

11/70-74

11/70 - 74 MEM MGMT
CEKBED0

AH-7976D-MC
COPYRIGHT 75-80
FICHE 1 OF 2

JAN 1980
digital
MADE IN USA

This microfiche card contains a grid of 100 frames of data, arranged in 10 rows and 10 columns. Each frame displays a table of information, likely related to personnel management. The data is organized into columns, with some frames showing names and other identifying information, while others show numerical data or dates. The text is small and difficult to read due to the high resolution of the microfiche format.

11/70-74

11/70 - 74 MEM MGMT
CEKBED0

AH-7976D-MC

COPYRIGHT 75-80
FICHE 2 OF 2

JAN 1980

digital

MADE IN USA

IDENTIFICATION

PRODUCT CODE: AC-7975D-MC
PRODUCT NAME: CEKBED0 PDP 11/70-74MP MEMORY MANAGEMENT DIAGNOSTIC
DATE CREATED: MAY, 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DALE A. ROEDGER
MODIFIED BY: GEOFFREY A. WHITE (20-MAY-78)
ERNEST M. PREISIG (01-NOV-78)

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 MANUAL.

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

COPYRIGHT (C) 1975, 1979 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	DECX/11

TABLE OF CONTENTS

- 1) ABSTRACT
- 2) REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3) LOADING PROCEDURE
 - 3.1 METHOD
- 4) STARTING PROCEDURE
 - 4.1 STARTING ADDRESSES
 - 4.2 PROGRAM AND OPERATOR ACTION
 - 4.3 SPECIAL STARTING PROCEDURE
- 5) OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUB-ROUTINE ABSTRACTS
 - 5.3 PROGRAM AND OPERATOR ACTION
- 6) ERRORS
 - 6.1 ERROR HALTS AND DESCRIPTION
 - 6.2 ERROR RECOVERY
 - 6.3 SAMPLE ERROR MESSAGES
- 7) RESTRICTIONS
 - 7.1 STARTING RESTRICTIONS
 - 7.2 OPERATING RESTRICTIONS
- 8) MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 SOME IDEAS ON HOW TO GET MORE FROM THE PROGRAM
 - 8.3 ROM STATE DESCRIPTIONS
- 9) PROGRAM DESCRIPTION

ABSTRACT

THIS PROGRAM WILL TEST ALL OF THE MEMORY MANAGEMENT LOGIC AND ENABLE THE FIELD SERVICE REPRESENTATIVE TO ISOLATE THE DETECTED FAILURES TO A REPLACABLE MODULE. IT IS ASSUMED THAT BOTH THE CPU AND THE CACHE HAVE BEEN TESTED, OR ARE KNOWN TO BE FUNCTIONING CORRECTLY, AND THAT THE PROGRAM IS STARTED FROM ADDRESS 200. THIS WILL PROVIDE THE EARLIEST DETECTION OF MEMORY MANAGEMENT RELATED ERRORS AND ENABLE LOOPING ON THE ERROR INVOLVING MINIMUM LOGIC. THIS PROGRAM MAY ALSO EXPOSE FAULTS THAT ARE ON THE INTERFACE BETWEEN MEMORY MANAGEMENT AND OTHER SECTIONS OF THE COMPUTER.

THIS PROGRAM HAS BEEN SEGMENTED IN THE FOLLOWING WAY: ALL DATA TABLES, ERROR MESSAGES, AND SUBROUTINES RESIDE IN LOW CORE (VIRTUAL PAGES 0 & 1 IE. ADDRESSES 001100 THRU 037776). RIGHT NOW THE END OF THE SUBROUTINES IS AROUND 030300, SO THERE IS SOME ROOM FOR FUTURE EXPANSION. THE TEST CODE STARTS AT VIRTUAL PAGE 2 (ADDRESS 040000) AND EXPANDS TOWARD PAGE 4 (ADDRESS 100000). THE END OF THE PROGRAM IS NOW AROUND ADDRESS 077600, SO SMALL MODIFICATIONS CAN BE MADE WITHOUT RE-SEGMENTING THE PROGRAM.

THE REASON FOR THIS SEGMENTATION IS TWO-FOLD, FIRST IT ENABLES THE OPERATOR TO TELL FROM THE ADDRESS LIGHTS EXACTLY WHERE THE PROGRAM HAS HALTED OR 'HUNG-UP'. THAT IS, DID IT HALT IN THE ERROR ROUTINE OR IN A TRAP ROUTINE BECAUSE OF A CONDITION IMPOSSIBLE TO RECOVER FROM (ON PAGE 0 OR 1), OR DID IT GET 'HUNG-UP' IN THE TEST CODE ON PAGE 2 OR 3. THE OTHER REASON IS THAT CERTAIN MEMORY MANAGEMENT FUNCTIONS LOCK UP THE VIRTUAL PC OF THE INSTRUCTION AND THE PROGRAM, IN ORDER TO OPERATE PROPERLY, MUST KNOW WHERE IT IS AT ALL TIMES. IT SEEMS MUCH SIMPLER FOR THE CODE TO START AT A PREDETERMINED BOUNDARY SO THAT IF THE MESSAGES CHANGE OR A NEW SUBROUTINE IS ADDED THE PAGE THAT THE CODE IS ON WILL REMAIN THE SAME.

EACH TEST WILL SET THE LOOP ON ERROR POINTER (\$LPERR) TO THE MINIMUM NECESSARY SETUP CODE, IF ANY, FOR THE FUNCTION UNDER TEST. A SYNCHRONIZATION INSTRUCTION (NOP) IS PROVIDED BEFORE THE INSTRUCTION(S) THAT TEST(S) EACH NEW FUNCTION. THIS WILL ENABLE THE FIELD SERVICE REPRESENTATIVE TO UTILIZE THE MICRO BREAK REGISTER TO GENERATE AN 'EXTERNAL SYNC' PULSE ON THE BACK PLANE FOR BETTER PULSE RESOLUTION.

SECTION 8.2 OF THIS DOCUMENT CONTAINS SOME IDEAS THAT I HAD WHEN I WAS WRITING THIS PROGRAM ON HOW TO EFFECTIVELY UTILIZE IT TO MAKE FAULT ISOLATION FASIER. IF THESE IDEAS ARE NOT CORRECT OR NEED TO BE EXPANDED TO PROVIDE MORE INFORMATION PLEASE WRITE DOWN YOUR SUGGESTIONS AND FORWARD THEM TO THE DIAGNOSTIC DEPARTMENT.

IT SHOULD BE NOTED THAT THIS PROGRAM DOES NOT CHECK OUT THE

EKBEDG.MAN

SEQ 0004

CONSOLE OR THE CONSOLE CABLES THAT PLUG INTO THE MEMORY MANAGEMENT BOARDS. THE PROGRAM ASSUMES THAT THOSE COMPONENTS HAVE BEEN TESTED OR ARE KNOWN TO BE GOOD.

THIS DIAGNOSTIC SUPPORTS THE KB11-B/C, AND KB11-CM PROCESSORS. TESTS HAVE BEEN INCLUDED TO VERIFY THE PROPER FUNCTIONALITY OF THE 'CACHE BYPASS' FEATURE OF THE 11/74.

2. REQUIREMENTS

- 2.1 EQUIPMENT
THE BASIC PDP-11/70-74MP COMPUTER, INCLUDING AN OPERATING CPU, CACHE, AND MEMORY. AN LA-30 OR EQUIVALENT DEVICE IS ALSO NEEDED FOR ERROR MESSAGES, AND END OF PASS REPORTS.
- 2.2 STORAGE
THIS PROGRAM REQUIRES 16K OF MEMORY TO LOAD AND AT LEAST 20K OF MEMORY TO RUN IN. IT WILL SCAN MEMORY FROM 16K TO 124K ON 2K BOUNDARIES, AND FROM 120K TO THE TOP OF MEMORY FOUND BY THE SIZE ROUTINE ON 8K BOUNDARIES.
- 2.3 PRELIMINARY PROGRAMS
THE CPU AND CACHE DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. MAIN MEMORY SHOULD BE SCANNED FOR AT LEAST THE FIRST 28K TO SEE THAT A PROGRAM WILL EXECUTE CORRECTLY BEFORE ANY PROGRAM IS RUN.

3. LOADING PROCEDURE

- 3.1 METHOD
THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS SUPPORTED BY XXDP, AND SHOULD BE LOADED USING THE XXDP PROCEDURE FOR THAT DEVICE.

