)	004000	BIT11=	4000
(1)	002000	BIT10=	2000
(1)	001000	BIT09=	1000
(1)	000400	BIT08=	400
(1)	000200	BIT07=	200
(1)	000100	BIT06=	100
(1)	000040	BIT05=	40
(1)	000020	BIT04=	20
(1)	000010	BIT03=	10
(1)	000004	BIT02=	4
(1)	000002	BIT01=	2
(1)	000001	BIT00=	1

(1)		.EQUIV	BIT09,BIT9
(1)		.EQUIV	BIT08,BIT8
(1)		.EQUIV	BIT07,BIT7
(1)		.EQUIV	BIT06,BIT6

```
(1) .EQUIV BIT05,BIT5
(1) .EQUIV BIT04,BIT4
(1) .EQUIV BIT03,BIT3
(1) .EQUIV BIT02,BIT2
(1) .EQUIV BIT01,BIT1
(1) .EQUIV BIT00,BIT0
(1)
(1) ;*BASIC "CPU" TRAP VECTOR ADDRESSES
(1) 000004 ERRVEC= 4 ;:TIME OUT AND OTHER ERRORS
(1) 000010 RESVEC= 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
(1) 000014 TBITVEC=14 ;: 'T' BIT
(1) 000014 TRTVEC= 14 ;:TRACE TRAP
(1) 000014 BPTVEC= 14 ;:BREAKPOINT TRAP (BPT)
(1) 000020 IOTVEC= 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1) 000024 PWRVEC= 24 ;:POWER FAIL
(1) 000030 EMTVEC= 30 ;:EMULATOR TRAP (EMT) **ERROR**
(1) 000034 TRAPVEC=34 ;: 'TRAP' TRAP
(1) 000060 TKVEC= 60 ;:TTY KEYBOARD VECTOR
(1) 000064 TPVEC= 64 ;:TTY PRINTER VECTOR
(1) 000240 PIRQVEC=240 ;:PROGRAM INTERRUPT REQUEST VECTOR
22 .SBTTL MEMORY MANAGEMENT DEFINITIONS
(1)
(1) ;*KT11 VECTOR ADDRESS
(1) 000250 MMVEC= 250
(1)
(1) ;*KT11 STATUS REGISTER ADDRESSES
(1) 177572 SR0= 177572
(1) 177574 SR1= 177574
(1) 177576 SR2= 177576
(1) 172516 SR3= 172516
(1)
(1) ;*USER "I" PAGE DESCRIPTOR REGISTERS
(1) 177600 UIPDR0= 177600
(1) 177602 UIPDR1= 177602
(1) 177604 UIPDR2= 177604
(1) 177606 UIPDR3= 177606
(1) 177610 UIPDR4= 177610
(1) 177612 UIPDR5= 177612
(1) 177614 UIPDR6= 177614
(1) 177616 UIPDR7= 177616
(1)
(1) ;*USER "I" PAGE ADDRESS REGISTERS
(1) 177640 UIPAR0= 177640
(1) 177642 UIPAR1= 177642
(1) 177644 UIPAR2= 177644
(1) 177646 UIPAR3= 177646
(1) 177650 UIPAR4= 177650
(1) 177652 UIPAR5= 177652
(1) 177654 UIPAR6= 177654
(1) 177656 UIPAR7= 177656
(1)
(1) ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
```



```
(1)
(1)      172300      KIPDR0= 172300
(1)      172302      KIPDR1= 172302
(1)      172304      KIPDR2= 172304
(1)      172306      KIPDR3= 172306
(1)      172310      KIPDR4= 172310
(1)      172312      KIPDR5= 172312
(1)      172314      KIPDR6= 172314
(1)      172316      KIPDR7= 172316
(1)
(1)
(1)      ;*KERNEL 'I' PAGE ADDRESS REGISTERS
(1)
(1)      172340      KIPAR0= 172340
(1)      172342      KIPAR1= 172342
(1)      172344      KIPAR2= 172344
(1)      172346      KIPAR3= 172346
(1)      172350      KIPAR4= 172350
(1)      172352      KIPAR5= 172352
(1)      172354      KIPAR6= 172354
(1)      172356      KIPAR7= 172356
(1)
23          .EQUIV SP,KSP
24          .EQUIV SP,USP
25          .EQUIV BIT4,TBIT
26          .EQUIV BIT6,WBIT
27          001100      KERSTK= STACK
28          000700      USESTK= STACK-200
29
30          ;*ADDITIONAL DEFINITIONS
31          ;*
32
256
258          .SBTTL TRAP CATCHER
(1)
(1)          .=0
(1)          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
(1)          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
(1)          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
(1)          .=174
(1) 000174 000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
(1) 000176 000000      SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER
(1)          .SBTTL STARTING ADDRESS(ES)
(1) 000200 000137 020000      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
259          .SBTTL ACT11 HOOKS
(1)
(2)          ;:*****
(1)          ;:HOOKS REQUIRED BY ACT11
(1)          $SVPC=.          ;SAVE PC
(1)          .=46
(1) 000046 034020      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1)          .=52
(1) 000052 000000      .WORD 0          ;;2)SET LOC.52 TO ZERO
(1)          .=$SVPC          ;; RESTORE PC
260          .SBTTL APT PARAMETER BLOCK
(1)
(2)          ;:*****
```

```
(1) ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2) ;*****
(1)      000204      .SX=.      ;;SAVE CURRENT LOCATION
(1)      000024      =24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200    200      ;;FOR APT START UP
(1)      000044      =44      ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 000204    $APTHDR ;;POINT TO APT HEADER BLOCK
(1)      000204      =.SX     ;;RESET LOCATION COUNTER
(2) ;*****
(1) ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1) ;INTERFACE SPEC.
(1)
(1) 000204 $APTHD:
(1) 000204 000000 $HIRTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 000206 001226 $MBADR: .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 000210 000014 $STMT: .WORD 14     ;;RUN TIM OF LONGEST TEST
(1) 000212 000020 $PASTM: .WORD 20    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 000214 000005 $UNITM: .WORD 5     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 000216 000016 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```



```

261      .SBTTL COMMON TAGS
(1)
(2)      ::*****
(1)      ::THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
(1)      ::USED IN THE PROGRAM.
(1)
(1)      001100      001100      $CMTAG:      .=1100      ::START OF COMMON TAGS
(1)      001100      000000      $STSTM:      .WORD      0      ::CONTAINS THE TEST NUMBER
(1)      001102      000      $ERFLG:      .BYTE      0      ::CONTAINS ERROR FLAG
(1)      001103      000      $ICNT:      .WORD      0      ::CONTAINS SUBTEST ITERATION COUNT
(1)      001104      000000      $LPADR:      .WORD      0      ::CONTAINS SCOPE LOOP ADDRESS
(1)      001106      000000      $LPERR:      .WORD      0      ::CONTAINS SCOPE RETURN FOR ERRORS
(1)      001110      000000      $ERTTL:      .WORD      0      ::CONTAINS TOTAL ERRORS DETECTED
(1)      001112      000000      $ITEMB:      .BYTE      0      ::CONTAINS ITEM CONTROL BYTE
(1)      001114      000      $ERMAX:      .BYTE      1      ::CONTAINS MAX. ERRORS PER TEST
(1)      001115      001      $ERRPC:      .WORD      0      ::CONTAINS PC OF LAST ERROR INSTRUCTION
(1)      001116      000000      $GDADR:      .WORD      0      ::CONTAINS ADDRESS OF 'GOOD' DATA
(1)      001120      000000      $BDADR:      .WORD      0      ::CONTAINS ADDRESS OF 'BAD' DATA
(1)      001122      000000      $GDDAT:      .WORD      0      ::CONTAINS 'GOOD' DATA
(1)      001124      000000      $BDDAT:      .WORD      0      ::CONTAINS 'BAD' DATA
(1)      001126      000000      .WORD      0      ::RESERVED--NOT TO BE USED
(1)      001130      000000      .WORD      0
(1)      001132      000000      .WORD      0
(1)      001134      000      $AUTOB:      .BYTE      0      ::AUTOMATIC MODE INDICATOR
(1)      001135      000      $INTAG:      .BYTE      0      ::INTERRUPT MODE INDICATOR
(1)      001136      000000      .WORD      0
(1)      001140      177570      SWR:      .WORD      DSWR      ::ADDRESS OF SWITCH REGISTER
(1)      001142      177570      DISPLAY:      .WORD      DDISP      ::ADDRESS OF DISPLAY REGISTER
(1)      001144      177560      $TKS:      177560      ::TTY KBD STATUS
(1)      001146      177562      $TKB:      177562      ::TTY KBD BUFFER
(1)      001150      177564      $TPS:      177564      ::TTY PRINTER STATUS REG. ADDRESS
(1)      001152      177566      $TPB:      177566      ::TTY PRINTER BUFFER REG. ADDRESS
(1)      001154      000      $NULL:      .BYTE      0      ::CONTAINS NULL CHARACTER FOR FILLS
(1)      001155      002      $FILLS:      .BYTE      2      ::CONTAINS # OF FILLER CHARACTERS REQUIRED
(1)      001156      012      $FILLC:      .BYTE      12      ::INSERT FILL CHARS. AFTER A 'LINE FEED'
(1)      001157      000      $TPFLG:      .BYTE      0      ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
(1)      001160      000000      $REGAD:      .WORD      0      ::CONTAINS THE ADDRESS FROM
(1)      ::WHICH ($REGO) WAS OBTAINED
(3)      001162      000000      $REG0:      .WORD      0      ::CONTAINS (($REGAD)+0)
(3)      001164      000000      $REG1:      .WORD      0      ::CONTAINS (($REGAD)+2)
(3)      001166      000000      $REG2:      .WORD      0      ::CONTAINS (($REGAD)+4)
(3)      001170      000000      $REG3:      .WORD      0      ::CONTAINS (($REGAD)+6)
(3)      001172      000000      $REG4:      .WORD      0      ::CONTAINS (($REGAD)+10)
(3)      001174      000000      $REG5:      .WORD      0      ::CONTAINS (($REGAD)+12)
(3)      001176      000000      $TMP0:      .WORD      0      ::USER DEFINED
(3)      001200      000000      $TMP1:      .WORD      0      ::USER DEFINED
(3)      001202      000000      $TMP2:      .WORD      0      ::USER DEFINED
(3)      001204      000000      $TMP3:      .WORD      0      ::USER DEFINED
(3)      001206      000000      $TMP4:      .WORD      0      ::USER DEFINED
(3)      001210      000000      $TMP5:      .WORD      0      ::USER DEFINED
(1)      001212      000000      $TIMES:      0      ::MAX. NUMBER OF ITERATIONS
(1)      001214      000000      $ESCAPE:      0      ::ESCAPE ON ERROR ADDRESS
(1)      001216      177607      000377      $BELL:      .ASCII      <207><377><377>      ::CODE FOR BELL
(1)      001222      077      $QUES:      .ASCII      /?/      ::QUESTION MARK
(1)      001223      015      $CRLF:      .ASCII      <15>      ::CARRIAGE RETURN

```

```
(1) 001224 000012 $LF: .ASCIZ <12> ;:LINE FEED
(2) ;:*****
(2) .SBTTL APT MAILBOX-ETABLE
(2) ;:*****
(2) .EVEN
(2) 001226 $MAIL: ;:APT MAILBOX
(2) 001226 000000 $MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE
(2) 001230 000000 $FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER
(2) 001232 000000 $TESTN: .WORD ATESTN ;:TEST NUMBER
(2) 001234 000000 $PASS: .WORD APASS ;:PASS COUNT
(2) 001236 000000 $DEVCT: .WORD ADEVCT ;:DEVICE COUNT
(2) 001240 000000 $UNIT: .WORD AUNIT ;:I/O UNIT NUMBER
(2) 001242 000000 $MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS
(2) 001244 000000 $MSGLG: .WORD AMSGLG ;:MESSAGE LENGTH
(2) 001246 $ETABLE: ;:APT ENVIRONMENT TABLE
(2) 001246 000 $ENV: .BYTE AENV ;:ENVIRONMENT BYTE
(2) 001247 000 $ENVM: .BYTE AENVM ;:ENVIRONMENT MODE BITS
(2) 001250 000000 $SWREG: .WORD ASWREG ;:APT SWITCH REGISTER
(2) 001252 000000 $USWR: .WORD AUSWR ;:USER SWITCHES
(2) 001254 000000 $CPUOP: .WORD ACPUOP ;:CPU TYPE,OPTIONS
(2) ;* BITS 15-11=CPU TYPE
(2) ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2) ;* 11/70=06,PDQ=07,Q=10
(2) ;* BIT 10=REAL TIME CLOCK
(2) ;* BIT 9=FLOATING POINT PROCESSOR
(2) ;* BIT 8=MEMORY MANAGEMENT
(2) 001256 000 $MAMS1: .BYTE AMAMS1 ;:HIGH ADDRESS,M.S. BYTE
(2) 001257 000 $MTYP1: .BYTE AMTYP1 ;:MEM. TYPE,BLK#1
(2) ;* MEM.TYPE BYTE -- (HIGH BYTE)
(2) ;* 900 NSEC CORE=001
(2) ;* 300 NSEC BIPOLAR=002
(2) ;* 500 NSEC MOS=003
(2) 001260 000000 $MADR1: .WORD AMADR1 ;:HIGH ADDRESS,BLK#1
(2) ;* MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
(2) 001262 $ETEND:
(2) .MEXIT
(3) 001262 000000 TESTNO: .WORD 0 ;:HOLDS TEST NUMBER FOR TYPEOUTS
(3) 001264 000000 WASR6: .WORD 0 ;:USED TO STORE THE STACK POINTER AFTER A TRAP
(3) 001266 000000 TRAPPC: .WORD 0 ;:USED TO STORE THE PC OF A TRAP OR ABORT
(3) 001270 000000 TRAPPS: .WORD 0 ;:USED TO STORE THE PS OF A TRAP OR ABORT
(3) 001272 000000 WASSR0: .WORD 0 ;:USED TO STORE CONTENTS OF SR0
(3) 001274 000000 WASSR2: .WORD 0 ;:USED TO STORE CONTENTS OF SR2
(3) 001276 000000 TBITPS: .WORD 0 ;:SAVES THE PSW THAT MAY HAVE ITS T-BIT ON
(3) 001300 000000 ANDADR: .WORD 0 ;:HOLDS RESULT OF ADDRESSES BEING AND-ED
(3) 001302 000000 ORADR: .WORD 0 ;:HOLDS RESULT OF ADDRESSES BEING OR-ED
(3) 001304 000000 TONUM: .WORD 0 ;:HOLDS NUMBER OF TIME-OUTS
(3) 001306 000000 VIRT1: .WORD 0 ;:HOLDS VIRTUAL ADDRESS TO BE CONVERTED
(3) 001310 000000 VIRT2: .WORD 0 ;:
(3) 001312 000000 PBALO: .WORD 0 ;:HOLDS BITS <15:00> OF PHYSICAL ADDRESS
(3) 001314 000000 PBAHI: .WORD 0 ;:HOLDS BITS <17:16> OF PHYSICAL ADDRESS
```



```
(1) .SBTTL ERROR POINTER TABLE
(1)
(1) ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
(1) ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
(1) ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
(1) ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
(1) ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
(1)
(1) ;* EM ;:POINTS TO THE ERROR MESSAGE
(1) ;* DH ;:POINTS TO THE DATA HEADER
(1) ;* DT ;:POINTS TO THE DATA
(1) ;* DF ;:POINTS TO THE DATA FORMAT
(1)
(1) $ERRTB:
(1)
(1) ;*ITEM 1
262
263
264 001316 040724 EM1 ;UNEXPECTED CPU TRAP TO LOC. 004
265 001320 044015 DH1 ;OLD PC OLD PSW R6 WAS TESTNO ERRORPC
266 001322 047222 DT1 ;TRAPPC, TRAPPS, WASR6, TESTNO, $ERPPC, 0
267 001324 050032 DF1 ;0,0,0,0,0
268
269 ;*ITEM 2
270 001326 040764 EM2 ;UNEXPECTED MEM. MGMT. TRAP TO LOC. 250
271 001330 044065 DH2 ;OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC
272 001332 047236 DT2 ;TRAPPC, TRAPPS, WASR6, WASSR0, WASSR2, TESTNO, $ERRPC,
273 001334 050037 DF2 ;0,0,0,0,0,0,0
274
275 ;*ITEM 3
276 001336 041033 EM3 ;PRIORITY BITS SET WRONG IN PSW
277 001340 044155 DH3 ;WROTE READ TESTNO ERRORPC
278 001342 047256 DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
279 001344 050046 DF3 ;0,0,0,0
280
281 ;*ITEM 4
282 001346 041072 EM4 ;MODE BITS SET WRONG IN PSW
283 001350 044155 DH3 ;WROTE READ TESTNO ERRORPC
284 001352 047256 DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
285 001354 050046 DF3 ;0,0,0,0
286
287 ;*ITEM 5
288 001356 041125 EM5 ;DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW
289 001360 044155 DH3 ;WROTE READ TESTNO ERRORPC
290 001362 047256 DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
291 001364 050046 DF3 ;0,0,0,0
292
293 ;*ITEM 6
294 001366 041200 EM6 ;KERNEL R6 CHANGED BY WRITING USER R6
295 001370 044155 DH3 ;WROTE READ TESTNO ERRORPC
296 001372 047256 DT3 ;$REG0,$REG1,TESTNO,$ERRPC,0
297 001374 050046 DF3 ;0,0,0,0
298
299 ;*ITEM 7
300 001376 041245 EM7 ;A MEMORY MGMT. REG. TIMED OUT
301 001400 044215 DH7 ;ADDRESS TESTNO ERRORPC
302 001402 047270 DT7 ;$REG0,TESTNO,$ERRPC,0
```

303	001404	050052	DF7	:0,0,0
304				
305			:*ITEM 10	
306	001406	041303	EM10	:SUMMARY OF MEM. MGMT. REG. TIMEOUTS
307	001410	044245	DH10	:REGISTER-ADDRS NUM. OF
308				:AND-ED OR-ED TIMOUTS TESTNO ERRORPC
309	001412	047300	DT10	:ANDADR,ORADR,TONUM,TESTNO,\$ERRPC,0
310	001414	050055	DF10	:0,0,1,0,0
311				
312			:*ITEM 11	
313	001416	041347	EM11	:MEM. MGMT. REG. WOULD NOT CLEAR
314	001420	044345	DH11	:REGISTR READ READ-(BINARY)
315				:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
316	001422	047314	DT11	:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
317	001424	050062	DF11	:0,0,2,0,0
318				
319			:*ITEM 12	
320	001426	041407	EM12	:MEM. MGMT. REG. BITS NOT SET CORRECTLY
321	001430	044465	DH12	:REGISTR WROTE READ READ
322				:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
323	001432	047330	DT12	:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
324	001434	050067	DF12	:0,0,0,2,0,0
325				
326			:*ITEM 13	
327	001436	041456	EM13	:SR0 EFFECTED BY WRITE TO PSW
328	001440	044625	DH13	:READ TESTNO ERRORPC
329	001442	047346	DT13	:\$REG0,TESTNO,\$ERRPC,0
330	001444	050075	DF13	:0,0,0
331				
332			:*ITEM 14	
333	001446	041513	EM14	:SR1 DID NOT READ ALL ZEROS
334	001450	044625	DH13	:READ TESTNO ERRORPC
335	001452	047346	DT13	:\$REG0,TESTNO,\$ERRPC,0
336	001454	050075	DF13	:0,0,0
337				
338			:*ITEM 15	
339	001456	041546	EM15	:DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR
340	001460	044465	DH12	:REGISTER WROTE READ READ
341				:ADDRESS (OCTAL) (OCTAL) (BINARY) TESTNO ERRORPC
342	001462	047330	DT12	:\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0
343	001464	050067	DF12	:0,0,0,2,0,0
344				
345			:*ITEM 16	
346	001466	041622	EM16	:DUAL ADDRESSING BETWEEN PAR-PDR'S
347	001470	044655	DH16	:PAR-PDR PAR-PDR
348				:CLEARED EFFECTD EXPECTD RECEIVD TESTNO ERRORPC
349	001472	047356	DT16	:\$REG0,\$REG1,\$REG5,\$REG2,TESTNO,\$ERRPC,0
350	001474	050100	DF16	:0,0,0,0,0,0
351				
352			:*ITEM 17	
353	001476	041664	EM17	:PHYS. ADDR. FORMED READ WRONG
354	001500	044755	DH17	:PHYSICAL VIRTUAL
355				:ADDRESS ADDRESS KIPAR4 TESTNO ERRORPC
356	001502	047374	DT17	:PBALO,VIRT1,\$REG4,TESTNO,\$ERRPC,0
357	001504	050106	DF17	:3,0,0,0,0
358				

Line	Code	Address	Item	Register	Description
359			*ITEM 20		
360	001506	041722		EM20	:PHYS. ADDR. FORMED READ WRONG IN RELOCATE MODE
361	001510	045045		DH20	:PHYSICL PAR 4 PAR 5
362					:ADDRESS VBA VBA PAR 4 PAR 5 PSW TESTNO
363	001512	047410		DT20	:PBALO,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
364	001514	050113		DF20	:3,0,0,0,0,0,0,0
365					
366			*ITEM 21		
367	001516	041774		EM21	:W-BIT DID NOT GET SET IN PDR
368	001520	045173		DH21	:PDR VIRTUAL
369					:TESTED ADDRESS TESTNO ERRORPC
370	001522	047432		DT21	:\$REG5,\$REG3,TESTNO,\$ERRPC,0
371	001524	050123		DF21	:0,0,0,0
372					
373			*ITEM 22		
374	001526	042031		EM22	:W-BIT SET IN MORE THAN ONE PDR
375	001530	045253		DH22	:PDR IN PDR VIRTUAL
376					:ERROR TESTED ADDRESS TESTNO ERRORPC
377	001532	047444		DT22	:\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
378	001534	050127		DF22	:0,0,0,0,0
379					
380			*ITEM 23		
381	001536	042070		EM23	:W-BIT NOT CLEARED BY WRITING TO PDR
382	001540	045352		DH23	:PDR TESTNO ERRORPC
383	001542	047460		DT23	:\$REG5,TESTNO,\$ERRPC,0
384	001544	050134		DF23	:0,0,0
385					
386			*ITEM 24		
387	001546	042134		EM24	:WRITING SRO SET W-BIT IN KIPDR7
388	001550	045402		DH24	:PDR WAS EXPECTD TESTNO ERRORPC
389	001552	047470		DT24	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
390	001554	050137		DF24	:0,0,0,0
391					
392			*ITEM 25		
393	001556	042174		EM25	:W-BIT GOT SET DURING TIMEOUT ABORT
394	001560	045402		DH24	:PDR WAS EXPECTD TESTNO ERRORPC
395	001562	047470		DT24	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
396	001564	050137		DF24	:0,0,0,0
397					
398			*ITEM 26		
399	001566	042237		EM26	:MEMORY MGMT. ACCESS ABORT DID NOT OCCUR
400	001570	045442		DH26	:PDR 4 PSW TESTNO ERRORPC
401	001572	047502		DT26	:\$REG2,\$TMP0,TESTNO,\$ERRPC,0
402	001574	050137		DF24	:0,0,0,0
403					
404			*ITEM 27		
405	001576	042307		EM27	:ACCESS ERROR DID NOT ABORT INSTRUCTION
406	001600	045442		DH26	:PDR 4 PSW TESTNO ERRORPC
407	001602	047502		DT26	:\$REG2,\$TMP0,TESTNO,\$ERRPC,0
408	001604	050137		DF24	:0,0,0,0
409					
410			*ITEM 30		
411	001606	042356		EM30	:SRO DID NOT REPORT ACCESS ERROR CORRECTLY
412	001610	045502		DH30	:SRO WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
413	001612	047514		DT30	:WASSRO,\$REG3,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
414	001614	050143		DF30	:0,0,0,0,0,0

415					
416			;*ITEM 31		
417	001616	042430	EM31		:SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
418	001620	045562	DH31		:SR2 WAS EXPECTD PDR 4 PSW TESTNO ERRORPC
419	001622	047532	DT31		:WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
420	001624	050143	DF30		:0,0,0,0,0,0
421					
422			;*ITEM 32		
423	001626	042475	EM32		:PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE
424	001630	045642	DH32		:V.B.A. KIPDR4 SR0 WAS SR2 WAS TESTNO ERRORPC
425	001632	047550	DT32		:\$REG0,\$REG4,WASSR0,WASSR2,TESTNO,\$ERRPC,0
426	001634	050143	DF30		:0,0,0,0,0,0
427					
428			;*ITEM 33		
429	001636	042556	EM33		:PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE
430	001640	045722	DH33		:V.B.A. KIPDR4 TESTNO ERRORPC
431	001642	047566	DT33		:\$REG0,\$REG4,TESTNO,\$ERRPC,0
432	001644	050137	DF24		:0,0,0,0
433					
434			;*ITEM 34		
435	001646	042641	EM34		:SR0 DID NOT REPORT PAGE LGTH. ABORT CORRECTLY
436	001650	045762	DH34		:V.B.A. KIPDR4 SR0 WAS EXPECTD TESTNO ERRORPC
437	001652	047600	DT34		:\$REG0,\$REG4,WASSR0,\$REG2,TESTNO,\$ERRPC,0
438	001654	050143	DF30		:0,0,0,0,0,0
439			;*ITEM 35		
440	001656	042430	EM31		:SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
441	001660	046042	DH35		:V.B.A. KIPDR4 SR2 WAS EXPECTD TESTNO ERRORPC
442	001662	047616	DT35		:\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
443	001664	050143	DF30		:0,0,0,0,0,0
444					
445			;*ITEM 36		
446	001666	042430	EM31		:SR2 DID NOT LOCKUP CORRECT VIRTUAL ADDR.
447	001670	046122	DH36		:SR2 WAS EXPECTD TESTNO ERRORPC
448	001672	047634	DT36		:WASSR2,\$REG1,TESTNO,\$ERRPC,0
449	001674	050137	DF24		:0,0,0,0
450					
451			;*ITEM 37		
452	001676	042717	EM37		:SR0 OR SR2 CHANGED BY A SECOND ABORT
453	001700	046162	DH37		:FIRST ABORT SECOND ABORT
454					:SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO ERRORPC
455	001702	047646	DT37		:\$TMP0,\$TMP2,WASSR0,WASSR2,TESTNO,\$ERRPC,0
456	001704	050143	DF30		:0,0,0,0,0,0
457					
458			;*ITEM 40		
459	001706	042764	EM40		:SR0 OR SR2 WAS NOT 'RESET' BY A RESET
460	001710	046277	DH40		:SR0 WAS SR2 WAS TESTNO ERRORPC
461	001712	047664	DT40		:WASSR0,WASSR2,TESTNO,\$ERRPC,0
462	001714	050137	DF24		:0,0,0,0
463					
464			;*ITEM 41		
465	001716	043033	EM41		:SR2 NOT TRACKING CORRECTLY
466	001720	046122	DH36		:SR2 WAS EXPECTD TESTNO ERROPC
467	001722	047634	DT36		:WASSR2,\$REG1,TESTNO,\$ERRPC,0
468	001724	050137	DF24		:0,0,0,0
469					
470			;*ITEM 42		

471	001726	043066	EM42	:DID NOT TRAP THRU KERNEL SPACE
472	001730	046337	DH42	:PSW WAS R6 WAS TESTNO ERRORPC
473	001732	047676	DT42	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
474	001734	050137	DF24	:0,0,0,0
475				
476			:*ITEM 43	
477	001736	043125	EM43	:KT ERROR NOT SERVICED ON TIMEOUT ERROR
478	001740	045352	DH23	:PDR TESTNO ERRORPC
479	001742	047460	DT23	:\$REG5,TESTNO,\$ERRPC,0
480	001744	050134	DF23	:0,0,0
481				
482			:*ITEM 44	
483				
484	001746	043174	EM44	:SRO OR SR2 CHANGED BY TIMEOUT ERROR
485	001750	046377	DH44	:EXPECTED RECEIVED
486				:SRO SR2 SRO WAS SR2 WAS TESTNO ERRORPC
487	001752	047710	DT44	:\$REG0,\$REG1,WASSRO,WASSR2,TESTNO,\$ERRPC,0
488	001754	050143	DF30	:0,0,0,0,0,0
489				
490			:*ITEM 45	
491	001756	043240	EM45	:ERROR DURING 'DOUBLE ERROR' (KT & ODD ADDR.)
492	001760	046511	DH45	:EXPECTED:
493				:PSW PC SRO SR2
494				:170017 (3\$+4) 020147 (3\$)
495				:RECEIVED
496				:PSW PC SRO SR2 TESTNO ERRORPC
497	001762	047726	DT45	:\$REG1,\$REG3,WASSRO,WASSR2,TESTNO,\$ERRPC,0
498	001764	050143	DF30	:0,0,0,0,0,0
499				
500			:*ITEM 46	
501	001766	043313	EM46	:MFPI INSTRUCTION PUSHED WRONG DATA
502	001770	046706	DH46	:DATA DATA
503				:EXPECTD RECEIVD TESTNO ERRORPC
504	001772	047744	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
505	001774	050151	DF46	:0,0,0,0
506				
507			:*ITEM 47	
508	001776	043356	EM47	:MTPI INSTRUCTION LOADED WRONG DATA
509	002000	046706	DH46	:DATA DATA
510				:EXPECTD RECEIVD TESTNO ERRORPC
511	002002	047744	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
512	002004	050151	DF46	:0,0,0,0
513				
514			:*ITEM 50	
515	002006	043421	EM50	:STACK NOT PUSHED BY MFPI-MTPI
516	002010	046763	DH50	:TESTNO ERRORPC
517	002012	047756	DT50	:TESTNO,\$ERRPC,0
518	002014	050155	DF50	:0,0
519				
520			:*ITEM 51	
521	002016	043457	EM51	:KERNEL PAGE ACCESSED INSTEAD OF USER: MFPI-MTPI
522	002020	047003	DH51	:SRO WAS SR2 WAS TESTNO ERRORPC
523	002022	047764	DT51	:WASSRO,WASSR2,TESTNO,\$ERRPC,0
524	002024	050157	DF51	:0,0,0,0
525				
526			:*ITEM 52	

527	002026	043535	EM52	:WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE
528	002030	047043	DH52	:PHYSICL PAR 4
529				:ADDRESS V.B.A. PAR 4 SRO WAS SR2 WAS PSW TESTNO
530	002032	047776	DT52	:PBALO,VIRT1,\$REG4,WASSRO,WASSR2,\$TMPO,TESTNO,\$ERRPC,0
531	002034	050163	DF52	:3,0,0,0,0,0,0,0
532				
533	002036	043613	EM53	:MFPD INSTRUCTION PUSHED WRONG DATA
534	002040	046706	DH46	:DATA DATA
535				:EXPECTD RECEIVD TESTNO ERRORPC
536	002042	047744	DT46	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
537	002044	050151	DF46	:0,0,0,0
538				
539				
540	002046	043656	EM54	:STACK NOT PUSHED BY MFPD-MTPD
541	002050	046763	DH50	:TESTNO ERRORPC
542	002052	047756	DT50	:TESTNO,\$ERRPC,0
543	002054	050155	DF50	:0,0
544				
545				
546	002056	043714	EM55	:PAR OR PDR WAS CHANGED BY A RESET
547	002060	044345	DH11	:REGISTR READ READ-(BINARY)
548				:ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC
549	002062	047314	DT11	:\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0
550	002064	050062	DF11	:0,0,2,0,0
551				
552				
553	002066	043752	EM56	:PSW CHANGED BY AN RTI IN USER MODE
554	002070	047161	DH56	:PSW WAS EXPECTD TESTNO ERRORPC
555	002072	050020	DT56	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
556	002074	050173	DF56	:0,0,0,0
557				
558				

560
561
562
563
564
565
566
567
568
569
570
571 002076 005227
572 002100 177777
573 002102 001403
574 002104 005237 001226
575 002110 000000
576
577
578
579
580 002112 012637 001266
581 002116 012637 001270
582 002122 010637 001264
583 002126 104001
584 002130 012737 177777 002100
585 002136 013746 001270
586 002142 013746 001266
587 002146 000006
588
589
590
591
592
593
594
595
596
597
598
599 002150 005227
600 002152 177777
601 002154 001403
602 002156 005237 001226
603 002162 000000
604
605
606
607
608 002164 012637 001266
609 002170 012637 001270
610 002174 010637 001264
611 002200 013737 177572 001272
612 002206 013737 177576 001274
613 002214 042737 160000 177572
614 002222 104002
615 002224 012737 177777 002152

```
.SBTTL ***** TRAP HANDLING ROUTINES *****  
.SBTTL CPU TRAP HANDLER ROUTINE  
:*****  
:*  
:* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS THRU  
:* 'ERRVEC' (LOC. 004). IF THIS SUBROUTINE IS ENTERED BY A  
:* SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A HALT IS  
:* EXECUTED.  
:*****  
TIMERR: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME THRU  
TIMFLG: .WORD -1 ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG  
BEQ 1$ ;BRANCH IF FIRST TIME IN  
INC $MSGTYPE ;TELL APT THERE WAS AN ERROR  
HALT ;STOP! - I'VE ENTERED THIS ROUTINE  
;A SECOND TIME BEFORE I FINISHED  
;REPORTING THE FIRST ERROR. THE  
;SECOND ENTRY ADDRESS SHOULD BE ON  
;THE KERNEL STACK.  
1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT  
MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT  
MOV KSP,WASR6 ;SAVE STACK POINTER VALUE  
ERROR 1 ;UNEXPECTED TRAP OR ABORT TO LOC. 4  
MOV #-1,TIMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME  
MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK  
MOV TRAPPC,-(KSP)  
RTT ;RETURN FROM INTERRUPT OR ABORT
```

```
.SBTTL MEMORY MANAGEMENT TRAP HANDLER ROUTINE  
:*****  
:*  
:* THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED MEMORY MANAGEMENT  
:* TRAPS AND ABORTS THRU 'MMVEC' (LOC. 250). IF THIS SUBROUTINE IS  
:* ENTERED BY A SECOND TRAP BEFORE THE FIRST HAS BEEN SERVICED, A  
:* HALT IS EXECUTED.  
:*****  
MGMERR: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME THRU  
MGMFLG: .WORD -1 ;NEGATIVE ONE FOR 'HAVE ENTERED' FLAG  
BEQ 1$ ;BRANCH IF FIRST TIME IN  
INC $MSGTYPE ;TELL APT THERE WAS AN ERROR  
HALT ;STOP! - I'VE ENTERED THIS ROUTINE  
;A SECOND TIME BEFORE I FINISHED  
;REPORTING THE FIRST ERROR. THE  
;SECOND ENTRY ADDRESS SHOULD BE ON  
;THE KERNEL STACK.  
1$: MOV (KSP)+,TRAPPC ;SAVE PC+2 AT TIME OF ABORT  
MOV (KSP)+,TRAPPS ;SAVE PS AT TIME OF ABORT  
MOV KSP,WASR6 ;SAVE STACK POINTER VALUE  
MOV SRO,WASSRO ;SAVE CONTENTS OF KT STATUS REG. 0  
MOV SR2,WASSR2 ;SAVE CONTENTS OF KT STATUS REG. 2  
BIC #160000,SRO ;CLEAR ERROR BITS IN STATUS REG 0  
ERROR 2 ;UNEXPECTED TRAP OR ABORT TO LOC. 250  
MOV #-1,MGMFLG ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
```

CJKDACO KTF11-AA MMU DIAG
CJKDAC.P11 12-MAR-80 07:56

MACY11 30A(1052) 12-MAR-80 08:00 ^{C 3} PAGE 1-14
MEMORY MANAGEMENT TRAP HANDLER ROUTINE

SEQ 0028

616 002232 013746 001270
617 002236 013746 001266
618 002242 000006
619

MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
MOV TRAPPC,-(KSP)
RTT ;RETURN FROM INTERRUPT OR ABORT.


```
621 .SBTTL
622 .SBTTL ***** STARTING POINT OF TEST *****
623 .SBTTL ***** STARTING ADDRESS OF 200 *****
624 .=20000
625
626 020000
(1)
(1) 020000 012706 001100
(1) 020004 005026
(1) 020006 022706 001140
(1) 020012 001374
(1) 020014 012706 001100
(1)
(1) 020020 012737 034136 000020
(1) 020026 012737 000340 000022
(1) 020034 012737 034416 000030
(1) 020042 012737 000340 000032
(1) 020050 012737 040410 000034
(1) 020056 012737 000340 000036
(1) 020064 012737 040476 000024
(1) 020072 012737 000340 000026
(1) 020100 013737 033650 033642
(1) 020106 005037 001212
(1) 020112 005037 001214
(1) 020116 112737 000001 001115
(2)
(2) 020124 012737 034122 000014
(2) 020132 012737 000340 000016
(2) 020140 012737 000002 034122
(2) 020146 012737 020174 000010
(2) 020154 005046
(2) 020156 012746 020164
(2) 020162 000006
(2) 020164 012737 000006 034122 64$:
(2) 020172 000402 BR 66$
(2) 020174 062706 000010 65$:
(2) 020200 012737 000012 000010 66$:
(2) 020206 005037 034130
(1) 020212 012737 020212 001106
(1) 020220 012737 020220 001110
(2)
(2) 020226 013746 000004
(2) 020232 012737 020266 000004
(2) 020240 012737 177570 001140
(2) 020246 012737 177570 001142
(2) 020254 022777 177777 160656
(2) 020262 001012
(2)
(2) 020264 000403
(2) 020266 012716 020274 67$:
(2) 020272 000002
(2) 020274 012737 000176 001140 68$:
(2) 020302 012737 000174 001142
```

```
.SBTTL
.SBTTL ***** STARTING POINT OF TEST *****
.SBTTL ***** STARTING ADDRESS OF 200 *****
.=20000

START:
.SBTTL INITIALIZE THE COMMON TAGS
::CLEAR THE COMMON TAGS ($CMTAG) AREA
MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
CLR (R6)+ ;;CLEAR MEMORY LOCATION
CMP #SWR,R6 ;;DONE?
BNE -6 ;;LOOP BACK IF NO
MOV #STACK,SP ;;SETUP THE STACK POINTER
::INITIALIZE A FEW VECTORS
MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
MOV #340,@IOTVEC+2 ;;LEVEL 7
MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
MOV #340,@EMTVEC+2 ;;LEVEL 7
MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
MOV #340,@TRAPVEC+2;LEVEL 7
MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
MOV #340,@PWRVEC+2 ;;LEVEL 7
MOV $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
MOVB #1,$ERMAX ;;ALLOW ONE ERROR PER TEST
::INITIALIZE THE 'T-BIT' TRAP VECTOR. THEN LOAD LOCATION '$RTRN', IN
::THE 'END-OF-PASS' ($EOP) ROUTINE, WITH A 'RTI' OR 'RTT'.
MOV #RTRN,@TBITVEC ;;SET 'T' BIT VECTOR TO $RTRN
MOV #340,@TBITVEC+2 ;;LEVEL 7
MOV #RTI,$RTRN ;;SET $RTRN TO A RTI
MOV #65$,@RESVEC ;;TRY TO DO A RTT
CLR -(SP) ;;DUMMY PS
MOV #64$,-(SP) ;;AND PC
RTT ;;TRY THE RTT
64$: MOV #RTT,$RTRN ;;RTT IS LEGAL--SET $RTRN TO A RTT
BR 66$
65$: ADD #10,SP ;;RTT ILLEGAL--CLEAN OFF THE STACK
66$: MOV #RESVEC+2,@RESVEC ;;RESTORE TRAP CATCHER
CLR $TBIT ;;CLEAR 'T' BIT SWITCH
MOV #,$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV #,$LPERR ;;SETUP THE ERROR LOOP ADDRESS
::SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
::EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
MOV #67$,@ERRVEC ;;SET UP ERROR VECTOR
MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
BNE 69$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
BR 68$ ;;BRANCH IF NO TIMEOUT
67$: MOV #68$,(SP) ;;SET UP FOR TRAP RETURN
68$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY
```

```
(2) 020310 012637 000004 69$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
(1)
(2) 020314 005037 001234 CLR $PASS ;;CLEAR PASS COUNT
(2) 020320 132737 000200 001247 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
(2) 020326 001403 BEQ 70$ ;;YES,USE NON-APT SWITCH
(2) 020330 012737 001250 001140 MOV #$$SWREG,SWR ;;NO,USE APT SWITCH REGISTER
(2) 020336 70$:
627 ;; INITIALIZE THE ERROR COUNTER FOR EOP REPORT($ERTTL).
628 020336 005037 001112 CLR $ERTTL ;CLEAR ERROR COUNTER
629 .SBTTL TYPE PROGRAM NAME
(1) ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
(1) 020342 005227 177777 INC #-1 ;;FIRST TIME?
(1) 020346 001046 BNE 71$ ;;BRANCH IF NO
(1) 020350 022737 034020 000042 CMP #$$ENDAD,@#42 ;;ACT-11?
(1) 020356 001442 BEQ 71$ ;;BRANCH IF YES
(1) 020360 104401 020426 TYPE ,72$ ;;TYPE ASCIZ STRING
(2) .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
(2) 020364 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
(2) 020370 001012 BNE 73$ ;;BRANCH IF YES
(2) 020372 123727 001246 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
(2) 020400 001406 BEQ 73$ ;;BRANCH IF YES
(2) 020402 023727 001140 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
(2) 020410 001005 BNE 74$ ;;BRANCH IF NO
(2) 020412 104407 GTSWR ;;GET SOFT-SWR SETTINGS
(2) 020414 000403 BR 74$
(2) 020416 112737 000001 001134 73$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
(2) 020424 74$:
(1) 020424 000417 BR 71$ ;;GET OVER THE ASCIZ
(1) ;;72$: .ASCIZ <CRLF>#CJKDACO KTF11-AA MMU DIAG.#<CRLF>
(1) 020464 71$:
630
631 020464 RESTR:
632 020464 012706 001100 LOOP: MOV #STACK,KSP ;INITIALIZE THE STACK POINTER
633 020470 012737 002076 000004 MOV #TIMERR,ERRVEC ;LOAD CPU SERVICE ROUTINE INTO TRAP VECTOR
634 020476 012737 000340 000006 MOV #340,ERRVEC+2 ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
635 020504 012737 002150 000250 MOV #MGMERR,MMVEC ;LOAD MEMORY MANAGENT ROUTINE INTO VECTOR
636 020512 012737 000340 000252 MOV #340,MMVEC+2 ;SET NEW PS TO PRIORITY LEVEL 7-KERNEL
637 020520 012700 177777 MOV #-1,R0 ;PUT -1 INTO R0 TO INITIALIZE FLAGS
638 020524 010037 002100 MOV R0,TIMFLG ;INITIALIZE CPU ERROR FLAG
639 020530 010037 002152 MOV R0,MGMFLG ;INITIALIZE MEMORY MANAGEMENT ERROR FLAG
640 020534 012737 000340 001276 MOV #340,TBITPS ;INITIALIZE LOG THAT HOLDS T-BIT PSW
641 020542 005037 177572 CLR SRO ;BE SURE MEM. MGMT IS OFF TO START WITH
642
```


644
645
652
(3)
(4)
(4)
(4)
(4)
(3)
(2) 020546 000004
653 020550 012737 020560 001110
654 020556 005000
655 020560 005001
656 020562 106400
657 020564 106701
658 020566 042701 177437
659 020572 020001
660 020574 001401
661 020576 104003
662
663
664
665 020600 062700 000040
666 020604 022700 000400
667 020610 001363
668 020612 012737 020550 001110
669
674
(3)
(4)
(4)
(3)
(2) 020620 000004
675 020622 012737 020632 001110
676 020630 005000
677 020632 005037 177776
678 020636 050037 177776
679 020642 013701 177776
680 020646 042701 007777
681 020652 020001
682 020654 001403
683 020656 005037 177776
684 020662 104004
685
686
687
688 020664 062700 010000
689 020670 001360
690 020672 012737 020622 001110
691 020700 005037 177776
692
699
(3)
(4)
(4)
(4)

```
*****  
*TEST 1 PSW PRIORITY BIT TEST  
*  
* THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <7:5> 'PRIORITY BITS'  
* TO SEE THAT SOME OF THE BASIC 'DATA PATH' LOGIC IS WORKING.  
*  
*****  
TST1: SCOPE  
1$: MOV #2,$LPERR ;SET LOOP ON ERROR POINTER TO 2$  
CLR R0 ;INITIALIZE R0 WITH PRIORITY=0 DATA  
2$: CLR R1 ;PREPARE R1 TO ACCEPT DATA READ  
MTPS R0 ;WRITE PRIORITY BITS IN THE PSW  
MFPS R1 ;READ BACK THE LOW BYTE OF PSW  
BIC #177437,R1 ;MASK OFF EVERYTHING EXCEPT PRIORITY BITS  
CMP R0,R1 ;WAS CORRECT PRIORITY SET IN THE PSW?  
BEQ 3$ ;BRANCH IF YES  
ERROR 3 ;PRIORITY BITS SET WRONG IN PSW  
;FOR TIGHTER SCOPE LOOP  
;REPLACE ERROR CALL WITH  
;'BR 2$' = 000770  
3$: ADD #40,R0 ;CHANGE DATA TO NEXT PRIORITY  
CMP #400,R0 ;HAVE PRIORITIES 0-7 ALL BEEN CHECKED?  
BNE 2$ ;BRANCH IF NO  
MOV #1,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$  
*****  
*TEST 2 PSW MODE BIT TEST  
* THIS TEST READS AND WRITES THE PROCESSOR STATUS WORD <15:12> 'MODE BITS'  
*  
*****  
TST2: SCOPE  
1$: MOV #2,$LPERR ;SET LOOP ON ERROR POINTER TO 2$  
CLR R0 ;INITIALIZE R0 WITH MODE BITS = 0000  
2$: CLR PSW ;INITIALIZE PSW  
BIS R0,PSW ;BIT SET THE PSW MODE BITS WITH R0  
MOV PSW,R1 ;READ BACK THE CONTENTS OF THE PSW  
BIC #007777,R1 ;MASK OFF EVERYTHING EXCEPT THE MODE BITS  
CMP R0,R1 ;WERE THE MODE BITS SET CORRECTLY?  
BEQ 3$ ;BRANCH IF YES  
CLR PSW ;CLEAR PSW FOR ERROR REPORT  
ERROR 4 ;MODE BITS SET WRONG IN PSW  
;FOR TIGHTER SCOPE LOOP  
;REPLACE ERROR CALL WITH  
;'BR 2$' = 000763  
3$: ADD #10000,R0 ;CHANGE MODE BIT DATA  
BNE 2$ ;BRANCH IF STILL MORE COMBINATIONS  
MOV #1,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$  
CLR PSW ;RESET PSW BEFORE LEAVING  
*****  
*TEST 3 BYTE ADDRESSING TEST FOR PSW  
*  
* THIS TEST WRITES THE HIGH AND LOW BYTES OF THE PROCESSOR STATUS WORD  
* AND READS THEM BACK TO BE SURE THEY CAN BE WRITTEN INDEPENDENTLY.  
*
```

```
(4)
(3)
(2) 020704 000004
700 020706 012737 020714 001110
701 020714 005037 177776
702 020720 012700 000360
703 020724 110037 177777
704 020730 013701 177776
705 020734 042701 007437
706 020740 000300
707 020742 020001
708 020744 001403
709 020746 005037 177776
710 020752 104005
711
712
713
714 020754 012737 020762 001110
715 020762 005037 177776
716 020766 012700 000340
717 020772 110037 177776
718 020776 013701 177776
719 021002 042701 007437
720 021006 020001
721 021010 001403
722 021012 005037 177776
723 021016 104005
724
725
726
727 021020 012737 020706 001110
728
737
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 021026 000004
738 021030 005037 177776
739 021034 012706 001100
740 021040 012737 140000 177776
741 021046 012706 000700
742 021052 005037 177776
743 021056 022706 001100
744 021062 001404
745 021064 012700 001100
746 021070 010601
747 021072 104006
748
749
750
751
```

```

:*****
:TST3: SCOPE
1$: MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
2$: CLR PSW ;CLEAR THE PSW
MOV #360,R0 ;PUT THE HIGH BYTE DATA INTO R0
MOVB R0,PSW+1 ;WRITE THE HIGH BYTE OF THE PSW
MOV PSW,R1 ;READ BACK THE ENTIRE PSW
BIC #007437,R1 ;MASK OFF THE T & CC BITS
SWAB R0 ;GET DATA WRITTEN IN HIGH BYTE OF R0
CMP R0,R1 ;WAS THE PSW WRITTEN TO CORRECTLY
BEQ 3$ ;BRANCH IF YES
CLR PSW ;CLEAR PSW FOR ERROR REPORT
ERROR 5 ;LOW BYTE EFFECTED BY WRITE TO HIGH BYTE OF PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000760
3$: MOV #4$,$LPERR ;SET LOOP ON ERROR POINTER TO 4$
4$: CLR PSW ;CLEAR THE PSW
MOV #340,R0 ;PUT THE LOW BYTE DATA INTO R0
MOVB R0,PSW ;WRITE THE LOW BYTE OF THE PSW
MOV PSW,R1 ;READ BACK THE ENTIRE PSW
BIC #007437,R1 ;MASK OFF THE T&CC BITS
CMP R0,R1 ;WAS PSW WRITTEN TO CORRECTLY
BEQ 5$ ;BRANCH IF YES
CLR PSW ;CLEAR PSW FOR ERROR REPORT
ERROR 5 ;HIGH BYTE EFFECTED BY WRITE TO LOW BYTE OF PSW
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 2$' = 000736
5$: MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
:*****
:TEST 4 TEST AND SETUP OF STACK POINTERS
:
: THIS TEST SETS THE USER AND KERNEL STACK POINTERS FOR THE
: REST OF THE PROGRAM AND MAKES SURE THEY ARE INDEPENDENT OF
: EACH OTHER. KERNEL R6 IS SET TO 1100, USER R6 IS SET TO 700, THEN
: KERNEL R6 IS READ TO BE SURE ITS STILL 1100.
:*****
:TST4: SCOPE
CLR PSW ;GO TO KERNEL MODE
MOV #KERSTK,KSP ;SET KERNEL STACK POINTER TO 1100
MOV #140000,PSW ;GO TO USER MODE
MOV #USESTK,USP ;SET USER STACK POINTER TO 700
CLR PSW ;BACK TO KERNEL MODE
CMP #KERSTK,KSP ;IS KERNEL R6 STILL 1100?
BEQ TST5 ;:BRANCH IF KERNEL R6 IS OKAY
MOV #KERSTK,R0 ;SAVE DATA WRITTEN FOR ERROR REPORT
MOV KSP,R1 ;SAVE DATA READ AFTER USER R6 WAS WRITTEN
ERROR 6 ;KERNEL R6 CHANGED BY WRITING USER R6
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
:000756
```


752
753
754
755
756
757
758
759
760
761
762
763
764
774
775
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2)
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

```
*****  
*  
* THE NEXT FIVE (5) TESTS WILL TRY TO ADDRESS ALL OF THE  
* MEMORY MANAGEMENT REGISTERS (SR0,SR1,SR2,SR3,KERNEL & USER PAR/PDR'S).  
* EVERY TIME A REGISTER TIMES OUT ITS ADDRESS WILL BE REPORTED.  
* AT THE END OF EACH TEST A SUMMARY OF THE ADDRESSES THAT TIMED  
* OUT DURING THAT TEST IS GIVEN. THE RESULTS OF 'AND-ING' AND 'OR-ING'  
* THEIR ADDRESSES IS GIVEN TO SHOW WHICH ADDRESS LINES MAY BE  
* STUCK AT 0 OR 1. THE PAR/PDR ADDRESS AND KT MUX'S ARE THE  
* THINGS BEING CHECKED.  
*  
*****
```

```
*****  
*TEST 5 SR0,SR1,SR2,SR3 TIMEOUT TEST  
*  
*****
```

```
* THIS TEST ADDRESSES THE MEMORY MANAGEMENT STATUS REGISTERS  
* 0,1,2, AND 3. STATUS REG. 1 IS NOT USED BUT SHOULD STILL  
* RESPOND TO ITS UNIBUS ADDRESS. DATA WILL BE WRITTEN OR READ  
* FROM THESE REGISTERS IN LATER TESTS, THIS TEST JUST CHECK  
* FOR A RESPONSE.  
*  
*****
```

```
TST5: SCOPE  
*****
```

```
1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$  
MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$  
MOV #SR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.  
MOV #3, R1 ;LOAD R1 WITH THE LOOP COUNT  
MOV #-1, ANDADR ;INITIALIZE 'AND' OF ADDRS. LOC.  
CLR ORADR ;INITIALIZE 'OR' OF ADDRS. LOC.  
CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER  
2$: TST (R0) ;TRY ADDRESSING A STATUS REGISTER  
;IF IT TIMES OUT GO TO 5$  
3$: ADD #2, R0 ;PUT NEXT ADDRESS IN R0  
SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED  
TST @#172516 ;CHECK SR3 FOR RESPONSE  
;IF IT TIMES OUT GO TO 5$  
MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$  
TST TONUM ;DID ANY OF THE STATUS REG.S TIMEOUT?  
BEQ 4$ ;BRANCH IF NO  
ERROR 10 ;SUMMARY OF STATUS REG. TIMEOUTS  
4$: MOV #TIMERR, @#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS  
BR TST6 ;GO TO NEXT TEST  
5$: ADD #4, KSP ;CLEAN UP THE STACK  
ERROR 7 ;ONE OF THE STATUS REGS. TIMED OUT  
;FOR TIGHTER SCOPE LOOP  
;REPLACE ERROR CALL WITH  
; 'BR 2$' = 000756  
MOV R0, R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2  
BIS R2, ORADR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT  
COM R2 ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT  
BIC R2, ANDADR  
INC TONUM ;INCREMENT THE TIMEOUT COUNTER
```

807 021230 000744 BR 3\$;BRANCH BACK TO TEST THE NEXT ADDR.

808
812 :*****
(3) :*TEST 6 KERNEL PAR'S TIMEOUT TEST
(5) :*
(5) :* THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE ADDRESS
(5) :* REGISTERS (KIPAR0-KIPAR7) AND CHECKS THAT SOMETHING
(5) :* RESPONDS TO THEIR ADDRESSES.
(5) :*
(3) :*****

(2) 021232 000004 TST6: SCOPE

813
(1) 021234 012737 021276 001110 1\$: MOV #2,\$,LPERR ;SET LOOP ON ERROR POINTER TO 2\$
(1) 021242 012737 021334 000004 MOV #5\$,@#4 ;SET TIMEOUT VECTOR TO 5\$
(1) 021250 012700 72340 MOV #KIPAR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
(1) 021254 012701 C J0010 MOV #10,R1 ;LOAD R1 WITH LOOP COUNT (8)
(1) 021260 012737 177777 001300 MOV #-1,ANDADR ;INITIALIZE 'AND' OF ADDR. LOC
(1) 021266 005037 001302 CLR ORADR ;INITIALIZE 'OR' OF ADDR. LOC.
(1) 021272 005037 001304 CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER
(1) 021276 005710 2\$: TST (R0) ;TRY ADDRESSING A KIPAR
(1) 3\$: ADD #2,R0 ;IF IT TIMES OUT, WILL GO TO 5\$
(1) 021304 077104 SOB R1,2\$;PUT NEXT KIPAR ADDRESS IN R0
(1) 021306 012737 021234 001110 MOV #1,\$,LPERR ;LOOP BACK TO 2\$ UNTIL ALL TESTED
(1) 021314 005737 001304 TST TONUM ;RESET LOOP ON ERROR POINTER TO 1\$
(1) 021320 001401 BEQ 4\$;DID ANY OF THE KIPARS TIME OUT?
(1) 021322 104010 ERROR 10 ;BRANCH IF NO
(1) 021324 012737 002076 000004 4\$: MOV #TIMERR,@#4 ;SUMMARY OF KIPAR TIMEOUTS
(3) 021332 000414 BR TST7 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
(1) 5\$: ADD #4,KSP ;GO TO NEXT TEST
(1) 021334 062706 000004 ERROR 7 ;CLEAN UP THE STACK
(1) 021340 104007 ;ONE OF THE KIPARS TIMED OUT
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 2\$' = 000756
(1) 021342 010002 MOV R0,R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
(1) 021344 050237 001302 BIS R2,ORADR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
(1) 021350 005102 COM R2 ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
(1) 021352 040237 001300 BIC R2,ANDADR
(1) 021356 005237 001304 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
(1) 021362 000746 BR 3\$;BRANCH BACK TO TEST THE NEXT KIPAR

814
818 :*****
(3) :*TEST 7 KERNEL PDR'S TIMEOUT TEST
(5) :*
(5) :* THIS TEST ADDRESSES THE EIGHT (8) KERNEL PAGE DESCRIPTOR
(5) :* REGISTERS (KIPDR0-KIPDR7) AND CHECKS THAT SOMETHING
(5) :* RESPONDS TO THEIR ADDRESSES.
(5) :*
(3) :*****

(2) 021364 000004 TST7: SCOPE

819
(1) 021366 012737 021430 001110 1\$: MOV #2,\$,LPERR ;SET LOOP ON ERROR POINTER TO 2\$
(1) 021374 012737 021466 000004 MOV #5\$,@#4 ;SET TIMEOUT VECTOR TO 5\$
(1) 021402 012700 172300 MOV #KIPDR0,R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
(1) 021406 012701 000010 MOV #10,R1 ;LOAD R1 WITH LOOP COUNT (8)


```
(1) 021412 012737 177777 001300      MOV    #-1,ANDADR      ;INITIALIZE 'AND' OF ADDR. LOC
(1) 021420 005037 001302              CLR    ORADR           ;INITIALIZE 'OR' OF ADDR. LOC.
(1) 021424 005037 001304              CLR    TONUM          ;INITIALIZE 'TIMEOUTS' COUNTER
(1) 021430 005710                      2$:   TST    (R0)         ;TRY ADDRESSING A KIPDR
(1)                                     ;IF IT TIMES OUT, WILL GO TO 5$
(1) 021432 062700 000002              3$:   ADD    #2,R0         ;PUT NEXT KIPDR ADDRESS IN R0
(1) 021436 077104                      SOB    R1,2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
(1) 021440 012737 021366 001110      MOV    #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
(1) 021446 005737 001304              TST    TONUM          ;DID ANY OF THE KIPDRS TIME OUT?
(1) 021452 001401                      BEQ    4$             ;BRANCH IF NO
(1) 021454 104010                      ERROR  10             ;SUMMARY OF KIPDR TIMEOUTS
(1) 021456 012737 002076 000004      4$:   MOV    #TIMERR,@#4  ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
(3) 021464 000414                      BR     TST10          ;GO TO NEXT TEST
(1) 021466 062706 000004              5$:   ADD    #4,KSP       ;CLEAN UP THE STACK
(1) 021472 104007                      ERROR  7              ;ONE OF THE KIPDRS TIMED OUT
(1)                                     ;FOR TIGHTER SCOPE LOOP
(1)                                     ;REPLACE ERROR CALL WITH
(1)                                     ;'BR 2$' = 000756
(1) 021474 010002                      MOV    R0,R2          ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
(1) 021476 050237 001302              BIS    R2,ORADR       ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
(1) 021502 005102                      COM    R2             ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
(1) 021504 040237 001300              BIC    R2,ANDADR      ;
(1) 021510 005237 001304              INC    TONUM          ;INCREMENT THE TIMEOUT COUNTER
(1) 021514 000746                      BR     3$             ;BRANCH BACK TO TEST THE NEXT KIPDR
```

820

824

```
(3) *****
*TEST 10      USER PAR'S TIMEOUT TEST
(5) *
(5) *          THIS TEST ADDRESSES THE EIGHT (8) USER PAGE ADDRESS
(5) *          REGISTERS (UIPAR0-UIPAR7) AND CHECKS THAT SOMETHING
(5) *          RESPONDS TO THEIR ADDRESSES.
(5) *
(3) *****
```

825

```
(2) 021516 000004      TST10: SCOPE
(1) 021520 012737 021562 001110      1$:   MOV    #2$,$LPERR     ;SET LOOP ON ERROR POINTER TO 2$
(1) 021526 012737 021620 000004      MOV    #5$,@#4        ;SET TIMEOUT VECTOR TO 5$
(1) 021534 012700 177640              MOV    #UIPAR0,R0     ;LOAD R0 WITH ADDRESS OF FIRST REG.
(1) 021540 012701 000010              MOV    #10,R1         ;LOAD R1 WITH LOOP COUNT (8)
(1) 021544 012737 177777 001300      MOV    #-1,ANDADR     ;INITIALIZE 'AND' OF ADDR. LOC
(1) 021552 005737 001302              CLR    ORADR          ;INITIALIZE 'OR' OF ADDR. LOC.
(1) 021556 005037 001304              CLR    TONUM          ;INITIALIZE 'TIMEOUTS' COUNTER
(1) 021562 005710                      2$:   TST    (R0)         ;TRY ADDRESSING A UIPAR
(1)                                     ;IF IT TIMES OUT, WILL GO TO 5$
(1) 021564 062700 000002              3$:   ADD    #2,R0         ;PUT NEXT UIPAR ADDRESS IN R0
(1) 021570 077104                      SOB    R1,2$          ;LOOP BACK TO 2$ UNTIL ALL TESTED
(1) 021572 012737 021520 001110      MOV    #1$,$LPERR     ;RESET LOOP ON ERROR POINTER TO 1$
(1) 021600 005737 001304              TST    TONUM          ;DID ANY OF THE UIPARS TIME OUT?
(1) 021604 001401                      BEQ    4$             ;BRANCH IF NO
(1) 021606 104010                      ERROR  10             ;SUMMARY OF UIPAR TIMEOUTS
(1) 021610 012737 002076 000004      4$:   MOV    #TIMERR,@#4  ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
(3) 021616 000414                      BR     TST11          ;GO TO NEXT TEST
(1) 021620 062706 000004              5$:   ADD    #4,KSP       ;CLEAN UP THE STACK
(1) 021624 104007                      ERROR  7              ;ONE OF THE UIPARS TIMED OUT
```

```
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 2$' = 000756
(1) 021626 010002 MOV R0,R2 ;LOAD THE ADDRESS THAT TIMED OUT INTO R2
(1) 021630 050237 001302 BIS R2,ORADR ;'OR' IT WITH OTHER ADDRS. THAT TIMED OUT
(1) 021634 005102 COM R2 ;'AND' IT WITH OTHER ADDRS. THAT TIMED OUT
(1) 021636 040237 001300 BIC R2,ANDADR
(1) 021642 005237 001304 INC TONUM ;INCREMENT THE TIMEOUT COUNTER
(1) 021646 000746 BR 3$ ;BRANCH BACK TO TEST THE NEXT UIPAR
```

826

830

```
(3) ;*****
(5) ;*TEST 11 USER PDR'S TIMEOUT TEST
(5) ;*
(5) ;* THIS TEST ADDRESSES THE EIGHT (8) USER PAGE DESCRIPTOR
(5) ;* REGISTERS (UIPDR0-UIPDR7) AND CHECKS THAT SOMETHING
(5) ;* RESPONDS TO THEIR ADDRESSES.
(5) ;*
(3) ;*****
```

(3)

(2)

831

```
(2) 021650 000004 TST11: SCOPE
(1) 021652 012737 021714 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
(1) 021660 012737 021752 000004 MOV #5$, @#4 ;SET TIMEOUT VECTOR TO 5$
(1) 021666 012700 177600 MOV #UIPDR0, R0 ;LOAD R0 WITH ADDRESS OF FIRST REG.
(1) 021672 012701 000010 MOV #10, R1 ;LOAD R1 WITH LOOP COUNT (8)
(1) 021676 012737 177777 001300 MOV #-1, ANDADR ;INITIALIZE 'AND' OF ADDR. LOC
(1) 021704 005037 001302 CLR ORADR ;INITIALIZE 'OR' OF ADDR. LOC.
(1) 021710 005037 001304 CLR TONUM ;INITIALIZE 'TIMEOUTS' COUNTER
(1) 021714 005710 2$: TST (R0) ;TRY ADDRESSING A UIPDR
(1) ;IF IT TIMES OUT, WILL GO TO 5$
(1) 021716 062700 000002 3$: ADD #2, R0 ;PUT NEXT UIPDR ADDRESS IN R0
(1) 021722 077104 SOB R1, 2$ ;LOOP BACK TO 2$ UNTIL ALL TESTED
(1) 021724 012737 021652 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
(1) 021732 005737 001304 TST TONUM ;DID ANY OF THE UIPDRS TIME OUT?
(1) 021736 001401 BEQ 4$ ;BRANCH IF NO
(1) 021740 104010 ERROR 10 ;SUMMARY OF UIPDR TIMEOUTS
(1) 021742 012737 002076 000004 4$: MOV #TIMERR, @#4 ;RESTORE NORMAL CPU TRAP ROUTINE ADDRESS
(3) 021750 000414 BR TST12 ;GO TO NEXT TEST
(1) ;
```

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

(1)

832

843

(3)

(4)

(4)

(4)

(4)

```
;*****
;*TEST 12 SR0(15:13) BIT TEST & SR2 TEST
;*
;* THIS TEST CHECKS BITS <15:13> OF STATUS REGISTER 0 TO SEE
;* THAT EACH CAN BE SET AND CLEARED AND THAT A 'RESET' WILL
;* CLEAR ALL OF THEM.
```



```
(4) ;* THE REST OF BITS IN SR0 WILL BE CHECKED LATER.
(4) ;* ALSO CHECK THAT SR2 IS TRACKING WITH MEM. MGMT.
(4) ;* OFF BUT LOCKS UP WHEN ANY OF SR0 ERROR BITS SET.
(4) ;*
(3) ;*****
(2) 022002 000004 TST12: SCOPE
844
845 022004 012700 177572 1$: MOV #SR0,R0 ;LOAD ADDRESS OF SR0 INTO R0
846 022010 012710 160000 MOV #160000,(R0) ;SET BITS <15:13> IN SR0 (ERROR BITS)
847 022014 000005 RESET ;ISSUE AND 'INIT' SIGNAL
848 022016 011001 MOV (R0),R1 ;READ SR0 INTO R1 TO SEE IF CLEAR
849 022020 001404 BEQ 2$ ;BRANCH IF SR0<15:13> CLEARED BY 'INIT'
850 022022 104011 ERROR 11 ;SR0<15:13> NOT CLEARED BY A 'RESET'
851 ;FOR TIGHTER SCOPE LOOP
852 ;REPLACE ERROR CALL WITH
853 ;'BR 1$' = 000770
854 022024 012737 022032 001110 MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 2$
855 022032 013737 177576 001274 2$: MOV SR2,WASSR2 ;READ CONTENTS OF SR2
856 022040 012701 022032 MOV #2$,R1 ;LOAD EXPECTED CONTENTS INTO R1
857 022044 020137 001274 CMP R1,WASSR2 ;IS SR2 TRACKING?
858 022050 001401 BEQ 3$ ;BRANCH IF YES
859 022052 104041 ERROR 41 ;SR2 NOT 'TRACKING' VIRTUAL ADDRESSES
860 ;FOR TIGHTER SCOPE LOOP
861 ;REPLACE ERROR CALL WITH
862 ;'BR 2$' = 000767
863 022054 012737 022072 001110 3$: MOV #4$,$LPERR ;SET LOOP ON ERROR POINTER TO 4$
864 022062 012701 100000 MOV #BIT15,R1 ;PUT DATA TO BE WRITTEN IN R1
865 022066 012703 000003 MOV #3,R3 ;SETUP R3 AS A LOOP COUNTER
866 022072 005010 4$: CLR (R0) ;CLEAR SR0
867 022074 050110 5$: BIS R1,(R0) ;SET ONE OF THE ERROR BITS IN SR0
868 022076 011002 MOV (R0),R2 ;READ SR0 INTO R2
869 022100 020102 CMP R1,R2 ;DID RIGHT ERROR BIT GET SET?
870 022102 001401 BEQ 6$ ;BRANCH IF YES
871 022104 104012 ERROR 12 ;BITS WERE SET WRONG IN SR0
872 ;FOR TIGHTER SCOPE LOOP
873 ;REPLACE ERROR CALL WITH
874 ;'BR 4$' = 000772
875 022106 012704 022074 6$: MOV #5$,R4 ;LOAD EXPECTED CONTENTS OF SR2 IN R4
876 022112 013737 177576 001274 MOV SR2,WASSR2 ;READ SR2
877 022120 020437 001274 CMP R4,WASSR2 ;DID SR2 LOCK UP WHEN ERROR
878 ;BIT SET IN SR1?
879 022124 001401 BEQ 7$ ;BRANCH IF YES
880 022126 104064 ERROR 64 ;SR2 DID NOT LOCK UP
881 ;FOR TIGHTER SCOPE LOOP
882 ;REPLACE ERROR CALL WITH
883 ;'BR 4$' = 000761
884 022130 006001 7$: ROR R1 ;CHANGE DATA TO CHECK NEXT ERROR BIT
885 022132 077321 SOB R3,4$ ;LOOP BACK UNTIL <15:13> ALL TESTED
886 022134 005010 CLR (R0) ;CLEAR SR0 BEFORE LEAVING
887 022136 012737 022004 001110 MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
888
898 ;*****
(3) ;*TEST 13 SR0 & PSW DUAL ADDRESSING TEST
(4) ;*
(4) ;* THIS TEST CHECKS MORE OF THE ADDRESS DETECTION LOGIC BY
(4) ;* VERIFYING THAT STATUS REGISTER 0 IS NOT EFFECTED BY WRITING
```

```
(4) :* TO THE PSW AND THAT THE LOW BYTE OF STATUS REGISTER 0
(4) :* IS NOT EFFECTED BY WRITING TO ITS HIGH BYTE. THIS IS TO
(4) :* SEE IF ADJACENT OUTPUTS ARE SHORTED ON THE ADDRESS DET. LOGIC.
(4) :*
(3) :*****
(2) 022144 000004 TST13: SCOPE
899
900 022146 005037 177776 1$: CLR PSW ;CLEAR THE PSW
901 022152 005037 177572 CLR SRO ;CLEAR STATUS REGISTER 0
902 022156 106427 000340 MTPS #340 ;SET PRIORITY 7 IN LOW BYTE OF PSW
903 022162 013700 177572 MOV SRO,R0 ;READ STATUS REGISTER 0
904 022166 001401 BEQ 2$ ;BRANCH IF IT WAS STILL 0
905 022170 104013 ERROR 13 ;SRO EFFECTED BY A WRITE TO THE PSW
906 ;FOR TIGHTER SCOPE LOOP
907 ;REPLACE ERROR CALL WITH
908 ;'BR 1$' = 000767
909 022172 005037 177572 2$: CLR SRO ;BE SURE SRO IS 0 BEFORE LEAVING
910 022176 005037 177776 CLR PSW ;BE SURE PSW IS 0 BEFORE LEAVING
911
919 :*****
(3) :*TEST 14 TEST THAT SR1 READS ALL ZEROS
(4) :*
(4) :* THIS TESTS CHECKS THAT STATUS REGISTER 1
(4) :* RESPONDS WITH ALL ZEROS, AND THAT ONLY BITS<5:4>
(4) :* OF STATUS REGISTER 3 ARE WRITEABLE.
(4) :*
(3) :*****
(2) 022202 000004 TST14: SCOPE
920 022204 012700 177777 1$: MOV #-1,R0 ;FILL R0 WITH ALL ONES
921 022210 013700 177574 MOV SR1,R0 ;READ SR1 INTO R0
922 022214 001401 BEQ 2$ ;BRANCH IF SR1 READS ALL ZEROS
923 022216 104014 ERROR 14 ;SR1 DID NOT READ ALL ZEROS
924 ;FOR TIGHTER SCOPE LOOP
925 ;REPLACE ERROR CALL WITH
926 ;000772
927 022220 012737 177777 172516 2$: MOV #-1,SR3 ;TRY TO WRITE ONES TO SR3
928 022226 022737 000060 172516 CMP #60,SR3 ;ONLY BITS <5:4> SHOULD BE ONES
929 022234 001401 BEQ 3$
930 022236 104012 ERROR 12 ;DIDN'T READ BACK A '60'
931 022240 000005 3$: RESET ;CLEARS SR3
932 022242 005737 172516 TST SR3 ;VERIFY THAT IT WAS CLEARED
933 022246 4$:
(2) 022246 001401 BEQ TST15 ;:BRANCH IF SR3 READ ALL ZEROS
934 022250 104012 ERROR 12 ;SR3 DIDN'T READ ALL ZEROS
935
943
944
945 :*****
(3) :*TEST 15 BIT TEST OF KERNEL & USER PAR'S
(4) :*
(4) :* THE FOLLOWING TEST CHECKS THE BITS <15:00> OF BOTH THE KERNEL
(4) :* AND USER PAGE ADDRESS REGISTERS. A '0' IS ROTATED THRU
(4) :* THE REGISTERS FROM LEFT TO RIGHT.
(4) :*
(3) :*****
(2) 022252 000004 TST15: SCOPE
```



```
946
947 022254 012700 172340 1$: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST PAR IN R0
948 022260 012703 000010 2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PAR'S
949 022264 012737 022272 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
950 022272 005010 ;CLR (R0) ;CLEAR THE PAR
951 022274 011001 ;MOV (R0),R1 ;READ THE PAR INTO R1
952 022276 001401 ;BEQ 4$ ;BRANCH IF PAR CLEARED OK
953 022300 104011 ;ERROR 11 ;PAR WOULD NOT CLEAR
954 ; ;FOR TIGHTER SCOPE LOOP
955 ; ;REPLACE ERROR CALL WITH
956 ; ;'BR 3$' = 000774
957 022302 012704 077777 4$: MOV #077777,R4 ;LOAD 'WALKING 0' TEST PATTERN IN R4
958 022306 012737 022314 001110 5$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
959 022314 005010 ;CLR (R0) ;CLEAR THE PAR BEFORE LOADING DATA
960 022316 050410 ;BIS R4,(R0) ;BIT SET THE TEST PATTERN INTO THE PAR
961 022320 011002 ;MOV (R0),R2 ;READ THE PAR INTO R2
962 022322 020402 ;CMP R4,R2 ;DOES DATA WRITTEN=DATA READ?
963 022324 001402 ;BEQ 6$ ;BRANCH IF YES
964 022326 010401 ;MOV R4,R1 ;SETUP FOR ERROR REPORTING
965 022330 104012 ;ERROR 12 ;PAR BITS DID NOT SET CORRECTLY
966 ; ;FOR TIGHTER SCOPE LOOP
967 ; ;REPLACE ERROR CALL WITH
968 ; ;'BR 5$' = 000767
969 022332 000261 6$: SEC ;SET THE C-BIT FOR THE ROTATE INST.
970 022334 006004 ;ROR R4 ;ROTATE THE TEST PATTERN IN R4
971 022336 103766 ;BCS 5$ ;BRANCH BACK IF MORE BITS TO TEST
972 022340 062700 000002 ;ADD #2,R0 ;GET NEXT PAR ADDRESS IN R0
973 022344 077326 ;SOB R3,3$ ;BRANCH BACK UNTIL ALL PAR'S TESTED
974 022346 022700 177660 ;CMP #UIPAR7+2,R0 ;HAVE USER PAR'S BEEN TESTED
975 022352 103003 ;BHIS 7$ ;BRANCH IF YES
976 022354 012700 177640 ;MOV #UIPAR0,R0 ;LOAD FIRST USER PAR ADDR. IN R0
977 022360 000737 ;BR 2$ ;BRANCH BACK TO TEST USER PAR'S
978 022362 012737 022254 001110 7$: MOV #1$, $LPERR ;RESET LOOP OR ERROR POINTER TO 1$
979 ; ;LEAVE TEST WITH BITS <11:1>=1 IN ALL PAR'S
980 ; ;*****
981 (3) ;*TEST 16 BIT TEST OF KERNEL & USER PDR'S
982 (4) ;*
983 (4) ;* THE FOLLOWING TEST CHECKS THE BITS <14:8> AND <3:1> OF BOTH THE
984 (4) ;* KERNEL AND USER PAGE DESCRIPTOR REGISTERS. A '0' IS ROTATED
985 (4) ;* THRU THE REGISTERS FROM LEFT TO RIGHT. SOME TEST PATTERNS WILL
986 (4) ;* BE LOADED MORE THAN ONCE DUE TO THE UNUSED BITS IN THE PDR'S.
987 (4) ;*
988 (3) ; ;*****
989 (2) 022370 000004 TST16: SCOPE
990 022372 012700 172300 1$: MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR IN R0
991 022376 012703 000010 2$: MOV #10,R3 ;SETUP R3 TO COUNT 8 PDR'S
992 022402 012737 022410 001110 3$: MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
993 022410 005010 ;CLR (R0) ;CLEAR THE PDR
994 022412 011001 ;MOV (R0),R1 ;READ THE PDR INTO R1
995 022414 001401 ;BEQ 4$ ;BRANCH IF PDR CLEARED OK
996 022416 104011 ;ERROR 11 ;PDR WOULD NOT CLEAR
997 ; ;FOR TIGHTER SCOPE LOOP
998 ; ;REPLACE ERROR CALL WITH
999 ; ;'BR 3$' = 000774
1000 022420 012704 077777 4$: MOV #077777,R4 ;LOAD 'WALKING '0' TEST PATTERN IN R4
```

```
1001 022424 012737 022432 001110      MOV    #5$, $LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1002 022432 005010                    CLR    (R0)              ;CLEAR THE PDR BEFORE LOADING DATA
1003 022434 010401                    MOV    R4, R1            ;LOAD DATA INTO R1
1004 022436 042701 100361              BIC    #100361, R1       ;MASK UNUSED BITS OUT OF THE DATA
1005 022442 050110                    BIS    R1, (R0)          ;BIT SET THE TEST PATTERN INTO THE PDR
1006 022444 011002                    MOV    (R0), R2         ;READ THE PDR INTO R2
1007 022446 020102                    CMP    R1, R2           ;DOES DATA WRITTEN=DATA READ?
1008 022450 001401                    BEQ    6$               ;BRANCH IF YES
1009 022452 104012                    ERROR  12               ;PDR BITS DID NOT SET CORRECTLY
1010                                     ;FOR TIGHTER SCOPE LOOP
1011                                     ;REPLACE ERROR CALL WITH
1012                                     ;'BR 5$' = 000767
1013 022454 000261                    6$:   SEC                ;SET THE C-BIT FOR THE ROTATE INST.
1014 022456 006004                    ROR    R4                ;ROTATE THE TEST PATTERN IN R4
1015 022460 103764                    BCS    5$               ;BRANCH BACK IF MORE BITS TO TEST
1016 022462 062700 000002              ADD    #2, R0            ;GET NEXT PDR ADDRESS IN R0
1017 022466 077330                    SOB    R3, 3$          ;BRANCH BACK UNTIL ALL PDR'S TESTED
1018 022470 022700 177620              CMP    #UIPDR7+2, R0    ;HAVE USER PDR'S BEEN TESTED?
1019 022474 103003                    BHIS   7$               ;BRANCH IF YES
1020 022476 012700 177600              MOV    #UIPDRO, R0      ;LOAD FIRST USER PDR ADDR. IN R0
1021 022502 000735                    BR     2$               ;BRANCH BACK TO TEST USER PDR'S
1022 022504 012737 022372 001110 7$:  MOV    #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1023                                     ;LEAVE TEST WITH ALL WRITEABLE BITS IN
1024                                     ;ALL PDR'S = 1
1025
1033                                     ;*****
1033 (3)                                ;*TEST 17      TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PAR'S
1034 (4)                                ;*
1035 (4)                                ;*      THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
1036 (4)                                ;*      PAR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
1037 (4)                                ;*      THE OTHER.
1038 (4)                                ;*
1039 (3)                                ;*
1040 (2) 022512 000004                TST17: SCOPE
1041
1036 022514 012700 172340              1$:   MOV    #KIPAR0, R0   ;LOAD ADDRESS OF FIRST PAR INTO R0
1037 022520 012737 022532 001110 2$:  MOV    #3$, $LPERR      ;SET LOOP ON ERROR POINTER TO 3$
1038 022526 012703 000010              MOV    #10, R3          ;LOAD LOOP COUNTER TO DO 8 PAR'S
1039 022532 012701 177777              3$:   MOV    #-1, R1        ;LOAD TEST PATTERN INTO R1
1040 022536 005010                    CLR    (R0)              ;CLEAR THE PAR
1041 022540 110110                    MOV    R1, (R0)          ;WRITE 1'S TO THE LOW BYTE OF THE PAR
1042 022542 011002                    MOV    (R0), R2         ;READ THE ENTIRE PAR INTO R2
1043 022544 042701 177400              BIC    #177400, R1       ;MASK HIGH BYTE & UNUSED BITS OUT OF THE DATA
1044 022550 020102                    CMP    R1, R2           ;WAS ONLY THE LOW BYTE WRITTEN TO
1045 022552 001401                    BEQ    4$               ;BRANCH IF YES
1046 022554 104015                    ERROR  15               ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PAR
1047                                     ;FOR TIGHTER SCOPE LOOP
1048                                     ;REPLACE ERROR CALL WITH
1049                                     ;'BR 3$' = 000766
1050 022556 012737 022564 001110 4$:  MOV    #5$, $LPERR      ;SET LOOP ON ERROR POINTER TO 5$
1051 022564 005010                    5$:   CLR    (R0)              ;CLEAR THE PAR
1052 022566 012701 177777              MOV    #-1, R1          ;LOAD TEST, PATTERN INTO R1
1053 022572 110160 000001              MOV    R1, 1(R0)        ;WRITE 1'S TO THE HIGH BYTE OF THE PAR
1054 022576 011002                    MOV    (R0), R2         ;READ THE ENTIRE PAR INTO R2
1055 022600 042701 000377              BIC    #000377, R1       ;MASK LOW BYTE
```