4. STARTING PROCEDURE

- 4.1 STARTING ADDRESSES
- | | |
|-----|---|
| 200 | THIS ADDRESS WILL RUN THE COMPLETE PROGRAM |
| 204 | THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 2
TEST THE READ/WRITE BITS IN THE MEMORY STATUS REGISTERS |
| 210 | THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 3
PAGE ADDRESS AND PAGE DESCRIPTOR TESTS |
| 214 | THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 4
RELOCATION AND ADDER TESTS |
| 220 | THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 5
MEMORY MANAGEMENT ABORTS AND TRAPS LOGIC TESTS |
| 224 | THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 6
D-SPACE TESTS, CORRECT TIMING OF I & D SPACE |
| 230 | THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 7
A & W BIT LOGIC TEST AND DUAL MAPPING TESTS |
| 234 | THIS ADDRESS WILL START THE PROGRAM AT ENTRY POINT 8
MOVE FROM AND MOVE TO PVIOUS MODE INSTRUCTION TESTS |
- 4.2 PROGRAM AND OPERATOR ACTION
AFTER THE PROGRAM IS LOADED, THE FIRST TIME IT IS RUN IT WILL IDENTIFY ITSELF AND RUN A QUICK VERIFY PASS. AT THE END OF EACH PASS THE PROGRAM WILL TYPE OUT THE PASS NUMBER AND THE TOTAL NUMBER OF ERRORS FOUND ON THAT PASS.

4.3

SPECIAL STARTING PROCEDURE

IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN THIS PROGRAM WITH THE CACHE DISABLED. SIMPLY LOAD THE CACHE CONTROL REGISTER (17777746) WITH THE DESIRED NUMBER, THEN LOAD THE PC (17777707) WITH THE STARTING ADDRESS AND PRESS CONTINUE. THE PROGRAM WILL NOW RUN AS IF YOU HAD LOADED THE STARTING ADDRESS AND PRESSED START BUT NOW THE CACHE CONTROL REGISTER IS DISABLING THE CACHE.

BIT00 -DISABLE TRAPS
BIT01 -DISABLE UNIBUS TRAPS
BIT02 -FORCE MISS ON READ GROUP 0
BIT03 -FORCE MISS ON READ GROUP 1
BIT04 -FORCE REPLACE GROUP 0
BIT05 -FORCE REPLACE GROUP 1

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SW15	1-	HALT ON ERROR
SW14	1=	LOOP ON THE TEST THAT YOU ARE IN
SW13	1=	INHIBIT ALL ERROR TYPE OUTS
SW12	1=	INHIBIT TRACE TRAP ON EVERY OTHER PASS
SW11	1-	INHIBIT ITERATIONS AFTER FIRST PASS
SW10	1-	RING BELL ON ERROR
SW09	1	LOOP ON ERROR
SW08	1=	LOOP ON TEST IN SWR<06:00>
SW07	1=	INHIBIT MULTIPLE ERROR TYPE OUTS

5.2 SUBROUTINE ABSTRACTS

ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE, BEFORE THEIR EXPANSION, AND IN THE DOCUMENT THAT IMMEDIATELY FOLLOWS THIS. THE FOLLOWING IS A LIST OF THEIR TITLES.

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

END OF PASS ROUTINE
SCOPE HANDLER ROUTINE
ERROR HANDLER ROUTINE
ERROR MESSAGE TYPE OUT ROUTINE
CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
SAVE & RESTORE R0-R5 ROUTINES
TYPE ROUTINE
BINARY TO OCTAL (ASCII) AND TYPE ROUTINE
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
TRAP DECODER
POWER DOWN AND UP ROUTINE
DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

TURN OFF AND SAVE T-BIT
RESTORE T-BIT TO ITS PREVIOUS CONDITION
CLEAR 16 PAR'S OR PDR'S STARTING FROM ADDRESS IN R5
CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
COUNT PATTERN ERROR IN P.A.R.'S OR P.D.R.'S

5.2.3 TRAP AND ABORT HANDLER ROUTINES

CPU TRAP HANDLER
CACHE TRAPS AND ABORTS HANDLER
MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER
TRAP ROUTINES FOR ABORT IN SUPERVISOR AND USER MODE

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION
WHEN THE PROGRAM DETECTS AN ERROR CONDITION IT ISSUES AN
'ERROR' (EMT) CALL. THIS CAUSES THE CPU TO TRAP TO THE
'ERROR HANDLER ROUTINE' WHICH PRINTS OUT THE ERROR MESSAGE,
IF ANY, AND CHECKS THE SWITCH REGISTER FOR THE MODE SELECTED.
THE PROGRAM WILL REACT AS FOLLOWS:

HALT ON THE ERROR	IF SW15=1, IS UP
INHIBIT ERROR TYPE OUT	IF SW13=1, IS UP
RING BELL ON THE ERROR	IF SW10=1, IS UP
LOOP ON THE ERROR	IF SW09=1, IS UP
INHIBIT MULTIPLE TYPE OUTS	IF SW07=1, IS UP

6.2 ERROR RECOVERY
IF SWITCH 09 IS UP, THE PROGRAM WILL LOOP BACK TO WHERE THE
LOOP ON ERROR POINTER (\$LPERR) IS SET. THIS WILL PROVIDE
THE TIGHTEST POSSIBLE SCOPING LOOP, AND PROVIDES ALL NECESSARY
SETUP CODE TO RECREATE THE ERROR. IF A SYNC POINT IS DESIRED
TO EXTERNAL SYNC THE SCOPE JUST PRIOR TO THE FAILING OPERATION
LOAD THE MICRO BREAK REGISTER WITH '044' AND USE THE 'NOP'
PROVIDED.

6.3 SAMPLE ERROR TYPE OUTS
CPU TRAP OR ABORT THRU 'ERRVEC' (004) HAD INCORRECT CONDITION
EXPECTD RECEIVD TESTNO PC AT ABORT
000040 000020 000047 XXXXXX

THIS ERROR MESSAGE INDICATES THAT THE CPU TIMED OUT OVER THE
JNIBUS WHEN IT WAS EXPECTING A CACHE NON-EXISTANT MEMORY TRAP.
THIS TEST IS CHECKING THE CARRY PROPAGATION, THAT IS DETERMINED FROM
LOOKING AT THE INDEX AT THE FRONT OF THE LISTING.

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS
IF A STARTING POINT OTHER THAN '200' IS USED AND ERRORS ARE REPORTED, THEY MAY BE DUE TO LOGIC THAT IS ASSUMED TO BE WORKING AS A RESULT OF TESTS THAT WERE NOT RUN.

7.2 OPERATING RESTRICTIONS
NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME
THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS IS APPROXIMATELY 10 SECONDS.

8.2 HINTS ON HOW TO GET MORE INFORMATION FROM THE PROGRAM

IF AN ERROR OCCURS THE FIRST THING THAT SHOULD BE NOTED IS WHAT PASS DID THE ERROR OCCUR ON. IF THE PASS HAS AN EVEN NUMBER AND SWITCH 12 IS DOWN THEN THE ERROR MIGHT BE T-BIT SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 12 UP, THIS WILL INHIBIT T-BIT TRAPPING.

IF THE PASS NUMBER IS GREATER THAN ONE THE ERROR MIGHT BE ITERATION SENSITIVE. TRY TO RUN AGAIN WITH SWITCH 11 UP, THIS WILL INHIBIT ITERATIONS.

NOW THAT YOU HAVE DETERMINED HOW TO MAKE THE MACHINE FAIL LOOK IN THE INDEX AT THE FRONT OF THE LISTING TO FIND THE TITLE OF THE TEST THAT WAS RUNNING WHEN THE ERROR CONDITION OCCURRED. GO TO THE LISTING AND READ THE PARAGRAPH AT THE BEGINNING OF THE TEST SO THAT YOU KNOW WHAT THE TEST IS TRYING TO DO. NOW READ THE ERROR MESSAGE AND IF THERE IS A COLUMN LABELED 'ERRORPC' GO TO THAT LOCATION IN THE TEST. THIS IS THE PC OF THE 'ERROR' STATEMENT. THE NUMBER IS THE ERROR MESSAGE NUMBER AND IN THE FRONT OF THE LISTING IS THE 'ERROR MESSAGE POINTER TABLE' WHICH WILL TELL YOU WHAT WORDS WERE TYPED OUT.

IF YOU WANT TO SCOPE THIS ERROR CONDITION, PUT UP SWITCH 09 (LOOP ON ERROR) OR IF YOU WANT TO LOOP ON THE ENTIRE TEST PUT UP SWITCH 14. YOU WILL PROBABLY WANT TO INHIBIT THE ERROR TYPE OUT AT THIS POINT, SWITCH 13 WILL DO THAT.

IF YOU NEED TO DO ACCURATE SCOPING AND NEED A GOOD SYNC POINT THEN MAKE SURE THAT THE MICRO BREAK REGISTER (1777770) HAS '044' IN IT. THIS WILL CAUSE A PULSE ON THE BACK PLANE AT PIN # AE1 (SLOT 10) WHENEVER A 'NOP' IS EXECUTED, AND THAT PULSE WILL MAKE AN EXCELLENT 'EXTERNAL SYNC' SIGNAL FOR YOUR SCOPE. THERE IS A 'NOP' JUST BEFORE EACH INSTRUCTION THAT TESTS A MEMORY MANAGEMENT FUNCTION FOR THE FIRST TIME, SO YOU SHOULD BE ABLE TO SCOPE ON ANY FAILURE THAT THIS PROGRAM CAN DETECT.

8.3

ROM STATE DESCRIPTIONS

THIS IS A LIST OF THE ROM OUTPUTS FROM THE MEMORY MANAGEMENT ROMS ON 'SSRA', WITH A SENTENCE OR TWO DESCRIBING THEIR MEANING AS IT RELATES TO MEMORY MANAGEMENT.

ROM OUT01 - ROM OUT03

THESE OUTPUTS ARE SENT TO MULTIPLEXERS AND REGISTERS ON 'SSRA' TO INDICATE A PARTICULAR MACHINE FUNCTION. THE ENCODING OF THESE OUTPUTS IS SHOWN IN A TRUTH TABLE ON 'SSRA'.

ROM OUT04

DESTINATION MODE -- THIS IS USED TO ENABLE RELOCATION IF BIT08 OF MMRO IS SET AND THIS IS THE DESTINATION CYCLE OF THE INSTRUCTION.

ROM OUT05

MFP + MTP -- THIS IS USED TO CLOCK THE PREVIOUS MODE INTO THE 'SPACE' FLIP-FLOPS ON 'SSRB', DURING A MFP + MTP INST.

ROM OUT06

TRAP OR ABORT -- THIS IS USED TO FORCE SETTING OF THE 'KERNEL SPACE' FLIP-FLOP ON 'SSRB' DURING A TRAP OR ABORT SEQUENCE.

ROM OUT07

BUST -- THIS IS USED TO CLOCK MOST OF THE LATCHES IN MEMORY MANAGEMENT SUCH AS: 'SPACE' F/F'S, STATUS REGISTERS,...

ROM OUT08

MFP + MTP -- THIS ASSERTS 'SSRB I SPACEB' TO FORCE I-SPACE ON MFPI + MTPI INSTRUCTIONS IF THE PSW IS NOT (USER MODE/ PREVIOUS USER MODE).

ROM OUT09

INST + INDEX FETCH -- THIS ASSERTS 'SSRB I SPACEA' WHICH FORCES I-SPACE DURING AN INSTRUCTION OR INDEX WORD FETCH.

ROM OUT10

THIS DOES NOT EXIST.

ROM OUT11

INST STARTED IN I-SPACE -- THIS ASSERTS 'SSRB I SPACEB' TO FORCE I-SPACE IF 'SSRB PREV-I' IS SET.

ROM OUT12

CONSOLE -- THIS ASSERTS 'SSRB I SPACEA' IF 'SSRK CNSL I SPACE H' IS TRUE AND YOU EXAMINE OR DEPOSIT.

ROM OUT13

SRCM - 1+2+3+4+5 -- IF THE SOURCE FIELD IS 7 THIS ASSERTS 'SSRB I SPACEB' WHICH FORCES I-SPACE.

ROM OUT14

DSTM - 1+2 -- IF THE DESTINATION FIELD IS 7 THIS ASSERTS 'SSRB I SPACEB' WHICH FORCES I-SPACE.

ROM OUT15
DSTM = 3 -- IF THE DESTINATION FIELD IS 7 THIS ASSERTS 'SSRB I SPACEA' WHICH FORCES I-SPACE DURING THE DESTINATION CYCLE.

ROM OUT16
FLOATING POINT -- IF THE DSTF = 7 AND THE DSTM = 2 THEN THIS ASSERTS 'SSRB I SPACEA' WHICH FORCES I-SPACE ON IMMEDIATE F.P.INST.

NOTE: THE FOLLOWING ROM STATES ARE NOT EXPLICITLY TESTED DUE TO THE FACT THAT I COULDN'T FIGURE OUT A WAY TO TEST THEM UNDER RUN CONDITIONS. THE LOGIC ASSOCIATED WITH THEM HAS BEEN TESTED BY OTHER ROM STATES BUT THESE STATES ARE NOT TESTED.

ROM OUT05
SHR.00, NEG.00, D12.90, D40.30, D12.30

ROM OUT09
FET.06, FET.08, FET.09
THESE NEXT ARE TESTED ON PASSES WITH T-BIT TRAPPING
(PASS 2, 4, ...)
FET.01, FET.02, FET.03

ROM OUT12
EXM.10, EXM.20, DEP.10, DEP.20

ROM OUT13
S45.10

ROM OUT16 FLOATING POINT
FSV.00, FSV.10

9. PROGRAM DESCRIPTION

THE LISTING THAT FOLLOWS THIS ONE HAS A PARAGRAPH DESCRIBING EACH OF THE TESTS. THERE IS ALSO A PARAGRAPH DESCRIBING EACH MAJOR GROUP OF TESTS.

13	OPERATIONAL SWITCH SETTINGS
27	BASIC DEFINITIONS
152	CACHE REGISTER DEFINITIONS
163	CPU REGISTER DEFINITIONS
177	MEMORY MANAGEMENT DEFINITIONS
326	UNIBUS MAP REGISTER DEFINITIONS
446	TRAP CATCHER
453	STARTING ADDRESS(ES)
474	ACT11 HOOKS
500	COMMON TAGS
627	ERROR POINTER TABLE
1431	ERROR TABLE MESSAGES AND DATA POINTERS
3152	END OF PASS ROUTINE
3224	SCOPE HANDLER ROUTINE
3292	ERROR HANDLER ROUTINE
3355	SPURIOUS ERROR HANDLER
3419	ERROR MESSAGE TYPE OUT ROUTINE
3549	CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
3589	SAVE AND RESTORE R0-R5 ROUTINES
3635	TYPE ROUTINE
3707	BINARY TO OCTAL (ASCII) AND TYPE
3785	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3853	TRAP DECODER
3868	TRAP TABLE
3888	POWER DOWN AND UP ROUTINES
3933	ROUTINE TO SIZE MEMORY
4027	DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
4067	***** SUBROUTINES UNIQUE TO THIS PROGRAM *****
4071	TURN OFF AND SAVE T-BIT
4088	RESTORE T-BIT TO ITS PREVIOUS CONDITION
4105	CLEAR 16 PARS OR PDRS STARTING FROM ADDRESS IN R5
4127	CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'
4147	P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED
4186	DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
4214	COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S
4248	***** TRAP HANDLING ROUTINES *****
4250	CPU TRAP HANDLER ROUTINE
4291	CACHE TRAPS AND ABORTS HANDLER ROUTINE
4351	MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
4392	D-SPACE TESTS MEMORY MANAGEMENT ABORT SERVICE ROUTINE
4434	TRAP ROUTINES FOR ABORT IN SUPERVISOR OR USER MODE
4596	
4597	***** ENTRY POINT 1 --- STARTING ADDRESS 200 *****
4598	***** TEST CODE STARTS AT ADDRESS 40000 *****
4714	T1 TRY TO READ ALL CPU REGISTERS
4751	T2 SYSTEM SIZE REGISTERS
4795	T3 CPU ERROR REGISTER
4819	T4 MICRO PROGRAM BREAK REGISTER
4867	T5 PROGRAM INTERRUPT REQUEST REGISTER
4897	T6 STACK LIMIT REGISTER
4925	T7 PROCESSOR STATUS WORD
4953	T10 SET UP THE STACK POINTERS FOR REST OF TESTS
4980	***** ENTRY POINT 2 --- STARTING ADDRESS 204 *****
4981	** TEST READ/WRITE BITS IN MEMORY MANAGEMENT STATUS REGISTERS **
4991	T11 BIT TEST OF MEMORY MANAGEMENT REGISTER 0
5050	T12 BIT TEST OF MEMORY MANAGEMENT REGISTER 1