```
1056 022604 020102      CMP      R1,R2      ;WAS ONLY THE HIGH BYTE WRITTEN 10?
1057 022606 001401      BEQ      6$         ;BRANCH IF YES
1058 022610 104015      ERROR    15         ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PAR
1059                      ;FOR TIGHTER SCOPE LOOP
1060                      ;REPLACE ERROR CALL WITH
1061                      ;'BR 5'' = 000765
1062 022612 062700 000002 6$:      ADD      #2,R0      ;PUT ADDRESS OF NEXT PAR IN R0
1063 022616 077333      SOB     R3,3$      ;BRANCH BACK UNTIL 8 PAR'S TESTED
1064 022620 022700 177660      CMP     #UIPAR7+2,R0 ;HAVE USER PAR'S BEEN TESTED
1065 022624 103003      BHIS   7$         ;BRANCH IF YES
1066 022626 012700 177640      MOV     #UIPAR0,R0 ;LOAD ADDRESS OF FIRST USER PAR IN R0
1067 022632 000732      BR     2$         ;BRANCH BACK TO TEST USER PAR'S
1068 022634 012737 022514 001110 7$:      MOV     #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1069
1077                      ;*****
(3)                      ;*TEST 20      TEST FOR DUAL BYTE ADDRESSING OF KERNEL & USER PDR'S
(4)                      ;*
(4)                      ;*      THE FOLLOWING TEST WRITES TO BOTH BYTES OF THE KERNEL & USER
(4)                      ;*      PDR'S SEPERATELY TO SEE THAT WRITING TO ONE DOES NOT EFFECT
(4)                      ;*      THE OTHER.
(4)                      ;*
(3)                      ;*****
(2) 022642 000004      TST20: SCOPE
1078
1079 022644 012700 172300 001110 1$:      MOV     #KIPDR0,R0 ;LOAD ADDRESS OF FIRST PDR INTO R0
1080 022650 012737 022662 001110 2$:      MOV     #3$,$LPERR ;SET LOOP ON ERROR POINTER TO 3$
1081 022656 012703 000010      MOV     #10,R3     ;LOAD LOOP COUNTER TO DO 8 PDR'S
1082 022662 012701 177777 3$:      MOV     #-1,R1     ;LOAD TEST PATTERN INTO R1
1083 022666 005010      CLR    (R0)        ;CLEAR THE PDR
1084 022670 110110      MOVB   R1,(R0)     ;WRITE 1'S TO THE LOW BYTE OF THE PDR
1085 022672 011002      MOV    (R0),R2     ;READ THE ENTIRE PDR INTO R2
1086 022674 042701 177761      BIC    #177761,R1  ;MASK HIGH BYTE & UNUSED BITS OUT OF DATA
1087 022700 020102      CMP    R1,R2      ;WAS ONLY THE LOW BYTE WRITTEN TO?
1088 022702 001401      BEQ    4$         ;BRANCH IF YES
1089 022704 104015      ERROR  15         ;HIGH BYTE EFFECTED BY WRITING LOW BYTE IN PDR
1090                      ;FOR TIGHTER SCOPE LOOP
1091                      ;REPLACE ERROR CALL WITH
1092                      ;'BR 3$' = 000766
1093 022706 012737 022714 001110 4$:      MOV     #5$,$LPERR ;SET LOOP ON ERROR POINTER TO 5$
1094 022714 005010 5$:      CLR    (R0)        ;CLEAR THE PDR
1095 022716 012701 177777      MOV     #-1,R1     ;LOAD TEST PATTERN INTO R1
1096 022722 110160 000001      MOVB   R1,1(R0)   ;WRITE 1'S TO THE HIGH BYTE OF THE PDR
1097 022726 011002      MOV    (R0),R2     ;READ THE ENTIRE PDR INTO R2
1098 022730 042701 100377      BIC    #100377,R1  ;MASK LOW BYTE & UNUSED BITS OUT OF DATA
1099 022734 020102      CMP    R1,R2      ;WAS ONLY THE HIGH BYTE WRITTEN TO?
1100 022736 001401      BEQ    6$         ;BRANCH IF YES
1101 022740 104015      ERROR  15         ;LOW BYTE EFFECTED BY WRITING HIGH BYTE IN PDR
1102                      ;FOR TIGHTER SCOPE LOOP
1103                      ;REPLACE ERROR CALL WITH
1104                      ;'BR 5$' = 000765
1105 022742 062700 000002 6$:      ADD     #2,R0      ;PUT ADDRESS OF NEXT PDR IN R0
1106 022746 077333      SOB   R3,3$       ;BRANCH BACK UNTIL 8 PDR'S TESTED
1107 022750 022700 177620      CMP   #UIPDR7+2,R0 ;HAVE USER PDR'S BEEN TESTED?
1108 022754 103003      BHIS  7$         ;BRANCH IF YES
1109 022756 012700 177600      MOV   #UIPDR0,R0  ;LOAD ADDRESS OF FIRST USER PDR IN R0
1110 022762 000732      BR    2$         ;BRANCH BACK TO TEST USER PDR'S
```

```
1111 022764 012737 022644 001110 7$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
1112
1124 (3) :*****
(4) :*TEST 21 PAR-PDR DUAL ADDRESSING TEST
(4) :*
(4) :* THE FOLLOWING TEST SETS ALL OF THE WRITEABLE BITS TO 1
(4) :* IN THE SIXTEEN (16) PAR'S AND PDR'S USING THE 'SETREG'
(4) :* SUBROUTINE AND THEN CLEARS JUST ONE OF THEM. THE 'CMPREG'
(4) :* SUBROUTINE IS USED TO READ ALL OF THE PAR'S AND PDR'S TO SEE
(4) :* THAT ONLY ONE REGISTER WAS CLEARED IN RESPONSE TO THAT ONE
(4) :* PAR OR PDR ADDRESS. THE 'CMPREG' SUBROUTINE REPORTS THE
(4) :* ADDRESS OF ANY REGISTER WHOSE BITS DID NOT REMAIN SET WHEN
(4) :* ANOTHER REGISTER WAS CLEARED.
(3) :*****
(2) 022772 000004 TST21: SCOPE
1125
1126 022774 012737 023016 001110 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER 2$
1127 023002 012703 000010 MOV #10, R3 ;LOAD LOOP COUNTER WITH AN 8
1128 023006 012700 172300 MOV #KIPDRO, RO ;LOAD ADDRESS OF FIRST KERNEL PDR AND RO
1129 023012 004737 035166 JSR PC, SETREG ;SET ALL BITS IN ALL PAR'S IN PDR'S
1130 023016 012706 001100 2$: MOV #KERSTK, KSP ;SETUP STACK POINTER
1131 023022 005010 CLR (RO) ;CLEAR ONE OF THE KERNEL PDR'S
1132 023024 004737 035260 JSR PC, CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
1133 023030 012720 177777 MOV #-1, (RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PDR
1134 023034 077310 SOB R3, 2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
1135 023036 012737 023054 001110 MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
1136 023044 012703 000010 MOV #10, R3 ;LOAD LOOP COUNTER WITH AN 8
1137 023050 012700 172340 MOV #KIPARO, RO ;LOAD ADDRESS OF FIRST KERNEL PAR IN RO
1138 023054 012706 001100 3$: MOV #KERSTK, KSP ;SETUP STACK POINTER
1139 023060 005010 CLR (RO) ;CLEAR ONE OF THE KERNEL PAR'S
1140 023062 004737 035260 JSR PC, CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
1141 023066 012720 177777 MOV #-1, (RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT PAR
1142 023072 077310 SOB R3, 3$ ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
1143 023074 012737 023112 001110 MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
1144 023102 012703 000010 MOV #10, R3 ;LOAD LOOP COUNTER WITH AN 8
1145 023106 012700 177600 MOV #UIPDRO, RO ;LOAD ADDRESS OF FIRST USER PDR IN RO
1146 023112 012706 001100 4$: MOV #KERSTK, KSP ;SETUP STACK POINTER
1147 023116 005010 CLR (RO) ;CLEAR ONE OF THE USER PDR'S
1148 023120 004737 035260 JSR PC, CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
1149 023124 012720 177777 MOV #-1, (RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT UPDR
1150 023130 077310 SOB R3, 4$ ;LOOP TO 4$ UNTIL ALL USER PDR'S CHECKED
1151 023132 012737 023150 001110 MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
1152 023140 012703 000010 MOV #10, R3 ;LOAD LOOP COUNTER WITH AN 8
1153 023144 012700 177640 MOV #UIPARO, RO ;LOAD ADDRESS OF FIRST USER PAR IN RO
1154 023150 012706 001100 5$: MOV #KERSTK, KSP ;SETUP STACK POINTER
1155 023154 005010 CLR (RO) ;CLEAR ONE OF THE USER PAR'S
1156 023156 004737 035260 JSR PC, CMPREG ;SEE IF OTHER PAR/PDR'S WERE EFFECTED
1157 023162 012720 177777 MOV #-1, (RO)+ ;RESTORE ALL ONES, AND SETUP FOR NEXT UPAR
1158 023166 077310 SOB R3, 5$ ;LOOP TO 5$ UNTIL ALL USER PAR'S CHECKED
1159 023170 012737 022774 001110 MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
```

```
1170 (3) :*****
(4) :*TEST 22 TEST THAT PAR-PDR'S NOT AFFECTED BY RESET
(4) :*
(4) :* THIS TEST CHECKS TO SEE THAT THE KERNEL OR USER PAR/PDR'S ARE
(4) :* NOT AFFECTED BY THE EXECUTION OF A 'RESET' INSTRUCTION. THE
```



```
(4) ;* 'SETREG' SUBROUTINE IS USED TO SET ALL WRITEABLE BITS TO A '1' IN
(4) ;* THE PAR/PDR'S. THEN THEY ARE READ TO SEE THAT THEY REMAINED
(4) ;* UNCHANGED
(4) ;*
(3) ;*****
(2) 023176 000004 TST22: SCOPE
1171
1172
1173 023200 004737 035166 1$: JSR PC,SETREG ;SET ALL BITS IN ALL PAR'S AND PDR'S
1174 023204 000005 RESET ;ISSUE AN 'INIT' BY EXECUTING A RESET
1175 023206 012700 172300 MOV #KIPDRO,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN RO
1176 023212 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
1177 023216 011001 2$: MOV (R0),R1 ;READ A KERNEL PDR INTO R1
1178 023220 022701 077416 CMP #77416,R1 ;ARE ALL THE BITS STILL SET?
1179 023224 001401 BEQ 3$ ;BRANCH IF YES
1180 023226 104055 ERROR 55 ;KERNEL PDR AFFECTED BY A RESET
1181 ;FOR TIGHTER SCOPE LOOP
1182 ;REPLACE ERROR CALL WITH
1183 ;'BR 2$' = 000773
1184 023230 062700 000002 3$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PDR
1185 023234 077410 SOB R4,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PDR'S CHECKED
1186 023236 012700 172340 MOV #KIPARO,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN RO
1187 023242 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
1188 023246 011001 4$: MOV (R0),R1 ;READ A KERNEL PAR INTO R1
1189 023250 022701 177777 CMP #177777,R1 ;ARE ALL THE BITS STILL SET?
1190 023254 001401 BEQ 5$ ;BRANCH IF YES
1191 023256 104055 ERROR 55 ;KERNEL PAR AFFECTED BY A RESET
1192 ;FOR TIGHTER SCOPE LOOP
1193 ;REPLACE ERROR CALL WITH
1194 ;'BR 4$' = 000773
1195 023260 062700 000002 5$: ADD #2,R0 ;FORM ADDRESS OF NEXT KERNEL PAR
1196 023264 077410 SOB R4,4$ ;LOOP TO 4$ UNTIL ALL KERNEL PAR'S CHECKED
1197 023266 012700 177600 MOV #UIPDRO,R0 ;LOAD ADDRESS OF FIRST USER PDR IN RO
1198 023272 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
1199 023276 011001 6$: MOV (R0),R1 ;READ A USER PDR INTO R1
1200 023300 022701 077416 CMP #77416,R1 ;ARE ALL THE BITS STILL SET?
1201 023304 001401 BEQ 7$ ;BRANCH IF YES
1202 023306 104055 ERROR 55 ;USER PDR AFFECTED BY A RESET
1203 ;FOR TIGHTER SCOPE LOOP
1204 ;REPLACE ERROR CALL WITH
1205 ;'BR 6$' = 000773
1206 023310 062700 000002 7$: ADD #2,R0 ;FORM ADDRESS OF NEXT USER PDR
1207 023314 077410 SOB R4,6$ ;LOOP TO 6$ UNTIL ALL USER PDR'S CHECKED
1208
1209 MOV #UIPARO,R0 ;LOAD ADDRESS OF FIRST USER PAR IN RO
1210 023322 012704 000010 MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
1211 023326 011001 8$: MOV (R0),R1 ;READ A USER PAR INTO R1
1212 023330 022701 177777 CMP #177777,R1 ;ARE ALL THE BITS STILL SET?
1213 023334 001401 BEQ 9$ ;BRANCH IF YES
1214 023336 104055 ERROR 55 ;USER PAR AFFECTED BY A RESET
1215 ;FOR TIGHTER SCOPE LOOP
1216 ;REPLACE ERROR CALL WITH
1217 ;'BR 8$' = 000773
1218 023340 062700 000002 9$: ADD #2,R0 ;FORM ADDRESS OF NEXT USER PAR
1219 023344 077410 SOB R4,8$ ;LOOP TO 8$ UNTIL ALL USER PAR'S CHECKED
1236 ;*****
```

```

(3) ;*TEST 23 RELOCATION & ADDER TEST (NO CARRIES)
(4) ;*
(4) ;* THE FOLLOWING TEST SETS UP THE KERNEL PAR'S AND PDR'S
(4) ;* FOR THE REST OF THE PROGRAM. IT THEN USES DIFFERENT
(4) ;* VIRTUAL ADDRESSES AND DIFFERENT VALUES FOR KERNEL PAR 4
(4) ;* TO PUT DIFFERENT PATTERNS AT THE INPUTS OF THE
(4) ;* MEMORY MANAGEMENT ADDER. THE VALUES ARE SUCH
(4) ;* THAT NO CARRIES ARE GENERATED OUT OF THE ADDER.
(4) ;*
(4) ;* THE METHOD USED TO SEE THAT THE RIGHT PHYSICAL BUS ADDRESS
(4) ;* IS FORMED BY THE ADDER IS TO WRITE A PATTERN TO VIRTUAL
(4) ;* LOCATION WITH MEMORY MGMT. AND
(4) ;* THEN READ THAT LOCATION USING THE PHYSICAL ADDRESS THAT SHOULD
(4) ;* HAVE BEEN FORMED TO SEE IF THE TEST PATTERN GOT THEIR.
(4) ;* 22-BIT AND 18-BIT ADDRESSING A.. USED.
(3) ;*****
(2) 023346 000004 TST23: SCOPE
1237
1238 023350 012700 172340 1$: MOV #KIPAR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R0
1239 023354 005001 CLR R1 ;CLEAR R1
1240 023356 012702 000007 MOV #7,R2 ;LOAD LOOP COUNTER WITH A 7
1241 023362 010120 2$: MOV R1,(R0)+ ;MAP KERNEL PAR'S TO PAGES 0-6 (4K EACH)
1242 023364 062701 000200 ADD #200,R1
1243 023370 077204 SOB R2,2$ ;LOOP UNTIL KIPAR0 - KIPAR6 ARE LOADED
1244 023372 012710 007600 MOV #7600,(R0) ;MAP KIPAR7 TO THE I/O PAGE
1245 023376 012700 172300 MOV #KIPDR0,R0 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R0
1246 023402 012701 077406 MOV #77406,R1 ;LOAD PDR DATA INTO R1
1247 023406 012702 000010 MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
1248 023412 010120 3$: MOV R1,(R0)+ ;MAP ALL 8 PAGES 128 BLOCKS, UPWARD
1249 023414 077202 SOB R2,3$ ; EXPANDABLE, READ/WRITE
1250
1251 023416 012737 023416 001110 4$: MOV #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
(1) 023424 012700 067776 MOV #67776,R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
(1) 023430 012701 107776 MOV #107776,R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
(1) 023434 012702 125250 MOV #125250,R2 ;LOAD TEST PATTERN INTO R2
(1) 023440 012704 000600 MOV #600,R4 ;LOAD R4 WITH PAR VALUE
(1) 023444 010437 172350 MOV R4,KIPAR4 ;LOAD KERNEL PAR 4 BITS <15:00>
(1) 023450 011037 001176 MOV (R0),$TMP0 ;SAVE CONTENTS AT TEST LOCATION
(1) 023454 052737 000021 177572 BIS #21,SRO ;TURN ON MEM. MGMT. 22-BIT ADDRESSING
(1) 023462 010211 MOV R2,(R1) ;LOAD 125250 USING ADDER (PAR4 + VIRT ADDR.)
(1) 023464 000005 RESET ;TURN OFF MEMORY MGMT.
(1) 023466 011003 MOV (R0),R3 ;READ 125250 BACK WITHOUT USING MEM. MGMT.
(1) 023470 013710 001176 MOV $TMP0,(R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
(1) 023474 020203 CMP R2,R3 ;WAS SAME PATTERN READ BACK THAT WAS
(1) ;WRITTEN USING MEMORY MANAGEMENT?
(1) BEQ 5$ ;BRANCH IF YES
(1) 023476 001405 MOV R1,VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
(1) 023500 010137 001306 JSR PC,FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
(1) 023504 004737 035452 JSR PC,FORMPA ;TEST LOCATION DID NOT HAVE PATTERN
(1) 023510 104017 ERROR 17 ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
(1) ;APPARENTLY PHYSICAL ADDR. WAS
(1) ;FORMED WRONG BY ADDERS USING
(1) ;THE VIRTUAL ADDR. AND KIPAR4
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 4$' = 000742
  
```



```

(1) 023512
1252 023512 012737 023512 001110 5$:
(1) 023520 012700 067776 6$: MOV #6$, $LPERR ;SET LOOP ON ERROR POINTER TO 6$
(1) 023524 012701 102576 MOV #67776, R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
(1) 023530 012702 125251 MOV #102576, R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
(1) 023534 012704 000652 MOV #125251, R2 ;LOAD TEST PATTERN INTO R2
(1) 023540 010437 172350 MOV #652, R4 ;LOAD R4 WITH PAR VALUE
(1) 023544 011037 001176 MOV R4, KIPAR4 ;LOAD KERNEL PAR 4 BITS <15:00>
(1) 023550 052737 000021 177572 MOV (R0), $TMP0 ;SAVE CONTENTS AT TEST LOCATION
(1) 023556 010211 BIS #21, SR0 ;TURN ON MEM. MGMT. 22-BIT ADDRESSING
(1) 023560 000005 MOV R2, (R1) ;LOAD 125251 USING ADDER (PAR4 + VIRT ADDR.)
(1) 023562 011003 RESET ;TURN OFF MEMORY MGMT.
(1) 023564 013710 001176 MOV (R0), R3 ;READ 125251 BACK WITHOUT USING MEM. MGMT.
(1) 023570 020203 MOV $TMP0, (R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
(1) CMP R2, R3 ;WAS SAME PATTERN READ BACK THAT WAS
(1) ;WRITTEN USING MEMORY MANAGEMENT?
(1) 023572 001405 BEQ 7$ ;BRANCH IF YES
(1) 023574 010137 001306 MOV R1, VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
(1) 023600 004737 035452 JSR PC, FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
(1) 023604 104017 ERROR 17 ;TEST LOCATION DID NOT HAVE PATTERN
(1) ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
(1) ;APPARENTLY PHYSICAL ADDR. WAS
(1) ;FORMED WRONG BY ADDERS USING
(1) ;THE VIRTUAL ADDR. AND KIPAR4
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 6$' = 000742
(1) 023606
1253 023606 012737 023606 001110 7$:
(1) 023614 012700 067776 8$: MOV #8$, $LPERR ;SET LOOP ON ERROR POINTER TO 8$
(1) 023620 012701 105276 MOV #67776, R0 ;LOAD PHYSICAL ADDR. PBA INTO R0
(1) 023624 012702 125252 MOV #105276, R1 ;LOAD VIRTUAL ADDR. VBA INTO R1
(1) 023630 012704 000625 MOV #125252, R2 ;LOAD TEST PATTERN INTO R2
(1) 023634 010437 172350 MOV #625, R4 ;LOAD R4 WITH PAR VALUE
(1) 023640 011037 001176 MOV R4, KIPAR4 ;LOAD KERNEL PAR 4 BITS <15:00>
(1) 023644 052737 000021 177572 MOV (R0), $TMP0 ;SAVE CONTENTS AT TEST LOCATION
(1) 023652 010211 BIS #21, SR0 ;TURN ON MEM. MGMT. 22-BIT ADDRESSING
(1) 023654 000005 MOV R2, (R1) ;LOAD 125252 USING ADDER (PAR4 + VIRT ADDR.)
(1) 023656 011003 RESET ;TURN OFF MEMORY MGMT.
(1) 023660 013710 001176 MOV (R0), R3 ;READ 125252 BACK WITHOUT USING MEM. MGMT.
(1) 023664 020203 MOV $TMP0, (R0) ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
(1) CMP R2, R3 ;WAS SAME PATTERN READ BACK THAT WAS
(1) ;WRITTEN USING MEMORY MANAGEMENT?
(1) 023666 001405 BEQ 9$ ;BRANCH IF YES
(1) 023670 010137 001306 MOV R1, VIRT1 ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
(1) 023674 004737 035452 JSR PC, FORMPA ;GO FORM PHYSICAL ADDRESS FOR TYPING
(1) 023700 104017 ERROR 17 ;TEST LOCATION DID NOT HAVE PATTERN
(1) ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
(1) ;APPARENTLY PHYSICAL ADDR. WAS
(1) ;FORMED WRONG BY ADDERS USING
(1) ;THE VIRTUAL ADDR. AND KIPAR4
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 8$' = 000742
(1) 023702
1254 023702 012737 023702 001110 9$:
1255 023702 012737 023702 10$: MOV #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
1256 023710 012700 177776 MOV #PSW, R0 ;LOAD PHYS. ADDR. OF PSW INTO R0

```

1257	023714	012701	100076			MOV	#100076,R1	:LOAD VIRTUAL ADDR. FOR PSW INTO R1
1258	023720	012702	030340			MOV	#030340,R2	:LOAD DATA FOR PSW IN R2
1259	023724	012704	007777			MOV	#7777,R4	:LOAD R4 WITH PAR VALUE
1260	023730	010437	172350			MOV	R4,KIPAR4	:LOAD KERNEL PAR 4 BITS <11:00>
1261	023734	005010				CLR	(R0)	:CLEAR THE PSW
1262	023736	052737	000001	177572		BIS	#BIT0,SRO	:TURN ON 'MEMORY MANAGEMENT' (18 BIT ADDRESSING)
1263	023744	010211				MOV	R2,(R1)	:LOAD PSW USING ADDER (PAR4 + VIRT ADDR.)
1264	023746	000005				RESET		:TURN OFF MEM. MGMT (SRO=0)
1265	023750	011003				MOV	(R0),R3	:READ PSW BACK WITHOUT USING MEM. MGMT.
1266	023752	005010				CLR	(R0)	:CLEAR THE PSW
1267	023754	042703	000037			BIC	#37,R3	:MASK T-BIT & CC BITS OUT OF DATA READ
1268	023760	020203				CMP	R2,R3	:WAS PSW WRITTEN?
1269	023762	001405				BEQ	11\$:BRANCH IF YES
1270	023764	010137	001306			MOV	R1,VIRT1	:SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
1271	023770	004737	035452			JSR	PC,FORMPA	:GO FORM PHYSICAL ADDR. FOR TYPING
1272	023774	104017				ERROR	17	:PSW DID NOT HAVE DATA THAT IT SHOULD HAVE,
1273								:APPARENTLY PHYS. ADDR. OF PSW WAS
1274								:NOT FORMED BY ADDERS USING THE
1275								:VIRTUAL ADDR. AND KIPAR4
1276								:FOR TIGHTER SCOPE LOOP
1277								:REPLACE ERROR CALL WITH
1278								: 'BR 10\$' = 000742
1279	023776	012737	023776	001110	11\$:	MOV	#11\$,\$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
1280	024004	012700	177776			MOV	#PSW,R0	:LOAD PHYS. ADDR. OF PSW INTO R0
1281	024010	012701	117776			MOV	#117776,R1	:LOAD VIRTUAL ADDR. FOR PSW INTO R1
1282	024014	012702	030240			MOV	#030240,R2	:LOAD DATA FOR PSW IN R2
1283	024020	012704	177600			MOV	#177600,R4	:LOAD R4 WITH PAR VALUE
1284	024024	010437	172350			MOV	R4,KIPAR4	:LOAD KERNEL PAR 4 BITS <15:00>
1285	024030	052737	000021	177572		BIS	#21,SRO	:TURN ON 'MEMORY MANAGEMENT' (22 BIT ADDRESSING)
1286	024036	010211				MOV	R2,(R1)	:LOAD PSW USING ADDER (PAR4 + VIRT. ADDR.)
1287	024040	000005				RESET		:TURN OFF MEM. MGMT (SRO=0)
1288	024042	011003				MOV	(R0),R3	:READ PSW BACK WITHOUT USING MEM. MGMT.
1289	024044	005010				CLR	(R0)	:CLEAR THE PSW
1290	024046	042703	000037			BIC	#37,R3	:MASK T-BIT & CC BITS OUT OF DATA READ
1291	024052	020203				CMP	R2,R3	:WAS PSW WRITTEN?
1292	024054	001405				BEQ	12\$:BRANCH IF YES
1293	024056	010137	001306			MOV	R1,VIRT1	:SAVE VIRTUAL ADDR. TO FORM PHYSICAL ADDR.
1294	024062	004737	035452			JSR	PC,FORMPA	:GO FORM PHYSICAL ADDR. FOR TYPING
1295	024066	104017				ERROR	17	:PSW DID NOT HAVE DATA THAT IT SHOULD
1296								:HAVE, APPARENTLY PHYS. ADDR. OF PSW WAS
1297								:NOT FORMED BY ADDERS USING THE
1298								:VIRTUAL ADDR. AND KIPAR4
1299								:FOR TIGHTER SCOPE LOOP
1300								:REPLACE ERROR CALL WITH
1301								: 'BR 11\$' = 000743
1302	024070	012737	023350	001110	12\$:	MOV	#1\$,\$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$
1303								

1315
(3) :*****
(4) :*TEST 24 RELOCATION & ADDER TEST (WITH CARRIES)
(4) :*
(4) :*
(4) :* THE FOLLOWING TEST USES THE SAME METHOD AS THE PREVIOUS
(4) :* TEST TO VERIFY MEMORY MANagements ABILITY TO CONSTRUCT
(4) :* PHYSICAL BUS ADDRESSES USING A VIRTUAL BUS ADDRESS AND THE
(4) :* CONTENTS OF A PAGE ADDRESS REGISTER. HOWEVER, THE VALUES
(4) :* AND PATTERNS USED IN THIS TEST WILL GENERATE CARRIES
(4) :* AND CHECK 'WRAPAROUND' TO ADDRESS 000000 BY


```

(1)                                     ;REPLACE ERROR CALL WITH
(1)                                     ;'BR 4$' = 000742
(1) 024270                               5$:
1321 024270 012737 024270 001110 6$:  MOV   #6$, $LPERR      ;SET LOOP ON ERROR POINTER TO 6$
(1) 024276 012700 062076                MOV   #62076, R0      ;LOAD PHYSICAL ADDR. PBA INTO R0
(1) 024302 012701 104576                MOV   #104576, R1    ;LOAD VIRTUAL ADDR. VBA INTO R1
(1) 024306 012702 125255                MOV   #125255, R2    ;LOAD TEST PATTERN INTO R2
(1) 024312 012704 000553                MOV   #553, R4 ;LOAD R4 WITH PAR VALUE
(1) 024316 010437 172350                MOV   R4, KIPAR4    ;LOAD KERNEL PAR 4 BITS <15:00>
(1) 024322 011037 001176                MOV   (R0), $TMP0   ;SAVE CONTENTS AT TEST LOCATION
(1) 024326 052737 000021 177572        BIS   #21, SR0      ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
(1) 024334 010211                        MOV   R2, (R1)      ;LOAD 125255 USING ADDER (PAR4 + VIRT ADDR.)
(1) 024336 000005                        RESET                      ;TURN OFF MEMORY MGNT.
(1) 024340 011003                        MOV   (R0), R3      ;READ 125255 BACK WITHOUT USING MEM. MGNT.
(1) 024342 013710 001176                MOV   $TMP0, (R0)   ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
(1) 024346 020203                        CMP   R2, R3        ;WAS SAME PATTERN READ BACK THAT WAS
(1)                                     ;WRITTEN USING MEMORY MANAGEMENT?
(1) 024350 001405                        BEQ   7$            ;BRANCH IF YES
(1) 024352 010137 001306                MOV   R1, VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
(1) 024356 004737 035452                JSR   PC, FORMPA    ;GO FORM PHYSICAL ADDRESS FOR TYPING
(1) 024362 104017                        ERROR  17           ;TEST LOCATION DID NOT HAVE PATTERN
(1)                                     ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
(1)                                     ;APPARENTLY PHYSICAL ADDR. WAS
(1)                                     ;FORMED WRONG BY ADDERS USING
(1)                                     ;THE VIRTUAL ADDR. AND KIPAR4
(1)                                     ;FOR TIGHTER SCOPE LOOP
(1)                                     ;REPLACE ERROR CALL WITH
(1)                                     ;'BR 6$' = 000742
(1) 024364                               7$:
1322 024364 012737 024364 001110 8$:  MOV   #8$, $LPERR      ;SET LOOP ON ERROR POINTER TO 8$
(1) 024372 012700 000000                MOV   #00000, R0    ;LOAD PHYSICAL ADDR. PBA INTO R0
(1) 024376 012701 111400                MOV   #111400, R1   ;LOAD VIRTUAL ADDR. VBA INTO R1
(1) 024402 012702 125256                MOV   #125256, R2   ;LOAD TEST PATTERN INTO R2
(1) 024406 012704 177664                MOV   #177664, R4   ;LOAD R4 WITH PAR VALUE
(1) 024412 010437 172350                MOV   R4, KIPAR4    ;LOAD KERNEL PAR 4 BITS <15:00>
(1) 024416 011037 001176                MOV   (R0), $TMP0   ;SAVE CONTENTS AT TEST LOCATION
(1) 024422 052737 000021 177572        BIS   #21, SR0      ;TURN ON MEM. MGNT. 22-BIT ADDRESSING
(1) 024430 010211                        MOV   R2, (R1)      ;LOAD 125256 USING ADDER (PAR4 + VIRT ADDR.)
(1) 024432 000005                        RESET                      ;TURN OFF MEMORY MGNT.
(1) 024434 011003                        MOV   (R0), R3      ;READ 125256 BACK WITHOUT USING MEM. MGNT.
(1) 024436 013710 001176                MOV   $TMP0, (R0)   ;RESTORE ORIGINAL CONTENTS TO TEST LOC.
(1) 024442 020203                        CMP   R2, R3        ;WAS SAME PATTERN READ BACK THAT WAS
(1)                                     ;WRITTEN USING MEMORY MANAGEMENT?
(1) 024444 001405                        BEQ   9$            ;BRANCH IF YES
(1) 024446 010137 001306                MOV   R1, VIRT1     ;SAVE VIRTUAL ADDR. TO FORM PHYS. ADDR
(1) 024452 004737 035452                JSR   PC, FORMPA    ;GO FORM PHYSICAL ADDRESS FOR TYPING
(1) 024456 104017                        ERROR  17           ;TEST LOCATION DID NOT HAVE PATTERN
(1)                                     ;THAT SHOULD HAVE BEEN WRITTEN TO IT.
(1)                                     ;APPARENTLY PHYSICAL ADDR. WAS
(1)                                     ;FORMED WRONG BY ADDERS USING
(1)                                     ;THE VIRTUAL ADDR. AND KIPAR4
(1)                                     ;FOR TIGHTER SCOPE LOOP
(1)                                     ;REPLACE ERROR CALL WITH
(1)                                     ;'BR 8$' = 000742
(1) 024460                               9$:
1323 024460 012737 024100 001110      MOV   #1$, $LPERR    ;RESET LOOP ON ERROR POINTER TO 1$
  
```


1379	024632	012700	100100		MOV	#100100,R0	:PUT VIRTUAL ADDR. THAT USER PAR4 IN R0	
1380	024636	012701	120000		MOV	#120000,R1	:PUT VIRTUAL ADDR. THAT USES PAR5 IN R1	
1381	024642	010010		5\$:	MOV	R0,(R0)	:WRITE TO TEST LOC. USING PAR4	
1382	024644	011102			MOV	(R1),R2	:READ THE SAME LOC., BUT USING PAR5	
1383	024646	020002			CMP	R0,R2	:DID WE READ WHAT WE WROTE?	
1384	024650	001411			BEQ	6\$:BRANCH IF YES	
1385	024652	010137	001310		MOV	R1,VIRT2	:SAVE VIRTUAL ADDR. THAT SELECTED PAR5	
1386	024656	010037	001306		MOV	R0,VIRT1	:SAVE VIRTUAL ADDR. THAT SELECTED PAR4	
1387	024662	004737	035452		JSR	PC,FORMPA	:GO FORM PHYSICAL ADDRESS BEING USED	
1388	024666	104020			ERROR	20	:READING LOC. USING PAR5 AND A VIRT.	
1389							:ADDR. DID NOT FIND DATA WRITTEN WHEN USING	
1390							:PAR4 AND VIRT. ADDRESS.	
1391							:FOR TIGHTER SCOPE LOOP	
1392							:REPLACE ERROR CALL WITH	
1393							: 'BR 5\$' = 000765	
1394	024670	013700	001306		MOV	VIRT1,R0	:RESTORE VBA IN R0	
1395	024674	062700	000100	6\$:	ADD	#100,R0	:CHANGE VIRTUAL ADDRS. TO POINT TO NEXT BLOCK	
1396	024700	062701	000100		ADD	#100,R1		
1397	024704	020127	127700		CMP	R1,#127700	:WERE BLOCKS FROM 60000-676000 ALL TRIED?	
1398	024710	001354			BNE	5\$:BRANCH IF NO	
1399	024712	032737	140000	177776	BIT	#140000,PSW	:HAVE WE DONE TEST IN USER MODE YET?	
1400	024720	001026			BNE	7\$:BRANCH IF YES	
1401	024722	010437	177650		MOV	R4,UIPAR4	:LOAD USER PAR4	
1402	024726	010537	177652		MOV	R5,UIPAR5	:LOAD USER PAR5	
1403	024732	112737	000006	177610	MOVB	#6,UIPDR4	:MAP USER SPACE R/W TO TEST IT	
1404	024740	112737	000006	177612	MOVB	#6,UIPDR5		
1405	024746	105037	172310		CLRB	KIPDR4	:MAP KERNEL SPACE NON-RESIDENT WHILE	
1406	024752	105037	172312		CLRB	KIPDR5	: TESTING USER SPACE	
1407	024756	010537	172350		MOV	R5,KIPAR4	:MAP KERNEL PAR'S OPPOSITE UIPAR'S	
1408	024762	010437	172352		MOV	R4,KIPAR5		
1409	024766	012737	140000	177776	MOV	#140000,PSW	:GO TO USER MODE	
1410	024774	000713			BR	4\$:GO BACK AND READ/WRITE IN USER MODE	
1411	024776	005037	177776	7\$:	CLR	PSW	:GO BACK TO KERNEL MODE BEFORE LEAVING	
1412								
1413	025002	020537	001260		CMP	R5,@#SMADR1	:HAVE WE CHECKED ALL MEMORY?	
1414	025006	002020			BGE	8\$:BRANCH IF YES	
1415	025010	062704	000200		ADD	#200,R4	:LOAD WITH NEXT VALUE FOR PAR4	
1416	025014	062705	000200		ADD	#200,R5	:LOAD WITH NEXT VALUE FOR PAR5	
1417	025020	010437	172350		MOV	R4,KIPAR4	:LOAD KERNAL PAR4	
1418	025024	010537	172352		MOV	R5,KIPAR5	:LOAD KERNAL PAR5	
1419	025030	112737	000006	172310	MOVB	#6,KIPDR4	:MAP KERNAL SPACE R/W WHILE TESTING	
1420	025036	112737	000006	172312	MOVB	#6,KIPDR5		
1421	025044	000137	024604		JMP	10\$:CONTINUE TEST	
1422								
1423	025050	012737	077406	172310	8\$:	MOV	#77406,KIPDR4	:REMAP KERNEL PAGES READ/WRITE
1424	025056	012737	077406	172312	MOV	#77406,KIPDR5		
1425	025064	012705	000600		MOV	#600,R5		
1426	025070	010537	172350		MOV	R5,KIPAR4	:MAP KERNEL AND USER PAR'S 4 & 5	
1427	025074	010537	172352		MOV	R5,KIPAR5	: BACK TO 12-16K	
1428	025100	010537	177650		MOV	R5,UIPAR4		
1429	025104	010537	177652		MOV	R5,UIPAR5		
1430	025110	012737	002150	000250	MOV	#MGMERR,MMVEC	:RESTORE ADDR. OF NORMAL M.M. TRAP ROUTINE	
1431	025116	012737	024470	001110	MOV	#1\$,\$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$	
1432	025124	000427			BR	TST26	:GO TO NEXT TEST	
1433								
1434	025126	012637	001266	9\$:	MOV	(KSP)+,TRAPPC	:SAVE PC & PS OF TRAP	


```
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 3$' = 000763
(1) 025314 000422 BR 8$ ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
(1) 025316 012702 000010 5$: MOV #10,R2 ;SET LOOP COUNTER TO 8
(1) 025322 012700 172300 MOV #KIPDR0,R0 ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
(1) 025326 031027 000100 6$: BIT (R0),#WBIT ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
(1) 025332 001403 BEQ 7$ ;BRANCH IF YES
(1) 025334 020500 CMP R5,R0 ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
(1) 025336 001401 BEQ 7$ ;BRANCH IF YES
(1) 025340 104022 ERROR 22 ;W-BIT GOT SET IN MORE THAN ONE PDR
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 3$' = 000750
(1) 025342 062700 000002 7$: ADD #2,R0 ;POINT R0 TO NEXT PDR TO BE CHECKED
(1) 025346 077211 SOB R2,6$ ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
(1) 025350 010115 MOV R1,(R5) ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
(1) 025352 031527 000100 BIT (R5),#WBIT ;DID WRITING PDR CLEAR THE W-BIT?
(1) 025356 001401 BEQ 8$ ;BRANCH IF YES
(1) 025360 104023 ERROR 23 ;W-BIT DID NOT CLEAR BY WRITING THE PDR
(1) ;FOR TIGHTER SCOPE LOOP
(1) ;REPLACE ERROR CALL WITH
(1) ;'BR 3$' = 000740
(1) 025362 062705 000002 8$: ADD #2,R5 ;POINT R5 TO THE NEXT PDR TO BE TESTED
(1) 025366 062703 020000 ADD #20000,R3 ;CHANGE VIRT. ADDR TO REF. NEXT PDR
(1) 025372 077445 SOB R4,3$ ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
(1) 025374 012737 025206 001110 MOV #1$,$LPERR ;RESET LOOP ON ERROR POINTER TO 1$
(1) 025402 004737 035134 JSR PC,TON ;TURN T-BIT BACK ON FOR NEXT TEST
```

1460
1464