5169	T13	BIT TEST OF MEMORY MANAGEMENT REGISTER 2
5220	T14	BIT TEST OF MEMORY MANAGEMENT REGISTER 3
5270		***** ENTRY POINT 3 --- STARTING ADDRESS 210 *****
5271		***** PAGE ADDRESS AND DESCRIPTOR REGISTER TESTS *****
5280		TEST THAT ALL P.A.R.'S & P.D.R.'S RESPOND
5286	T15	READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
5324	T16	READ ALL SUPERVISOR PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
5357	T17	READ ALL USER PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
5390	T20	READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
5423	T21	READ ALL SUPERVISOR PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
5456	T22	READ ALL USER PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT
5489		TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS
5498	T23	DUAL ADDRESS KERNEL PAGE ADDRESS REGISTERS, ON LOADING
5562	T24	DUAL ADDRESS SUPERVISOR PAGE ADDRESS REGISTERS, ON LOADING
5626	T25	DUAL ADDRESS USER PAGE ADDRESS REGISTERS, ON LOADING
5689	T26	DUAL ADDRESS KERNEL PAGE DESCRIPTOR REGISTERS, ON LOADING
5752	T27	DUAL ADDRESS SUPERVISOR PAGE DESCRIPTOR REGISTERS, ON LOADING
5815	T30	DUAL ADDRESS USER PAGE DESCRIPTOR REGISTERS, ON LOADING
5878		TEST FOR BAD READ/WRITE BITS IN P.A.R.'S & P.D.R.'S
5887	T31	COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS
5931	T32	COUNT PATTERN IN SUPERVISOR PAGE ADDRESS REGISTERS
5975	T33	COUNT PATTERN IN USER PAGE ADDRESS REGISTERS
6019	T34	COUNT PATTERN IN KERNEL PAGE DESCRIPTOR REGISTERS
6069	T35	COUNT PATTERN IN SUPERVISOR PAGE DESCRIPTOR REGISTERS
6118	T36	COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS
6167		TEST FOR CORRECT BYTE ADDRESSING OF P.A.R.'S & P.D.R.'S
6176	T37	BYTE ADDRESSING OF KERNEL PAGE ADDRESS REGISTERS
6212	T40	BYTE ADDRESSING OF SUPERVISOR PAGE ADDRESS REGISTERS
6248	T41	BYTE ADDRESSING OF USER PAGE ADDRESS REGISTERS
6284	T42	BYTE ADDRESSING OF KERNEL PAGE DESCRIPTOR REGISTERS
6320	T43	BYTE ADDRESSING OF SUPERVISOR PAGE DESCRIPTOR REGISTERS
6356	T44	BYTE ADDRESSING OF USER PAGE DESCRIPTOR REGISTERS
6392		DUAL ADDRESSING BETWEEN GROUPS OF REGISTERS
6393	T45	DUAL ADDRESSING FOR ALL PAR'S AND PDR'S
6430		***** ENTRY POINT 4 --- STARTING ADDRESS 214 *****
6431		***** RELOCATION AND ADDER TESTS *****
6439	T46	18-BIT MAPPING ADDER TESTING
6655	T47	18-BIT MAPPING CARRY PROPAGATION
6713		TEST 'SAPN UNIBUS ADRS L' IN 18-BIT MAPPING
6739		TEST 'SAPN NOT CACHE ADRS H' 18-BIT MAPPING
6817	T50	22-BIT MAPPING CARRY PROPAGATION
6875		TEST 'SAPN UNIBUS ADRS L' IN 22-BIT MAPPING
6899		TEST 'SAPN NOT CACHE ADRS H' 22-BIT MAPPING
6978		***** ENTRY POINT 5 --- STARTING ADDRESS 220 *****
6979		***** MEMORY MANAGEMENT ABORTS AND TRAPS LOGIC TESTS *****
6989	T51	PAGE LENGTH FAULTS - UPWARD EXPANSION
7060	T52	PAGE LENGTH FAULTS - DOWNWARD EXPANSION
7115	T53	ACCESS CONTROL FIELD = 0, 3, OR 7 (ABORT ALL ACCESSES)
7193	T54	ACCESS CONTROL FIELD = 2 (ABORT ON WRITE)
7230	T55	ACCESS CONTROL FIELD = 1 (ABORT ON WRITE, TRAP ON READ)
7301	T56	ACCESS CONTROL FIELD = 4 (TRAP ON READ OR WRITE)
7359	T57	ACCESS CONTROL FIELD = 5 (TRAP ON WRITE)
7409	T60	NO TRAP WHEN TRAP BIT IS SET
7444	T61	NO TRAPPING WHEN REFERENCING A MEMORY MANAGEMENT REG
7503	T62	ONLY ONE VECTOR TAKEN IF TRAP AND ABORT
7556	T63	PROPER TIMING OF MEMORY MANAGEMENT TRAPS

7605	T64	ABORT ON ILLEGAL MODE
7648	T65	MEMORY MANAGEMENT REGISTERS ONLY CLOCKED ONCE IF MMRO NOT CLEARED
7711	T66	SUPERVISOR MODE, ABORT VECTOR FROM KERNEL SPACE
7802	T67	M.M. ABORT DURING AN ODD ADDRESS ABORT SEQUENCE
7898	T70	USER MODE, ABORT VECTOR FROM KERNEL SPACE
7993	T71	COUNT PATTERN THRU MMR2, TO TEST ALL BITS
8054	***** ENTRY POINT 6 --- STARTING ADDRESS 224 *****	
8055	***** D-SPACE TESTS, CORRECT TIMING OF I & D SPACE *****	
8074	T72	ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS FORCED
8232	T73	ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS NOT FORCED
8313	T74	PROPER ENABLING OF SUPERVISOR D-SPACE
8417	T75	PROPER ENABLING OF USER D-SPACE
8518	T76	TRAPPING IN D-SPACE KERNEL MODE
8571	***** ENTRY POINT 7 --- STARTING ADDRESS 230 *****	
8572	***** A & W BIT LOGIC TEST AND DUAL MAPPING TESTS *****	
8582	T77	TEST A-BIT AND W-BIT LOGIC
8698	T100	DUAL MAPPING KERNEL MODE I-SPACE
8888	T101	DUAL MAPPING SUPERVISOR MODE I-SPACE
9022	T102	DUAL MAPPING USER MODE I-SPACE
9156	T103	DUAL MAPPING KERNEL MODE D-SPACE
9292	T104	DUAL MAPPING SUPERVISOR MODE D-SPACE
9428	T105	DUAL MAPPING USER MODE D-SPACE
9564	***** ENTRY POINT 8 --- STARTING ADDRESS 234 *****	
9565	***** MOVE FROM AND MOVE TO PREVIOUS MODE INSTRUCTION TEST *****	
9574	T106	MOVE FROM PREVIOUS (SUPERVISOR) I-SPACE
9742	T107	MOVE TO PREVIOUS (SUPERVISOR) I-SPACE
9879	T110	MFPI (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
10019	T111	MTPI (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
10150	T112	MOVE FROM PREVIOUS (USER) I-SPACE
10288	T113	MOVE FROM PREVIOUS (KERNEL) I-SPACE TO SUPERVISOR MODE
10439	T114	MFPD (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
10577	T115	MFPI (USER/PREV.USER) WITH USER D-SPACE ENABLED
10721	T116	CHECK DPARS READ BACK CORRECTLY
10828	MEMORY MANAGEMENT SETUP	
10857	T117	MEMORY MANAGEMENT ABORT
10899	T120	MEMORY MANAGEMENT TRAP
10939	T121	NON EXISTANT MEMORY ABORT
10978	T122	KT BEND
11044	T123	SL REGISTER COMPARATOR TEST 2
11169	T124	PS RESTORE