```
::*****
:*TEST 27 W-BIT LOGIC TEST, USER PDR'S
:*
```

```
:* THIS TEST WRITES TO EIGHT (8) DIFFERENT VIRTUAL ADDRESSES
:* (VBA'S = 17776,37776,57776,77776,117776,137776,157776, & 177776
:* & PBA'S CONSTRUCTED = 17776,37776,57776,77776,77776,
:* 77776,77776, & 777776 RESPECTIVELY).
:* WHICH SHOULD CAUSE THE 'W-BIT' TO SET IN EACH OF THE
:* EIGHT (8) USER PAGE DESCRIPTOR REGISTERS. THE PDR'S
:* ARE CHECKED TO SEE THAT IT'S W-BIT DOES SET WHEN THE
:* PAGE IT IS MAPPED TO IS WRITTEN TO AND THAT THE W-BIT
:* DOES NOT SET IN ANY OF THE OTHER PDR'S. USER PDR'S 3,4,5,6
:* ARE MAPPED TO 12-16K FOR THIS TEST. ALSO THE W-BIT
:* SHOULD BE CLEARED WHEN THE PDR IS WRITTEN TO. THE
:* W-BIT PORTION OF THE PDR'S IS BEING CHECKED.
```

```
(3) *****
(2) 025406 000004 TST27: SCOPE
1465 025410 012737 140000 177776 1$: MOV #140000,PSW ;GO TO USER MODE FOR THIS TEST
1466 025416 004737 035100 JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
(1) 025422 012702 000004 MOV #4,R2 ;SET LOOP COUNTER TO 4
(1) 025426 012700 177646 MOV #UIPAR3,R0 ;LOAD ADDRESS OF PAR3 INTO R0
(1) 025432 012701 000600 MOV #600,R1 ;LOAD '12-16K' PAR VALUE INTO R1
(1) 025436 010120 2$: MOV R1,(R0)+ ;MAP PARS 3-6 TO 12-16K
(1) 025440 077202 SOB R2,2$ ;LOOP TIL ALL 4 OF THEM LOADED
(1) 025442 012705 177600 MOV #UIPDR0,R5 ;LOAD ADDRESS OF FIRST PDR TO BE TESTED IN R5
(1) 025446 012704 000010 MOV #10,R4 ;SET LOOP COUNTER TO 8
(1) 025452 012703 017776 MOV #17776,R3 ;INITIALIZE VIRTUAL ADDRESS TO BE IN R3
```



```
(1) 025456 012737 025464 001110      MOV    #3$, $LPERR      ;SET LOOP ON ERROR POINTER TO 3$
(1) 025464 012700 177600      3$:  MOV    #UIPDRO, R0      ;LOAD ADDR. OF FIRST PDR TO BE SETUP IN R0
(1) 025470 012702 000010      MOV    #10, R2          ;SET LOOP COUNTER TO 8
(1) 025474 012701 077406      MOV    #77406, R1       ;PUT 'W-BIT OFF DATA' INTO R1
(1) 025500 010120      4$:  MOV    R1, (R0)+        ;CLEAR ALL W-BITS BY WRITING TO ALL PDRS
(1) 025502 077202      SOB    R2, 4$          ;LOOP UNTIL ALL OF THEM SETUP
(1) 025504 011313      MOV    (R3), (R3)       ;DO 'DATO' TO VIRTUAL ADDR.-SETTING A W-BIT
(1) 025506 031527 000100      BIT    (R5), #WBIT      ;DID THAT CAUSE W-BIT TO BE SET?
(1) 025512 001002      BNE    5$              ;BRANCH IF YES
(1) 025514 104021      ERROR  21              ;W-BIT DID NOT GET SET IN PDR
(1)                                     ;FOR TIGHTER SCOPE LOOP
(1)                                     ;REPLACE ERROR CALL WITH
(1)                                     ;'BR 3$' = 000763
(1) 025516 000422      BR     8$              ;SKIP CHECKING OTHER PDR'S-ERROR WILL SET W-BITS
(1) 025520 012702 000010      5$:  MOV    #10, R2          ;SET LOOP COUNTER TO 8
(1) 025524 012700 177600      MOV    #UIPDRO, R0      ;LOAD ADDR. OF FIRST PDR TO BE CHECKED IN R0
(1) 025530 031027 000100      6$:  BIT    (R0), #WBIT      ;DID W-BIT IN OTHER PDRS REMAIN CLEAR?
(1) 025534 001403      BEQ    7$              ;BRANCH IF YES
(1) 025536 020500      CMP    R5, R0          ;IF W-BIT SET, THEN WAS IT PDR UNDER TEST?
(1) 025540 001401      BEQ    7$              ;BRANCH IF YES
(1) 025542 104022      ERROR  22              ;W-BIT GOT SET IN MORE THAN ONE PDR
(1)                                     ;FOR TIGHTER SCOPE LOOP
(1)                                     ;REPLACE ERROR CALL WITH
(1)                                     ;'BR 3$' = 000750
(1) 025544 062700 000002      7$:  ADD    #2, R0          ;POINT R0 TO NEXT PDR TO BE CHECKED
(1) 025550 077211      SOB    R2, 6$          ;LOOP UNTIL ALL 8 CHECKED FOR CLEAR W-BIT
(1) 025552 010115      MOV    R1, (R5)        ;WRITE TO THE PDR TESTED TO CLEAR W-BIT
(1) 025554 031527 000100      BIT    (R5), #WBIT      ;DID WRITING PDR CLEAR THE W-BIT?
(1) 025560 001401      BEQ    8$              ;BRANCH IF YES
(1) 025562 104023      ERROR  23              ;W-BIT DID NOT CLEAR BY WRITING THE PDR
(1)                                     ;FOR TIGHTER SCOPE LOOP
(1)                                     ;REPLACE ERROR CALL WITH
(1)                                     ;'BR 3$' = 000740
(1) 025564 062705 000002      8$:  ADD    #2, R5          ;POINT R5 TO THE NEXT PDR TO BE TESTED
(1) 025570 062703 020000      ADD    #20000, R3       ;CHANGE VIRT. ADDR TO REF. NEXT PDR
(1) 025574 077445      SOB    R4, 3$          ;LOOP BACK TO 3$ UNTIL ALL 8 PDR'S TESTED
(1) 025576 012737 025410 001110      MOV    #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
(1) 025604 004737 035134      JSR    PC, TON          ;TURN T-BIT BACK ON FOR NEXT TEST
1467 025610 005037 177776      CLR    PSW             ;BACK TO KERNEL MODE BEFORE LEAVING
```

```
1468
1478
(3) *****
(4) *TEST 30          TEST 'W-BIT' SPECIAL CASES
(4) *
(4) * THIS TEST CHECKS TWO SPECIAL CASES OF THE W-BIT. FIRST CASE IS
(4) * THAT THE W-BIT SHOULD NOT SET IN PDR 7 WHEN WRITING TO
(4) * STATUS REG SRO (KERNEL PDR 7 IS USED). SECOND CASE IS THAT
(4) * THE W-BIT IS STILL SET IF THE 'DATO' IS ABORTED DUE TO A
(4) * TIMEOUT ERROR (KERNEL PDR6 & VIRTUAL ADDR 140000 ARE USED).
(4) *
(3) *****
```

```
(2) 025614 000004      TST30: SCOPE
1479
1480 025616 004737 035100      1$:  JSR    PC, TOFF        ;TURN OFF T-BIT TRAPPING FOR THIS TEST
1481 025622 012701 077406      MOV    #77406, R1       ;PUT 'W-BIT OFF' VALUE FOR PDR IN R1
1482 025626 012737 025634 001110      MOV    #2$, $LPERR      ;SET LOOP ON ERROR POINTER TO 2$
1483
```

```

1484 025634 010137 172316      2$:  MOV    R1,KIPDR7      ;LOAD KERNEL PDR 7 TO CLEAR W-BIT
1485 025640 013700 177572      MOV    SR0,R0          ;READ PRESENT CONTENTS OF STATUS REG. 0
1486 025644 010037 177572      MOV    R0,SR0          ;WRITE PRESENT CONTENTS OF SR0 BACK TO ITSELF
1487 025650 013702 172316      MOV    KIPDR7,R2       ;READ CONTENTS OF KIPDR7 INTO R2
1488 025654 020102              CMP    R1,R2           ;WAS W-BIT LEFT CLEARED?
1489 025656 001401              BEQ    3$              ;BRANCH IF YES
1490 025660 104024              ERROR  24             ;W-BIT IN KIPDR7 SET WHEN SR0 WAS WRITTEN TO
1491                                     ;FOR TIGHTER SCOPE LOOP
1492                                     ;REPLACE ERROR CALL WITH
1493                                     ;'BR 2$' = 000765
1494 025662 012737 025662 001110 3$:  MOV    #3$,SLPERR      ;SET LOOP ON ERROR POINTER TO 3$
1495 025670 010137 172314      MOV    R1,KIPDR6       ;LOAD KERNEL PDR6 WITH 77406 TO CLEAR W-BIT
1496 025674 012737 025706 000004  MOV    #4$,ERRVEC      ;SET UP LOC. 4 TO 4$ FOR ODD ADDR. ABORT
1497 025702 005037 140000      CLR    @#140000        ;CAUSE TIMEOUT ABORT THRU LOC. 4
1498 025706 012706 001100      4$:  MOV    #KERSTK,KSP     ;RESTORE THE STACK POINTER
1499 025712 013702 172314      MOV    KIPDR6,R2       ;READ KIPDR6 INTO R2
1500 025716 052701 000100      BIS    #100,R1         ;R1-77506
1501 025722 020102              CMP    R1,R2           ;WAS W-BIT SET?
1502 025724 001401              BEQ    5$              ;BRANCH IF YES
1503 025726 104025              ERROR  25             ;W-BIT WAS NOT SET DURING A TIMEOUT ABORT
1504                                     ;FOR TIGHTER SCOPE LOOP
1505                                     ;REPLACE ERROR CALL WITH
1506                                     ;'BR 3$' = 000757
1507 025730 010137 172314      5$:  MOV    R1,KIPDR6       ;RESTORE KIPDR6 TO 77406
1508 025734 012737 001400 172354  MOV    #1400,KIPAR6    ;RESTORE KIPAR6 TO 1400
1509 025742 012737 002076 000004  MOV    #TIMERR,ERRVEC  ;RESTORE NORMAL CPU TRAP ROUTINE TO LOC.4
1510 025750 012737 025616 001110  MOV    #1$,SLPERR      ;RESET LOOP ON ERROR POINTER TO 1$
1511 025756 004737 035134      JSR    PC,TON          ;TURN T-BIT TRAPPING BACK ON

```

```

1512
1513 :*****
1514 :*
1515 :* THE NEXT THREE (3) TESTS CAUSE MEMORY MANAGEMENT ERRORS
1516 :* TO CHECK THE ABILITY OF STATUS REGISTER 0 TO RECORD KT
1517 :* ERRORS AND THE ABILITY OF STATUS REGISTER 2 TO LOCK UP THE
1518 :* VIRTUAL ADDR. OF THE INSTRUCTION THAT CAUSED THE ERROR.
1519 :* THE BITS OF SR2 ARE CHECKED AND BITS <15:13>, <6:5>, AND <3:0>
1520 :* ARE CHECKED IN SR0. SO THE SR0 AND SR2 LOGIC AND THE
1521 :* KT ERROR LOGIC ARE CHECKED.
1522 :*
1523 :*****

```

```

1533 :*****
(3) :*TEST 31 NON-RESIDENT ABORT TEST (ACF=0&4)
(4) :*
(4) :* THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
(4) :* LOGIC BY CAUSING NON-RESIDENT ABORTS IN BOTH KERNEL AND
(4) :* USER MODES. PDR 4 IS LOADED WITH ACF'S = 0&4 AND
(4) :* THEN PHYSICAL ADDR. 60000 IS ACCESSED TO CAUSE THE ABORT.
(4) :*
(3) :*****

```

```

(2) 025762 000004 TST31: SCOPE
1534
1535 025764 012700 000600 1$:  MOV    #600,R0        ;LOAD DATA FOR PAR'S INTO R0
1536 025770 010037 172346      MOV    R0,KIPAR3      ;MAP KERNEL PAR'S 3&4 TO 12-16K
1537 025774 010037 172350      MOV    R0,KIPAR4
1538 026000 010037 177646      MOV    R0,UIPAR3      ;MAP USER PAR'S 3&4 TO 12-16K

```


1539	026004	010037	177650			MOV	R0,UIPAR4				
1540	026010	012737	077406	172306		MOV	#77406,KIPDR3				:MAP KERNEL PDR 3 128 BLKS, READ-WRITE
1541	026016	012737	077406	177606		MOV	#77406,UIPDR3				:MAP USER PDR 3 128 BLKS, READ-WRITE
1542	026024	012700	060000			MOV	#60000,R0				:LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
1543	026030	012701	100000			MOV	#100000,R1				:LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
1544	026034	012703	100011			MOV	#100011,R3				:LOAD R3 WITH WHAT SRO SHOULD READ - N.R., KERNEL, PG.4
1545	026040	012702	077400			MOV	#77400,R2				:LOAD ACF=0 (NON-RESIDENT) PDR VALUE IN R2
1546	026044	012737	026106	000250	2\$:	MOV	#5\$,MMVEC				:POINT MEM. MGMT. TRAP VECTOR TO 5\$ BELOW
1547	026052	010237	172310			MOV	R2,KIPDR4				:LOAD ACF TEST VALUE INTO KIPDR4
1548	026056	010237	177610			MOV	R2,UIPDR4				:LOAD ACF TEST VALUE INTO UIPDR4
1549	026062	012737	026070	001110		MOV	#3\$,SLPERR				:SET LOOP ON ERROR POINTER TO 3\$
1550	026070	005010			3\$:	CLR	(R0)				:CLEAR PHYS. LOC. 60000 USING PDR3
1551	026072	013737	177776	001176		MOV	PSW,\$TMP0				:SAVE PSW IN CASE OF ERROR
1552	026100	005211			4\$:	INC	(R1)				:TRY TO REF. IT USING PDR4 - SHOULD TRAP TO 5\$
1553	026102	104026				ERROR	26				:MEM. MGMT. ABORT DID NOT OCCUR
1554											:FOR TIGHTER SCOPE LOOP
1555											:REPLACE ERROR CALL WITH
1556											: 'BR 3\$' = 000772
1557	026104	000425				BR	8\$:BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
1558	026106	062706	000004		5\$:	ADD	#4,SP				:RESTORE STACK POINTER
1559	026112	005710				TST	(R0)				:DID INSTRUCTION GET ABORTED & NOT EXECUTE
1560	026114	001401				BEQ	6\$:BRANCH IF YES
1561	026116	104027				ERROR	27				:INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
1562											:FOR TIGHTER SCOPE LOOP
1563											:REPLACE ERROR CALL WITH
1564											: 'BR 3\$' = 000764
1565	026120	013737	177572	001272	6\$:	MOV	SRO,WASSRO				:READ STATUS REGISTER 0
1566	026126	013737	177576	001274		MOV	SR2,WASSR2				:READ STATUS REGISTER 2
1567	026134	020337	001272			CMP	R3,WASSRO				:DID SRO REPORT NON-RESIDENT ERROR CORRECTLY?
1568	026140	001401				BEQ	7\$:BRANCH IF YES
1569	026142	104030				ERROR	30				:SRO DID NOT REPORT NON-RES. ERROR CORRECTLY
1570											:FOR TIGHTER SCOPE LOOP
1571											:REPLACE ERROR CALL WITH
1572											: 'BR 3\$' = 000752
1573	026144	012704	026100		7\$:	MOV	#4\$,R4				:LOAD R4 WITH WHAT SR2 SHOULD READ
1574	026150	020437	001274			CMP	R4,WASSR2				:DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4\$)?
1575	026154	001401				BEQ	8\$:BRANCH IF YES
1576	026156	104031				ERROR	31				:SR2 DID NOT LOCK VIRTUAL ADDR. OF NON-RES. ERROR
1577											:FOR TIGHTER SCOPE LOOP
1578											:REPLACE ERROR CALL WITH
1579											: 'BR 3\$' = 000744
1580	026160	042737	160000	177572	8\$:	BIC	#160000,SRO				:CLEAR THE ERROR BITS IN SRO
1581	026166	032737	140000	001176		BIT	#140000,\$TMP0				:HAS ACF=084 BEEN TESTED IN USER YET
1582	026174	001006				BNE	9\$:BRANCH IF YES
1583	026176	012703	100151			MOV	#100151,R3				:LOAD R3 WITH WHAT SRO SHOULD READ - N.R., USER, PG.4
1584	026202	012737	140000	177776		MOV	#140000,PSW				:GO TO USER MODE
1585	026210	000715				BR	2\$:REPEAT TEST IN USER MODE
1586	026212	022702	077404		9\$:	CMP	#77404,R2				:HAS ACF=4 BEEN TESTED YET?
1587	026216	001407				BEQ	10\$:BRANCH IF YES
1588	026220	012702	077404			MOV	#77404,R2				:THEN LOAD ACF=4 (NON-RES) PDR VALUE IN R2
1589	026224	012703	100011			MOV	#100011,R3				:LOAD R3 WITH WHAT SRO SHOULD READ-N.R.,KERNEL,PG. 4
1590	026230	005037	177776			CLR	PSW				:GO BACK TO KERNEL MODE
1591	026234	000703				BR	2\$:GO BACK & TEST ACF=4 IN SAME MODE
1592	026236	005037	177776		10\$:	CLR	PSW				:GO BACK TO KERNEL MODE BEFORE LEAVING
1593	026242	012737	025764	001110		MOV	#1\$,SLPERR				:RESET LOOP ON ERROR POINTER TO 1\$
1594	026250	012737	002150	000250		MOV	#MGMERR,MMVEC				:RESTORE ADDRESS OF NORMAL MEMORY

;MANAGEMENT ERROR ROUTINE TO MMVEC

1595
1596
1605
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(3)

:TEST 32 READ-ONLY ABORT TEST (ACF=2)
:*****
: THIS TEST CHECKS THE ACCESS CONTROL FIELD (ACF) COMPARATOR
: LOGIC BY CAUSING READ-ONLY ABORTS IN BOTH KERNEL AND
: USER MODES. PDR 4 IS LOAD WITH ACF=2 AND THEN
: PHYSICAL ADDR. 60000 IS WRITTEN TO CAUSE THE ABORT.
:*****

(2) 026256 000004
1606 026260
1607
1608 026260 012700 060000
1609 026264 012701 100000
1610 026270 012703 020011
1611 026274 012702 077402
1612 026300 012737 026342 000250 2\$:
1613 026306 010237 172310
1614 026312 010237 177610
1615 026316 012737 026324 001110
1616 026324 005010 3\$:
1617 026326 013737 177776 001176
1618 026334 005211 4\$:
1619 026336 104026
1620
1621
1622

TST32: SCOPE
1\$:

:KERNEL & USER PAR'S 3 & 4 AND PDR 3
:ARE SETUP FROM LAST TEST
:LOAD VIRTUAL ADDR. TO REFERENCE PDR3 INTO R0
:LOAD VIRTUAL ADDR. TO REFERENCE PDR4 INTO R1
:LOAD R3 WITH WHAT SR0 SHOULD READ - R/O, KERNEL, PG.4
:LOAD ACF=2 (READ-ONLY) PDR VALUE IN R2
:POINT MEM. MGMT. TRAP VECTOR TO 5\$ BELOW
:LOAD ACF=2 INTO KIPDR4
:LOAD ACF=2 INTO UIPDR4
:SET LOOP ON ERROR POINTER TO 3\$
:CLEAR PHYS. LOC. 60000 USING PDR3
:SAVE PSW IN CASE OF ERROR
:TRY TO WRITE USING PDR4 - SHOULD TRAP TO 5\$
:MEM. MGMT. ABORT DID NOT OCCUR
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH

1623 026340 000425
1624 026342 062706 000004 5\$:
1625 026346 005710
1626 026350 001401
1627 026352 104027
1628
1629
1630
1631 026354 013737 177572 001272 6\$:
1632 026362 013737 177576 001274
1633 026370 020337 001272
1634 026374 001401
1635 026376 104030
1636
1637
1638

BR 8\$
ADD #4,SP
TST (R0)
BEQ 6\$
ERROR 27
MOV SR0,WASSR0
MOV SR2,WASSR2
CMP R3,WASSR0
BEQ 7\$
ERROR 30

:MEM. MGMT. ABORT DID NOT OCCUR
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
: 'BR 3\$' = 000772
:BRANCH AROUND STATUS REG. CHECKS IF NO ABORT
:RESTORE STACK POINTER
:DID INSTRUCTION GET ABORTED & NOT EXECUTE
:BRANCH IF YES
:INSTRUCTION WAS NOT ABORTED, LOC. GOT CHANGED
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
: 'BR 3\$' = 000764
:READ STATUS REG. 0
:READ STATUS REG. 2
:DID SR0 REPORT READ-ONLY ERROR CORRECTLY?
:BRANCH IF YES
:SR0 DID NOT REPORT R/O ERROR CORRECTLY
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
: 'BR 3\$' = 000752

1639 026400 012704 026334 7\$:
1640 026404 020437 001274
1641 026410 001401
1642 026412 104031
1643
1644
1645
1646 026414 042737 160000 177572 8\$:
1647 026422 032737 140000 001176
1648 026430 001006
1649 026432 012703 020151

MOV #4\$,R4
CMP R4,WASSR2
BEQ 8\$
ERROR 31
BIC #160000,SR0
BIT #140000,\$TMP0
BNE 9\$
MOV #20151,R3

: 'BR 3\$' = 000752
:LOAD R4 WITH WHAT SR2 SHOULD READ
:DID SR2 LOCKUP RIGHT VIRTUAL ADDR. (=4\$)?
:BRANCH IF YES
:SR2 DID NOT LOCKUP VIRTUAL ADDR. OF R/O ERROR
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
: 'BR 3\$' = 000744
:CLEAR THE ERROR BITS IN SR0
:HAS ACF=2 BEEN TESTED IN USER MODE?
:BRANCH IF YES
:LOAD R3 WITH WHAT SR0 SHOULD READ-R/O, USER, PG.4

1650 026436 012737 140000 177776
1651 026444 000715
1652 026446 005037 177776
1653 026452 012737 026260 001110
1654 026460 012737 002150 000250
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
1696
1697

9\$: MOV #140000,PSW ;GO TO USER MODE
BR 2\$;REPEAT TEST IN USER MODE
CLR PSW ;GO BACK TO KERNEL MODE BEFORE LEAVING
MOV #1\$, \$LPERR ;RESET LOOP ON ERROR POINTER TO 1\$
MOV #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
;MANAGEMENT ERROR ROUTINE TO MMVEC.

*
* THE NEXT TWO (2) TESTS WILL BE CHECKING THE PAGE LENGTH
* COMPARATORS AND SOME MORE OF THE KT ERROR DETECTION
* AND STATUS LOGIC. THE PAGE LENGTH FIELD (PLF) IN KERNEL
* PDR 4 IS VARIED AND FOR EVERY PLF, THREE (3) VIRTUAL
* ADDRESSES ARE READ. WHILE USING BOTH UPWARD & DOWNWARD PAGE
* EXPANSION, ONE OF THOSE THREE VIRTUAL ADDRESSES WILL CAUSE A
* 'PAGE LENGTH ABORT' WHILE THE OTHER TWO WON'T.
*
* STATUS REGISTER 0 & 2 ARE CHECKED WHEN THE PAGE LENGTH
* ABORT DOES OCCUR TO SEE THAT THE ABORT IS REPORTED AND THAT
* THE VIRTUAL ADDRESS OF THE INSTRUCTION THAT CAUSED THE ABORT
* IS LOCKED UP.
*

(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 026466 000004
1698 026470 012737 077406 172306
1699 026476 012737 077406 172312
1700 026504 012700 026764
1701 026510 012704 027002
1702 026514 012701 000006
1703 026520 012737 026676 000250
1704 026526 012737 026540 001110

*TEST 33 PAGE LENGTH FAULTS-UPWARD EXPANSION
*
* THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR 4
* FROM 1 TO 177 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
* ARE ACCESSED. WHEN VBA <12:6> IS LESS THAN OR EQUAL TO PDR <14:8>
* NO ABORT SHOULD OCCUR. WHEN VBA <12:6> IS GREATER THAN PDR <14:8>,
* A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
* THE PAGE EXPANSION DIRECTION IN THIS TEST IS UPWARD, (THE ED BIT
* (BIT 3) OF PDR 4 = 0).
*

TST33: SCOPE
1\$: MOV #77406,KIPDR3 ;MAKE SURE PDR3 IS DESCRIBED AS R/W
MOV #77406,KIPDR5 ;MAKE SURE PDR5 IS DESCRIBED AS R/W
MOV #DALTB1,R0 ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.
MOV #PDRTB1,R4 ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).
MOV #6,R1 ;SET UP LOOP COUNTER.
MOV #9\$,MMVEC ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
MOV #2\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 2\$

```

1705 026534 012706 001100          MOV    #KERSTK,KSP    ;MAKE SURE STACK POINTER IS ALL SET UP
1706
1707          ;TEST NON-ABORT CASES (VBA < OR = PLF)
1708 026540 012437 172310          2$:   MOV    (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
1709 026544 005730          TST    @ (R0)+        ;ACCESS VIRTUAL ADDR. (VBA < OR = PLF)
1710
1711 026546 077104          SOB    R1,2$         ;NO ABORT SHOULD OCCUR!!!
1712
1713          ;TEST ABORT CASES (VBA > PLF)
1714 026550 012701 000005          3$:   MOV    #5,R1        ;SET UP LOOP COUNTER.
1715 026554 012700 027020          MOV    #DALTB2,R0    ;DAL TABLE
1716 026560 012704 027034          MOV    #PDRTB2,R4    ;PDR TABLE
1717 026564 012737 026600 001110          MOV    #4$,$LPERR    ;SET LOOP ON ERROR POINTER TO 4$
1718 026572 012737 026612 000250          MOV    #6$,MMVEC     ;SETUP M.M. TRAP VECTOR FOR EXPECTED ABORT
1719
1720 026600 012437 172310          4$:   MOV    (R4)+,KIPDR4 ;LOAD KIPDR4 WITH PAGE LENGTH VALUE
1721 026604 005730          5$:   TST    @ (R0)+        ;ACCESS VIRTUAL ADDR. (VBA > PLF - ABORT TO 6$)
1722 026606 104033          ERROR  33            ;EXPECTED PAGE LENGTH ABORT DID NOT OCCUR
1723
1724          ;FOR TIGHTER SCOPE LOOP
1725          ;REPLACE ERROR CALL WITH
1726          ;'BR 5$' = 000776
1726 026610 000424          BR     8$            ;BRANCH AROUND ABORT CHECKS
1727 026612 012706 001100          6$:   MOV    #KERSTK,KSP  ;RESTORE STACK POINTER FOLLOWING ABORT
1728 026616 013737 177572 001272          MOV    SR0,WASSR0    ;READ M.M. STATUS REG. 0
1729 026624 013737 177576 001274          MOV    SR2,WASSR2    ;READ M.M. STATUS REG. 2
1730 026632 012702 040011          MOV    #40011,R2     ;PUT EXPECTED SR0 CONTENTS IN R2
1731 026636 020237 001272          CMP    R2,WASSR0     ;DID SR0 REPORT PG. LENGTH ABORT, PAGE 4, KERNEL?
1732 026642 001401          BEQ    7$            ;BRANCH IF YES
1733 026644 104034          ERROR  34            ;SR0 DID NOT REPORT PG. LENGTH ABORT CORRECTLY
1734
1735          ;FOR TIGHTER SCOPE LOOP
1736          ;REPLACE ERROR CALL WITH
1737          ;'BR 5$' = 000757
1737 026646 012703 026604          7$:   MOV    #5$,R3        ;PUT EXPECTED SR2 CONTENTS IN R3
1738 026652 020337 001274          CMP    R3,WASSR2     ;DID SR2 LOCKUP VIRT. ADDR. OF ABORTED INSTRUCTION?
1739 026656 001401          BEQ    8$            ;BRANCH IF YES
1740 026660 104035          ERROR  35            ;SR2 DID NOT LOCKUP VIRT. ADDR. OF ABORT CORRECTLY
1741
1742          ;FOR TIGHTER SCOPE LOOP
1743          ;REPLACE ERROR CALL WITH
1744          ;'BR 5$' = 000751
1744 026662 042737 160000 177572          8$:   BIC    #160000,SR0   ;CLEAR ERROR BITS IN SR0
1745 026670 077135          SOB    R1,4$         ;DONE?...NO - GET NEXT DAL & PDR PAIR
1746 026672 000137 026744          JMP    10$           ;YES...
1747 026676 012637 001266          9$:   MOV    (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
1748 026702 012637 001270          MOV    (KSP)+,TRAPPS
1749 026706 013737 177572 001272          MOV    SR0,WASSR0    ;SAVE CONTENTS OF SR0 FOR ERROR
1750 026714 013737 177576 001274          MOV    SR2,WASSR2    ;SAVE CONTENTS OF SR2 FOR ERROR
1751 026722 042737 160000 177572          BIC    #160000,SR0   ;CLEAR ERROR BITS IN SR0
1752 026730 104032          ERROR  32            ;GOT PG. LENGTH ABORT BEFORE IT WAS EXPECTED
1753
1754          ;FOR TIGHTER SCOPE LOOP
1755          ;REPLACE ERROR CALL WITH
1756          ;A 'NOP' = 000240
1756 026732 013746 001270          MOV    TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
1757 026736 013746 001266          MOV    TRAPPC,-(KSP)
1758 026742 000002          RTI                    ;RETURN FROM UNEXPECTED ABORT
1759
1760 026744 012737 026470 001110          10$:  MOV    #1$,$LPERR    ;RESET LOOP ON ERROR POINTER TO 1$

```



```
1761 026752 012737 002150 000250      MOV      #MGMERR,MMVEC      ;RESTORE NORMAL M.M. TRAP HANDLER
1762                                     ;ADDRESS TO M.M. TRAP VECTOR
1763 026760 000137 027050      JMP      TST34
```

```
1764
1765      ;DAL TABLE FOR UPWARD EXPANSION (NON-ABORT CASES)
1766 026764 100000      DALTB1: 100000
1767 026766 106100      106100
1768 026770 102300      102300
1769 026772 102500      102500
1770 026774 113700      113700
1771 026776 104600      104600
1772 027000 117700      117700
```

```
1773
1774      ;PDR TABLE FOR KPDR4 (NON-ABORT CASES)
1775 027002 000006      PDRTB1: 000006
1776 027004 052006      052006
1777 027006 045006      045006
1778 027010 052006      052006
1779 027012 074406      074406
1780 027014 025006      025006
1781 027016 077406      077406
```

```
1782
1783      ;DAL TABLE (ABORT CASES)
1784 027020 100100      DALTB2: 100100
1785 027022 110100      110100
1786 027024 116600      116600
1787 027026 112700      112700
1788 027030 117000      117000
1789 027032 117700      117700
```

```
1790
1791      ;PDR TABLE (ABORT CASES)
1792 027034 000006      PDRTB2: 000006
1793 027036 030406      030406
1794 027040 046406      046406
1795 027042 042006      042006
1796 027044 073406      073406
1797 027046 077006      077006
```

```
1810
1811      ;*****
      ;*TEST 34      PAGE LENGTH FAULTS-DOWNWARD EXPANSION
      ;*
      ;*      THIS TEST VARIES THE PAGE LENGTH FIELD (PLF) IN KERNEL PDR4
      ;*      FROM 176 TO 0 AND FOR EACH PLF, THREE VIRTUAL ADDRESSES (VBA'S)
      ;*      ARE ACCESSED. WHEN VBA <12:6> IS GREATER THAN OR EQUAL TO PDR <14:8>
      ;*      NO PAGE ABORT SHOULD OCCUR. WHEN VBA <12:6> IS LESS THAN PDR <14:8>
      ;*      A PAGE LENGTH ABORT SHOULD OCCUR AND BE REPORTED BY SRO & SR2.
      ;*      THE PAGE EXPANSION DIRECTION IN THIS TEST IS DOWNWARD, (THE ED BIT
      ;*      (BIT 3) OF PDR4=1).
      ;*
      ;*****
```

```
1812 027050 000004      TST34: SCOPE
1813 027052 012700 027332 1$: MOV      #DALTB3,R0      ;DAL TABLE FOR VIRTUAL ADDR'S. TO SELECT PDR4.
1814 027056 012704 027350      MOV      #PDRTB3,R4      ;PDR TABLE FOR PDR4 (COINCIDES WITH DAL TABLE).
1815 027062 012701 000006      MOV      #6,R1           ;SET UP LOOP COUNTER.
      MOV      #9$,MMVEC      ;SETUP M.M. TRAP VECTOR FOR UNEXPECTED ABORTS
```



```

1927 027420 012737 000600 172346 1$: MOV #600,KIPAR3 ;BE SURE PAR3 IS MAPPED TO 12-16k
1928 027426 012737 000600 172350 MOV #600,KIPAR4 ;BE SURE PAR4 IS MAPPED TO 12-16k
1929 027434 012737 077406 172306 MOV #77406,KIPDR3 ;MAP PAGE 3 128 BLOCKS, R/W
1930 027442 012737 077402 172310 MOV #77402,KIPDR4 ;MAP PAGE 4 128 BLOCKS, READ-ONLY
1931 027450 012700 060002 MOV #60002,R0 ;LOAD R0 WITH VIRTUAL ADDR. WHICH USES PDR3
1932 027454 012701 100002 MOV #100002,R1 ;LOAD R1 WITH VIRTUAL ADDR. WHICH USES PDR4
1933 027460 012737 027514 000250 MOV #3$,MMVEC ;SET M.M. TRAP VECTOR TO 3$
1934 027466 012737 027474 001110 MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
1935 027474 012720 010727 2$: MOV #010727,(R0)+ ;LOAD 'MOV PC,(PC)+' INSTRUCTION AT ADDR.
1936 027500 005020 CLR (R0)+ ; REACHED THRU PDR/PAR 4.
1937 027502 012720 000137 MOV #000137,(R0)+ ;LOAD 'JMP @#3$' INSTRUCTION AT VIRT. ADDR.
1938 027506 012710 027514 MOV #3$,(R0) ; IN CASE R/O VIOL. DOES NOT ABORT
1939 027512 010107 MOV R1,PC ;TRANSFER PROGRAM EXECUTION TO 'PAGE 4 INSTRUCTIONS'
1940 027514 012706 001100 3$: MOV #KERSTK,KSP ;RESTORE STACK POINTER
1941 027520 013737 177576 001274 MOV SR2,WASSR2 ;READ CONTENTS OF STATUS REG 2
1942 027526 020137 001274 CMP R1,WASSR2 ;WAS ADDR. OF 'RELOCATED - R/O ABORT' LOCKED UP?
1943 027532 001401 BEQ 4$ ;BRANCH IF YES
1944 027534 104036 ERROR 36 ;SR2 DID NOT LOCK UP VIRTUAL ADDR. OF R/O VIOL.
1945 ;FOR TIGHTER SCOPE LOOP
1946 ;REPLACE ERROR CALL WITH
1947 ;'BR 2$' = 000757
1948 027536 042737 160000 177572 4$: BIC #160000,SRO ;CLEAR THE ERROR BITS IN SRO
1949 027544 060101 ADD R1,R1 ;SETUP TO FORM NEXT VIRTUAL ADDRESS
1950 027546 010100 MOV R1,R0 ;SETUP R0 TO FORM NEXT VIRT. ADDR. TO LOAD
1951 027550 052701 100000 BIS #100000,R1 ;FORM VIRTUAL ADDR. THAT SHOULD BE LOCKED UP NEXT
1952 027554 052700 060000 BIS #60000,R0 ;POINT R0 TO NEXT VIRT. ADDR. TO LOAD
1953 027560 020127 110000 CMP R1,#110000 ;HAVE ALL VBA'S 100000-110000 BEEN TESTED?
1954 027564 101743 BLOS 2$ ;BRANCH IF NO
1955
1956 027566 012737 027420 001110 5$: MOV #1$,SLPERR ;RESET LOOP ON ERROR POINTER TO 1$
1957 027574 012737 077406 172310 MOV #77406,KIPDR4 ;RESTORE PDR4 TO R/W ACCESS
1958 027602 012737 002150 000250 MOV #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL M.M.
1959 ;TRAP HANDLER TO M.M. VECTOR
1960
1974
1975

```

```

(3) ;*****
(4) ;*TEST 36 MORE CHECKS OF SRO & SR2
(4) ;*
(4) ;* THIS TEST PERFORMS SOME ADDITIONAL CHECKS OF THE SRO & SR2 LOGIC.
(4) ;* FIRST IT CHECKS THAT SR2 'TRACKS' ALONG ACTING AS A VIRTUAL ADDRESS
(4) ;* PROGRAM COUNTER. ALSO SRO & SR2 ARE LOCKED UP BY A PAGE LENGTH
(4) ;* ABORT, THEN WITHOUT CLEARING SRO'S ERROR BITS, A R/O ABORT IS CAUSED.
(4) ;* SRO & SR2 SHOULD NOT BE CHANGED BY THE SECOND ABORT AND THE
(4) ;* INFORMATION ABOUT THE PAGE LENGTH ABORT SHOULD STILL BE LOCKED UP.
(4) ;* IN ADDITION A 'RESET' IS EXECUTED TO VERIFY THAT SRO IS CLEARED
(4) ;* AND SR2 IS UNLOCKED BY A RESET. AFTER MEMORY MANAGEMENT IS TURNED BACK ON,
(4) ;* SR2 IS CHECKED TO SEE THAT IT IS TRACKING AGAIN.
(4) ;*
(3) ;*****

```