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

```
.TITLE PDP-11/70-74MP MEMORY MANAGEMENT DIAGNOSTIC
;*COPYRIGHT (C) 1975,1978
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
*
;*PROGRAM BY DALE A. ROEDGER
*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQA(-A3-1).
*

.SBTTL OPERATIONAL SWITCH SETTINGS
*
*      SWITCH          USE
*      -----
*      15             HALT ON ERROR
*      14             LOOP ON TEST
*      13             INHIBIT ERROR TYPEOUTS
*      12             INHIBIT TRACE TRAP
*      11             INHIBIT ITERATIONS
*      10             BELL ON ERROR
*      9              LOOP ON ERROR
*      8              LOOP ON TEST IN SWR<6:0>
*      7              INHIBIT MULTIPLE ERROR TYPEOUTS

.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100          ;;FIRST ADDRESS OF THE STACK
KERSTK= STACK       ;;KERNEL STACK
SUPSTK= STACK-200   ;;SUPERVISOR STACK
USESTK= STACK-300   ;;USER STACK
.EQUIV EMT,ERROR    ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE    ;;BASIC DEFINITION OF SCOPE CALL
PS= 177776          ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774      ;;STACK LIMIT REGISTER
PIRQ= 177772        ;;PROGRAM INTERRUPT REQUEST REGISTER
SWR= 177570         ;;SWITCH REGISTER
DISPLAY=SWR

;*MISCELLANEOUS DEFINITIONS
HT= 11              ;;CODE FOR HORIZONTAL TAB
LF= 12              ;;CODE LINE FEED
CR= 15              ;;CODE CARRIAGE RETURN
CRLF= 200           ;;CODE FOR CARRIAGE RETURN-LINE FEED

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0              ;;GENERAL REGISTER
R1= %1              ;;GENERAL REGISTER
R2= %2              ;;GENERAL REGISTER
R3= %3              ;;GENERAL REGISTER
R4= %4              ;;GENERAL REGISTER
R5= %5              ;;GENERAL REGISTER
R6= %6              ;;GENERAL REGISTER
R7= %7              ;;GENERAL REGISTER
```

```

57      .EQUIV R0,R10      ;;GENERAL REGISTER
58      .EQUIV R1,R11      ;;GENERAL REGISTER
59      .EQUIV R2,R12      ;;GENERAL REGISTER
60      .EQUIV R3,R13      ;;GENERAL REGISTER
61      .EQUIV R4,R14      ;;GENERAL REGISTER
62      .EQUIV R5,R15      ;;GENERAL REGISTER
63      000006      SP=%6
64      .EQUIV SP,KSP      ;;KERNEL STACK POINTER
65      .EQUIV SP,SSP      ;;SUPERVISOR STACK POINTER
66      .EQUIV SP,USP      ;;USER STACK POINTER
67      000007      PC=%7
68
69      ;*PRIORITY LEVEL DEFINITIONS
70      000000      PR0= 0      ;;PRIORITY LEVEL 0
71      000040      PR1= 40      ;;PRIORITY LEVEL 1
72      000100      PR2= 100     ;;PRIORITY LEVEL 2
73      000140      PR3= 140     ;;PRIORITY LEVEL 3
74      000200      PR4= 200     ;;PRIORITY LEVEL 4
75      000240      PR5= 240     ;;PRIORITY LEVEL 5
76      000300      PR6= 300     ;;PRIORITY LEVEL 6
77      000340      PR7= 340     ;;PRIORITY LEVEL 7
78
79      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
80      100000      SW15= 100000
81      040000      SW14= 40000
82      020000      SW13= 20000
83      010000      SW12= 10000
84      004000      SW11= 4000
85      002000      SW10= 2000
86      001000      SW09= 1000
87      000400      SW08= 400
88      000200      SW07= 200
89      000100      SW06= 100
90      000040      SW05= 40
91      000020      SW04= 20
92      000010      SW03= 10
93      000004      SW02= 4
94      000002      SW01= 2
95      000001      SW00= 1
96      .EQUIV SW09,SW9
97      .EQUIV SW08,SW8
98      .EQUIV SW07,SW7
99      .EQUIV SW06,SW6
100     .EQUIV SW05,SW5
101     .EQUIV SW04,SW4
102     .EQUIV SW03,SW3
103     .EQUIV SW02,SW2
104     .EQUIV SW01,SW1
105     .EQUIV SW00,SW0
106
107     ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
108     100000      BIT15= 100000
109     040000      BIT14= 40000
110     020000      BIT13= 20000
111     010000      BIT12= 10000
112     004000      BIT11= 4000
    
```

```

113      002000      BIT10= 2000
114      001000      BIT09= 1000
115      000400      BIT08= 400
116      000200      BIT07= 200
117      000100      BIT06= 100
118      000040      BIT05= 40
119      000020      BIT04= 20
120      000010      BIT03= 10
121      000004      BIT02= 4
122      000002      BIT01= 2
123      000001      BIT00= 1
124      .EQUIV      BIT09,BIT9
125      .EQUIV      BIT08,BIT8
126      .EQUIV      BIT07,BIT7
127      .EQUIV      BIT06,BIT6
128      .EQUIV      BIT05,BIT5
129      .EQUIV      BIT04,BIT4
130      .EQUIV      BIT03,BIT3
131      .EQUIV      BIT02,BIT2
132      .EQUIV      BIT01,BIT1
133      .EQUIV      BIT00,BIT0
    
```

```

135      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
136      000004      ERRVEC= 4          ;;TIME OUT AND OTHER ERRORS
137      000010      RESVEC= 10         ;;RESERVED AND ILLEGAL INSTRUCTIONS
138      000014      TBITVEC=14         ;;'T' BIT
139      000014      TRTVEC= 14         ;;TRACE TRAP
140      000014      BPTVEC= 14         ;;BREAKPOINT TRAP (BPT)
141      000020      IOTVEC= 20         ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
142      000024      PWRVEC= 24         ;;POWER FAIL
143      000030      EMTVEC= 30         ;;EMULATOR TRAP (EMT) **ERROR**
144      000034      TRAPVEC=34         ;;'TRAP' TRAP
145      000060      TKVEC= 60          ;;TTY KEYBOARD VECTOR
146      000064      TPVEC= 64          ;;TTY PRINTER VECTOR
147      000114      CACHVEC=114        ;;CACHE ERROR INTERRUPT VECTOR
148      000240      PIRQVEC=240        ;;PROGRAM INTERRUPT REQUEST VECTOR
149      000250      MMVEC= 250         ;;MEMORY MANAGEMENT VECTOR
    
```

151 .SBTTL CACHE REGISTER DEFINITIONS

```

154      177740      LOADRS = 177740    ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
155      177742      HIADRS = 177742    ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
156      177744      MEMERR = 177744    ;;CACHE ERROR REGISTER
157      177746      CONTRL = 177746    ;;MEMORY CONTROL REGISTER
158      177750      MAINT = 177750     ;;MEMORY MAINTENANCE REGISTER
159      177752      HITMIS = 177752    ;;HIT MISS REGISTER '1' IMPLIES HIT IN CACHE
    
```

162 .SBTTL CPU REGISTER DEFINITIONS

```

165      177760      SIZELO = 177760    ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
166      177762      SIZEHI = 177762    ;;TO GET TO THE LAST 32 WORDS OF MEMORY
167      ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
168      ;;CURRENTLY ALL ZERO
    
```

169 177764 SYSTID = 177764 ;;SYSTEM ID REGISTER
170 177766 CPUERR = 177766 ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
171 ;;THE TRAP TO ERRVEC (000004)
172
173
174
175

176 .SBTTL MEMORY MANAGEMENT DEFINITIONS
177

178 ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
179

180
181 177572 MMR0= 177572
182 177574 MMR1= 177574
183 177576 MMR2= 177576
184 172516 MMR3= 172516
185 .EQUIV MMR0,SR0
186 .EQUIV MMR1,SR1
187 .EQUIV MMR2,SR2
188 .EQUIV MMR3,SR3
189

190 ;*USER 'I' PAGE DESCRIPTOR REGISTERS
191

192 177600 UIPDR0= 177600
193 177602 UIPDR1= 177602
194 177604 UIPDR2= 177604
195 177606 UIPDR3= 177606
196 177610 UIPDR4= 177610
197 177612 UIPDR5= 177612
198 177614 UIPDR6= 177614
199 177616 UIPDR7= 177616
200

201 ;*USER 'D' PAGE DESCRIPTOR REGISTORS
202

203 177620 UDPDR0= 177620
204 177622 UDPDR1= 177622
205 177624 UDPDR2= 177624
206 177626 UDPDR3= 177626
207 177630 UDPDR4= 177630
208 177632 UDPDR5= 177632
209 177634 UDPDR6= 177634
210 177636 UDPDR7= 177636
211

212 ;*USER 'I' PAGE ADDRESS REGISTERS
213

214 177640 UIPAR0= 177640
215 177642 UIPAR1= 177642
216 177644 UIPAR2= 177644
217 177646 UIPAR3= 177646
218 177650 UIPAR4= 177650
219 177652 UIPAR5= 177652
220 177654 UIPAR6= 177654
221 177656 UIPAR7= 177656
222

223 ;*USER 'D' PAGE ADDRESS REGISTERS
224

225	177660	UDPAR0= 177660
226	177662	UDPAR1= 177662
227	177664	UDPAR2= 177664
228	177666	UDPAR3= 177666
229	177670	UDPAR4= 177670
230	177672	UDPAR5= 177672
231	177674	UDPAR6= 177674
232	177676	UDPAR7= 177676

;*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS

233		
234		
235		
236	172200	SIPDR0= 172200
237	172202	SIPDR1= 172202
238	172204	SIPDR2= 172204
239	172206	SIPDR3= 172206
240	172210	SIPDR4= 172210
241	172212	SIPDR5= 172212
242	172214	SIPDR6= 172214
243	172216	SIPDR7= 172216

;*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS

244		
245		
246		
247	172220	SDPDR0= 172220
248	172222	SDPDR1= 172222
249	172224	SDPDR2= 172224
250	172226	SDPDR3= 172226
251	172230	SDPDR4= 172230
252	172232	SDPDR5= 172232
253	172234	SDPDR6= 172234
254	172236	SDPDR7= 172236

;*SUPERVISOR 'I' PAGE ADDRESS REGISTERS

255		
256		
257		
258	172240	SIPAR0= 172240
259	172242	SIPAR1= 172242
260	172244	SIPAR2= 172244
261	172246	SIPAR3= 172246
262	172250	SIPAR4= 172250
263	172252	SIPAR5= 172252
264	172254	SIPAR6= 172254
265	172256	SIPAR7= 172256

;*SUPERVISOR 'D' PAGE ADDRESS REGISTERS

266		
267		
268		
269	172260	SDPAR0= 172260
270	172262	SDPAR1= 172262
271	172264	SDPAR2= 172264
272	172266	SDPAR3= 172266
273	172270	SDPAR4= 172270
274	172272	SDPAR5= 172272
275	172274	SDPAR6= 172274
276	172276	SDPAR7= 172276

;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS

277		
278		
279		
280	172300	KIPDR0= 172300

281	172302	KIPDR1= 172302
282	172304	KIPDR2= 172304
283	172306	KIPDR3= 172306
284	172310	KIPDR4= 172310
285	172312	KIPDR5= 172312
286	172314	KIPDR6= 172314
287	172316	KIPDR7= 172316

;*KERNEL 'D' PAGE DESCRIPTOR REGISTERS

291	172320	KDPDR0= 172320
292	172322	KDPDR1= 172322
293	172324	KDPDR2= 172324
294	172326	KDPDR3= 172326
295	172330	KDPDR4= 172330
296	172332	KDPDR5= 172332
297	172334	KDPDR6= 172334
298	172336	KDPDR7= 172336

;*KERNFL 'I' PAGE ADDRESS REGISTERS

302	172340	KIPAR0= 172340
303	172342	KIPAR1= 172342
304	172344	KIPAR2= 172344
305	172346	KIPAR3= 172346
306	172350	KIPAR4= 172350
307	172352	KIPAR5= 172352
308	172354	KIPAR6= 172354
309	172356	KIPAR7= 172356

;*KERNEL 'D' PAGE ADDRESS REGISTERS

313	172360	KDPAR0= 172360
314	172362	KDPAR1= 172362
315	172364	KDPAR2= 172364
316	172366	KDPAR3= 172366
317	172370	KDPAR4= 172370
318	172372	KDPAR5= 172372
319	172374	KDPAR6= 172374
320	172376	KDPAR7= 172376

.SBTTL UNIBUS MAP REGISTER DEFINITIONS

;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'

332	170200	MAPL00 = 170200
333	170202	MAPH00 = 170202
334	170204	MAPL01 = 170204
335	170206	MAPH01 = 170206
336	170210	MAPL02 = 170210

337	170212	MAPH02 = 170212
338	170214	MAPL03 = 170214
339	170216	MAPH03 = 170216
340	170220	MAPL04 = 170220
341	170222	MAPH04 = 170222
342	170224	MAPL05 = 170224
343	170226	MAPH05 = 170226
344	170230	MAPL06 = 170230
345	170232	MAPH06 = 170232
346	170234	MAPL07 = 170234
347	170236	MAPH07 = 170236
348	170240	MAPL10 = 170240
349	170242	MAPH10 = 170242
350	170244	MAPL11 = 170244
351	170246	MAPH11 = 170246
352	170250	MAPL12 = 170250
353	170252	MAPH12 = 170252
354	170254	MAPL13 = 170254
355	170256	MAPH13 = 170256
356	170260	MAPL14 = 170260
357	170262	MAPH14 = 170262
358	170264	MAPL15 = 170264
359	170266	MAPH15 = 170266
360	170270	MAPL16 = 170270
361	170272	MAPH16 = 170272
362	170274	MAPL17 = 170274
363	170276	MAPH17 = 170276
364	170300	MAPL20 = 170300
365	170302	MAPH20 = 170302
366	170304	MAPL21 = 170304
367	170306	MAPH21 = 170306
368	170310	MAPL22 = 170310
369	170312	MAPH22 = 170312
370	170314	MAPL23 = 170314
371	170316	MAPH23 = 170316
372	170320	MAPL24 = 170320
373	170320	MAPH24 = 170320
374	170324	MAPL25 = 170324
375	170326	MAPH25 = 170326
376	170330	MAPL26 = 170330
377	170332	MAPH26 = 170332
378	170334	MAPL27 = 170334
379	170336	MAPH27 = 170336
380	170340	MAPL30 = 170340
381	170342	MAPH30 = 170342
382	170344	MAPL31 = 170344
383	170346	MAPH31 = 170346
384	170350	MAPL32 = 170350
385	170352	MAPH32 = 170352
386	170354	MAPL33 = 170354
387	170356	MAPH33 = 170356
388	170360	MAPL34 = 170360
389	170362	MAPH34 = 170362
390	170364	MAPL35 = 170364
391	170366	MAPH35 = 170366
392	170370	MAPL36 = 170370

393	170372	MAPH36 = 170372
394	170374	MAPL37 = 170374
395	170376	MAPH37 = 170376
396		.EQUIV MAPL00,MAPL0
397		.EQUIV MAPH00,MAPH0
398		.EQUIV MAPL01,MAPL1
399		.EQUIV MAPH01,MAPH1
400		.EQUIV MAPL02,MAPL2
401		.EQUIV MAPH02,MAPH2
402		.EQUIV MAPL03,MAPL3
403		.EQUIV MAPH03,MAPH3
404		.EQUIV MAPL04,MAPL4
405		.EQUIV MAPH04,MAPH4
406		.EQUIV MAPL05,MAPL5
407		.EQUIV MAPH05,MAPH5
408		.EQUIV MAPL06,MAPL6
409		.EQUIV MAPH06,MAPH6
410		.EQUIV MAPL07,MAPL7
411		.EQUIV MAPH07,MAPH7

:DEFINITIONS

418	100000	VSPE=BIT15
419	040000	IVSS=BIT14
420	020000	VSIU=BIT13
421	010000	VCIP=BIT12
422	004000	DMMA=BIT11
423	002000	FVPE=BIT10
424	001000	UCB=BIT9
425	000400	FCAC=BIT8
426	000040	S1=BIT5
427	000020	S0=BIT4
428	000010	M1=BIT3
429	000004	M0=BIT2
430	000002	DUT=BIT1
431	000001	DT=BIT0
433	100000	BYP=BIT15
435	000054	S1MOM1=BIT5+BIT3+BIT2
436	000034	S0MOM1=BIT4+BIT3+BIT2
437	000014	MOM1=BIT3+BIT2
439	177746	CONTRL=177746
440	177752	HITMIS=177752
441	177744	MSER=177744