```

(2) 027610 000004 TST36: SCOPE
1976 027612 012737 000600 172352 1$: MOV #600,KIPAR5 ;MAP KERNEL PAGE 5 TO 12-16K
1977 027620 012737 000406 172310 MOV #406,KIPDR4 ;SETUP PDR4 FOR PAGE LENGTH ABORT
1978 027626 012737 077402 172312 MOV #77402,KIPDR5 ;SETUP PDR5 FOR R/O ABORT
1979 027634 012737 027642 001110 MOV #2$,SLPERR ;SET LOOP ON ERROR POINTER TO 2$
1980 027642 013737 177576 001274 2$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING
1981 027650 012701 027642 MOV #2$,R1 ;PUT EXPECTED VIRTUAL PC IN R1

```


1982	027654	020137	001274			CMP	R1,WASSR2	:DID SR2 CONTAIN VIRTUAL PC AT 2\$?
1983	027660	001401				BEQ	3\$:BRANCH IF YES
1984	027662	104041				ERROR	41	:SR2 NOT TRACKING CORRECTLY
1985								:FOR TIGHTER SCOPE LOOP
1986								:REPLACE ERROR CALL WITH
1987								: 'BR 2\$' = 000767
1988	027664	012737	027672	001110	3\$:	MOV	#4\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 4\$
1989	027672	013737	177576	001274	4\$:	MOV	SR2,WASSR2	:READ SR2 TO SEE IF ITS TRACKING
1990	027700	012701	027672			MOV	#4\$,R1	:PUT EXPECTED VIRTUAL PC IN R1
1991	027704	020137	001274			CMP	R1,WASSR2	:DID SR2 CONTAIN VIRTUAL PC AT 4\$
1992	027710	001401				BEQ	5\$:BRANCH IF YES
1993	027712	104041				ERROR	41	:SR2 NOT TRACKING CORRECTLY
1994								:FOR TIGHTER SCOPE LOOP
1995								:REPLACE ERROR CALL WITH
1996								: 'BR 4\$' = 000767
1997	027714	012737	027722	001110	5\$:	MOV	#6\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 6\$
1998	027722	012737	027740	000250	6\$:	MOV	#7\$,MMVEC	:PUT ADDRESS OF 7\$ IN M.M. TRAP VECTOR
1999	027730	005037	001200			CLR	\$TMP1	:CLEAR ERROR INDICATOR
2000	027734	005237	100500			INC	@#100500	:CAUSE PAGE LENGTH ABORT - TRAP TO 7\$
2001	027740	012706	001100		7\$:	MOV	#KERSTK,KSP	:RESTORE STACK POINTER AFTER ABORT
2002	027744	013737	177572	001176		MOV	SRO,\$TMP0	:SAVE SRO'S INFORMATION ON PG. LGTH. ABORT
2003	027752	013737	177576	001202		MOV	SR2,\$TMP2	:SAVE SR2'S INFORMATION ON PG. LGTH. ABORT
2004	027760	012737	027772	000250		MOV	#8\$,MMVEC	:PUT ADDRESS OF 8\$ IN M.M. TRAP VECTOR
2005	027766	005237	120000			INC	@#120000	:CAUSE R/O ABORT - TRAP TO 8\$
2006	027772	012706	001100		8\$:	MOV	#KERSTK,KSP	:RESTORE STACK POINTER AFTER ABORT
2007	027776	013737	177572	001272		MOV	SRO,WASSRO	:READ SRO FOLLOWING SECOND KT ABORT
2008	030004	013737	177576	001274		MOV	SR2,WASSR2	:READ SR2 FOLLOWING SECOND KT ABORT
2009	030012	023737	001176	001272		CMP	\$TMP0,WASSRO	:IS SRO STILL HOLDING INFO ON FIRST ABORT?
2010	030020	001402				BEQ	9\$:BRANCH IF YES
2011	030022	005237	001200			INC	\$TMP1	:SET ERROR INDICATOR
2012	030026	023737	001202	001274	9\$:	CMP	\$TMP2,WASSR2	:DOES SR2 STILL HOLD PC OF FIRST ABORT?
2013	030034	001402				BEQ	10\$:BRANCH IF YES
2014	030036	005237	001200			INC	\$TMP1	:SET ERROR INDICATOR
2015	030042	005737	001200		10\$:	TST	\$TMP1	:WERE SRO OR SR2 CHANGED BY A SECOND ABORT?
2016	030046	001401				BEQ	11\$:BRANCH IF NO
2017	030050	104037				ERROR	37	:ONE OF STATUS REGS. CHANGED BY SECOND ABORT
2018								:FOR TIGHTER SCOPE LOOP
2019								:REPLACE ERROR CALL WITH
2020								: 'BR 6\$' = 000726
2021	030052	005037	001200		11\$:	CLR	\$TMP1	:CLEAR ERROR INDICATOR
2022	030056	000005				RESET		:EXECUTE A RESET, APPLYING AN 'INIT'
2023	030060	013737	177572	001272		MOV	SRO,WASSRO	:READ SRO
2024	030066	005737	001272			TST	WASSRO	:WAS SRO CLEARED BY THE RESET?
2025	030072	001402				BEQ	12\$:BRANCH IF YES
2026	030074	005237	001200			INC	\$TMP1	:SRO NOT CLEARED BY A RESET
2027	030100	013737	177576	001274	12\$:	MOV	SR2,WASSR2	:READ SR2
2028	030106	022737	030100	001274		CMP	#12\$,WASSR2	:WAS SR2 UNLOCKED BY A RESET?
2029	030114	001402				BEQ	13\$:BRANCH IF YES
2030	030116	005237	001200			INC	\$TMP1	:SR2 NOT UNLOCKED BY A RESET
2031	030122	005737	001200		13\$:	TST	\$TMP1	:WERE SRO & SR2 BOTH 'RESET' BY A RESET?
2032	030126	001401				BEQ	14\$:BRANCH IF YES
2033	030130	104040				ERROR	40	:SRO OR SR2 NOT 'RESET' BY A RESET
2034								:FOR TIGHTER SCOPE LOOP
2035								:REPLACE ERROR CALL WITH
2036								: 'BR 6\$' = 000676
2037	030132	012737	000001	177572	14\$:	MOV	#1,SRO	:TURN MEMORY MANAGEMENT BACK ON

```
2038 030140 013737 177576 001274 15$: MOV SR2,WASSR2 ;READ SR2 TO SEE IF ITS TRACKING AGAIN
2039 030146 012701 030140 MOV #15$,R1 ;PUT EXPECTED VIRTUAL PC IN R1
2040 030152 020137 001274 CMP R1,WASSR2 ;DID SR2 CONTAIN VIRTUAL PC AT 15$
2041 030156 001401 BEQ 16$ ;BRANCH IF YES
2042 030160 104041 ERROR 41 ;SR2 NOT TRACKING CORRECTLY
2043 ;FOR TIGHTER SCOPE LOOP
2044 ;REPLACE ERROR CALL WITH
2045 ;'BR 6$' = 000663
2046 030162 012737 027612 001110 16$: MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$
2047 030170 012737 077406 172310 MOV #77406,KIPDR4 ;RESET PDR4 TO 128 BLKS, R/W
2048 030176 012737 077406 172312 MOV #77406,KIPDR5 ;RESET PDR5 TO 128 BLKS, R/W
2049 030204 012737 002150 000250 MOV #MGMERR,MMVEC ;RESTORE ADDRESS OF NORMAL MEMORY
2050 ;MANAGEMENT TRAP ROUTINE TO M.M. VECTOR
2051
2063
2064
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 030212 000004
2065 030214 004737 035100 TST37: SCOPE
2066 030220 012737 030226 001110 1$: JSR PC,TOFF ;TURN OFF T-BIT TRAPPING FOR THIS TEST
2067 030226 005037 177776 2$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
2068 030232 012706 001100 CLR PSW ;GO TO KERNEL MODE
2069 030236 012737 000600 177640 MOV #KERSTK,KSP ;SETUP KERNEL STACK PTR.
2070 030244 012737 030326 000004 MOV #600,UIPARO ;MAP USER PAGE 0 TO 12K
2071 030252 012737 000340 000006 MOV #4$,@#4 ;LOAD KERNEL VECTOR 4 (LOC.4) WITH 4$
2072 030260 012737 140000 177776 MOV #340,@#6 ;LOAD VECTOR+2 WITH NEW PSW
2073 030266 012706 000700 MOV #140000,PSW ;GO TO USER MODE
2074 030272 012737 030312 000004 MOV #USESTK,USP ;SETUP USER STACK PTR.
2075 030300 012737 000340 000006 MOV #3$,@#4 ;LOAD USER VECTOR 4 (LOC. 60004) WITH 3$
2076 030306 005737 160000 MOV #340,@#6 ;LOAD VECTOR+2 WITH NEW PSW
2077 TST 160000 ;CAUSE TIMEOUT ERROR TRAP TO '4'
2078 ;SHOULD PICK UP NEW PC=4$ FROM KERNEL
2079 030312 013701 177776 3$: MOV PSW,R1 ;LOC. 4, NOT PC=3$ FROM USER LOC. 4 (=60004)
2080 030316 010602 MOV SP,R2 ;SAVE PSW FOR ERROR
2081 030320 005037 177776 CLR PSW ;SAVE VALUE OF STACK POINTER FOR ERROR
2082 030324 104042 ERROR 42 ;BE SURE BACK IN KERNEL MODE
2083 ;DID NOT TRAP THRU KERNEL SPACE
2084 ;FOR TIGHTER SCOPE LOOP
2085 ;REPLACE ERROR CALL WITH
2086 030326 005037 177776 4$: CLR PSW ;'BR 2$' = 000740
2087 030332 012706 001100 MOV #KERSTK,KSP ;BE SURE BACK IN KERNEL MODE
2088 030336 005037 177640 CLR UIPARO ;RESTORE KERNEL S.P. IN CASE IT CHANGED
2089 030342 012737 140000 177776 MOV #140000,PSW ;REMAP USER PAGE 0 TO 0-4K
2090 030350 012706 000700 MOV #USESTK,USP ;GO TO USER MODE
2091 030354 005037 177776 CLR PSW ;RESTORE USER STACK POINTER
2092 030360 012737 002076 000004 MOV #TIMERR,@#4 ;GO BACK TO KERNEL MODE
;RESTORE ADDR. OF NORMAL CPU TRAP HANDLER TO 4
```



```
2093 030366 012737 030214 001110      MOV    #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2094 030374 004737 035134      JSR    PC, TON          ;TURN T-BIT TRAPPING BACK ON
2095
2102      ;*****
(3)      ;*TEST 40          RTI IN USER MODE DOES NOT CHANGE PSW
(4)      ;*
(4)      ;*          THIS TEST CHECKS TO SEE THAT WHEN AN RTI IS EXECUTED IN USER
(4)      ;*          MODE, THE MODE OR PRIORITY BITS OF THE PSW ARE NOT CHANGED.
(4)      ;*
(3)      ;*****
(2) 030400 000004      TST40: SCOPE
2103
2104 030402 012737 030414 001110 1$:  MOV    #2$, $LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2105 030410 012702 170000      MOV    #170000, R2      ;LOAD 'PRESENT & EXPECTED' PSW VALUE INTO R2
2106 030414 010237 177776      2$:  MOV    R2, PSW         ;GO TO USER MODE-PRIORITY 0
2107 030420 012746 000340      MOV    #340, -(SP)      ;PUT A NEW PSW (PRIORITY=7) ON STACK
2108 030424 012746 030432      MOV    #3$, -(SP)      ;PUT NEW PC ON THE STACK
2109 030430 000002      RTI                    ;DO AN RTI FROM USER MODE
2110 030432 013701 177776      3$:  MOV    PSW, R1        ;READ NEW PSW INTO R1
2111 030436 042701 007437      BIC    #7437, R1        ;MASK OFF COND. CODE, T-BIT, AND UNUSED BITS
2112 030442 005037 177776      CLR    PSW             ;GO BACK TO KERNEL MODE
2113 030446 020201      CMP    R2, R1          ;DID PSW STAY IN USER, PRIORITY=0?
2114 030450 001401      BEQ    4$              ;BRANCH IF YES
2115 030452 104060      ERROR  60              ;PSW CHANGED BY AN RTI FROM USER
2116      ;FOR A TIGHTER SCOPE LOOP
2117      ;REPLACE ERROR CALL WITH
2118      ;'BR=2$' = 000760
2119 030454 012737 030402 001110 4$:  MOV    #1$, $LPERR      ;RESET LOOP ON ERROR POINTER TO 1$
2120
2132
2133
2134
```

```
(3)      ;*****
(4)      ;*TEST 41          KT ERROR SERVICED BEFORE TIMEOUT ERROR
(4)      ;*
(4)      ;*          THIS TEST CHECKS TO SEE THAT IF A CERTAIN VIRTUAL ADDRESS THAT
(4)      ;*          WOULD CAUSE A MEMORY MANAGEMENT ERROR CAUSES A TIMEOUT
(4)      ;*          ERROR FIRST, THE TIMEOUT ERROR IS SERVICED BUT THE MEMORY
(4)      ;*          MANAGEMENT ERROR ISN'T. THIS MEANS THAT SR0 AND SR2
(4)      ;*          SHOULD NOT REPORT THE ERROR OR LOCK UP ITS VIRTUAL ADDRESS.
(4)      ;*          A READ-ONLY VIOLATION IS USED AS THE POTENTIAL MEMORY MANAGEMENT
(4)      ;*          ERROR
(3)      ;*****
(2) 030462 000004      TST41: SCOPE
2135 030464 012705 077006      1$:  MOV    #77006, R5      ;LOAD PDR7 DATA INTO R5
2136 030470 010537 172316      MOV    R5, KIPDR7      ;MAP PAGE 7 R/W PLF=176
2137 030474 012737 030522 000004      MOV    #3$, @#4        ;SET CPU TRAP VECTOR TO ADDRESS OF 3$
2138 030502 012737 030524 000250      MOV    #4$, @#250      ;SET M.M. TRAP VECTOR TO ADDRESS OF 4$
2139 030510 012737 030516 001110      MOV    #2$, $LPERR      ;SET LOOP ON ERROR POINTER TO 2$
2140 030516 005237 177700      2$:  INC    @#177700      ;CAUSE PLF ABORT AND POTENTIAL TIMEOUT
2141 030522 104043      3$:  ERROR  43          ;TRAPPED THRU CPU TRAP VECTOR BUT SHOULDN'T HAVE
2142      ;FOR TIGHTER SCOPE LOOP
2143      ;REPLACE ERROR CALL WITH
2144      ;'BR 2$' = 000776
2145 030524 012706 001100      4$:  MOV    #KERSTK, KSP    ;RESTORE STACK POINTER AFTER TRAPPING
2146 030530 013737 177572 001272      MOV    SR0, WASSRO     ;READ STATUS REG.0
```

```

2147 030536 013737 177576 001274 5$: MOV SR2,WASSR2 ;READ STATUS REG. 2
2148 030544 012700 040017 MOV #40017,R0 ;LOAD EXPECTED SRO CONTENTS INTO R0
2149 030550 020037 001272 CMP R0,WASSRO ;SRO PLF ERROR BIT SET?
2150 030554 001401 BEQ 6$ ;BRANCH IF YES
2151 030556 104044 ERROR 44 ;SRO DIDN'T REPORT PLF ERROR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
; 'BR 2$' = 000741
2155 030560 012701 030516 6$: MOV #2$,R1 ;LOAD EXPECTED SR2 CONTENTS INTO R1
2156 030564 020137 001274 CMP R1,WASSR2 ;WAS SR2 LOCKED BY PLF ABORT?
2157 030570 001401 BEQ 7$ ;BRANCH IF YES
2158 030572 104044 ERROR 44 ;SR2 DIDN'T LOCK UP VIRTUAL ADDRESS
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
; 'BR 2$' = 000741
2162 030574 042737 160000 177572 7$: BIC #160000,SRO ;CLEAR ERROR BITS THAT WERE SET IN SRO
2163 030602 012737 002076 000004 MOV #TIMERR,@#4 ;RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
2164 030610 012737 002150 000250 MOV #MMGMERR,@#250 ;RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
2165 030616 012737 077406 172316 MOV #77406,KIPDR7 ;REMAP PAGE 7 TO READ/WRITE PLF=177
2166 030624 012737 030464 001110 MOV #1$, $LPERR ;RESET LOOP ON ERROR POINTER TO 1$

```

```

2167
2182
2183
(3) *****
*TEST 42 PC & PSW SAVED FOR KT ERROR DURING SERVICE OF TIMEOUT ERROR
(4)
(4)
(4) * THIS TEST CHECKS THE PC AND PROCESSOR STATUS WORD SAVED WHEN
(4) * A KT ERROR OCCURS DURING THE SECOND PUSH ON THE STACK DURING
(4) * SERVICING OF A TIMEOUT ERROR. DURING A 'DOUBLE ERROR'
(4) * SEQUENCE SUCH AS THIS, THE PSW SAVED WILL BE THE ONE PICKED UP
(4) * FROM VECTOR+2 (LOC. 6 IN THIS CASE) AFTER THE FIRST TRAP,
(4) * NOT THE PSW PRESENT BEFORE THE FIRST TRAP. SRO AND SR2
(4) * SHOULD RECORD THE KT ERROR (A R/O VIOLATION BY THE USER STACK PTR.)
(4)
(4) * NOTE THAT THE PREVIOUS MODE BITS <13:12> OF THE PSW
(4) * WILL BE SET IN THE PSW THAT IS SAVED.
(4)
(3) *****

```

```

(2) 030632 000004
2184 030634 004737 035100 TST42: SCOPE
1$: JSR PC,TOFF ;TURN T-BIT TRAPPING OFF FOR THIS TEST
2185 030640 012737 000600 177646 MOV #600,UIPAR3 ;MAP USER PAGE 3 TO 12-16K
2186 030646 012737 000600 177650 MOV #600,UIPAR4 ;MAP USER PAGE 4 TO 12-16K
2187 030654 012737 077402 177606 MOV #77402,UIPDR3 ;MAP USER PAGE 3 READ-ONLY
2188 030662 012737 077406 177610 MOV #77406,UIPDR4 ;MAP USER PAGE 4 READ/WRITE
2189 030670 012737 030744 000004 MOV #4$,@#4 ;LOAD ADDRESS OF 4$ IN CPU (TIMEOUT) VECTOR
2190 030676 012737 140017 000006 MOV #140017,@#6 ;LOAD PSW THAT SHOULD BE PUT ON STACK IN VECTOR+2
2191 030704 012737 030744 000250 MOV #4$,@#250 ;LOAD ADDRESS OF 4$ IN M.M. TRAP VECTOR
2192 030712 012737 000340 000252 MOV #340,@#252 ;LOAD A KERNEL PSW IN MMVEC+2
2193 030720 012737 030726 001110 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
2194 030726 012737 140000 177776 2$: MOV #140000,PSW ;GO TO USER MODE
2195 030734 012706 100002 3$: MOV #100002,USP ;SET USER STACK PTR. SO SECOND PUSH IS IN PG. 3
2196 030740 005737 177700 TST @#177700 ;CAUSE TIMEOUT ERROR THAT WILL CAUSE
;R/O ERROR WHEN TRY TO SAVE OLD PC
2198 030744 016601 000002 4$: MOV 2(KSP),R1 ;PUT PSW SAVED ON KERNEL STACK INTO R1
2199 030750 011603 MOV (KSP),R3 ;PUT PC SAVED ON KERNEL STACK INTO R3
2200 030752 013737 177572 001272 MOV SRO,WASSRO ;READ THE CONTENTS OF M.M. STATUS REG. 0
2201 030760 013737 177576 001274 MOV SR2,WASSR2 ;READ THE CONTENTS OF M.M. STATUS REG. 2

```


2202	030766	042737	160000	177572		BIC	#160000,SRO	:CLEAR THE ERROR BITS IN SRO
2203	030774	005037	177776			CLR	PSW	:BE SURE IN KERNEL MODE
2204	031000	012706	001100			MOV	#KERSTK,KSP	:RESTORE KERNEL STACK POINTER
2205	031004	012737	140000	177776		MOV	#140000,PSW	:GO TO USER MODE
2206	031012	012706	000700			MOV	#USESTK,USP	:RESTORE USER STACK POINTER
2207	031016	005037	177776			CLR	PSW	:GO BACK TO KERNEL MODE
2208	031022	005037	001176			CLR	\$TMP0	:CLEAR ERROR INDICATOR
2209	031026	020127	170017			CMP	R1,#170017	:WAS THE PSW SAVED THE ONE PICKED UP BY THE
2210								:TIMEOUT TRAP FROM ERRVEC+2?
2211								:VALUE 170017 = PSW FROM LOC. 6 WITH
2212								:PREVIOUS MODE BITS = USER
2213	031032	001402				BEQ	5\$:BRANCH IF YES
2214	031034	005237	001176			INC	\$TMP0	:WRONG PSW SAVED DURING 'DOUBLE ERROR' SEQUENCE
2215	031040	020327	030744	5\$:		CMP	R3,#3\$+4	:WAS THE PC AT THE TIME OF THE TIMEOUT ERROR
2216								:SAVED ON THE STACK?
2217	031044	001402				BEQ	6\$:BRANCH IF YES
2218	031046	005237	001176			INC	\$TMP0	:WRONG PC SAVED DURING TRAP SEQUENCE
2219	031052	023727	001272	020147	6\$:	CMP	WASSR0,#20147	:DID SRO REPORT - USER, PAGE 3, R/O ABORT?
2220	031060	001402				BEQ	7\$:BRANCH IF YES
2221	031062	005237	001176			INC	\$TMP0	:SRO DID NOT REPORT R/O ABORT
2222	031066	023727	001274	030740	7\$:	CMP	WASSR2,#3\$:DID SR2 LOCK UP VIRTUAL ADDR. OF LAST
2223								:INSTRUCTION SUCCESSFULLY FETCHED?
2224	031074	001402				BEQ	8\$:BRANCH IF YES
2225	031076	005237	001176			INC	\$TMP0	:SR2 DID NOT LOCK UP ADDR. OF TIMEOUT INST.
2226	031102	005737	001176	8\$:		TST	\$TMP0	:ANY 'ERRORS' DURING TRAP SEQUENCE?
2227	031106	001401				BEQ	9\$:BRANCH IF NO
2228	031110	104045				ERROR	45	:THE WRONG PC OR PSW WERE SAVED
2229								:OR SRO OR SR2 DID NOT REPORT R/O
2230								:ERROR DURING TIMEOUT - KT TRAP
2231								:SEQUENCE
2232								:FOR TIGHTER SCOPE LOOP
2233								:REPLACE ERROR CALL WITH
2234								: 'BR 2\$' = 000710
2235	031112	012737	002076	000004	9\$:	MOV	#TIMERR,@#4	:RESTORE ADDRESS OF NORMAL CPU TRAP HANDLER
2236	031120	012737	000340	000006		MOV	#340,@#6	:RELOAD ERRVEC+2 WITH KERNEL PSW
2237	031126	012737	002150	000250		MOV	#MGMERR,@#250	:RESTORE ADDRESS OF NORMAL M.M. TRAP HANDLER
2238	031134	012737	077406	177606		MOV	#77406,UIPDR3	:REMAP USER PAGE 3 READ/WRITE
2239	031142	012737	030634	001110		MOV	#1\$, \$LPERR	:RESET LOOP ON ERROR POINTER TO 1\$
2240	031150	004737	035134			JSR	PC,TON	:TURN T-BIT TRAPPING BACK ON

```
2241 *****
2242 *
2243 * THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
2244 * THE 'MOVE FROM PREVIOUS' AND MOVE TO PREVIOUS' INSTRUCTIONS.
2245 *
2246 *****
2247
2248 *****
2260 *TEST 43 MOVE FROM PREVIOUS (USER) I-SPACE
(3) *
(4) *
(4) * THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE
(4) * PREVIOUS MODE IS CLOKED CORRECTLY
(4) * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
(4) *
(4) *
(4) * IF THE CORRECT MODE (USER) IS NOT ENABLED A NON-RESIDENT ABORT
```

```
(4)
(4)
(3)
(2) 031154 000004
2261 031156 005037 172340
2262 031162 012737 000200 172342
2263 031170 012737 000400 172344
2264 031176 012737 000600 172346
2265 031204 012737 000600 172350
2266 031212 012737 007600 172356
2267 031220 012700 077406
2268
2269 031224 012702 000010
2270 031230 012701 172300
2271 031234 010021
2272 031236 077202
2273 031240 012702 000010
2274 031244 012701 177600
2275 031250 010021
2276 031252 077202
2277 031254 012737 000000 177640
2278 031262 012737 000200 177642
2279 031270 012737 000400 177644
2280 031276 012737 000600 177646
2281 031304 012737 007600 177656
2282 031312 012737 031320 001110
2283 031320
2284 031320 012737 077406 172310
2285 031326 012737 000600 172350
2286 031334 012737 000600 177650
2287 031342 012700 036514
2288 031346 010037 100000
2289 031352 012737 031754 000250
2290 031360 105037 172310
2291
2292
2293 031364 012737 031372 001110
2294 031372 012737 030340 177776
2295 031400 006506
2296
2297 031402 022706 001100
2298 031406 001407
2299 031410 012600
2300 031412 012701 000700
2301 031416 020001
2302 031420 001403
2303 031422 104046
2304
2305
2306
2307 031424 000401
2308 031426 104050
2309
2310
2311
2312 031430
```

```

;* WILL OCCUR AND TRAP TO 23$, WHERE THE ERRORS ARE REPORTED.
;*
:*****
TST43: SCOPE
1$: CLR KIPAR0 ;MAP KERNEL PAGE 0 TO 0-4K
MOV #200,KIPAR1 ;MAP KERNEL PAGE 1 TO 4-8K
MOV #400,KIPAR2 ;MAP KERNEL PAGE 2 TO 8-12K
MOV #600,KIPAR3 ;MAP KERNEL PAGE 3 TO 12-16K
MOV #600,KIPAR4 ;MAP KERNEL PAGE 4 TO 12-16K
MOV #7600,KIPAR7 ;MAP KERNEL PAGE 7 TO THE I/O PAGE
MOV #77406,R0 ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT
;READ/WRITE, LENGTH 200 BLOCKS
MOV #10,R2 ;SET LOOP COUNTER TO 8
MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
2$: MOV R0,(R1)+ ;LOAD PDR WITH 77406
SOB R2,2$ ;LOOP TO 2$ UNTIL ALL PDRS LOADED
MOV #10,R2 ;SET LOOP COUNTER TO 8
MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
3$: MOV R0,(R1)+ ;LOAD PDR WITH 77406
SOB R2,3$ ;LOOP TO 3$ UNTIL ALL PDRS LOADED
MOV #000,UIPAR0 ;MAP USER I PAGE 0 TO 0-4K
MOV #200,UIPAR1 ;MAP USER I PAGE 1 TO 4-8K
MOV #400,UIPAR2 ;MAP USER I PAGE 2 TO 8-12K
MOV #600,UIPAR3 ;MAP USER I PAGE 3 TO 12-16K
MOV #7600,UIPAR7 ;MAP USER I PAGE 7 TO THE I/O PAGE
4$: MOV #4$,$LPERR ;SET LOOP ON ERROR TO 4$
MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
MOV #600,KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K
MOV #600,UIPAR4 ;MAP USER I PAGE 4 TO 12K
MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 60000
MOV #23$,MMVEC ;SET M.M. VECTOR TO 23$
CLRB KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
;THE FOLLOWING WILL TEST DSTM=0 MFPI
MOV #5$,$LPERR ;SET LOOP ON ERROR POINTER TO 5$
5$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
6$: MFPI USP ;PUT USER STACK POINTER ON KERNEL
;STACK
;WAS SOMETHING PUSHED ON STACK AT 6$
7$: BEQ 7$ ;BRANCH IF NOTHING WAS PUSHED
MOV (KSP)+,R0 ;POP KERNEL STACK INTO R0
MOV #USESTK,R1 ;EXPECTING TO GET 700 AS USP
CMP R0,R1 ;DID YOU GET THE RIGHT POINTER?
8$: BEQ 8$ ;BRANCH IF YOU DID
ERROR 46 ;WRONG THING WAS PUSHED ON STACK
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 5$' = 000763
BR 8$ ;BRANCH TO NEXT TRY
7$: ERROR 50 ;NOTHING PUSHED ON STACK
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;'BR 5$' = 000761
8$: ;THE FOLLOWING WILL TEST DSTM=1 MFPI.
```



```
2313 031430 012737 031442 001110      MOV    #9$, $LPERR      ;SET LOOP ON ERROR POINTER TO 9$
2314 031436 012700 036514      MOV    #36514, R0      ;RELOAD DATA PATTERN IN R0
2315 031442 012737 030340 177776 9$:    MOV    #030340, PSW    ;MAKE PREVIOUS MODE USER
2316 031450 012702 100000      MOV    #100000, R2     ;LOAD VIRTUAL ADDRESS INTO R2
2317 031454 006512      MFPI   (R2)            ;READ FROM PHYSICAL 60000
2318 031456 012601      MOV    (KSP)+, R1     ;POP KERNEL STACK INTO R1
2319 031460 020001      CMP    R0, R1         ;WAS DATA FETCHED SAME AS STORED
2320 031462 001401      BEQ    10$            ;BRANCH IF CORRECT DATA WAS FETCHED
2321 031464 104046      ERROR  46            ;WRONG DATA WAS FETCHED
2322      ;FOR TIGHTER SCOPE LOOP
2323      ;REPLACE ERROR CALL WITH
2324      ;'BR 9$' = 000766
2325 031466      10$:    ;THE FOLLOWING WILL TEST DSTM=2 MFPI.
2326 031466 012737 031474 001110      MOV    #11$, $LPERR    ;SET LOOP ON ERROR POINTER TO 11$
2327 031474 012737 030340 177776 11$:    MOV    #030340, PSW    ;MAKE PREVIOUS MODE USER
2328 031502 012702 100000      MOV    #100000, R2     ;LOAD VIRTUAL ADDRESS INTO R2
2329 031506 006522      MFPI   (R2)+          ;READ FROM PHYSICAL 60000
2330 031510 012601      MOV    (KSP)+, R1     ;POP KERNEL STACK INTO R1
2331 031512 020001      CMP    R0, R1         ;WAS DATA FETCHED SAME AS STORED
2332 031514 001401      BEQ    12$            ;BRANCH IF CORRECT DATA WAS FETCHED
2333 031516 104046      ERROR  46            ;WRONG DATA WAS FETCHED
2334      ;FOR TIGHTER SCOPE LOOP
2335      ;REPLACE ERROR CALL WITH
2336      ;'BR 11$' = 000766
2337 031520      12$:    ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
2338 031520 012737 031526 001110      MOV    #13$, $LPERR    ;SET LOOP ON ERROR POINTER TO 13$
2339 031526 012737 030340 177776 13$:    MOV    #030340, PSW    ;MAKE PREVIOUS MODE USER
2340 031534 006537 100000      MFPI   @#100000        ;READ FROM PHYSICAL 60000
2341 031540 012601      MOV    (KSP)+, R1     ;POP KERNEL STACK INTO R1
2342 031542 020001      CMP    R0, R1         ;WAS DATA FETCHED SAME AS STORED
2343 031544 001401      BEQ    14$            ;BRANCH IF CORRECT DATA WAS FETCHED
2344 031546 104046      ERROR  46            ;WRONG DATA WAS FETCHED
2345      ;FOR TIGHTER SCOPE LOOP
2346      ;REPLACE ERROR CALL WITH
2347      ;'BR 13$' = 000767
2348 031550      14$:    ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
2349 031550 012737 031556 001110      MOV    #15$, $LPERR    ;SET LOOP ON ERROR POINTER TO 15$
2350 031556 012737 030340 177776 15$:    MOV    #030340, PSW    ;MAKE PREVIOUS MODE USER
2351 031564 012702 100002      MOV    #100002, R2     ;LOAD VIRTUAL ADDRESS INTO R2
2352 031570 006542      MFPI   -(R2)          ;READ FROM PHYSICAL 60000
2353 031572 012601      MOV    (KSP)+, R1     ;POP KERNEL STACK INTO R1
2354 031574 020001      CMP    R0, R1         ;WAS DATA FETCHED SAME AS STORED
2355 031576 001401      BEQ    16$            ;BRANCH IF CORRECT DATA WAS FETCHED
2356 031600 104046      ERROR  46            ;WRONG DATA WAS FETCHED
2357      ;FOR TIGHTER SCOPE LOOP
2358      ;REPLACE ERROR CALL WITH
2359      ;'BR 15$' = 000766
2360 031602      16$:    ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
2361      ;
2362      ;
2363 031602 012737 031610 001110      MOV    #17$, $LPERR    ;SET LOOP ON ERROR POINTER TO 17$
2364 031610 012737 030340 177776 17$:    MOV    #030340, PSW    ;MAKE PREVIOUS MODE USER
2365 031616 012737 100000 001202      MOV    #100000, $TMP2  ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
2366 031624 012702 001204      MOV    #<$TMP2+2>, R2  ;LOAD ADDR. OF $TMP2+2 INTO R2
2367 031630 006552      MFPI   @-(R2)         ;READ FROM PHYSICAL 60000
2368 031632 012601      MOV    (KSP)+, R1     ;POP KERNEL STACK INTO R1
```

```
2369 031634 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2370 031636 001401      BEQ      18$        ;BRANCH IF CORRECT DATA WAS FETCHED
2371 031640 104046      ERROR   46         ;WRONG DATA WAS FETCHED
2372                                     ;FOR TIGHTER SCOPE LOOP
2373                                     ;REPLACE ERROR CALL WITH
2374                                     ;'BR 17$' = 000763
2375 031642          18$:  ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
2376                                     ;
2377 031642 012737 031650 001110      MOV      #19$, $LPERR ;SET LOOP ON ERROR POINTER TO 19$
2378 031650 012737 030340 177776 19$:  MOV      #030340,PSW  ;MAKE PREVIOUS MODE USER
2379 031656 005002          CLR      R2         ;MAKE REGISTER 2 A ZERO
2380 031660 006562 100000      MFPI    100000(R2)  ;READ FROM PHYSICAL 60000
2381 031664 012601          MOV      (KSP)+,R1  ;POP KERNEL STACK INTO R1
2382 031666 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2383 031670 001401      BEQ      20$        ;BRANCH IF CORRECT DATA WAS FETCHED
2384 031672 104046      ERROR   46         ;WRONG DATA WAS FETCHED
2385                                     ;FOR TIGHTER SCOPE LOOP
2386                                     ;REPLACE ERROR CALL WITH
2387                                     ;'BR 19$' = 000766
2388 031674          20$:  ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
2389                                     ;
2390 031674 012737 031702 001110      MOV      #21$, $LPERR ;SET LOOP ON ERROR POINTER TO 21$
2391 031702 012737 030340 177776 21$:  MOV      #030340,PSW  ;MAKE PREVIOUS MODE USER
2392 031710 012737 100000 001202      MOV      #100000,$TMP2 ;LOAD TEST LOC. V.A. INTO $TMP2
2393 031716 012702 001202          MOV      #$TMP2,R2  ;LOAD ADDRESS OF $TMP2 INTO R2
2394 031722 006572 000000      MFPI    @0(R2)     ;USE $TMP2 TO FETCH VIRTUAL
2395                                     ;ADDRESS OF 60000
2396 031726 012601          MOV      (KSP)+,R1  ;POP KERNEL STACK INTO R1
2397 031730 020001      CMP      R0,R1      ;WAS DATA FETCHED SAME AS STORED
2398 031732 001401      BEQ      22$        ;BRANCH IF CORRECT DATA WAS FETCHED
2399 031734 104046      ERROR   46         ;WRONG DATA WAS FETCHED
2400                                     ;FOR TIGHTER SCOPE LOOP
2401                                     ;REPLACE ERROR CALL WITH
2402                                     ;'BR 21$' = 000762
2403 031736 012737 002150 000250 22$:  MOV      #MGMERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
2404 031744 012737 031156 001110      MOV      #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
2405 031752 000423          BR       TST44     ;BRANCH TO NEXT TEST
2406
2407
2408 031754 012637 001266          23$:  MOV      (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
2409 031760 012637 001270          MOV      (KSP)+,TRAPPS
2410 031764 013737 177572 001272      MOV      SR0,WASSR0 ;SAVE SR0 FOR ERROR TYPEOUT
2411 031772 013737 177576 001274      MOV      SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
2412 032000 042737 160000 177572      BIC     #160000,SR0 ;CLEAR ERROR BITS IN SR0 AND LEAVE
2413 032006 104051          ERROR   51         ;TRIED TO READ NON-RESIDENT PAGE
2414                                     ;FOR TIGHTER SCOPE LOOP
2415                                     ;REPLACE ERROR CALL WITH
2416                                     ;A 'NOP' = 000240
2417 032010 013746 001270          MOV      TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2418 032014 013746 001266          MOV      TRAPPC,-(KSP)
2419 032020 000002          RTI
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
(3)
(4)
*****
:*TEST 44 MOVE TO PREVIOUS (USER) I-SPACE
:*
```



```
(4) : * THIS TEST USES THE 'MTPI' INSTRUCTION TO ENSURE THAT THE
(4) : * PREVIOUS MODE IS CLOKED CORRECTLY
(4) : * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
(4) : *
(4) : * IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
(4) : * WILL OCCUR AND TRAP TO 20$, WHERE THE ERRORS ARE REPORTED.
(4) : *
(3) : *****
(2) 032022 000004 TST44: SCOPE
2434 032024 012737 077406 172310 1$: MOV #77406,KIPDR4 :KERNEL I-SPACE PAGE 4 READ/WRITE
2435 032032 012737 077406 177610 MOV #77406,UIPDR4 :USER I-SPACE PAGE 4 READ/WRITE
2436 032040 012737 000600 172350 MOV #600,KIPAR4 :MAP KERNEL I PAGE 4 TO 12K
2437 032046 012737 000600 177650 MOV #600,UIPAR4 :MAP USER I PAGE 4 TO 12K
2438 032054 012737 032634 000250 MOV #20$,MMVEC :SET M.M. VECTOR TO 20$
2439 :THE FOLLOWING WILL TEST DSTM=0 MTPI
2440 :
2441 032062 012737 030340 177776 2$: MOV #030340,PSW :MAKE PREVIOUS MODE USER
2442 032070 012746 007777 MOV #7777,-(KSP) :PUSH DATA ON KERNEL STACK
2443 032074 006606 MTPI USP :LOAD USER STACK POINTER
2444 032076 006506 MFPI USP :READ USER STACK POINTER
2445 032100 012601 MOV (KSP)+,R1 :POP KERNEL STACK INTO R1
2446 032102 022701 007777 CMP #7777,R1 :WAS USER STACK POINTER CHANGED
2447 032106 001401 BEQ 3$ :BRANCH IF IT WAS
2448 032110 104050 ERROR 50 :USER STACK POINTER NOT CHANGED
2449 :FOR TIGHTER SCOPE LOOP
2450 :REPLACE ERROR CALL WITH
2451 :'BR 2$' = 000764
2452 032112 012737 030340 177776 3$: MOV #030340,PSW :MAKE PREVIOUS MODE USER
2453 032120 012746 000700 MOV #USESTK,-(KSP) :GET READY TO RESTORE USER S. POINT
2454 032124 006606 MTPI USP :RESTORE USER STACK POINTER
2455 032126 4$: :THIS WILL TEST DSTM = 1 MTPI.
2456 032126 012737 032144 001110 MOV #5$,SLPERR :SET LOOP ON ERROR POINTER TO 5$
2457 032134 012702 100000 MOV #100000,R2 :LOAD VIRTUAL ADDRESS INTO R2
2458 032140 012700 125252 MOV #125252,R0 :LOAD TEST DATA INTO R0
2459 032144 010046 5$: MOV R0,-(KSP) :PUSH TEST DATA ON KERNEL STACK
2460 032146 105037 172310 CLR B KIPDR4 :MAKE KERNEL I PAGE 4 NON-RESIDENT
2461 032152 006612 MTPI (R2) :LOAD TEST DATA INTO PHYSICAL 60000
2462 032154 112737 000006 172310 MOV B #006,KIPDR4 :MAKE KERNEL PAGE 4 RESIDENT
2463 032162 011201 MOV (R2),R1 :READ FROM ADDRESS 60000
2464 032164 020001 CMP R0,R1 :SEE IF DATA WAS STORED AT CORRECT PLACE
2465 032166 001401 BEQ 6$ :BRANCH IF STORE WAS CORRECT
2466 032170 104047 ERROR 47 :INCORRECT STORE
2467 :FOR TIGHTER SCOPE LOOP
2468 :REPLACE ERROR CALL WITH
2469 :'BR 5$' = 000765
2470 032172 6$: :THE FOLLOWING WILL TEST DSTM=2 MTPI.
2471 :
2472 032172 012737 032216 001110 MOV #8$,SLPERR :SET LOOP ON ERROR POINTER TO 8$
2473 032200 012737 030340 177776 MOV #030340,PSW :MAKE PREVIOUS MODE USER
2474 032206 012700 125252 MOV #125252,R0 :LOAD TEST DATA INTO R0
2475 032212 012702 100000 MOV #100000,R2 :LOAD VIRTUAL ADDRESS INTO R2
2476 032216 010046 8$: MOV R0,-(KSP) :PUSH TEST DATA ON KERNEL STACK
2477 032220 105037 172310 CLR B KIPDR4 :MAKE KERNEL PAGE 4 NON-RESIDENT
2478 032224 006612 MTPI (R2) :LOAD TEST DATA INTO PHYSICAL 60000
2479 032226 112737 000006 172310 MOV B #006,KIPDR4 :MAKE KERNEL PAGE 4 RESIDENT
```