.SBTTL TRAP CATCHER

447 000000 . = 0

;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '-2,HALT'

```

449          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
450          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
451
452          .SBTTL  STARTING ADDRESS(ES)
453             =200
454
455 000200 000137 040000      JMP      @#STRT1      ;; JUMP TO STARTING ADDRESS OF PROGRAM
456                                     ;; RUN ENTIRE PROGRAM (ALL TESTS)
457 000204 000137 040010      JMP      STRT2      ;; STARTING AT ENTRY POINT 2
458                                     ;; MEMORY MANAGEMENT STATUS REGISTERS
459 000210 000137 040020      JMP      STRT3      ;; STARTING AT ENTRY POINT 3
460                                     ;; PAGE ADDRESS AND DESCRIPTOR REGISTERS
461 000214 000137 040030      JMP      STRT4      ;; STARTING AT ENTRY POINT 4
462                                     ;; RELOCATION AND ADDER TESTS
463 000220 000137 040040      JMP      STRT5      ;; STARTING AT ENTRY POINT 5
464                                     ;; MEMORY MANAGEMENT TRAP AND ABORT LOGIC
465 000224 000137 040050      JMP      STRT6      ;; STARTING AT ENTRY POINT 6
466                                     ;; D-SPACE ENABLING
467 000230 000137 040060      JMP      STRT7      ;; STARTING AT ENTRY POINT 7
468                                     ;; A & W BIT LOGIC & DUAL MAPPING
469 000234 000137 040070      JMP      STRT8      ;; STARTING AT ENTRY POINT 8
470                                     ;; MFP & MTP LOGIC TESTS
471          ;:*****
472
473          .SBTTL  ACT11 HOOKS
474
475          ;*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11
476          ;*
477          ;*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL
478          ;*END OF THE PROGRAM.
479          ;*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
480          ;*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
481          ;*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:
482          ;*
483          ;*      BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
484          ;*          =0 NO POWER FAIL DESIRED
485          ;*
486          ;*      BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
487          ;*          =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT
488          ;*
489          ;*      BITS 13-0 MUST BE ZERO'S
490
491          000240      $SVPC=      ;; SAVE LOCATION COUNTER
492          000046      =46        ;; SET LOCATION COUNTER
493 000046 025412      .WORD  $ENDAD  ;; SET LOC.46 TO ADDRESS $ENDAD
494          000052      =52        ;; SET LOCATION COUNTER
495 000052 000000      .WORD  0      ;; SET LOC.52 TO ZERO
496          000240      .-$SVPC    ;; RESTORE LOCATION COUNTER
    
```

```
497      ::*****
498
499      .SBTTL COMMON TAGS
500
501      :*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
502      :*USED IN THE PROGRAM.
503
504      001100      .=1100
505
506      001100      $CMTAG:      ::START OF COMMON TAGS
507      001100      000000      $PASS: .WORD 0      ::CONTAINS PASS COUNT
508      001102      000      $STSTM: .BYTE 0      ::CONTAINS THE TEST NUMBER
509      001103      000      $ERFLG: .BYTE 0      ::CONTAINS ERROR FLAG
510      001104      000000      $$TSTM: .WORD 0      ::TEST NUMBER STORAGE
511      001106      000000      $ICNT: .WORD 0      ::CONTAINS SUBTEST ITERATION COUNT
512      001110      000000      $LPADR: .WORD 0      ::CONTAINS SCOPE LOOP
513      001112      000000      $LPERR: .WORD 0      ::CONTAINS SCOPE RETURN FOR ERRORS
514      001114      000000      $ERTTL: .WORD 0      ::CONTAINS TOTAL ERRORS DETECTED
515      001116      000      $ITEMB: .BYTE 0      ::CONTAINS ITEM CONTROL BYTE
516      001117      001      $ERMAX: .BYTE 1      ::CONTAINS MAX. ERRORS PER TEST
517      001120      000000      $ERRPC: .WORD 0      ::CONTAINS PC OF LAST ERROR INSTRUCTION
518      001122      000000      $GDADR: .WORD 0      ::CONTAINS OF 'GOOD' DATA
519      001124      000000      $BDADR: .WORD 0      ::CONTAINS OF 'BAD' DATA
520      001126      000000      $GDDAT: .WORD 0      ::CONTAINS 'GOOD' DATA
521      001130      000000      $BDDAT: .WORD 0      ::CONTAINS 'BAD' DATA
522      001132      000000 000000 000000      .WORD 0,0,0      ::RESERVED--NOT TO BE USED
523      001140      177560      $TKS: 177560      ::TTY KBD STATUS
524      001142      177562      $TKB: 177562      ::TTY KBD BUFFER
525      001144      177564      $TPS: 177564      ::TTY PRINTER STATUS REG.
526      001146      177566      $TPB: 177566      ::TTY PRINTER BUFFER REG.
527      001150      000      $NULL: .BYTE 0      ::CONTAINS NULL CHARACTER FOR FILLS
528      001151      002      $FILLS: .BYTE 2      ::CONTAINS # OF FILLER CHARACTERS REQUIRED
529      001152      012      $FILLC: .BYTE 12      ::INSERT FILL CHARS. AFTER A 'LINE FEED'
530      001153      000      $TPFLG: .BYTE 0      ::'TERMINAL AVAILABLE' FLAG (BIT<07>-0=YES)
531      001154      000000      $REGAD: .WORD 0      ::CONTAINS THE FROM
532      ::WHICH ($REG0) WAS OBTAINED
533      001156      000000      $REG0: .WORD 0      ::CONTAINS (($REGAD)+0)
534      001160      000000      $REG1: .WORD 0      ::CONTAINS (($REGAD)+2)
535      001162      000000      $REG2: .WORD 0      ::CONTAINS (($REGAD)+4)
536      001164      000000      $REG3: .WORD 0      ::CONTAINS (($REGAD)+6)
537      001166      000000      $REG4: .WORD 0      ::CONTAINS (($REGAD)+10)
538      001170      000000      $REG5: .WORD 0      ::CONTAINS (($REGAD)+12)
539      001172      000000      $TMP0: .WORD 0      ::USER DEFINED
540      001174      000000      $TMP1: .WORD 0      ::USER DEFINED
541      001176      000000      $TMP2: .WORD 0      ::USER DEFINED
542      001200      000000      $TMP3: .WORD 0      ::USER DEFINED
543      001202      000000      $TMP4: .WORD 0      ::USER DEFINED
544      001204      000000      $TMP5: .WORD 0      ::USER DEFINED
545      001206      000000      $TIMES: 0      ::MAX. NUMBER OF ITERATIONS
546      001210      000000      $ESCAPE: 0      ::ESCAPE ON ERROR
547      001212      177607 000377      $BELL: .ASCIZ <207><377><377>      ::CODE FOR BELL
548      001216      077      $QUES: .ASCII /?/      ::QUESTION MARK
549      001217      015      $CRLF: .ASCII <15>      ::CARRIAGE RETURN
550      001220      000012      $LF: .ASCIZ <12>      ::LINE FEED
551      001222      000000      $STTST: .WORD 0      ::HOLDS THE INDEX TO
552      ::THE STARTING ADDRESS TABLE
```


553	001224	000000	CPUEXP: .WORD	0	:HOLDS EXPECTED CPU ERROR CONDITION
554	001226	000000	MMEXP: .WORD		:HOLDS EXPECTED MEMORY MANAGEMENT ABORT CONDITION
555	001230	000000	PADRSL: .WORD	0	:HOLDS THE LOWER 16 BITS OF A 22-BIT
556					:ADDRESS GENERATED FOR TYPE OUT
557	001232	000000	PADRSH: .WORD	0	:HOLDS THE UPPER 6 BITS OF A 22-BIT
558					:ADDRESS GENERATED FOR TYPE OUT
559	001234	000000	PLOADR: .WORD	0	:HOLDS CACHE LO ADDR REG
560	001236	000000	PHIADR: .WORD	0	:HOLDS CACHE HI ADDR REG
561	001240	000000	PPARER: .WORD	0	:HOLDS PARITY ERROR REG
562	001242	000000	PCONTR: .WORD	0	:HOLDS CACHE CONTROL REG
563	001244	000000	PMAINT: .WORD	0	:HOLDS CACHE MAINTENANCE REG
564	001246	000000	PHITMI: .WORD	0	:HOLDS CACHE HIT MISS REG
565	001250	000000	PMMR0: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 0
566	001252	000000	PMMR1: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 1
567	001254	000000	PMMR2: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 2
568	001256	000000	PMMR3: .WORD	0	:HOLDS MEMORY MANAGEMENT REGISTER 3
569	001260	000000	PCPUER: .WORD	0	:HOLDS CPU ERROR REGISTER.
570	001262	000000	BADPC: .WORD	0	:HOLDS PC AT ABORT OR TRAP TIME.
571	001264	000000	TESTNO: .WORD	0	:HOLDS TEST NUMBER OF LAST ERROR.
572	001266	000000	DATAOR: .WORD	0	:HOLDS LOGICAL OR OF BAD DATA
573	001270	000000	DATAND: .WORD	0	:HOLDS LOGICAL AND OF BAD DATA
574	001272	000000	PATTOR: .WORD	0	:HOLDS LOGICAL OR OF PATTERN LOADED
575	001274	000000	PATAND: .WORD	0	:HOLDS LOGICAL AND OF PATTERN LOADED
576	001276	000000	ADDROR: .WORD	0	:HOLDS LOGICAL OR OF ADDRESS
577	001300	000000	ADRAND: .WORD	0	:HOLDS LOGICAL AND OF ADDRESS
578	001302	000000	ERRCNT: .WORD	0	:HOLDS NUMBER OF ERRORS ON TEST
579	001304	000000	HOLFLG: .WORD	0	:HOLDS NUMBER OF CONSECUTIVE TIME OUTS
580					:OCCURRING IN A HOLE IN MEMORY
581	001306	000000	OLDPC: .WORD	0	:HOLDS THE RETURN ADDRESS
582					:IN CASE OF A LOOP ON ERROR
583	001310	000000	OLDPS: .WORD	0	:HOLDS THE OLD PROCESSOR STATUS
584					:IN CASE OF A LOOP ON ERROR
585	001312	000340	OLDPSW: .WORD	000340	:HOLDS THE OLD PSW SO THAT THE T-BIT
586					:CAN BE RESTORED PROPERLY
587	001314	000000	RETRY: .WORD	0	:FLAG TO DECIDE IF ONE PARITY HAS
588					:HAS BEEN ATTEMPTED
589	001316	000000	NXTTST: .WORD	0	:HOLDS ADDRESS OF THE NEXT TEST
590					
591			:: THIS AREA STARTS THE DATA TABLE WHERE ANY TABLE REFERENCE WILL		
592			:: REFER TO. ALL DATA WORDS WILL BE LOADED HERE IN ONE SPOT SO THAT		
593			:: IT WILL BE EASIER FOR YOU TO FIND OUT WHAT THAT WORD IS.		
594					
595	001320	020000	READON: .WORD	20000	:READ ONLY BIT IN MMRO
596					
597					
598	001322		PARTAB:		:THIS IS THE TABLE OF THE FIRST
599					:PAR OR PDR OF EACH GROUP. THEY
600					:WILL BE USED FOR A DUAL ADDRESSING
601					:TEST BETWEEN GROUPS.
602	001322	172200	.WORD	172200	:SIPDR0
603	001324	172240	.WORD	172240	:SIPAR0
604	001326	172300	.WORD	172300	:KIPDR0
605	001330	172340	.WORD	172340	:KIPAR0
606	001332	177600	.WORD	177600	:UIPDR0
607	001334	177640	.WORD	177640	:UIPAR0
608	001336	040670	STRTAB: .WORD	TST1	:STARTING ADDRESS OF TEST ONE