```

2480 032234 013701 100000      MOV    @#100000,R1      ;READ FROM ADDRESS 60000
2481 032240 020001      CMP    R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
2482 032242 001401      BEQ    9$              ;BRANCH IF STORE WAS CORRECT
2483
2484 032244 104047      ERROR  47             ;INCORRECT STORE
2485
2486
2487
2488 032246
2489 032246 012737 032266 001110 9$:      ;THIS WILL TEST DSTM = 3 MTPI.
2490 032254 012737 030340 177776      MOV    #10$, $LPERR    ;SET LOOP ON ERROR POINTER TO 10$
2491 032262 012700 052525      MOV    #030340,PSW     ;MAKE PREVIOUS MODE USER
2492 032266 010046      MOV    #52525,R0       ;LOAD TEST DATA INTO R0
2493 032270 105037 172310 10$:      MOV    R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
2494 032274 006637 100000      CLRB   KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2495 032300 112737 000006 172310      MTPI   @#100000        ;LOAD TEST DATA INTO PHYSICAL 60000
2496 032306 013701 100000      MOV    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
2497 032312 020001      MOV    @#100000,R1     ;READ FROM ADDRESS 60000
2498 032314 001401      CMP    R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
2499 032316 104047      BEQ    11$             ;BRANCH IF STORE WAS CORRECT
2500
2501
2502
2503 032320
2504 032320 012737 032340 001110 11$:      ;THIS WILL TEST DSTM = 4 MTPI.
2505 032326 012737 030340 177776      MOV    #12$, $LPERR    ;SET LOOP ON ERROR POINTER TO 12$
2506 032334 012700 125252      MOV    #030340,PSW     ;MAKE PREVIOUS MODE USER
2507 032340 010046      MOV    #125252,R0       ;LOAD TEST DATA INTO R0
2508 032342 012702 100002 12$:      MOV    R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
2509 032346 105037 172310      MOV    #100002,R2      ;LOAD VIRTUAL ADDRESS INTO R2
2510 032352 006642      CLRB   KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2511 032354 112737 000006 172310      MTPI   -(R2)           ;LOAD TEST DATA INTO PHYSICAL 60000
2512 032362 013701 100000      MOV    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
2513 032366 020001      MOV    @#100000,R1     ;READ FROM ADDRESS 60000
2514 032370 001401      CMP    R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
2515 032372 104047      BEQ    13$             ;BRANCH IF STORE WAS CORRECT
2516
2517
2518
2519 032374
2520
2521 032374 012737 032426 001110 13$:      ;THE FOLLOWING WILL TEST DSTM=5 MTPI.
2522 032402 012737 030340 177776      MOV    #14$, $LPERR    ;SET LOOP ON ERROR POINTER TO 14$
2523 032410 012700 052525      MOV    #030340,PSW     ;MAKE PREVIOUS MODE USER
2524 032414 012702 001204      MOV    #52525,R0       ;LOAD TEST DATA INTO R0
2525 032420 012737 100000 001202      MOV    #<$TMP2+2>,R2   ;LOAD ADDR. OF LOC. $TMP2+2 INTO R2
2526 032426 010046      MOV    #100000,$TMP2   ;LOAD VIRT. ADDR. OF TEST LOC. INTO $TMP2
2527 032430 105037 172310 14$:      MOV    R0,-(KSP)       ;PUSH TEST DATA ON KERNEL STACK
2528 032434 006652      CLRB   KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
2529 032436 112737 000006 172310      MTPI   @-(R2)          ;LOAD TEST DATA INTO PHYSICAL 60000
2530 032444 013701 100000      MOV    #006,KIPDR4     ;MAKE KERNEL PAGE 4 RESIDENT
2531 032450 020001      MOV    @#100000,R1     ;READ FROM ADDRESS 60000
2532 032452 001401      CMP    R0,R1           ;SEE IF DATA WAS STORED CORRECTLY
2533 032454 104047      BEQ    15$             ;BRANCH IF STORE WAS CORRECT
2534
2535
    ERROR  47             ;INCORRECT STORE
                        ;FOR TIGHTER SCOPE LOOP
                        ;REPLACE ERROR CALL WITH
    
```



```
2536                                     ;'BR 14$' = 000764
2537 032456 15$: ;THIS WILL TEST DSTM = 6 MTPI.
2538
2539 032456 012737 032500 001110      MOV    #16$, $LPERR      ;SET LOOP ON ERROR POINTER TO 16$
2540 032464 012737 030340 177776      MOV    #030340, PSW     ;MAKE PREVIOUS MODE USER
2541 032472 012700 052525              MOV    #52525, R0      ;LOAD TEST DATA INTO R0
2542 032476 005002                    CLR    R2              ;MAKE REGISTER 2 ZERO
2543 032500 010046 16$: MOV    R0, -(KSP)      ;PUSH TEST DATA ON KERNEL STACK
2544 032502 105037 172310              CLRB  KIPDR4          ;MAKE KERNEL I PAGE 4 NON-RESIDENT
2545 032506 006662 100000              MTPI  100000(R2)      ;LOAD TEST DATA INTO PHYSICAL 60000
2546 032512 112737 000006 172310      MOV    #006, KIPDR4   ;MAKE KERNEL PAGE 4 RESIDENT
2547 032520 013701 100000              MOV    @#100000, R1    ;READ FROM ADDRESS 60000
2548 032524 020001                    CMP    R0, R1         ;SEE IF DATA WAS STORED CORRECTLY
2549 032526 001401                    BEQ    17$            ;BRANCH IF STORE WAS CORRECT
2550 032530 104047                    ERROR  47            ;INCORRECT STORE
2551                                     ;FOR TIGHTER SCOPE LOOP
2552                                     ;REPLACE ERROR CALL WITH
2553                                     ;'BR 16$' = 000763
2554 032532 17$: ;THE FOLLOWING WILL TEST DSTM=7 MTPI.
2555
2556 032532 012737 032564 001110      MOV    #18$, $LPERR      ;SET LOOP ON ERROR POINTER TO 18$
2557 032540 012737 030340 177776      MOV    #030340, PSW     ;MAKE PREVIOUS MODE USER
2558 032546 012700 125252              MOV    #125252, R0     ;LOAD TEST DATA INTO R0
2559 032552 012737 100000 001202      MOV    #100000, $TMP2  ;LOAD VIRT. ADDR. OF TEST LOCATION
2560                                     ;INTO LOCATION $TMP2
2561 032560 012702 001202 18$: MOV    #$TMP2, R2      ;LOAD ADDRESS OF $TMP2 INTO R2
2562 032564 010046                    MOV    R0, -(KSP)     ;PUSH TEST DATA ON KERNEL STACK
2563 032566 105037 172310              CLRB  KIPDR4          ;MAKE KERNEL PAGE 4 NON-RESIDENT
2564 032572 006672 000000              MTPI  @0(R2)         ;LOAD TEST DATA INTO PHYSICAL 60000
2565 032576 112737 000006 172310      MOV    #006, KIPDR4   ;MAKE KERNEL PAGE 4 RESIDENT
2566 032604 013701 100000              MOV    @#100000, R1    ;READ FROM ADDRESS 60000
2567 032610 020001                    CMP    R0, R1         ;SEE IF DATA WAS STORED CORRECTLY
2568 032612 001401                    BEQ    19$            ;BRANCH IF STORE WAS CORRECT
2569 032614 104047                    ERROR  47            ;INCORRECT STORE
2570                                     ;FOR TIGHTER SCOPE LOOP
2571                                     ;REPLACE ERROR CALL WITH
2572                                     ;'BR 18$' = 000763
2573 032616 012737 032024 001110 19$: MOV    #1$, $LPERR      ;SET LOOP POINTER TO START OF TEST
2574 032624 012737 002150 000250      MOV    #MGERR, MMVEC   ;RESTORE M.M. VECTOR TO NORMAL ROUTINE
2575 032632 000423                    BR     TST45          ;BRANCH TO NEXT TEST
2576
2577
2578 032634 012637 001266 20$: MOV    (KSP)+, TRAPPC  ;SAVE PC & PS OF TRAP
2579 032640 012637 001270              MOV    (KSP)+, TRAPPS
2580 032644 013737 177572 001272      MOV    SR0, WASSR0    ;SAVE SR0 FOR ERROR TYPEOUT
2581 032652 013737 177576 001274      MOV    SR2, WASSR2    ;SAVE SR2 FOR ERROR TYPEOUT
2582 032660 042737 160000 177572      BIC   #160000, SR0    ;CLEAR ERROR BITS IN SR0
2583 032666 104051                    ERROR  51            ;TRIED TO LOAD A N.R. PAGE 4
2584                                     ;FOR TIGHTER SCOPE LOOP
2585                                     ;REPLACE ERROR CALL WITH
2586                                     ;A 'NOP' = 000240
2587 032670 013746 001270              MOV    TRAPPS, -(KSP) ;PUT PC & PS OF TRAP ON STACK
2588 032674 013746 001266              MOV    TRAPPC, -(KSP)
2589 032700 000002                    RTI                    ;RETURN TO TEST
2590
2591
```

2592
2604
(3)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(4)
(3)
(2)
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646

032702 000004
032704 012700 077406
032710 012702 000010
032714 012701 177600
032720 010021
032722 077202
032724 012737 032732 001110
032732 012737 140340 177776
032740 012737 077406 172310
032746 012737 000600 172350
032754 012737 000600 177650
032762 012700 036514
032766 010037 100000
032772 012702 100000
032776 012737 033400 000250
033004 105037 177610
033010 012737 140340 177776
033016 006506
033020 022706 000700
033024 001407
033026 012600
033030 012701 001100
033034 020001
033036 001403
033040 104046
033042 000401
033044 104050
033046
033046 012737 033054 001110
033054 012737 140340 177776
033062 012700 036514
033066 012702 100000
033072 006512
033074 012601

```
*****  
*TEST 45 MOVE FROM PREVIOUS (KERNEL) I-SPACE TO USER MODE  
*  
* THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE  
* FETCH IS FROM KERNEL SPACE.  
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED.  
*  
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT  
* WILL OCCUR AND TRAP TO 21$, WHERE THE ERRORS ARE REPORTED.  
*****  
TST45: SCOPE  
1$: MOV #77406,R0 ;MAKE ALL USER I-SPACE PAGES RESIDENT  
;READ/WRITE, LENGTH 200 BLOCKS  
MOV #10,R2 ;SET LOOP COUNTER TO 8  
MOV #UIPDR0,R1 ;LOAD ADDRESS OF FIRST PDR IN R1  
2$: MOV R0,(R1)+ ;LOAD PDR WITH 77406  
SOB R2,2$ ;LOOP UNTIL 8 USER PDRS LOADED  
MOV #3$, $LPERR ;SET LOOP ON ERROR TO 3$  
3$: MOV #140340,PSW ;GO TO USER MODE FOR THIS TEST  
MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE  
MOV #600,KIPAR4 ;MAP KERNEL I PAGE 4 TO 12K  
MOV #600,UIPAR4 ;MAP USER I PAGE 4 TO 12K  
MOV #36514,R0 ;LOAD DATA PATTERN INTO R0  
MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 60000  
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2  
;THE FOLLOWING WILL TEST DSTM=0 MFPI  
MOV #21$,MMVEC ;SET M.M. VECTOR TO 21$  
CLRB UIPDR4 ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT  
MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER  
4$: MFPI KSP ;PUT KERNEL STACK POINTER ON USER STACK  
CMP #USESTK,USP ;WAS SOMETHING PUSHED ON STACK AT 1$  
BEQ 5$ ;BRANCH IF NOTHING WAS PUSHED  
MOV (USP)+,R0 ;POP USER STACK INTO R0  
MOV #KERSTK,R1 ;EXPECTING 1100 AS KSP  
CMP R0,R1 ;DID YOU GET THE RIGHT POINTER?  
BEQ 6$ ;BRANCH IF YOU DID  
ERROR 46 ;WRONG THING WAS PUSHED ON STACK  
;FOR TIGHTER SCOPE LOOP  
;REPLACE ERROR CALL WITH  
; 'BR 4$' = 000766  
5$: BR 6$ ;BRANCH TO NEXT TRY  
ERROR 50 ;NOTHING PUSHED ON STACK  
;FOR TIGHTER SCOPE LOOP  
;REPLACE ERROR CALL WITH  
; 'BR 4$' = 000764  
6$: ;THE FOLLOWING WILL TEST DSTM=1 MFPI.  
MOV #7$, $LPERR ;SET LOOP ON ERROR POINTER TO 7$  
7$: MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER  
MOV #36514,R0 ;LOAD DATA EXPECTED INTO R0  
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2  
MFPI (R2) ;READ FROM PHYSICAL 60000  
MOV (USP)+,R1 ;POP USER STACK INTO R1
```



```
2647 033076 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
2648 033100 001401          BEQ    8$             ;BRANCH IF CORRECT DATA WAS FETCHED
2649 033102 104046          ERROR  46            ;WRONG DATA WAS FETCHED
2650                                     ;FOR TIGHTER SCOPE LOOP
2651                                     ;REPLACE ERROR CALL WITH
2652                                     ;'BR 7$' = 000764
2653 033104                                     8$:      ;THE FOLLOWING WILL TEST DSM=2 MFPI.
2654 033104 012737 033112 001110  MOV    #9$, $LPERR    ;SET LOOP ON ERROR POINTER TO 9$
2655 033112 012737 140340 177776 9$:      MOV    #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
2656 033120 012702 100000          MOV    #100000,R2     ;LOAD VIRTUAL ADDRESS INTO R2
2657 033124 006522          MFPI   (R2)+          ;READ FROM PHYSICAL 60000
2658 033126 012601          MOV    (USP)+,R1     ;POP USER STACK INTO R1
2659 033130 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
2660 033132 001401          BEQ    10$            ;BRANCH IF CORRECT DATA WAS FETCHED
2661 033134 104046          ERROR  46            ;WRONG DATA WAS FETCHED
2662                                     ;FOR TIGHTER SCOPE LOOP
2663                                     ;REPLACE ERROR CALL WITH
2664                                     ;'BR 9$' = 000766
2665 033136                                     10$:     ;THE FOLLOWING WILL TEST DSTM=3 MFPI.
2666 033136 012737 033144 001110  MOV    #11$, $LPERR    ;SET LOOP ON ERROR POINTER TO 11$
2667 033144 012737 140340 177776 11$:     MOV    #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
2668 033152 006537 100000          MFPI   @#100000       ;READ FROM PHYSICAL 60000
2669 033156 012601          MOV    (USP)+,R1     ;POP USER STACK INTO R1
2670 033160 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
2671 033162 001401          BEQ    12$            ;BRANCH IF CORRECT DATA WAS FETCHED
2672 033164 104046          ERROR  46            ;WRONG DATA WAS FETCHED
2673                                     ;FOR TIGHTER SCOPE LOOP
2674                                     ;REPLACE ERROR CALL WITH
2675                                     ;'BR 11$' = 000767
2676 033166                                     12$:     ;THE FOLLOWING WILL TEST DSTM=4 MFPI.
2677 033166 012737 033174 001110  MOV    #13$, $LPERR    ;SET LOOP ON ERROR POINTER TO 13$
2678 033174 012737 140340 177776 13$:     MOV    #140340,PSW    ;MAKE PREVIOUS MODE DERNEL PRESENT USER
2679 033202 012702 100002          MOV    #100002,R2     ;LOAD VIRTUAL ADDRESS INTO R2
2680 033206 006542          MFPI   -(R2)         ;READ FROM PHYSICAL 60000
2681 033210 012601          MOV    (USP)+,R1     ;POP USER STACK INTO R1
2682 033212 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
2683 033214 001401          BEQ    14$            ;BRANCH IF CORRECT DATA WAS FETCHED
2684 033216 104046          ERROR  46            ;WRONG DATA WAS FETCHED
2685                                     ;FOR TIGHTER SCOPE LOOP
2686                                     ;REPLACE ERROR CALL WITH
2687                                     ;'BR 13$' = 000766
2688 033220                                     14$:     ;THE FOLLOWING WILL TEST DSTM=5 MFPI.
2689                                     ;
2690 033220 012737 033226 001110  MOV    #15$, $LPERR    ;SET LOOP ON ERROR POINTER TO 15$
2691 033226 012737 140340 177776 15$:     MOV    #140340,PSW    ;MAKE PREVIOUS MODE KERNEL PRESENT USER
2692 033234 012737 100000 001202  MOV    #100000,$TMP2  ;LOAD TEST LOC. VIRT. ADDR INTO LOC. $TMP2
2693 033242 012702 001204          MOV    #<$TMP2+2>,R2  ;LOAD ADDRESS OF $TMP2+2 INTO R2
2694 033246 006552          MFPI   @-(R2)        ;READ FROM PHYSICAL 60000
2695 033250 012601          MOV    (USP)+,R1     ;POP USER STACK INTO R1
2696 033252 020001          CMP    R0,R1          ;WAS DATA FETCHED SAME AS STORED
2697 033254 001401          BEQ    16$            ;BRANCH IF CORRECT DATA WAS FETCHED
2698 033256 104046          ERROR  46            ;WRONG DATA WAS FETCHED
2699                                     ;FOR TIGHTER SCOPE LOOP
2700                                     ;REPLACE ERROR CALL WITH
2701                                     ;'BR 15$' = 000763
2702 033260                                     16$:     ;THE FOLLOWING WILL TEST DSTM=6 MFPI.
```

```
2703
2704 033260 012737 033266 001110      :
2705 033266 012737 140340 177776 17$: MOV #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$.
2706 033274 005002                :MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
2707 033276 006562 100000          :CLR R2 ;MAKE REGISTER 2 A ZERO
2708 033302 012601                :MFPI 100000(R2) ;READ FROM PHYSICAL 60000
2709 033304 020001                :MOV (USP)+,R1 ;POP USER STACK INTO R1
2710 033306 001401                :CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2711 033310 104046                :BEQ 18$ ;BRANCH IF CORRECT DATA FETCHED
2712                                :ERROR 46 ;WRONG DATA WAS FETCHED
2713                                ;FOR TIGHTER SCOPE LOOP
2714                                ;REPLACE ERROR CALL WITH
2715 033312                                ;'BR 17$' = 000766
2716                                ;THE FOLLOWING WILL TEST DSTM=7 MFPI.
2717 033312 012737 033320 001110      :
2718 033320 012737 140340 177776 19$: MOV #19$, $LPERR ;SET LOOP ON ERROR POINTER TO 19$
2719 033326 012737 100000 001202      :MOV #140340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT USER
2720 033334 012702 001202                :MOV #100000,$TMP2 ;LOAD TEST LOC. VIRT. ADDR. INTO $TMP2
2721 033340 006572 000000          :MOV #$TMP2,R2 ;LOAD ADDRESS OF $TMP2 INTO R2
2722 033344 012601                :MFPI @0(R2) ;READ FROM PHYSICAL 60000
2723 033346 020001                :MOV (USP)+,R1 ;POP USER STACK INTO R1
2724 033350 001401                :CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
2725 033352 104046                :BEQ 20$ ;BRANCH IF CORRECT DATA FETCHED
2726                                :ERROR 46 ;WRONG DATA WAS FETCHED
2727                                ;FOR TIGHTER SCOPE LOOP
2728                                ;REPLACE ERROR CALL WITH
2729 033354 012737 002150 000250 20$: MOV #MGERR,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
2730 033362 012737 000340 177776      :MOV #00340,PSW ;GO BACK TO KERNEL MODE, PREVIOUS KERNEL
2731 033370 012737 032704 001110      :MOV #1$, $LPERR ;SET LOOP POINTER TO START OF TEST
2732 033376 000423                :BR TST46 ;:BRANCH TO NEXT TEXT
2733
2734
2735 033400 012637 001266          21$: MOV (KSP)+,TRAPPC ;SAVE PC & PS OF TRAP
2736 033404 012637 001270          :MOV (KSP)+,TRAPPS
2737 033410 013737 177572 001272      :MOV SR0,WASSRO ;SAVE SR0 FOR ERROR TYPEOUT
2738 033416 013737 177576 001274      :MOV SR2,WASSR2 ;SAVE SR2 FOR ERROR TYPEOUT
2739 033424 042737 160000 177572      :BIC #160000,SR0 ;CLEAR ERROR BITS IN SR0
2740 033432 104051                :ERROR 51 ;TRIED TO READ NON-RESIDENT PAGE
2741                                ;FOR TIGHTER SCOPE LOOP
2742                                ;REPLACE ERROR CALL WITH
2743                                ;A 'NOP' = 000240
2744 033434 013746 001270          :MOV TRAPPS,-(KSP) ;PUT PC & PS OF TRAP ON STACK
2745 033440 013746 001266          :MOV TRAPPC,-(KSP)
2746 033444 000002                :RTI ;RETURN TO TEST
2747
2755
(3)
(4)
(4)
(4)
(4)
(4)
(3)
(2) 033446 000004
2756 033450 012737 030340 177776 1$: TST46: SCOPE
2757 033456 106506                :MOV #030340,PSW ;MAKE PREVIOUS MODE=USER,CURRENT=KERNEL
:MFPD USP ;MFPD SHOULD ACT LIKE MFPI PUTTING
```



```
2758                                     :USER STACK POINTER ON THE KERNEL STACK
2759 033460 022706 001100                CMP    #KERSTK,KSP    :WAS SOMETHING PUSHED ON KERNEL STACK?
2760 033464 001407                       BEQ    2$             :BRANCH IF NO
2761 033466 012600                       MOV    (KSP)+,R0     :POP KERNEL STACK INTO R0
2762 033470 012701 000700                MOV    #USESTK,R1   :EXPECTING TO GET 700 AS USP
2763 033474 020001                       CMP    R0,R1        :DID GET RIGHT POINTER VALUE?
2764 033476 001403                       BEQ    3$             :BRANCH IF YES
2765 033500 104053                       ERROR   53           :WRONG THING WAS PUSHED ON STACK
2766                                     :FOR TIGHTER SCOPE LOOP
2767                                     :REPLACE ERROR CALL WITH
2768                                     :'BR 1$' = 000763
2769 033502 000401                       BR     3$             :BRANCH TO NEXT TRY
2770 033504 104054                       2$:   ERROR   54     :NOTHING PUSHED ON STACK
2771                                     :FOR TIGHTER SCOPE LOOP
2772                                     :REPLACE ERROR CALL WITH
2773                                     :'BR 1$' = 000761
2774 033506 012737 033514 001110          3$:   MOV    #4$,$LPERR   :SET LOOP ON ERROR POINTER TO 4$
2775 033514 012746 007777 001110          4$:   MOV    #7777,-(KSP) :PUSH DATA ON KERNEL STACK
2776 033520 106606                       MTPD   USP          :LOAD THE USER STACK POINTER
2777 033522 106506                       MFDP   USP          :READ USER STACK POINTER
2778 033524 012601                       MOV    (KSP)+,R1    :POP KERNEL STACK INTO R1
2779 033526 022701 007777                CMP    #7777,R1    :WAS USER STACK POINTER CHANGED?
2780 033532 001401                       BEQ    5$             :BRANCH IF YES
2781 033534 104054                       ERROR   54           :USER STACK POINTER NOT CHANGED
2782                                     :FOR TIGHTER SCOPE LOOP
2783                                     :REPLACE ERROR CALL WITH
2784                                     :'BR 4$' = 000767
2785 033536 012746 000700          5$:   MOV    #USESTK,-(KSP) :GET READY TO RESTORE USER STK. PTR.
2786 033542 106606                       MTPD   USP          :RESTORE USER STACK POINTER
2787 033544 012737 033450 001110          MOV    #1$,$LPERR   :SET LOOP POINTER TO START OF TEST
2788
2797                                     :*****
(3)                                     :*TEST 47      MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)
(4)                                     :*
(4)                                     :* THIS TEST CHECKS THAT IF BOTH PREVIOUS AND CURRENT MODES
(4)                                     :* ARE KERNEL, AND THE SOURCE MODE IS 0, THE DESTINATION
(4)                                     :* STACK IS NOT DECREMENTED BEFORE ACCESS.
(4)                                     :* 'MFP! KSP' SHOULD PUSH THE NON-DECREMENTED VALUE
(4)                                     :* OF KSP (1100) ONTO THE STACK (AT LOC. 1076).
(3)                                     :*****
(2) 033552 000004                       TST47: SCOPE
2798 033554 005037 177776          1$:   CLR    @#PSW        :SET PREVIOUS = CURRENT = KERNEL
2799 033560 012700 001100                MOV    #STACK,R0    :SETUP VALUE FOR STACK POINTER
2800 033564 010006                       MOV    R0,KSP       :LOAD STACK POINTER
2801 033566 006506                       MFPI   KSP          :THE VALUE 'STACK' SHOULD BE PUSHED
2802                                     :BEFORE BEING DECREMENTED
2803 033570 011601                       MOV    (KSP),R1     :READ DATA WHICH WAS PUSHED
2804 033572 020001                       CMP    R0,R1        :WAS THE ORIGINAL VALUE OF THE
2805                                     :STACK POINTER PUSHED?
2806 033574 001401                       BEQ    2$             :BRANCH IF YES
2807 033576 104046                       ERROR   46           :MFPI FETCHED WRONG DATA
2808                                     :FOR TIGHTER SCOPE LOOP
2809                                     :REPLACE ERROR CALL WITH
2810                                     :'BR 1$' = 000766
2811 033600 005740          2$:   TST    -(R0)        :SETUP EXPECTED STACK POINTER VALUE
2812 033602 020600                CMP    KSP,R0       :WAS THE STACK POINTER DECREMENTED?
```

CJKDACO KTF11-AA MMU DIAG
CJKDAC.P11 12-MAR-80 07:56

MACY11 30A(1052)
T47

12-MAR-80 08:00
MOVE FROM PREVIOUS I=SPACE (PREVIOUS=CURRENT=KERNEL)

N 6
PAGE 1-64

SEQ 0078

2813 033604 001401
2814 033606 104050
2815
2816
2817
2818 033610 012706 001100

BEQ 3\$
ERROR 50

3\$: MOV #STACK,KSP

:BRANCH IF YES
:STACK NOT PUSHED BY THE MFPI
:FOR TIGHTER SCOPE LOOP
:REPLACE ERROR CALL WITH
: 'BR 1\$' = 000762
:RESTORE STACK POINTER


```
2820 .SBTTL END OF PASS ROUTINE
2821
2822 ::*****
2823 ::*INCREMENT THE PASS NUMBER ($PASS)
2824 ::*TYPE 'END OF PASS #XXXX'
2825 ::*TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYY''
2826 ::*WHERE XXXX AND YYYY ARE DECIMAL NUMBERS
2827 ::*IF SW12=1 INHIBIT TRACE TRAP
2828 ::*IF THERES A MONITOR GO TO IT
2829 ::*IF THERE ISN'T JUMP TO LOOP
2830
2831 $EOP:
2832 033614 000004 SCOPE
2833 033616 005037 001102 CLR $TSTNM ;;ZERO THE TEST NUMBER
2834 033622 005037 001212 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
2835 033626 005237 001234 INC $PASS ;;INCREMENT THE PASS NUMBER
2836 033632 042737 100000 001234 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
2837 033640 005327 DEC (PC)+ ;;LOOP?
2838 033642 000001 $EOPCT: .WORD 1
2839 033644 003110 BGT $DOAGN ;;YES
2840 033646 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
2841 033650 000001 $ENDCT: .WORD 1
2842 033652 033642 $EOPCT
2843 033654 104401 033662 TYPE ,65$ ;;TYPE ASCIZ STRING
2844 033660 000407 BR 64$ ;;GET OVER THE ASCIZ
2845 033662 006412 047105 020104 65$: .ASCIZ <12><15>/END PASS #/
033670 040520 051523 021440
033676 000

2846 .EVEN
2847 033700 64$:
2848 033700 013746 001234 MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
2849
2850 033704 104405 TYPDS ;;TYPE PASS NUMBER
2851 033706 104401 033714 TYPE ,67$ ;;GO TYPE--DECIMAL ASCIZ WITH SIGN
2852 033712 000425 BR 66$ ;;TYPE ASCII STRING
2853 033714 035411 047524 040524 67$: .ASCIZ / ;TOTAL ERRORS SINCE LAST START AT 200 /
033722 020114 051105 047522
033730 051522 051440 047111
033736 042503 046040 051501
033744 020124 052123 051101
033752 020124 052101 031040
033760 030060 020040 000

2854 .EVEN
2855 033766 66$:
2856 033766 013746 001112 MOV $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
2857
2858 033772 104405 TYPDS ;;TOTAL NUMBER OF ERRORS
2859 033774 104401 001223 TYPE , $CRLF ;;GO TYPE--DECIMAL ASCII WITH SIGN
2860 034000 013700 000042 $GET42: MOV @#42,R0 ;;TYPE CARRIAGE RETURN, LINE FEED
2861 034004 001411 BEQ DOAGIN ;;GET MONITOR ADDRESS
2862 034006 005046 CLR -(SP) ;;BRANCH IF NO MONITOR
2863 034010 012746 034016 MOV #$CLR.T,-(SP) ;;INSURE THE 'T' BIT IS CLEAR
2864 034014 000442 BR $RTRN ;;SETUP FOR AN RTI OR RTT
2865
2866 034016 000005 $CLR.T: RESET ;;GO DO AN RTI OR RTT TO LOAD THE PSW
2867 034020 004710 $ENDAD: JSR PC,(R0) ;;WITH A CLEARED 'T' BIT
;;CLEAR THE WORLD
;;GO TO MONITOR
```



```
(3) 034440 010437 001172      MOV      R4,$REG4      ;SAVE THE CONTENTS OF R4
(3) 034444 010537 001174      MOV      R5,$REG5      ;SAVE THE CONTENTS OF R5
(3) 034450 113737 001102 001262  MOVB     $TSTNM,TESTNO ;SAVE THE TEST NUMBER
(1) 034456 105237 001103      7$:     INCB     $ERFLG    ;SET THE ERROR FLAG
(1) 034462 001775              BEQ      7$            ;DON'T LET THE FLAG GO TO ZERO
(1) 034464 013777 001102 144450  MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
(1) 034472 032777 002000 144440  BIT     #BIT10,@SWR    ;BELL ON ERROR?
(1) 034500 001402              BEQ      1$            ;NO - SKIP
(1) 034502 104401 001216      TYPE    ,SBELL        ;RING BELL
(1) 034506 005237 001112      1$:     INC     $ERTTL    ;COUNT THE NUMBER OF ERRORS
(1) 034512 011637 001116      MOV     (SP),$ERRPC    ;GET ADDRESS OF ERROR INSTRUCTION
(1) 034516 162737 000002 001116  SUB     #2,$ERRPC
(1) 034524 117737 144366 001114  MOVB    @ $ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
(1) 034532 032777 020000 144400  BIT     #BIT13,@SWR    ;SKIP TYPEOUT IF SET
(1) 034540 001004              BNE     20$           ;SKIP TYPEOUTS
(1) 034542 004737 034654      JSR     PC,ERRTYP     ;GO TO USER ERROR ROUTINE
(1) 034546 104401 001223      TYPE    ,SCRLF
(1) 034552              20$:
(1) 034552 122737 000001 001246  CMPB    #APTENV,$ENV   ;:RUNNING IN APT MODE
(1) 034560 001007              BNE     2$            ;:NO,SKIP APT ERROR REPORT
(1) 034562 113737 001114 034574  MOVB    $ITEMB,21$    ;:SET ITEM NUMBER AS ERROR NUMBER
(1) 034570 004737 037216      JSR     PC,$ATY4     ;:REPORT FATAL ERROR TO APT
(1) 034574      000          21$:     .BYTE    0
(1) 034575      000          .BYTE    0
(1) 034576 000777          22$:     BR      22$           ;:APT ERROR LOOP
(1) 034600 005777 144334      2$:     TST     @SWR      ;:HALT ON ERROR
(1) 034604 100002          BPL     3$            ;:SKIP IF CONTINUE
(1) 034606 000000          HALT    ;:HALT ON ERROR!
(1) 034610 104410          CKSWR   ;:TEST FOR CHANGE IN SOFT-SWR
(1) 034612 032777 001000 144320  3$:     BIT     #BIT09,@SWR ;:LOOP ON ERROR SWITCH SET?
(1) 034620 001402          BEQ     4$            ;:BR IF NO
(1) 034622 013716 001110      MOV     $LPERR,(SP)   ;:FUDGE RETURN FOR LOOPING
(1) 034626 005737 001214      4$:     TST     $ESCAPE   ;:CHECK FOR AN ESCAPE ADDRESS
(1) 034632 001402          BEQ     5$            ;:BR IF NONE
(1) 034634 013716 001214      MOV     $ESCAPE,(SP) ;:FUDGE RETURN ADDRESS FOR ESCAPE
(1) 034640          5$:
(1) 034640 022737 034020 000042  CMP     # $ENDAD,@#42 ;:ACT-11 AUTO-ACCEPT?
(1) 034646 001001          BNE     6$            ;:BRANCH IF NO
(1) 034650 000000          HALT    ;:YES
(1) 034652          6$:
(1) 034652 000002          RTI     ;:RETURN
```

2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912

```
.SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
;*****
;THIS ROUTINE USES THE 'ITEM CONTROL BYTE' ($ITEMB) TO DETERMINE WHICH
;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
;
;NOTES:
;*1) THIS ROUTINE PROVIDES AN AUTOMATIC 'CARRIAGE RETURN-LINE FEED'
;    FOR 'EM', 'DH', AND 'DT'.
;*2) TWO SPACES ARE TYPED AFTER EACH NUMBER FOR 'DT'
;*3) FOR $ITEMB=0, JUST THE ERROR PC IS TYPED
;*4) THE AVAILABLE FORMATS FOR TYPING DATA ARE:
;    DF      FORMAT
;    0       TYPE A 6 DIGIT OCTAL NUMBER (FROM 16-BIT BINARY)
```



```

2913      :*      1      TYPE A DECIMAL NUMBER WITHOUT LEADING ZEROS
2914      :*      2      TYPE A 16 DIGIT BINARY NUMBER
2915      :*      3      TYPE A 6 DIGIT OCTAL NUMBER (FROM 18-BIT BINARY)
2916      :*
2917      :*
2918      034654      ERRRTYP:
2919      034654      104401      001223      TYPE      , $CRLF      ;'CARRIAGE RETURN' & 'LINE FEED'
2920      034660      010046      MOV      R0,-(SP)      ;SAVE R0
2921      034662      005000      CLR      R0      ;PICKUP THE ITEM INDEX
2922      034664      153700      001114      BISB     @#$ITEMB,R0
2923      034670      001004      BNE     1$      ;IF ITEM NUMBER IS ZERO, JUST
2924      ;TYPE THE PC OF THE ERROR
2925      034672      013746      001116      MOV      $ERRPC,-(SP)      ;SAVE $ERRPC FOR TYPEOUT
2926      ;ERROR ADDRESS
2927      ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2928      (1)      034676      104402      TYPDC
2929      034700      000471      BR      13$      ;GET OUT
2930      034702      005300      1$:      DEC      R0      ;ADJUST THE INDEX SO THAT IT WILL
2931      034704      006300      ASL     R0      ;WORK FOR THE ERROR TABLE
2932      034706      006300      ASL     R0
2933      034710      006300      ASL     R0
2934      034712      062700      001316      ADD     #$ERRTB,R0      ;FORM TABLE POINTER
2935      034716      012037      034726      MOV     (R0)+,2$      ;PICKUP 'ERROR MESSAGE' POINTER
2936      034722      001404      BEQ     3$      ;SKIP TYPEOUT IF NO POINTER
2937      034724      104401      TYPE
2938      034726      000000      2$:      .WORD  0      ;TYPE THE 'ERROR MESSAGE'
2939      034730      104401      001223      TYPE   , $CRLF      ;'ERROR MESSAGE' POINTER GOES HERE
2940      034734      012037      034744      3$:      MOV     (R0)+,4$      ;'CARRIAGE RETURN' & 'LINE FEED'
2941      034740      001404      BEQ     5$      ;PICKUP 'DATA HEADER' POINTER
2942      034742      104401      TYPE
2943      034744      000000      4$:      .WORD  0      ;SKIP TYPEOUT IF 0
2944      034746      104401      001223      TYPE   , $CRLF      ;TYPE THE 'DATA HEADER'
2945      034752      010146      5$:      MOV     R1,-(SP)      ;'DATA HEADER' POINTER GOES HERE
2946      034754      012001      MOV     (R0)+,R1      ;'CARRIAGE RETURN' & 'LINE FEED'
2947      034756      001441      BEQ     12$      ;SAVE R1
2948      034760      012000      MOV     (R0)+,R0      ;PICKUP 'DATA TABLE' POINTER
2949      034762      105710      6$:      TSTB   (R0)      ;BR IF NO DATA TO BE TYPED
2950      034764      001003      BNE     7$      ;PICKUP 'DATA FORMAT' POINTER
2951      ;IS IT FORMAT 0?
2952      ;BR IF NO
2953      ;*THIS CODE IS FOR OCTAL (16-BIT) FORMAT (DF=0)
2954      034766      013146      MOV     @ (R1)+,-(SP)      ;SAVE @ (R1)+ FOR TYPEOUT
2955      (1)      034770      104402      TYPDC      ;GO TYPE--OCTAL ASCII(ALL DIGITS)
2956      034772      000425      BR      11$
2957      ;*THIS CODE IS FOR DECIMAL FORMAT (DF=1)
2958      7$:      CMPB   (R0),#1      ;IS IT FORMAT 1?
2959      034774      121027      000001      BNE     8$      ;BRANCH IF NO
2960      035000      001003      MOV     @ (R1)+,-(SP)      ;SAVE @ (R1)+ FOR TYPEOUT
2961      035002      013146      TYPDS      ;GO TYPE--DECIMAL ASCII WITH SIGN
2962      (1)      035004      104405      BR      11$
2963      035006      000417      BR      11$
2964      ;*THIS CODE IS FOR BINARY FORMAT (DF=2)
2965      8$:      CMPB   (R0),#2      ;IS IT FORMAT 2?
2966      035010      121027      000002      BNE     9$      ;BRANCH IF NO
2967      035014      001003      MOV     @ (R1)+,-(SP)      ;SAVE @ (R1)+ FOR TYPEOUT
2968      035016      013146      TYPBN      ;GO TYPE--BINARY ASCII
2969      (1)      035020      104406      BR      11$
2970      035022      000411      BR      11$

```

2964
2965
2966 035024 012146
2967 035026 004737 040270
2968 035032 062716 000005
2969 035036 012637 035044
2970 035042 104401
2971 035044 000000
2972
2973 035046 005711
2974 035050 001404
2975 035052 104401 035074
2976 035056 105720
2977 035060 000740
2978
2979 035062 012601
2980 035064 012600
2981 035066 104401 001223
2982 035072 000207
2983 035074 020040 000
2984 035100
2985

```
.*THIS CODE IS FOR OCTAL (18-BIT) FORMAT (DF=3)
9$:  MOV      (R1)+,-(SP)      ;PUT ADDRESS OF FIRST LOC. ON STACK
     JSR      PC,$DB20        ;CONVERT TWO LOCS. TO AN ASCII STRING
     ADD      #5,(SP)         ;ONLY NEED 6 CHARACTERS NOT 11
     MOV      (SP)+,10$       ;PUT ADDRESS OF ASCII CHARS. AT 10$
     TYPE     TYPE            ;TYPE OCTAL VALUE OF 18-BIT BINARY NO.
10$:  .WORD   0
11$:  TST     (R1)             ;IS THERE ANOTHER NUMBER?
     BEQ     12$              ;BR IF NO
     TYPE     ,14$            ;TYPE TWO(2) SPACES
     TSTB    (R0)+           ;POINT TO NEW 'DATA FORMAT'
     BR      6$               ;LOOP
12$:  MOV     (SP)+,R1         ;RESTORE R1
13$:  MOV     (SP)+,R0         ;RESTORE R0
     TYPE     ,CRLF           ;'CARRIAGE RETURN' & 'LINE FEED'
     RTS     PC               ;RETURN
14$:  .ASCIZ  / /             ;TWO(2) SPACES
     .EVEN
```


2987
2988
2989
2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042

035100 033727 177776 000020
035106 001411
035110 013746 177776
035114 011637 001276
035120 042716 000020
035124 012746 035132
035130 000006
035132 000207

035134 033727 001276 000020
035142 001410
035144 013746 001276
035150 012737 000340 001276
035156 012746 035164
035162 000006
035164 000207

035166 012702 000010
035172 012701 172300
035176 012721 177777
035202 077203
035204 012702 000010
035210 012701 172340
035214 012721 177777
035220 077203

```
.SBTTL ***** SUBROUTINES USED BY THIS PROGRAM *****  
.SBTTL TURN OFF T-BIT AND SAVE CURRENT PSW  
:*****  
:* THIS SUBROUTINE IS USED TO TURN OFF THE TRACE TRAP BIT IN THE PSW  
:* IF IT IS ON. THE PROCESSOR STATUS IS SAVED IN 'TBITPS' SO THAT  
:* THE PSW CAN BE RESTORED TO ITS PREVIOUS CONDITION WHEN CONDITIONS  
:* WARRANT T-BIT TRAPPING.  
:*****  
TOFF: BIT PSW,#TBIT ;IS THE T-BIT SET IN THE PSW?  
BEQ 1$ ;EXIT IF NO  
MOV PSW,-(SP) ;PUSH PRESENT PSW ON THE STACK  
MOV (SP),TBITPS ;ALSO SAVE IT IN 'TBITPS' FOR  
;RESTORING LATER  
BIC #TBIT,(SP) ;CLEAR THE T-BIT (BIT 4) IN THE PSW  
MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK  
RTT ;'RETURN' TO 1$ WITH T-BIT OFF  
1$: RTS PC ;RETURN TO PROGRAM  
  
.SBTTL TURN ON T-BIT AND RESTORE PREVIOUS PSW  
:*****  
:* THIS SUBROUTINE IS USED TO RESTORE THE PROCESSOR STATUS TO ITS  
:* PREVIOUS CONDITION BY RESTORING THE 'T-BIT PSW' SAVED BY THE  
:* 'TOFF' SUBROUTINE IN THE 'TBITPS' LOCATION.  
:*****  
TON: BIT TBITPS,#TBIT ;WAS T-BIT ON IN THE PREVIOUS PSW?  
BEQ 1$ ;EXIT IF NO  
MOV TBITPS,-(SP) ;PUSH PREVIOUS PSW ON THE STACK  
MOV #340,TBITPS ;RESET THE 'TBITPS' LOCATION  
MOV #1$,-(SP) ;PUSH PC OF 'RTS' ON STACK  
RTT ;'RETURN' TO 1$ WITH T-BIT RESTORED  
1$: RTS PC ;RETURN TO PROGRAM  
  
.SBTTL SET ALL WRITEABLE BITS IN ALL PAR/PDR'S  
:*****  
:* THIS SUBROUTINE IS USED BY THE PAR/PDR DUAL ADDRESSING TEST  
:* TO SET ALL WRITEABLE BITS IN ALL KERNEL AND USE PAR'S AND  
:* PDR'S TO A 1. THE "INITIAL STATE" OF HAVING ALL BITS=1 IS  
:* USED TO SEE THAT ONLY ONE REGISTER IS CLEARED IN RESPONSE TO  
:* A SINGLE PAR OR PDR ADDRESS.  
:*****  
SETREG: MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8  
MOV #KIPDRO,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1  
1$: MOV #-1,(R1)+ ;SET BITS IN KERNEL PDR TO 1  
SOB R2,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S LOADED  
MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8  
MOV #KIPARO,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1  
2$: MOV #-1,(R1)+ ;SET BITS IN A KERNEL PAR TO 1  
SOB R2,2$ ;LOOP TO 2$ UNTIL ALL KERNEL PAR'S LOADED
```

3043 035222 012702 000010
3044 035226 012701 177600
3045 035232 012721 177777
3046 035236 077203
3047 035240 012702 000010
3048 035244 012701 177640
3049 035250 012721 177777
3050 035254 077203
3051 035256 000207

```
MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPDR0,R1 ;LOAD ADDRESS OF FIRST PDR INTO R1
3$: MOV #-1,(R1)+ ;SET BITS IN A USER PDR TO 1
SOB R2,3$ ;LOOP TO 3$ UNTIL ALL USER PDR'S LOADED
MOV #10,R2 ;LOAD LOOP COUNTER WITH AN 8
MOV #UIPAR0,R1 ;LOAD ADDRESS OF FIRST PAR INTO R1
4$: MOV #-1,(R1)+ ;SET BITS IN A USER PAR TO 1
SOB R2,4$ ;LOOP TO 4$ UNTIL ALL USER PAR'S LOADED
RTS PC ;RETURN TO TEST
```

.SBTTL READ & COMPARE KERNEL & USER PAR/PDR'S

```
*****
*
* THIS SUBROUTINE IS USED BY PAR/PDR DUAL ADDRESSING TEST TO
* READ ALL THE PAR'S AND PDR'S TO SEE THAT ONLY ONE REGISTER
* WAS CLEARED IN RESPONSE TO A SINGLE PAR OR PDR ADDRESS.
* ANY FAILURES FOUND BY THE PAR/PDR DUAL ADDRESSING TEST WILL
* BE REPORTED BY THIS SUBROUTINE.
*
*****
```

3063 035260
3064 035260 012701 172300
3065 035264 012704 000010
3066 035270 012705 077416
3067 035274 021105
3068 035276 001404
3069 035300 020100
3070 035302 001402
3071 035304 011102
3072 035306 104016

```
CMPREG:
MOV #KIPDR0,R1 ;LOAD ADDRESS OF FIRST KERNEL PDR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
1$: CMP (R1),R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 2$ ;BRANCH IF YES
CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 2$ ;BRANCH IF YES
MOV (R1),R2 ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
ERROR 16 ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN 'RTS PC' = 000207
```

3076 035310 062701 000002
3077 035314 077411
3078 035316 012701 172340
3079 035322 012704 000010
3080 035326 012705 177777
3081 035332 021105
3082 035334 001404
3083 035336 020100
3084 035340 001402
3085 035342 011102
3086 035344 104016

```
2$: ADD #2,R1 ;FORM NEXT ADDRESS
SOB R4,1$ ;LOOP TO 1$ UNTIL ALL KERNEL PDR'S CHECKED
MOV #KIPAR0,R1 ;LOAD ADDRESS OF FIRST KERNEL PAR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #177777,R5 ;PUT EXPECTED PAR CONTENTS IN R5
3$: CMP (R1),R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 4$ ;BRANCH IF YES
CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 4$ ;BRANCH IF YES
MOV (R1),R2 ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
ERROR 16 ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
;FOR TIGHTER SCOPE LOOP
;REPLACE ERROR CALL WITH
;AN 'RTS PC' = 000207
```

3090 035346 062701 000002
3091 035352 077411
3092 035354 012701 177600
3093 035360 012704 000010
3094 035364 012705 077416
3095 035370 021105
3096 035372 001404
3097 035374 020100
3098 035376 001402

```
4$: ADD #2,R1 ;FORM NEXT ADDRESS
SOB R4,3$ ;LOOP TO 3$ UNTIL ALL KERNEL PAR'S CHECKED
MOV #UIPDR0,R1 ;LOAD ADDRESS OF FIRST USER PDR IN R1
MOV #10,R4 ;LOAD LOOP COUNTER WITH AN 8
MOV #77416,R5 ;PUT EXPECTED PDR CONTENTS IN R5
5$: CMP (R1),R5 ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
BEQ 6$ ;BRANCH IF YES
CMP R1,R0 ;WAS IT THE REG. THAT WAS CLEARED?
BEQ 6$ ;BRANCH IF YES
```



```

3099 035400 011102          MOV    (R1),R2          ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
3100 035402 104016          ERROR  16              ;A PDR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
3101                                     ;FOR TIGHTER SCOPE LOOP
3102                                     ;REPLACE ERROR CALL WITH
3103                                     ;AN 'RTS PC' = 000207
3104 035404 062701 000002    6$:   ADD    #2,R1          ;FORM NEXT ADDRESS
3105 035410 077411          SOB    R4,5$           ;LOOP TO 5$ UNTIL ALL USER PDR'S CHECKED
3106 035412 012701 177640    MOV    #UIPAR0,R1      ;LOAD ADDRESS OF FIRST USER PAR IN R1
3107 035416 012704 000010    MOV    #10,R4          ;LOAD LOOP COUNTER WITH AN 8
3108 035422 012705 177777    MOV    #177777,R5      ;PUT EXPECTED PAR CONTENTS IN R5
3109 035426 021105          7$:   CMP    (R1),R5         ;ARE ALL WRITEABLE BITS SET AS EXPECTED?
3110 035430 001404          BEQ    8$              ;BRANCH IF YES
3111 035432 020100          CMP    R1,R0           ;WAS IT THE REG. THAT WAS CLEARED?
3112 035434 001402          BEQ    8$              ;BRANCH IF YES
3113 035436 011102          MOV    (R1),R2          ;SAVE CONTENTS OF IMPROPERLY CLEARED REGISTER
3114 035440 104016          ERROR  16              ;A PAR WAS EFFECTED BY CLEARING A DIFFERENT PAR/PDR
3115                                     ;FOR TIGHTER SCOPE LOOP
3116                                     ;REPLACE ERROR CALL WITH
3117                                     ;AN 'RTS PC' = 000207
3118 035442 062701 000002    8$:   ADD    #2,R1          ;FORM NEXT ADDRESS
3119 035446 077411          SOB    R4,7$           ;LOOP TO 7$ UNTIL ALL USER PAR'S CHECKED
3120 035450 000207          RTS    PC              ;RETURN TO TEST

```

.SBTTL CONVERT VIRTUAL ADDRESS TO PHYSICAL ADDRESS

THIS SUBROUTINE IS USED TO FORM AN 18-BIT PHYSICAL ADDRESS (PBA) FROM THE 16-BIT VIRTUAL ADDRESS (VBA) AND THE APPROPRIATE PAGE ADDRESS REGISTER (PAR). THE SAME METHOD USED BY THE MEMORY MANAGEMENT LOGIC IS USED. VBA <15:13> SELECTS WHICH PAR/PDR IS TO BE USED, VBA <5:0>+PBA <5:0>, AND VBA <12:6> IS ADDED TO PAR <11:00> TO GIVE PBA <17:6>. BITS <17:16> OF THE PHYSICAL ADDRESS ARE LEFT IN LOC. 'PBAHI' AND BITS <15:00> ARE LEFT IN LOC. 'PBALO'. THE PSW'S 'CURRENT MODE' BITS ARE USED TO SELECT THE KERNEL OR USER PAR/PDR'S. THE ROUTINE IS ENTERED WITH LOC. 'VIRT1' CONTAINING THE 16-BIT VIRTUAL ADDRESS.

FORMPA:

```

3139 035452          (2) 035452 010046          MOV    R0,-(SP)         ;;PUSH R0 ON STACK
3140 035456          (2) 035454 010246          MOV    R2,-(SP)         ;;PUSH R2 ON STACK
3141 035462          012702 172340          MOV    #KIPAR0,R2      ;LOAD ADDRESS OF FIRST KERNEL PAR IN R2
3142 035470          032737 140000 177776    BIT    #140000,PSW      ;IN USER MODE?
3143 035472          001402          BEQ    1$              ;BRANCH IF NO
3144 035476          012702 177640          MOV    #UIPAR0,R2      ;LOAD ADDRESS OF FIRST USER PAR IN R2
3145 035502          013700 001306          1$:   MOV    VIRT1,R0        ;LOAD VIRTUAL ADDR. (VBA) INTO R0
3146 035506          072027 177764          ASH    #-14,R0         ;GET BITS <15:13> DOWN TO BITS <3:1>
3147 035512          042700 177761          BIC    #177761,R0      ;MASK OF ALL BITS BUT BITS <3:1>
3148 035514          060002          ADD    R0,R2           ;ADD OFFSET TO BASE PAR ADDRESS
3149 035516          011200          MOV    (R2),R0         ;GET BITS <11:00> FROM APPROPRIATE PAR
3150 035520          013737 001306 001312    MOV    R0,R2           ;COPY PAR BITS <11:00> INTO R2
3151 035526          042737 160000 001312    MOV    VIRT1,PBALO     ;PUT VIRTUAL ADDR. IN LOC. 'PBALO'
3152 035534          072227 177766          BIC    #160000,PBALO   ;CLEAR OFF BITS <15:13> OF ORIGINAL VBA
3152 035534          072227 177766          ASH    #-12,R2        ;GET PAR <11:00> DOWN TO BITS <1:0> OF R2

```

3153	035540	042702	177774	BIC	#177774,R2	:CLEAR OFF ALL BITS BUT BITS <1:0>
3154	035544	072027	000006	ASH	#6,R0	:SHIFT PAR<9:0> TO <15:6> OF R0
3155	035550	042700	000077	BIC	#77,R0	:CLEAR BITS <5:0> OF R0
3156	035554	060037	001312	ADD	R0,PBALO	:IN EFFECT, ADD VBA<12:0> TO PAR<9:0>
3157						:(PAR<9:0> IN BITS <15:6> OF R0)
3158	035560	005502		ADC	R2	:ADD ANY CARRY TO R2
3159	035562	010237	001314	MOV	R2,PBAHI	:PUT BITS <17:16> OF PHYSICAL ADDR. IN PBAHI
3160	035566	012602		MOV	(SP)+,R2	::POP STACK INTO R2
(2)	035570	012600		MOV	(SP)+,R0	::POP STACK INTO R0
3161	035572	000207		RTS	PC	:RETURN TO PROGRAM
3162						
3163						


```
(1) 036006 062706 000006 11$: ADD #6,SP ;:CLEAR UP STACK
(1) 036012 104401 001223 14$: TYPE ,SCLRF ;:ECHO <CR> AND <LF>
(1) 036016 123727 001135 000001 CMPB $INTAG,#1 ;:RE-ENABLE TTY KBD INTERRUPTS?
(1) 036024 001003 BNE 15$ ;:BRANCH IF NOT
(1) 036026 012777 000100 143110 MOV #100,@$TKS ;:RE-ENABLE TTY KBD INTERRUPTS
(1) 036034 000002 15$: RTI ;:RETURN
(1) 036036 004737 037130 16$: JSR PC,$TYPEC ;:ECHO CHAR
(1) 036042 021627 000060 CMP (SP),#60 ;:CHAR < 0?
(1) 036046 002420 BLT 18$ ;:BRANCH IF YES
(1) 036050 021627 000067 CMP (SP),#67 ;:CHAR > 7?
(1) 036054 003015 BGT 18$ ;:BRANCH IF YES
(1) 036056 042726 000060 BIC #60,(SP)+ ;:STRIP-OFF ASCII
(1) 036062 005766 000002 TST 2(SP) ;:IS THIS THE FIRST CHAR
(1) 036066 001403 BEQ 17$ ;:BRANCH IF YES
(1) 036070 006316 ASL (SP) ;:NO, SHIFT PRESENT
(1) 036072 006316 ASL (SP) ;: CHAR OVER TO MAKE
(1) 036074 006316 ASL (SP) ;: ROOM FOR NEW ONE.
(1) 036076 005266 000002 17$: INC 2(SP) ;:KEEP COUNT OF CHAR
(1) 036102 056616 177776 BIS -2(SP),(SP) ;:SET IN NEW CHAR
(1) 036106 000667 BR 7$ ;:GET THE NEXT ONE
(1) 036110 104401 001222 18$: TYPE ,SQUES ;:TYPE ?<CR><LF>
(1) 036114 000720 BR 20$ ;:SIMULATE CONTROL-U
(1) .DSABL LSB
```

```
(1) ;:*****
(1) ;:*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1) ;:*CALL:
(1) ;:* RDCHR ;:INPUT A SINGLE CHARACTER FROM THE TTY
(1) ;:* RETURN HERE ;:CHARACTER IS ON THE STACK
(1) ;:* ;:WITH PARITY BIT STRIPPED OFF
(1) ;:
(1) ;:
```

```
(1) 036116 011646 $RDCHR: MOV (SP),-(SP) ;:PUSH DOWN THE PC
(1) 036120 016666 000004 000002 MOV 4(SP),2(SP) ;:SAVE THE PS
(1) 036126 105777 143012 1$: TSTB @$TKS ;:WAIT FOR
(1) 036132 100375 BPL 1$ ;:A CHARACTER
(1) 036134 117766 143006 000004 MOVB @$TKB,4(SP) ;:READ THE TTY
(1) 036142 042766 177600 000004 BIC #^C<177>,4(SP) ;:GET RID OF JUNK IF ANY
(1) 036150 026627 000004 000023 CMP 4(SP),#23 ;:IS IT A CONTROL-S?
(1) 036156 001013 BNE 3$ ;:BRANCH IF NO
(1) 036160 105777 142760 2$: TSTB @$TKS ;:WAIT FOR A CHARACTER
(1) 036164 100375 BPL 2$ ;:LOOP UNTIL ITS THERE
(1) 036166 117746 142754 MOVB @$TKB,-(SP) ;:GET CHARACTER
(1) 036172 042716 177600 BIC #^C177,(SP) ;:MAKE IT 7-BIT ASCII
(1) 036176 022627 000021 CMP (SP)+,#21 ;:IS IT A CONTROL-Q?
(1) 036202 001366 BNE 2$ ;:IF NOT DISCARD IT
(1) 036204 000750 BR 1$ ;:YES, RESUME
(1) 036206 026627 000004 000140 3$: CMP 4(SP),#140 ;:IS IT UPPER CASE?
(1) 036214 002407 BLT 4$ ;:BRANCH IF YES
(1) 036216 026627 000004 000175 CMP 4(SP),#175 ;:IS IT A SPECIAL CHAR?
(1) 036224 003003 BGT 4$ ;:BRANCH IF YES
(1) 036226 042766 000040 000004 BIC #40,4(SP) ;:MAKE IT UPPER CASE
(1) 036234 000002 4$: RTI ;:GO BACK TO USER
```

```
(2) ;:*****
(1) ;:*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
```



```

(1)          ;*CALL:
(1)          ;*      RDLIN          ;;INPUT A STRING FROM THE TTY
(1)          ;*      RETURN HERE    ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)          ;*                               ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)          $RDLIN: MOV      R3,-(SP)    ;;SAVE R3
(1)          036236 010346          CLR      -(SP)    ;;CLEAR THE RUBOUT KEY
(1)          036240 005046          1$:      MOV      #$TTYIN,R3    ;;GET ADDRESS
(1)          036242 012703 036514    2$:      CMP      #$TTYIN+8.,R3    ;;BUFFER FULL?
(1)          036246 022703 036524    BLOS     4$      ;;BR IF YES
(1)          036252 101467          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
(1)          036254 104411          MOVVB    (SP)+,(R3)    ;;GET CHARACTER
(1)          036256 112613          CMPB    #3,(R3)    ;;IS IT A CONTROL-C?
(1)          036260 122713 000003    BNE     10$     ;;BRANCH IF NO
(1)          036264 001006          TYPE    ,SCNTLC    ;;TYPE A CONTROL-C (^C)
(1)          036266 104401 036524    TST     (SP)+      ;;CLEAN RUBOUT KEY OFF OF THE STACK
(1)          036272 005726          MOV     (SP)+,R3    ;;RESTORE R3
(1)          036274 012603          JMP     CNTRLC     ;;GOTO CONTROL-C RESTART
(1)          036276 000137 036566    10$:    CMPB    #177,(R3)    ;;IS IT A RUBOUT
(1)          036302 122713 000177    BNE     5$      ;;BR IF NO
(1)          036306 001022          TST     (SP)      ;;IS THIS THE FIRST RUBOUT?
(1)          036310 005716          BNE     6$      ;;BR IF NO
(1)          036312 001007          MOVVB   #' \ ,9$    ;;TYPE A BACK SLASH
(1)          036314 112737 000134 036512    TYPE    ,9$
(1)          036322 104401 036512    MOV     #-1,(SP)    ;;SET THE RUBOUT KEY
(1)          036326 012716 177777    6$:      DEC     R3      ;;BACKUP BY ONE
(1)          036332 005303          CMP     R3,$TTYIN  ;;STACK EMPTY?
(1)          036334 020327 036514    BLO     4$      ;;BR IF YES
(1)          036340 103434          MOVVB   (R3),9$    ;;SETUP TO TYPEOUT THE DELETED CHAR.
(1)          036342 111337 036512    TYPE    ,9$      ;;GO TYPE
(1)          036346 104401 036512    BR      2$      ;;GO READ ANOTHER CHAR.
(1)          036352 000735          TST     (SP)      ;;RUBOUT KEY SET?
(1)          036354 005716          BEQ     7$      ;;BR IF NO
(1)          036356 001406          MOVVB   #' \ ,9$    ;;TYPE A BACK SLASH
(1)          036360 112737 000134 036512    TYPE    ,9$
(1)          036366 104401 036512    CLR     (SP)      ;;CLEAR THE RUBOUT KEY
(1)          036372 005016          CMPB    #25,(R3)    ;;IS CHARACTER A CTRL U?
(1)          036374 122713 000025    7$:      BNE     8$      ;;BR IF NO
(1)          036400 001003          TYPE    ,SCNTLU    ;;TYPE A CONTROL 'U'
(1)          036402 104401 036531    BR      1$      ;;GO START OVER
(1)          036406 000715          CMPB    #22,(R3)    ;;IS CHARACTER A '^R'?
(1)          036410 122713 000022    8$:      BNE     3$      ;;BRANCH IF NO
(1)          036414 001011          CLRB   (R3)      ;;CLEAR THE CHARACTER
(1)          036416 105013          TYPE    ,SCRLF     ;;TYPE A 'CR' & 'LF'
(1)          036420 104401 001223    TYPE    ,TTYIN     ;;TYPE THE INPUT STRING
(1)          036424 104401 036514    BR      2$      ;;GO PICKUP ANOTHER CHACTER
(1)          036430 000706          TYPE    ,SQUES     ;;TYPE A '?'
(1)          036432 104401 001222    4$:      BR      1$      ;;CLEAR THE BUFFER AND LOOP
(1)          036436 000701          MOVVB   (R3),9$    ;;ECHO THE CHARACTER
(1)          036440 111337 036512    3$:      TYPE    ,9$
(1)          036444 104401 036512    CMPB    #15,(R3)+  ;;CHECK FOR RETURN
(1)          036450 122723 000015    BNE     2$      ;;LOOP IF NOT RETURN
(1)          036454 001274          CLRB   -1(R3)     ;;CLEAR RETURN (THE 15)
(1)          036456 105063 177777    TYPE    ,SLF      ;;TYPE A LINE FEED
(1)          036462 104401 001224    TST     (SP)+      ;;CLEAN RUBOUT KEY FROM THE STACK
(1)          036466 005726          MOV     (SP)+,R3    ;;RESTORE R3
(1)          036470 012603

```

```
(1) 036472 011646          MOV    (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1) 036474 016666 000004 000002  MOV    4(SP),2(SP)    ;;      FIRST ASCII CHARACTER ON IT
(1) 036502 012766 036514 000004  MOV    #$TTYIN,4(SP)
(1) 036510 000002          RTI                    ;;RETURN
(1) 036512 000          9$: .BYTE 0              ;;STORAGE FOR ASCII CHAR. TO TYPE
(1) 036513 000          .BYTE 0              ;;TERMINATOR
(1) 036514 000010  $TTYIN: .BLKB 8.      ;;RESERVE 8 BYTES FOR TTY INPUT
(1) 036524 041536 005015 000  $CNTLC: .ASCIZ /^C/<15><12> ;;CONTROL 'C'
(1) 036531 136 006525 000012  $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
(1) 036536 043536 005015 000  $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
(1) 036543 015 051412 051127  $MSWR: .ASCIZ <15><12>/SWR = /
(1) 036550 036440 000040          $MNEW: .ASCIZ / NEW = /
(1) 036554 020040 042516 020127
(1) 036562 020075 000
(1) 036566          .EVEN
3166          .SBTTL CONTROL-C SERVICING ROUTINE
3167
3168
3169          ;* THE FOLLOWING CODE IS EXECUTED WHEN A CONTROL-C HAS
3170          ;* BEEN TYPED INSTEAD OF A NEW SWITCH REG. VALUE.
3171
3172
3173          ;* (IN OTHER WORDS, AFTER A CONTROL-G WAS TYPED).
3174          ;* A NEW SWITCH REG. VALUE WILL BE ASKED FOR,
3175          ;* THE TEST NUMBER AND PASS NUMBER WILL BE TYPED,
3176          ;* AND THEN THE PROGRAM WILL GO TO 'END-OF-PASS' AND CONTINUE
3177
3178 036566 013737 001234 001210  CNTRLC: MOV    $PASS,$TMP5    ;GET THE VALUE OF '$PASS'
3179 036574 005237 001210          INC    $TMP5          ;FORM CURRENT PASS NO.
3180 036600 104401 036645          TYPE  CMSG          ;TYPE THE TEST STOPS MESSAGE
3181 036604 113737 001102 036640  MOV    $TSTNM,1$     ;SAVE THE TEST NUMBER
3182 036612 013746 036640          MOV    1$,-(SP)     ;SAVE 1$ FOR TYPEOUT
(1) 036616 104402          TYPOC                ;GO TYPE--OCTAL ASCII(ALL DIGITS)
3183 036620 104401 036642          TYPE  2$            ;TYPE 2 SPACES
3184 036624 013746 001210          MOV    $TMP5,-(SP)  ;SAVE $TMP5 FOR TYPEOUT
(1) 036630 104405          TYPDS                ;GO TYPE--DECIMAL ASCII WITH SIGN
3185 036632 104407          GTSWR                ;ASK FOR NEW SWR VALUE
3186 036634 000137 033616          JMP    $EOP+2        ;CONTINUE AT $EOP+2
3187 036640 000000          1$: .WORD 0          ;BUFFER FOR TEST NUMBER
3188 036642 020040 000          2$: .ASCIZ / /      ;TWO SPACES AND THE STOP MESSAGE
3189 036645 112 046525 044520  CMSG: .ASCIZ /JUMPING TO END-OF-PASS/<15><12>
      036652 043516 052040 020117
      036660 047105 026504 043117
      036666 050055 051501 006523
      036674 012
3190 036675 124 051505 047124          .ASCIZ /TESTNO PASSNO/<15><12>
      036702 020117 050040 051501
      036710 047123 006517 000012
3191          .EVEN
3192          .SBTTL TYPE ROUTINE
(1)
(2)          ;*****
(1)          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
(1)          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
(1)          ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
(1)          ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
```



```
(1) ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
(1) ;*
(1) ;*CALL:
(1) ;*1) USING A TRAP INSTRUCTION
(1) ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
(1) ;*OR
(1) ;* TYPE
(1) ;* MESADR
(1) ;*
(1) 036716 105737 001157 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
(1) 036722 100002 BPL 1$ ;;BR IF YES
(1) 036724 000000 HALT ;;HALT HERE IF NO TERMINAL
(1) 036726 000430 BR 3$ ;;LEAVE
(1) 036730 010046 1$: MOV R0,-(SP) ;;SAVE R0
(1) 036732 017600 000002 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
(1) 036736 122737 000001 001246 CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
(1) 036744 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
(1) 036746 132737 000100 001247 BITB #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1) 036754 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
(1) 036756 010037 036766 MOV R0,61$ ;;SETUP MESSAGE ADDRESS FOR APT
(1) 036762 004737 037206 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
(1) 036766 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
(1) 036770 132737 000040 001247 62$: BITB #APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
(1) 036776 001003 BNE 60$ ;;YES,SKIP TYPE OUT
(1) 037000 112046 2$: MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1) 037002 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
(1) 037004 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
(1) 037006 012600 60$: MOV (SP)+,R0 ;;RESTORE R0
(1) 037010 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
(1) 037014 000002 RTI ;;RETURN
(1) 037016 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
(1) 037022 001430 BEQ 8$
(1) 037024 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
(1) 037030 001006 BNE 5$
(1) 037032 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
(1) 037034 104401 TYPE ;;TYPE A CR AND LF
(1) 037036 001223 $CRLF
(1) 037040 105037 037174 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
(1) 037044 000755 BR 2$ ;;GET NEXT CHARACTER
(1) 037046 004737 037130 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
(1) 037052 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
(1) 037056 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
(1) 037060 013746 001154 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
(1) ;;AND THE NULL CHAR.
(1) 037064 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
(1) 037070 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1) 037072 004737 037130 JSR PC,$TYPEC ;;GO TYPE A NULL
(1) 037076 105337 037174 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
(1) 037102 000770 BR 7$ ;;LOOP
(1)
(1) ;HORIZONTAL TAB PROCESSOR
(1)
(1) 037104 112716 000040 8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
(1) 037110 004737 037130 9$: JSR PC,$TYPEC ;;TYPE A SPACE
(1) 037114 132737 000007 037174 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
```

```
(1) 037122 001372          BNE      9$          ::TAB STOP
(1) 037124 005726          TST      (SP)+      ::POP SPACE OFF STACK
(1) 037126 000724          BR       2$          ::GET NEXT CHARACTER
(1) 037130 105777 142014  $TYPEPC: TSTB     @$TPS      ::WAIT UNTIL PRINTER IS READY
(1) 037134 100375          BPL      $TYPEPC
(1) 037136 116677 000002 142006  MOVB     2(SP),@$TPB  ::LOAD CHAR TO BE TYPED INTO DATA REG.
(1) 037144 122766 000015 000002  CMPB     #CR,2(SP)   ::IS CHARACTER A CARRIAGE RETURN?
(1) 037152 001003          BNE     1$          ::BRANCH IF NO
(1) 037154 105037 037174  CLRB     $CHARCNT   ::YES--CLEAR CHARACTER COUNT
(1) 037160 000406          BR       $TYPEPC   ::EXIT
(1) 037162 122766 000012 000002 1$:  CMPB     #LF,2(SP)   ::IS CHARACTER A LINE FEED?
(1) 037170 001402          BEQ     $TYPEPC   ::BRANCH IF YES
(1) 037172 105227          INCB     (PC)+      ::COUNT THE CHARACTER
(1) 037174 000000          $CHARCNT: .WORD    0  ::CHARACTER COUNT STORAGE
(1) 037176 000207          $TYPEPC: RTS      PC
```

3193 .SBTTL APT COMMUNICATIONS ROUTINE

```
(1)
(2)
(1) 037200 112737 000001 037444 $ATY1:  MOVB     #1,$FFLG      ::TO REPORT FATAL ERROR
(1) 037206 112737 000001 037442 $ATY3:  MOVB     #1,$MFLG      ::TO TYPE A MESSAGE
(1) 037214 000403          BR       $ATYC
(1) 037216 112737 000001 037444 $ATY4:  MOVB     #1,$FFLG      ::TO ONLY REPORT FATAL ERROR
(1) 037224          $ATYC:
(3) 037224 010046          MOV      R0,-(SP)    ::PUSH R0 ON STACK
(3) 037226 010146          MOV      R1,-(SP)    ::PUSH R1 ON STACK
(1) 037230 105737 037442          TSTB     $MFLG      ::SHOULD TYPE A MESSAGE?
(1) 037234 001450          BEQ     5$          ::IF NOT: BR
(1) 037236 122737 000001 001246  CMPB     #APTENV,$ENV  ::OPERATING UNDER APT?
(1) 037244 001031          BNE     3$          ::IF NOT: BR
(1) 037246 132737 000100 001247  BITB     #APTSPOOL,$ENVM ::SHOULD SPOOL MESSAGES?
(1) 037254 001425          BEQ     3$          ::IF NOT: BR
(1) 037256 017600 000004          MOV      @4(SP),R0   ::GET MESSAGE ADDR.
(1) 037262 062766 000002 000004          ADD      #2,4(SP)    ::BUMP RETURN ADDR.
(1) 037270 005737 001226          1$:  TST      $MSGTYPE   ::SEE IF DONE W/ LAST XMISSION?
(1) 037274 001375          BNE     1$          ::IF NOT: WAIT
(1) 037276 010037 001242          MOV      R0,$MSGAD   ::PUT ADDR IN MAILBOX
(1) 037302 105720          2$:  TSTB     (R0)+      ::FIND END OF MESSAGE
(1) 037304 001376          BNE     2$
(1) 037306 163700 001242          SUB      $MSGAD,R0   ::SUB START OF MESSAGE
(1) 037312 006200          ASR      R0          ::GET MESSAGE LNGTH IN WORDS
(1) 037314 010037 001244          MOV      R0,$MSGGLT  ::PUT LENGTH IN MAILBOX
(1) 037320 012737 000004 001226  MOV      #4,$MSGTYPE ::TELL APT TO TAKE MSG.
(1) 037326 000413          BR       5$
(1) 037330 017637 000004 037354 3$:  MOV      @4(SP),4$   ::PUT MSG ADDR IN JSR LINKAGE
(1) 037336 062766 000002 000004          ADD      #2,4(SP)    ::BUMP RETURN ADDRESS
(3) 037344 013746 177776          MOV      177776,-(SP) ::PUSH 177776 ON STACK
(1) 037350 004737 036716          JSR     PC,$TYPE    ::CALL TYPE MACRO
(1) 037354 000000          4$:  .WORD    0
(1) 037356          5$:
(1) 037356 105737 037444          10$: TSTB     $FFLG      ::SHOULD REPORT FATAL ERROR?
(1) 037362 001416          BEQ     12$         ::IF NOT: BR
(1) 037364 005737 001246          TST     $ENV        ::RUNNING UNDER APT?
(1) 037370 001413          BEQ     12$         ::IF NOT: BR
(1) 037372 005737 001226          11$: TST     $MSGTYPE   ::FINISHED LAST MESSAGE?
(1) 037376 001375          BNE     11$         ::IF NOT: WAIT
```



```

(1)          ;*$TYPOS OR $TYPOC
(1)          ;*$CALL:
(1)          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
(1)          ;*      TYPON                    ;;CALL FOR TYPEOUT
(1)          ;*
(1)          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)          ;*$CALL:
(1)          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
(1)          ;*      TYPOC                    ;;CALL FOR TYPEOUT
(1)
(1) 037522 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
(1) 037526 116637 000001 037745  MOVB     1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
(1) 037534 112637 037747          MOVB     (SP)+, $OMODE+1    ;;NUMBER OF DIGITS TO TYPE
(1) 037540 062716 000002          ADD      #2, (SP)         ;;ADJUST RETURN ADDRESS
(1) 037544 000406          BR       $TYPON
(1) 037546 112737 000001 037745  $TYPOC: MOVB     #1, $OFILL      ;;SET THE ZERO FILL SWITCH
(1) 037554 112737 000006 037747  MOVB     #6, $OMODE+1    ;;SET FOR SIX(6) DIGITS
(1) 037562 112737 000005 037744  $TYPON: MOVB     #5, $OCNT      ;;SET THE ITERATION COUNT
(1) 037570 010346          MOV      R3, -(SP)       ;;SAVE R3
(1) 037572 010446          MOV      R4, -(SP)       ;;SAVE R4
(1) 037574 010546          MOV      R5, -(SP)       ;;SAVE R5
(1) 037576 113704 037747          MOVB     $OMODE+1, R4    ;;GET THE NUMBER OF DIGITS TO TYPE
(1) 037602 005404          NEG      R4
(1) 037604 062704 000006          ADD      #6, R4          ;;SUBTRACT IT FOR MAX. ALLOWED
(1) 037610 110437 037746          MOVB     R4, $OMODE      ;;SAVE IT FOR USE
(1) 037614 113704 037745          MOVB     $OFILL, R4      ;;GET THE ZERO FILL SWITCH
(1) 037620 016605 000012          MOV      12(SP), R5     ;;PICKUP THE INPUT NUMBER
(1) 037624 005003          CLR      R3             ;;CLEAR THE OUTPUT WORD
(1) 037626 006105          1$:     ROL      R5          ;;ROTATE MSB INTO 'C'
(1) 037630 000404          BR       3$             ;;GO DO MSB
(1) 037632 006105          2$:     ROL      R5          ;;FORM THIS DIGIT
(1) 037634 006105          ROL      R5
(1) 037636 006105          ROL      R5
(1) 037640 010503          MOV      R5, R3
(1) 037642 006103          3$:     ROL      R3          ;;GET LSB OF THIS DIGIT
(1) 037644 105337 037746          DECB     $OMODE         ;;TYPE THIS DIGIT?
(1) 037650 100016          BPL      7$             ;;BR IF NO
(1) 037652 042703 177770          BIC      #177770, R3    ;;GET RID OF JUNK
(1) 037656 001002          BNE      4$             ;;TEST FOR 0
(1) 037660 005704          TST      R4             ;;SUPPRESS THIS 0?
(1) 037662 001403          BEQ      5$             ;;BR IF YES
(1) 037664 005204          4$:     INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
(1) 037666 052703 000060          BIS      #'0, R3        ;;MAKE THIS DIGIT ASCII
(1) 037672 052703 000040          5$:     BIS      #' , R3    ;;MAKE ASCII IF NOT ALREADY
(1) 037676 110337 037742          MOVB     R3, 8$         ;;SAVE FOR TYPING
(1) 037702 104401 037742          TYPE     , 8$          ;;GO TYPE THIS DIGIT
(1) 037706 105337 037744          7$:     DECB     $OCNT      ;;COUNT BY 1
(1) 037712 003347          BGT      2$             ;;BR IF MORE TO DO
(1) 037714 002402          BLT      6$             ;;BR IF DONE
(1) 037716 005204          INC      R4             ;;INSURE LAST DIGIT ISN'T A BLANK
(1) 037720 000744          BR       2$             ;;GO DO THE LAST DIGIT
(1) 037722 012605          6$:     MOV      (SP)+, R5    ;;RESTORE R5
(1) 037724 012604          MOV      (SP)+, R4      ;;RESTORE R4
(1) 037726 012603          MOV      (SP)+, R3      ;;RESTORE R3
(1) 037730 016666 000002 000004  MOV      2(SP), 4(SP)    ;;SET THE STACK FOR RETURNING
(1) 037736 012616          MOV      (SP)+, (SP)

```


3200

.SBTTL TRAP DECODER

(1)
(2)
(1)
(1)
(1)
(1)
(1)

: *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
: *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
: *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
: *GO TO THAT ROUTINE.

(1) 040410 010046
(1) 040412 016600 000002
(1) 040416 005740
(1) 040420 111000
(1) 040422 006300
(1) 040424 016000 040444
(1) 040430 000200

\$TRAP: MOV R0,-(SP) ;;SAVE R0
MOV 2(SP),R0 ;;GET TRAP ADDRESS
TST -(R0) ;;BACKUP BY 2
MOVB (R0),R0 ;;GET RIGHT BYTE OF TRAP
ASL R0 ;;POSITION FOR INDEXING
MOV \$TRPAD(R0),R0 ;;INDEX TO TABLE
RTS R0 ;;GO TO ROUTINE

(1)
(1)

;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO

(1) 040432 011646
(1) 040434 016666 000004 000002
(1) 040442 000002

\$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
RTI ;;RESTORE THE PSW

(1)
(3)

.SBTTL TRAP TABLE

(3)
(3)

: *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
: *BY THE 'TRAP' INSTRUCTION.

(3)
(3)

: ROUTINE
: -----

(3) 040444 040432
(3) 040446 036716
(3) 040450 037516
(3) 040452 037522
(3) 040454 037562
(3) 040456 037750
(3) 040460 037446
(1)
(3) 040462 035644
(1)
(3) 040464 035574
(3) 040466 036116
(3) 040470 036236
(3) 040472 040174
(3) 040474 040232

\$TRPAD: .WORD \$TRAP2
\$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
\$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
\$TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
\$TYPBN ;;CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER

\$GTSWR ;;CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING

\$CKSWR ;;CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
\$RDCHR ;;CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
\$RDLIN ;;CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
\$SAVREG ;;CALL=SAVREG TRAP+13(104413) SAVE R0-R5 ROUTINE
\$RESREG ;;CALL=RESREG TRAP+14(104414) RESTORE R0-R5 ROUTINE

3201

.SBTTL POWER DOWN AND UP ROUTINES

(1)
(2)

: POWER DOWN ROUTINE

(1) 040476 012737 040654 000024
(1) 040504 012737 000340 000026
(3) 040512 010046
(3) 040514 010146
(3) 040516 010246
(3) 040520 010346
(3) 040522 010446

\$PWRDN: MOV #SILLUP,@#PWRVEC ;;SET FOR FAST UP
MOV #340,@#PWRVEC+2 ;;PRIO:7
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK


```
(3) 040524 010546          MOV      R5,-(SP)          ;;PUSH R5 ON STACK
(3) 040526 017746 140406    MOV      @SWR,-(SP)       ;;PUSH @SWR ON STACK
(1) 040532 010637 040660    MOV      SP,$SAVR6       ;;SAVE SP
(1) 040536 012737 040550 000024  MOV      #$PWRUP,@#PWRVEC ;;SET UP VECTOR
(1) 040544 000000          HALT
(1) 040546 000776          BR       .-2              ;;HANG UP
(1)
(2)
(1)
(1) 040550 012737 040654 000024  $PWRUP: MOV      #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
(1) 040556 013706 040660    MOV      $SAVR6,SP       ;;GET SP
(1) 040562 005037 040660    CLR      $SAVR6         ;;WAIT LOOP FOR THE TTY
(1) 040566 005237 040660    1$:     INC      $SAVR6         ;;WAIT FOR THE INC
(1) 040572 001375          BNE     1$              ;;OF WORD
(3) 040574 012677 140340    MOV      (SP)+,@SWR      ;;POP STACK INTO @SWR
(3) 040600 012605          MOV      (SP)+,R5       ;;POP STACK INTO R5
(3) 040602 012604          MOV      (SP)+,R4       ;;POP STACK INTO R4
(3) 040604 012603          MOV      (SP)+,R3       ;;POP STACK INTO R3
(3) 040606 012602          MOV      (SP)+,R2       ;;POP STACK INTO R2
(3) 040610 012601          MOV      (SP)+,R1       ;;POP STACK INTO R1
(3) 040612 012600          MOV      (SP)+,R0       ;;POP STACK INTO R0
(1) 040614 012737 040476 000024  MOV      #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
(1) 040622 012737 000340 000026  MOV      #340,@#PWRVEC+2 ;;PRIO:7
(1) 040630 104401          TYPE
(1) 040632 040662          $PWRMG: .WORD  PWRMSG      ;;REPORT THE POWER FAILURE
(1) 040634 012716          MOV      (PC)+,(SP)     ;;POWER FAIL MESSAGE POINTER
(1) 040636 020464          $PWRAD: .WORD  RESTR     ;;RESTART AT RESTR
(1) 040640 042766 000020 000002  BIC      #20,2(SP)      ;;RESTART ADDRESS
(1) 040646 005037 034130    CLR      $TBIT         ;;CLEAR 'T' BIT
(1) 040652 000002          RTI                    ;;CLEAR THE 'T' BIT FLAG
(1) 040654 000000          $ILLUP: HALT
(1) 040656 000776          BR       .-2              ;;THE POWER UP SEQUENCE WAS STARTED
(1) 040660 000000          $SAVR6: 0                ;;BEFORE THE POWER DOWN WAS COMPLETE
3202 040662 006412 050040 053517  PWRMSG: .ASCIZ <12><15>? POWER FAILURE - RESTARTING ?<12><15>
      040670 051105 043040 044501
      040676 052514 042522 026440
      040704 051040 051505 040524
      040712 052122 047111 020107
      040720 006412 000
3203 040724          .EVEN
3204
3205
```

3207					.SBTTL	ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS
3208	040724	047125	054105	042520	EM1:	.ASCIZ /UNEXPECTED CPU TRAP TO LOC. 004/
	040732	052103	042105	041440		
	040740	052520	052040	040522		
	040746	020120	047524	046040		
	040754	041517	020056	030060		
	040762	000064				
3209	040764	047125	054105	042520	EM2:	.ASCIZ /UNEXPECTED MEM. MGMT. TRAP TO LOC. 250/
	040772	052103	042105	046440		
	041000	046505	020056	043515		
	041006	052115	020056	051124		
	041014	050101	052040	020117		
	041022	047514	027103	031040		
	041030	030065	000			
3210	041033	120	044522	051117	EM3:	.ASCIZ /PRIORITY BITS SET WRONG IN PSW/
	041040	052111	020131	044502		
	041046	051524	051440	052105		
	041054	053440	047522	043516		
	041062	044440	020116	051520		
	041070	000127				
3211	041072	047515	042504	041040	EM4:	.ASCIZ /MODE BITS SET WRONG IN PSW/
	041100	052111	020123	042523		
	041106	020124	051127	047117		
	041114	020107	047111	050040		
	041122	053523	000			
3212	041125	104	040525	020114	EM5:	.ASCIZ /DUAL ADDRESSING BETWEEN HI&LO BYTES OF PSW/
	041132	042101	051104	051505		
	041140	044523	043516	041040		
	041146	052105	042527	047105		
	041154	044040	023111	047514		
	041162	041040	052131	051505		
	041170	047440	020106	051520		
	041176	000127				
3213	041200	042513	047122	046105	EM6:	.ASCIZ /KERNEL R6 CHANGED BY WRITING USER R6/
	041206	051040	020066	044103		
	041214	047101	042507	020104		
	041222	054502	053440	044522		
	041230	044524	043516	052440		
	041236	042523	020122	033122		
	041244	000				
3214	041245	101	046440	046505	EM7:	.ASCIZ /A MEMORY MGMT. REG. TIMED OUT/
	041252	051117	020131	043515		
	041260	052115	020056	042522		
	041266	027107	052040	046511		
	041274	042105	047440	052125		
	041302	000				
3215	041303	123	046525	040515	EM10:	.ASCIZ /SUMMARY OF MEM. MGMT. REG. TIMEOUTS/
	041310	054522	047440	020106		
	041316	042515	027115	046440		
	041324	046507	027124	051040		
	041332	043505	020056	044524		
	041340	042515	052517	051524		
	041346	000				
3216	041347	115	046505	020056	EM11:	.ASCIZ /MEM. MGMT. REG. WOULD NOT CLEAR/
	041354	043515	052115	020056		
	041362	042522	027107	053440		

	041370	052517	042114	047040		
	041376	052117	041440	042514		
	041404	051101	000			
3217	041407	115	046505	020056	EM12:	.ASCIZ /MEM. MGMT. REG. BITS NOT SET CORRECTLY/
	041414	043515	052115	020056		
	041422	042522	027107	041040		
	041430	052111	020123	047516		
	041436	020124	042523	020124		
	041444	047503	051122	041505		
	041452	046124	000131			
3218	041456	051123	020060	043105	EM13:	.ASCIZ /SRO EFFECTED BY WRITE TO PSW/
	041464	042506	052103	042105		
	041472	041040	020131	051127		
	041500	052111	020105	047524		
	041506	050040	053523	000		
3219	041513	123	030522	042040	EM14:	.ASCIZ /SR1 DID NOT READ ALL ZEROS/
	041520	042111	047040	052117		
	041526	051040	040505	020104		
	041534	046101	020114	042532		
	041542	047522	000123			
3220	041546	052504	046101	040440	EM15:	.ASCIZ /DUAL ADDRESSING BETWEEN BYTES OF PAR OR PDR/
	041554	042104	042522	051523		
	041562	047111	020107	042502		
	041570	053524	042505	020116		
	041576	054502	042524	020123		
	041604	043117	050040	051101		
	041612	047440	020122	042120		
	041620	000122				
3221	041622	052504	046101	040440	EM16:	.ASCIZ /DUAL ADDRESSING BETWEEN PAR-PDR'S/
	041630	042104	042522	051523		
	041636	047111	020107	042502		
	041644	053524	042505	020116		
	041652	040520	026522	042120		
	041660	023522	000123			
3222	041664	044120	051531	041511	EM17:	.ASCIZ /PHYSICAL ADDRESS FORMED WRONG/
	041672	046101	040440	042104		
	041700	042522	051523	043040		
	041706	051117	042515	020104		
	041714	051127	047117	000107		
3223	041722	044120	051531	020056	EM20:	.ASCIZ /PHYS. ADDR. FORMED WRONG IN RELOCATE MODE/
	041730	042101	051104	020056		
	041736	047506	046522	042105		
	041744	053440	047522	043516		
	041752	044440	020116	042522		
	041760	047514	040503	042524		
	041766	046440	042117	000105		
3224	041774	026527	044502	020124	EM21:	.ASCIZ /W-BIT DID NOT GET SET IN PDR/
	042002	044504	020104	047516		
	042010	020124	042507	020124		
	042016	042523	020124	047111		
	042024	050040	051104	000		
3225	042031	127	041055	052111	EM22:	.ASCIZ /W-BIT SET IN MORE THAN ONE PDR/
	042036	051440	052105	044440		
	042044	020116	047515	042522		
	042052	052040	040510	020116		
	042060	047117	020105	042120		

3226	042066	000122					
	042070	026527	044502	020124	EM23:	.ASCIZ	/W-BIT NOT CLEARED BY WRITING TO PDR/
	042076	047516	020124	046103			
	042104	040505	042522	020104			
	042112	054502	053440	044522			
	042120	044524	043516	052040			
	042126	020117	042120	000122			
3227	042134	051127	052111	047111	EM24:	.ASCIZ	/WRITING SRO SET W-BIT IN KIPDR7/
	042142	020107	051123	020060			
	042150	042523	020124	026527			
	042156	044502	020124	047111			
	042164	045440	050111	051104			
	042172	000067					
3228	042174	026527	044502	020124	EM25:	.ASCIZ	/W-BIT GOT SET DURING TIMEOUT ABORT/
	042202	047507	020124	042523			
	042210	020124	052504	044522			
	042216	043516	052040	046511			
	042224	047505	052125	040440			
	042232	047502	052122	000			
3229	042237	115	046505	051117	EM26:	.ASCIZ	/MEMORY MGMT. ACCESS ABORT DID NOT OCCUR/
	042244	020131	043515	052115			
	042252	020056	041501	042503			
	042260	051523	040440	047502			
	042266	052122	042040	042111			
	042274	047040	052117	047440			
	042302	041503	051125	000			
3230	042307	101	041503	051505	EM27:	.ASCIZ	/ACCESS ERROR DID NOT ABORT INSTRUCTION/
	042314	020123	051105	047522			
	042322	020122	044504	020104			
	042330	047516	020124	041101			
	042336	051117	020124	047111			
	042344	052123	052522	052103			
	042352	047511	000116				
3231	042356	051123	020060	044504	EM30:	.ASCIZ	/SRO DID NOT REPORT ACCESS ERROR CORRECTLY/
	042364	020104	047516	020124			
	042372	042522	047520	052122			
	042400	040440	041503	051505			
	042406	020123	051105	047522			
	042414	020122	047503	051122			
	042422	041505	046124	000131			
3232	042430	044504	020104	047516	EM31:	.ASCIZ	/DID NOT LOCKUP CORRECT VIRTUAL ADDR./
	042436	020124	047514	045503			
	042444	050125	041440	051117			
	042452	042522	052103	053040			
	042460	051111	052524	046101			
	042466	040440	042104	027122			
	042474	000					
3233	042475	120	043501	020105	EM32:	.ASCIZ	/PAGE LGTH. ABORT OCCURRED WHEN IT SHOULDN'T HAVE/
	042502	043514	044124	020056			
	042510	041101	051117	020124			
	042516	041517	052503	051122			
	042524	042105	053440	042510			
	042532	020116	052111	051440			
	042540	047510	046125	047104			
	042546	052047	044040	053101			
	042554	000105					

3234	042556	040520	042507	046040	EM33:	.ASCIZ /PAGE LGTH. ABORT DID NOT OCCUR WHEN IT SHOULD HAVE/
	042564	052107	027110	040440		
	042572	047502	052122	042040		
	042600	042111	047040	052117		
	042606	047440	041503	051125		
	042614	053440	042510	020116		
	042622	052111	051440	047510		
	042630	046125	020104	040510		
	042636	042526	000			
3235	042641	123	030122	042040	EM34:	.ASCIZ /SRO DID NOT REPORT PAGE LGTH. ABORT CORRECTLY/
	042646	042111	047040	052117		
	042654	051040	050105	051117		
	042662	020124	040520	042507		
	042670	046040	052107	027110		
	042676	040440	047502	052122		
	042704	041440	051117	042522		
	042712	052103	054514	000		
3236	042717	123	030122	047440	EM37:	.ASCIZ /SRO OR SR2 CHANGED BY A SECOND ABORT/
	042724	020122	051123	020062		
	042732	044103	047101	042507		
	042740	020104	054502	040440		
	042746	051440	041505	047117		
	042754	020104	041101	051117		
	042762	000124				
3237	042764	051123	020060	051117	EM40:	.ASCIZ /SRO OR SR2 WERE NOT 'RESET' BY A RESET/
	042772	051440	031122	053440		
	043000	051105	020105	047516		
	043006	020124	051042	051505		
	043014	052105	020042	054502		
	043022	040440	051040	051505		
	043030	052105	000			
3238	043033	123	031122	047040	EM41:	.ASCIZ /SR2 NOT TRACKING CORRECTLY/
	043040	052117	052040	040522		
	043046	045503	047111	020107		
	043054	047503	051122	041505		
	043062	046124	000131			
3239	043066	044504	020104	047516	EM42:	.ASCIZ /DID NOT TRAP THRU KERNEL SPACE/
	043074	020124	051124	050101		
	043102	052040	051110	020125		
	043110	042513	047122	046105		
	043116	051440	040520	042503		
	043124	000				
3240	043125	113	020124	051105	EM43:	.ASCIZ /KT ERROR NOT SERVICED ON TIMEOUT ERROR/
	043132	047522	020122	047516		
	043140	020124	042523	053122		
	043146	041511	042105	047440		
	043154	020116	044524	042515		
	043162	052517	020124	051105		
	043170	047522	000122			
3241	043174	051123	020060	051117	EM44:	.ASCIZ /SRO OR SR2 CHANGED BY TIMEOUT ERROR/
	043202	051440	031122	041440		
	043210	040510	043516	042105		
	043216	041040	020131	044524		
	043224	042515	052517	020124		
	043232	051105	047522	000122		
3242	043240	051105	047522	020122	EM45:	.ASCIZ /ERROR DURING 'DOUBLE ERROR' (KT & TIMEOUT)/

Address	Hex 1	Hex 2	Hex 3	Hex 4	Message
3243	043246	052504	044522	043516	EM46: .ASCIZ /MFPI INSTRUCTION PUSHED WRONG DATA/
	043254	021040	047504	041125	
	043262	042514	042440	051122	
	043270	051117	020042	045450	
	043276	020124	020046	044524	
	043304	042515	052517	024524	
	043312	000			
	043313	115	050106	020111	EM47: .ASCIZ /MTPI INSTRUCTION LOADED WRONG DATA/
	043320	047111	052123	052522	
	043326	052103	047511	020116	
	043334	052520	044123	042105	
	043342	053440	047522	043516	
	043350	042040	052101	000101	
3244	043356	052115	044520	044440	EM47: .ASCIZ /MTPI INSTRUCTION LOADED WRONG DATA/
	043364	051516	051124	041525	
	043372	044524	047117	046040	
	043400	040517	042504	020104	
	043406	051127	047117	020107	
	043414	040504	040524	000	
3245	043421	123	040524	045503	EM50: .ASCIZ /STACK NOT PUSHED BY MFPI-MTPI/
	043426	047040	052117	050040	
	043434	051525	042510	020104	
	043442	054502	046440	050106	
	043450	026511	052115	044520	
	043456	000			
3246	043457	113	051105	042516	EM51: .ASCIZ /KERNEL PAGE ACCESS INSTEAD OF USER: MFPI-MTPI/
	043464	020114	040520	042507	
	043472	040440	041503	051505	
	043500	020123	047111	052123	
	043506	040505	020104	043117	
	043514	052440	042523	035122	
	043522	046440	050106	026511	
3247	043530	052115	044520	000	EM52: .ASCIZ /WRONG PDR'S REFERENCED WHILE IN RELOCATE MODE/
	043535	127	047522	043516	
	043542	050040	051104	051447	
	043550	051040	043105	051105	
	043556	047105	042503	020104	
	043564	044127	046111	020105	
	043572	047111	051040	046105	
	043600	041517	052101	020105	
3248	043606	047515	042504	000	EM53: .ASCIZ /MFPD INSTRUCTION PUSHED WRONG DATA/
	043613	115	050106	020104	
	043620	047111	052123	052522	
	043626	052103	047511	020116	
	043634	052520	044123	042105	
	043642	053440	047522	043516	
	043650	042040	052101	000101	
3249	043656	052123	041501	020113	EM54: .ASCIZ /STACK NOT PUSHED BY MFPD-MTPD/
	043664	047516	020124	052520	
	043672	044123	042105	041040	
	043700	020131	043115	042120	
	043706	046455	050124	000104	
	043714	040520	020122	051117	
3250	043722	050040	051104	041440	EM55: .ASCIZ /PAR OR PDR CHANGED BY A RESET/
	043730	040510	043516	042105	
	043736	041040	020131	020101	

Line	Address	PC	PSW	Register	Message
3251	043744	042522	042523	000124	EM56: .ASCIZ /PSW CHANGED BY AN RTI IN USER MODE/
	043752	051520	020127	044103	
	043760	047101	042507	020104	
	043766	054502	040440	020116	
	043774	052122	020111	047111	
	044002	052440	042523	020122	
	044010	047515	042504	000	
3252					
3253	044015	117	042114	050040	DH1: .ASCIZ /OLD PC OLD PSW R6 WAS TESTNO ERRORPC/
	044022	020103	047440	042114	
	044030	050040	053523	051040	
	044036	020066	040527	020123	
	044044	052040	051505	047124	
	044052	020117	042440	051122	
	044060	051117	041520	000	
3254	044065	117	042114	050040	DH2: .ASCIZ /OLD PC OLD PSW R6 WAS SR0 SR2 TESTNO ERRORPC/
	044072	020103	047440	042114	
	044100	050040	053523	051040	
	044106	020066	040527	020123	
	044114	051440	030122	020040	
	044122	020040	051440	031122	
	044130	020040	020040	052040	
	044136	051505	047124	020117	
	044144	042440	051122	051117	
	044152	041520	000		
3255	044155	127	047522	042524	DH3: .ASCIZ /WROTE READ TESTNO ERRORPC/
	044162	020040	051040	040505	
	044170	020104	020040	052040	
	044176	051505	047124	020117	
	044204	042440	051122	051117	
	044212	041520	000		
3256	044215	101	042104	042522	DH7: .ASCIZ /ADDRESS TESTNO ERRORPC/
	044222	051523	052040	051505	
	044230	047124	020117	042440	
	044236	051122	051117	041520	
	044244	000			
3257	044245	122	043505	051511	DH10: .ASCII /REGISTER-ADDRS NUM OF/<CRLF>
	044252	042524	026522	042101	
	044260	051104	020123	047040	
	044266	046525	020040	043117	
	044274	200			
3258	044275	101	042116	042455	.ASCIZ /AND-ED OR-ED TIMOUTS TESTNO ERRORPC/
	044302	020104	047440	026522	
	044310	042105	020040	052040	
	044316	046511	052517	051524	
	044324	052040	051505	047124	
	044332	020117	042440	051122	
	044340	051117	041520	000	
	044345	122	043505	051511	
3259	044352	051124	051040	040505	DH11: .ASCII /REGISTR READ READ-(BINARY)/<CRLF>
	044360	020104	020040	051040	
	044366	040505	026504	041050	
	044374	047111	051101	024531	
	044402	200			
3260	044403	101	042104	042522	.ASCIZ /ADDRESS (OCTAL) 5432109876543210 TESTNO ERRORPC/
	044410	051523	024040	041517	

	045576	052103	020104	042120					
	045604	020122	020064	020040					
	045612	051520	020127	020040					
	045620	020040	042524	052123					
	045626	047516	020040	051105					
3279	045634	047522	050122	000103					
	045642	027126	027102	027101	DH32:	.ASCIZ	/V.B.A. KIPDR4	SR0 WAS SR2 WAS TESTNO	ERRORPC/
	045650	020040	044513	042120					
	045656	032122	020040	051123					
	045664	020060	040527	020123					
	045672	051123	020062	040527					
	045700	020123	042524	052123					
	045706	047516	020040	051105					
3280	045714	047522	050122	000103					
	045722	027126	027102	027101	DH33:	.ASCIZ	/V.B.A. KIPDR4	TESTNO	ERRORPC/
	045730	020040	044513	042120					
	045736	032122	020040	042524					
	045744	052123	047516	020040					
	045752	051105	047522	050122					
3281	045760	000103							
	045762	027126	027102	027101	DH34:	.ASCIZ	/V.B.A. KIPDR4	SR0 WAS EXPECTD TFSTNO	ERRORPC/
	045770	020040	044513	042120					
	045776	032122	020040	051123					
	046004	020060	040527	020123					
	046012	054105	042520	052103					
	046020	020104	042524	052123					
	046026	047516	020040	051105					
3282	046034	047522	050122	000103					
	046042	027126	027102	027101	DH35:	.ASCIZ	/V.B.A. KIPDR4	SR2 WAS EXPECTD TESTNO	ERRORPC/
	046050	020040	044513	042120					
	046056	032122	020040	051123					
	046064	020062	040527	020123					
	046072	054105	042520	052103					
	046100	020104	042524	052123					
	046106	047516	020040	051105					
3283	046114	047522	050122	000103					
	046122	051123	020062	040527	DH36:	.ASCIZ	/SR2 WAS EXPECTD TESTNO	ERRORPC/	
	046130	020123	054105	042520					
	046136	052103	020104	042524					
	046144	052123	047516	020040					
	046152	051105	047522	050122					
3284	046160	000103							
	046162	044506	051522	020124	DH37:	.ASCII	/FIRST ABORT	SECOND ABORT/<CRLF>	
	046170	041101	051117	020124					
	046176	020040	020040	042523					
	046204	047503	042116	040440					
	046212	047502	052122	200					
3285	046217	123	030122	053440		.ASCIZ	/SR0 WAS SR2 WAS SR0 WAS SR2 WAS TESTNO	ERRORPC/	
	046224	051501	051440	031122					
	046232	053440	051501	051440					
	046240	030122	053440	051501					
	046246	051440	031122	053440					
	046254	051501	052040	051505					
	046262	047124	020117	042440					
	046270	051122	051117	041520					
	046276	000							

3286	046277	123	030122	053440	DH40:	.ASCIZ	/SRO WAS SR2 WAS TESTNO ERRORPC/
	046304	051501	051440	031122			
	046312	053440	051501	052040			
	046320	051505	047124	020117			
	046326	042440	051122	051117			
	046334	041520	000				
3287	046337	120	053523	053440	DH42:	.ASCIZ	/PSW WAS R6 WAS TESTNO ERRORPC/
	046344	051501	051040	020066			
	046352	040527	020123	052040			
	046360	051505	047124	020117			
	046366	042440	051122	051117			
	046374	041520	000				
3288	046377	105	050130	041505	DH44:	.ASCII	/EXPECTED RECEIVED/<CRLF>
	046404	042524	020104	020040			
	046412	020040	020040	020040			
	046420	042522	042503	053111			
	046426	042105	200				
3289	046431	123	030122	020040		.ASCIZ	/SRO SR2 SRO WAS SR2 WAS TESTNO ERRORPC/
	046436	020040	051440	031122			
	046444	020040	020040	051440			
	046452	030122	053440	051501			
	046460	051440	031122	053440			
	046466	051501	052040	051505			
	046474	047124	020117	042440			
	046502	051122	051117	041520			
	046510	000					
3290	046511	105	050130	041505	DH45:	.ASCII	/EXPECTED:<CRLF>
	046516	042524	035104	200			
3291	046523	120	053523	020040		.ASCII	/PSW PC SRO SR2/<CRLF>
	046530	020040	050040	020103			
	046536	020040	020040	051440			
	046544	030122	020040	020040			
	046552	051440	031122	200			
3292	046557	061	030067	030460		.ASCII	/170017 (3\$+4) 020147 (3\$)/<CRLF>
	046564	020067	024040	022063			
	046572	032053	020051	030040			
	046600	030062	032061	020067			
	046606	024040	022063	100051			
3293	046614	042522	042503	053111		.ASCII	/RECEIVED:<CRLF>
	046622	042105	100072				
3294	046626	051520	020127	020040		.ASCIZ	/PSW PC SRO SR2 TESTNO ERRORPC/
	046634	020040	041520	020040			
	046642	020040	020040	051123			
	046650	020060	020040	020040			
	046656	051123	020062	020040			
	046664	020040	042524	052123			
	046672	047516	020040	051105			
	046700	047522	050122	000103			
3295	046706	040504	040524	020040	DH46:	.ASCII	/DATA DATA/<CRLF>
	046714	020040	040504	040524			
	046722	200					
3296	046723	105	050130	041505		.ASCIZ	/EXPECTD RECEIVD TESTNO ERRORPC/
	046730	042124	051040	041505			
	046736	044505	042126	052040			
	046744	051505	047124	020117			
	046752	042440	051122	051117			

CJKDACO KTF11-AA MMU DIAG
CJKDAC.P11 12-MAR-80 07:56

MACY:11 30A(1052) 12-MAR-80 08:00 PAGE 1-98
ERROR MESSAGES, DATA HEADERS-TABLES & FORMATS

SEQ 0112

3297	046760	041520	000	047124	DH50:	.ASCIZ	/TESTNO	ERRORPC/	
	046763	124	051505	051122					
	046770	020117	042440	051122					
3298	046776	051117	041520	000	DH51:	.ASCIZ	/SRO WAS SR2 WAS TESTNO	ERRORPC/	
	047003	123	030122	053440					
	047010	051501	051440	031122					
	047016	053440	051501	052040					
	047024	051505	047124	020117					
	047032	042440	051122	051117					
	047040	041520	000						
3299	047043	120	054510	044523	DH52:	.ASCII	/PHYSICL PAR 4/<CRLF>		
	047050	046103	050040	051101					
	047056	032040	200						
3300	047061	101	042104	042522		.ASCIZ	/ADDRESS V.B.A. PAR 4	SRO WAS SR2 WAS PSW	TESTNO ERRORPC/
	047066	051523	053040	041056					
	047074	040456	020056	050040					
	047102	051101	032040	020040					
	047110	051440	030122	053440					
	047116	051501	051440	031122					
	047124	053440	051501	050040					
	047132	053523	020040	020040					
	047140	052040	051505	047124					
	047146	020117	042440	051122					
	047154	051117	041520	000					
3301	047161	120	053523	053440	DH56:	.ASCIZ	/PSW WAS EXPECTD TESTNO	ERRORPC/	
	047166	051501	042440	050130					
	047174	041505	042124	052040					
	047202	051505	047124	020117					
	047210	042440	051122	051117					
	047216	041520	000						
3302									
3303		047222				.EVEN			
3304									
3305	047222	001266	001270	001264	DT1:	.WORD	TRAPPC,TRAPPS,WASR6,TESTNO,\$ERRPC,0		
	047230	001262	001116	000000					
3306	047236	001266	001270	001264	DT2:	.WORD	TRAPPC,TRAPPS,WASR6,WASSR0,WASSR2,TESTNO,\$ERRPC,0		
	047244	001272	001274	001262					
	047252	001116	000000						
3307	047256	001162	001164	001262	DT3:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0		
	047264	001116	000000						
3308	047270	001162	001262	001116	DT7:	.WORD	\$REG0,TESTNO,\$ERRPC,0		
	047276	000000							
3309	047300	001300	001302	001304	DT10:	.WORD	ANDADR,ORADR,TONUM,TESTNO,\$ERRPC,0		
	047306	001262	001116	000000					
3310	047314	001162	001164	001164	DT11:	.WORD	\$REG0,\$REG1,\$REG1,TESTNO,\$ERRPC,0		
	047322	001262	001116	000000					
3311	047330	001162	001164	001166	DT12:	.WORD	\$REG0,\$REG1,\$REG2,\$REG2,TESTNO,\$ERRPC,0		
	047336	001166	001262	001116					
	047344	000000							
3312	047346	001162	001262	001116	DT13:	.WORD	\$REG0,TESTNO,\$ERRPC,0		
	047354	000000							
3313	047356	001162	001164	001174	DT16:	.WORD	\$REG0,\$REG1,\$REG5,\$REG2,TESTNO,\$ERRPC,0		
	047364	001166	001262	001116					
	047372	000000							
3314	047374	001312	001306	001172	DT17:	.WORD	PBALO,VIRT1,\$REG4,TESTNO,\$ERRPC,0		
	047402	001262	001116	000000					

3315	047410	001312	001306	001310	DT20:	.WORD	PBALO,VIRT1,VIRT2,\$REG4,\$REG5,\$TMP0,TESTNO,\$ERRPC,0
	047416	001172	001174	001176			
	047424	001262	001116	000000			
3316	047432	001174	001170	001262	DT21:	.WORD	\$REG5,\$REG3,TESTNO,\$ERRPC,0
	047440	001116	000000				
3317	047444	001162	001174	001170	DT22:	.WORD	\$REG0,\$REG5,\$REG3,TESTNO,\$ERRPC,0
	047452	001262	001116	000000			
3318	047460	001174	001262	001116	DT23:	.WORD	\$REG5,TESTNO,\$ERRPC,0
	047466	000000					
3319	047470	001166	001164	001262	DT24:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0
	047476	001116	000000				
3320	047502	001166	001176	001262	DT26:	.WORD	\$REG2,\$TMP0,TESTNO,\$ERRPC,0
	047510	001116	000000				
3321	047514	001272	001170	001166	DT30:	.WORD	WASSRO,\$REG3,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
	047522	001176	001262	001116			
	047530	000000					
3322	047532	001274	001172	001166	DT31:	.WORD	WASSR2,\$REG4,\$REG2,\$TMP0,TESTNO,\$ERRPC,0
	047540	001176	001262	001116			
	047546	000000					
3323	047550	001162	001172	001272	DT32:	.WORD	\$REG0,\$REG4,WASSRO,WASSR2,TESTNO,\$ERRPC,0
	047556	001274	001262	001116			
	047564	000000					
3324	047566	001162	001172	001262	DT33:	.WORD	\$REG0,\$REG4,TESTNO,\$ERRPC,0
	047574	001116	000000				
3325	047600	001162	001172	001272	DT34:	.WORD	\$REG0,\$REG4,WASSRO,\$REG2,TESTNO,\$ERRPC,0
	047606	001166	001262	001116			
	047614	000000					
3326	047616	001162	001172	001274	DT35:	.WORD	\$REG0,\$REG4,WASSR2,\$REG3,TESTNO,\$ERRPC,0
	047624	001170	001262	001116			
	047632	000000					
3327	047634	001274	001164	001262	DT36:	.WORD	WASSR2,\$REG1,TESTNO,\$ERRPC,0
	047642	001116	000000				
3328	047646	001176	001202	001272	DT37:	.WORD	\$TMP0,\$TMP2,WASSRO,WASSR2,TESTNO,\$ERRPC,0
	047654	001274	001262	001116			
	047662	000000					
3329	047664	001272	001274	001262	DT40:	.WORD	WASSRO,WASSR2,TESTNO,\$ERRPC,0
	047672	001116	000000				
3330	047676	001164	001166	001262	DT42:	.WORD	\$REG1,\$REG2,TESTNO,\$ERRPC,0
	047704	001116	000000				
3331	047710	001162	001164	001272	DT44:	.WORD	\$REG0,\$REG1,WASSRO,WASSR2,TESTNO,\$ERRPC,0
	047716	001274	001262	001116			
	047724	000000					
3332	047726	001164	001170	001272	DT45:	.WORD	\$REG1,\$REG3,WASSRO,WASSR2,TESTNO,\$ERRPC,0
	047734	001274	001262	001116			
	047742	000000					
3333	047744	001162	001164	001262	DT46:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0
	047752	001116	000000				
3334	047756	001262	001116	000000	DT50:	.WORD	TESTNO,\$ERRPC,0
3335	047764	001272	001274	001262	DT51:	.WORD	WASSRO,WASSR2,TESTNO,\$ERRPC,0
	047772	001116	000000				
3336	047776	001312	001306	001172	DT52:	.WORD	PBALO,VIRT1,\$REG4,WASSRO,WASSR2,\$TMP0,TESTNO,\$ERRPC,0
	050004	001272	001274	001176			
	050012	001262	001116	000000			
3337	050020	001164	001166	001262	DT56:	.WORD	\$REG1,\$REG2,TESTNO,\$ERRPC,0
	050026	001116	000000				
3338							

3339	050032	000	000	000	DF1:	.BYTE	0,0,0,0,0
	050035	000	000				
3340	050037	000	000	000	DF2:	.BYTE	0,0,0,0,0,0,0
	050042	000	000	000			
	050045	000					
3341	050046	000	000	000	DF3:	.BYTE	0,0,0,0
	050051	000					
3342	050052	000	000	000	DF7:	.BYTE	0,0,0
3343	050055	000	000	001	DF10:	.BYTE	0,0,1,0,0
	050060	000	000				
3344	050062	000	000	002	DF11:	.BYTE	0,0,2,0,0
	050065	000	000				
3345	050067	000	000	000	DF12:	.BYTE	0,0,0,2,0,0
	050072	002	000	000			
3346	050075	000	000	000	DF13:	.BYTE	0,0,0
3347	050100	000	000	000	DF16:	.BYTE	0,0,0,0,0,0
	050103	000	000	000			
3348	050106	003	000	000	DF17:	.BYTE	3,0,0,0,0
	050111	000	000				
3349	050113	003	000	000	DF20:	.BYTE	3,0,0,0,0,0,0,0
	050116	000	000	000			
	050121	000	000				
3350	050123	000	000	000	DF21:	.BYTE	0,0,0,0
	050126	000					
3351	050127	000	000	000	DF22:	.BYTE	0,0,0,0,0
	050132	000	000				
3352	050134	000	000	000	DF23:	.BYTE	0,0,0
3353	050137	000	000	000	DF24:	.BYTE	0,0,0,0
	050142	000					
3354	050143	000	000	000	DF30:	.BYTE	0,0,0,0,0,0,0
	050146	000	000	000			
3355	050151	000	000	000	DF46:	.BYTE	0,0,0,0
	050154	000					
3356	050155	000	000		DF50:	.BYTE	0,0
3357	050157	000	000	000	DF51:	.BYTE	0,0,0,0
	050162	000					
3358	050163	003	000	000	DF52:	.BYTE	3,0,0,0,0,0,0,0
	050166	000	000	000			
	050171	000	000				
3359	050173	000	000	000	DF56:	.BYTE	0,0,0,0
	050176	000					
3360							
3361		000001				.END	

DF7	050052	303	3342#			
DH1	044015	265	3253#			
DH10	044245	307	3257#			
DH11	044345	314	547	3259#		
DH12	044465	321	340	3261#		
DH13	044625	328	334	3263#		
DH16	044655	347	3264#			
DH17	044755	354	3266#			
DH2	044065	271	3254#			
DH20	045045	361	3268#			
DH21	045173	368	3270#			
DH22	045253	375	3272#			
DH23	045352	382	478	3274#		
DH24	045402	388	394	3275#		
DH26	045442	400	406	3276#		
DH3	044155	277	283	289	295	3255#
DH30	045502	412	3277#			
DH31	045562	418	3278#			
DH32	045642	424	3279#			
DH33	045722	430	3280#			
DH34	045762	436	3281#			
DH35	046042	441	3282#			
DH36	046122	447	466	3283#		
DH37	046162	453	3284#			
DH40	046277	460	3286#			
DH42	046337	472	3287#			
DH44	046377	485	3288#			
DH45	046511	492	3290#			
DH46	046706	502	509	534	3295#	
DH50	046763	516	541	3297#		
DH51	047003	522	3298#			
DH52	047043	528	3299#			
DH56	047161	554	3301#			
DH7	044215	301	3256#			
DISPLA	001142	261#	626*	2896*	2897*	
DISPRE	000174	258#	626			
DOAGIN	034030	2861	2872#			
DSWR =	177570	21#	261	626		
DT1	047222	266	3305#			
DT10	047300	309	3309#			
DT11	047314	316	549	3310#		
DT12	047330	323	342	3311#		
DT13	047346	329	335	3312#		
DT16	047356	349	3313#			
DT17	047374	356	3314#			
DT2	047236	272	3306#			
DT20	047410	363	3315#			
DT21	047432	370	3316#			
DT22	047444	377	3317#			
DT23	047460	383	479	3318#		
DT24	047470	389	395	3319#		
DT26	047502	401	407	3320#		
DT3	047256	278	284	290	296	3307#
DT30	047514	413	3321#			
DT31	047532	419	3322#			
DT32	047550	425	3323#			

DT33	047566	431	3324#		
DT34	047600	437	3325#		
DT35	047616	442	3326#		
DT36	047634	448	467	3327#	
DT37	047646	455	3328#		
DT40	047664	461	3329#		
DT42	047676	473	3330#		
DT44	047710	487	3331#		
DT45	047726	497	3332#		
DT46	047744	504	511	536	3333#
DT50	047756	517	542	3334#	
DT51	047764	523	3335#		
DT52	047776	530	3336#		
DT56	050020	555	3337#		
DT7	047270	302	3308#		
EMTVEC=	000030	21#	626*		
EM1	040724	264	3208#		
EM10	041303	306	3215#		
EM11	041347	313	3216#		
EM12	041407	320	3217#		
EM13	041456	327	3218#		
EM14	041513	333	3219#		
EM15	041546	339	3220#		
EM16	041622	346	3221#		
EM17	041664	353	3222#		
EM2	040764	270	3209#		
EM20	041722	360	3223#		
EM21	041774	367	3224#		
EM22	042031	374	3225#		
EM23	042070	381	3226#		
EM24	042134	387	3227#		
EM25	042174	393	3228#		
EM26	042237	399	3229#		
EM27	042307	405	3230#		
EM3	041033	276	3210#		
EM30	042356	411	3231#		
EM31	042430	417	440	446	3232#
EM32	042475	423	3233#		
EM33	042556	429	3234#		
EM34	042641	435	3235#		
EM37	042717	452	3236#		
EM4	041072	282	3211#		
EM40	042764	459	3237#		
EM41	043033	465	3238#		
EM42	043066	471	3239#		
EM43	043125	477	3240#		
EM44	043174	484	3241#		
EM45	043240	491	3242#		
EM46	043313	501	3243#		
EM47	043356	508	3244#		
EM5	041125	288	3212#		
EM50	043421	515	3245#		
EM51	043457	521	3246#		
EM52	043535	527	3247#		
EM53	043613	533	3248#		
EM54	043656	540	3249#		

	145#	1251	1252	1253	1319	1320	1321	1322								
ADDTST	21#															
COMMEN	21#															
ENDCOM	21#															
ERROR	21#	583	614	661	684	710	723	747	793	798	813	819	825	831	850	
	859	871	880	905	923	930	934	953	965	996	1009	1046	1058	1089	1101	
	1180	1191	1202	1214	1251	1252	1253	1272	1295	1319	1320	1321	1322	1388	1444	
	1459	1466	1490	1503	1553	1561	1569	1576	1619	1627	1635	1642	1722	1733	1740	
	1752	1834	1845	1852	1864	1944	1984	1993	2017	2033	2042	2082	2115	2141	2151	
	2158	2228	2303	2308	2321	2333	2344	2356	2371	2384	2399	2413	2448	2466	2484	
	2499	2515	2533	2550	2569	2583	2631	2636	2649	2661	2672	2684	2698	2711	2725	
	2740	2765	2770	2781	2807	2814	3072	3086	3100	3114						
ESCAPE	21#															
GETPRI	21#															
GETSWR	21#	629#														
MSG1	646#	652														
MSG10	821#	824														
MSG11	827#	830														
MSG12	833#	843														
MSG13	889#	898														
MSG14	912#	919														
MSG15	936#	945														
MSG16	980#	988														
MSG17	1026#	1033														
MSG2	670#	674														
MSG20	1070#	1077														
MSG21	1113#	1124														
MSG21A	1161#	1170														
MSG22	1220#	1236														
MSG23	1304#	1315														
MSG24	1325#	1352														
MSG25	1454#	1457														
MSG26	1461#	1464														
MSG27	1469#	1478														
MSG3	693#	699														
MSG30	1525#	1533														
MSG31	1597#	1605														
MSG32	1685#	1697														
MSG33	1799#	1811														
MSG34	1912#	1926														
MSG35	1961#	1975														
MSG36	2052#	2064														
MSG36A	2096#	2102														
MSG37	2121#	2134														
MSG4	729#	737														
MSG40	2168#	2183														
MSG41	2249#	2260														
MSG42	2422#	2433														
MSG43	2593#	2604														
MSG44	2748#	2755														
MSG45	2789#	2797														
MSG5	765#	775														
MSG6	809#	812														
MSG7	815#	818														
MULT	21#															
NEWTST	21#	652	674	699	737	775	812	818	824	830	843	898	919	945	988	
	1033	1077	1124	1170	1236	1315	1352	1457	1464	1478	1533	1605	1697	1811	1926	

CJKDACO KTF11-AA MMU DIAG
CJKDAC.P11 12-MAR-80 07:56

MACY11 30A(1052) 12-MAR-80 08:00 PAGE 3-2
CROSS REFERENCE TABLE -- MACRO NAMES

L 10

SEQ 0128

.\$DB20	14#	3198
.\$EOP	10#	
.\$ERRO	11#	2897
.\$ERRT	11#	
.\$POWE	12#	3201
.\$READ	13#	3165
.\$SAVE	14#	3197
.\$SCOP	11#	2896
.\$STRAP	12#	3200
.\$TYPB	13#	3194
.\$TYPD	12#	3196
.\$TYPE	11#	3192
.\$TYPO	12#	3195

. ABS. 050177 000

ERRORS DETECTED: 0

CJKDAC.BIN,CJKDAC.LST/CRF=CJKDAC.P11
RUN-TIME: 98 54 4 SECONDS
RUN-TIME RATIO: 332/157=2.1
CORE USED: 32K (63 PAGES)