609	001340	041636	.WORD	ENTPT2	:ADDRESS OF ENTRY POINT 2
610	001342	043014	.WORD	ENTPT3	:ADDRESS OF ENTRY POINT 3
611	001344	047524	.WORD	ENTPT4	:ADDRESS OF ENTRY POINT 4
612	001346	052652	.WORD	ENTPT5	:ADDRESS OF ENTRY POINT 5
613	001350	060024	.WORD	ENTPT6	:ADDRESS OF ENTRY POINT 6
614	001352	062514	.WORD	ENTPT7	:ADDRESS OF ENTRY POINT 7
615	001354	070214	.WORD	ENTPT8	:ADDRESS OF ENTRY POINT 8
616	001356	001000	ENMMTR: .WORD	BIT9	:ENABLE MEMORY MANAGEMENT TRAPS BIT
617	001360	000	KB11E: .BYTE	0	:KB-11E WITHOUT MP CACHE FLAG
618	001361	000	KB11EM: .BYTE	0	:KB-11EM WITH MP CACHE FLAG
619	001362	000	KB11CM: .BYTE	0	:MODIFIED PROCESSOR FLAG (11/70 WITH MP MODS)
620	001363	000	CISP: .BYTE	0	:CISP OPTION FLAG (NOT PRESENTLY NEEDED)
621					
622					
623	000007				:OPCODE FOR MFPT INSTRUCTION (AVAILABLE ON KB11-E AND KB11-EM ONLY) MFPT=7

624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679

```
*****  
.SBTTL ERROR POINTER TABLE  
:*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.  
:*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN  
:*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.  
:*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).  
:*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:  
  
:* EM ;;POINTS TO THE ERROR MESSAGE  
:* DH ;;POINTS TO THE DATA HEADER  
:* DT ;;POINTS TO THE DATA  
:* DF ;;POINTS TO THE DATA FORMAT  
  
$ERRTB:  
  
:ITEM 1  
EM1 ;NOT THE CORRECT CPU ABORT CONDITION THRU 'ERRVEC' (004)  
DH1 ;EXPECTD RECEIVD TESTNO PC AT ABORT  
DT1 ;CPUEXP,PCPUER,TESTNO,BADPC,0  
DF1 ;0,0,0,0  
  
:ITEM 2  
EM2 ;UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)  
DH2 ;RECEIVD TESTNO PC AT ABORT  
DT2 ;PCPUER,TESTNO,BADPC,0  
DF2 ;0,0,0  
  
:ITEM 3  
EM3 ;UNEXPECTED CACHE PARITY ERROR THRU 'CACHVEC' (114)  
DH3 ;PARITY ADDRESS CONTROL MAINTEN  
DT3 ;RECEIVD REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT  
DF3 ;PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0  
;0,2,0,0,0,0  
  
:ITEM 4  
EM4 ;UNEXPECTED MAIN MEMORY PARITY ERROR THRU 'CACHVEC' (114)  
DH3 ;PARITY ADDRESS CONTROL MAINTEN  
DT3 ;RECEIVD REFERENC'D REGISTR REGISTR TESTNO PC AT ABORT  
DF3 ;PPARER,PLOADR,PCONTR,PMAINT,TESTNO,BADPC,0  
;0,2,0,0,0,0  
  
:ITEM 5  
EM5 ;UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT  
DH5 ;ERROR AUTOI/D VIRTUAL  
DT5 ;REGISTR REGISTR ADDRESS TESTNO PC AT ABORT  
DF5 ;PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0  
;0,0,0,0,0  
  
:ITEM 6  
EM6 ;MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION  
DH6 ;EXPECTD ERROR AUTOI/D VIRTUAL  
;CONDIIN REGISTR REGISTR ADDRESS TESTNO PC AT ABORT
```

001364

001364 003447
001366 017003
001370 023532
001372 024650

001374 003537
001376 017013
001400 023534
001402 024651

001404 003620
001406 017047

001410 023544
001412 024654

001414 003736
001416 017047

001420 023544
001422 024654

001424 004061
001426 017177

001430 023562
001432 024662

001434 004134
001436 017303

680	001440	023576	DT6	:MMEXP,PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
681	001442	024667	DF6	:0,0,0,0,0,0
682				
683			: ITEM 7	
684	001444	004224	EM7	:MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR
685	001446	017427	DH7	:(MMR0) TESTNO ERRORPC
686	001450	023614	DT7	:\$REG1,TESTNO,\$ERRPC,0
687	001452	024675	DF7	:0,0,0
688				
689			: ITEM 10	
690	001454	004300	EM10	:CAN'T SET 171000 INTO MMR0
691	001456	017427	DH7	:(MMR0) TESTNO ERRORPC
692	001460	023614	DT7	:\$REG1,TESTNO,\$ERRPC,0
693	001462	024675	DF7	:0,0,0
694				
695			: ITEM 11	
696	001464	004331	EM11	:GOT THE WRONG DATA BACK FROM MMR0
697	001466	017457	DH11	:LOADED RECEIVED TESTNO ERRORPC
698	001470	023624	DT11	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
699	001472	024700	DF11	:0,0,0,0
700				
701			: ITEM 12	
702	001474	004373	EM12	:MEMORY MANAGEMENT REGISTER 1 WILL NOT CLEAR
703	001476	017527	DH12	:(MMR1) TESTNO ERRORPC
704	001500	023636	DT12	:\$REG2,TESTNO,\$ERRPC,0
705	001502	024704	DF12	:0,0,0
706				
707			: ITEM 13	
708	001504	004447	EM13	:MMR1 DID NOT TRACK PROPERLY
709	001506	017517	DH13	:EXPECTED (MMR1) TESTNO ERRORPC
710	001510	023646	DT13	:\$REG3,\$REG2,TESTNO,\$ERRPC,0
711	001512	024707	DF13	:0,0,0,0
712				
713			: ITEM 14	
714	001514	004503	EM14	:MMR2 DID NOT TRACK PROPERLY
715	001516	017557	DH14	:EXPECTED (MMR2) TESTNO ERRORPC
716	001520	023646	DT13	:\$REG3,\$REG2,TESTNO,\$ERRPC
717	001522	024707	DF13	:0,0,0,0
718				
719			: ITEM 15	
720	001524	004537	EM15	:MEMORY MANAGEMENT REGISTER 3 WILL NOT CLEAR
721	001526	017627	DH15	:(MMR3) TESTNO ERRORPC
722	001530	023636	DT12	:\$REG,TESTNO,\$ERRPC,0
723	001532	024704	DF12	:0,0,0
724				
725			: ITEM 16	
726	001534	004613	EM16	:MMR3 IS HOLDING THE WRONG DATA
727	001536	017617	DH16	:LOADED (MMR3) TESTNO ERRORPC
728	001540	023660	DT16	:\$REG1,\$REG2,TESTNO,\$ERRPC,0
729	001542	024713	DF16	:0,0,0,0
730				
731			: ITEM 17	
732	001544	004646	EM17	:SUMMARY OF PAR/PDR REFERENCE TIMEOUTS
733	001546	017657	DH17	:ADDROR ADDRAND TESTNO #ERRGRS
734	001550	023672	DT17	:ADDROR,ADDRAND,TESTNO,ERRCNT,0
735	001552	024722	DF17	:0,0,0,1

736				
737			: ITEM 20	
738	001554	004714	EM20	: FOLLOWING PAR/PDR WILL NOT ZERO
739	001556	017717	DH20	: ADDRESS DATA TESTNO ERRORPC
740	001560	023704	DT20	: \$REG0, \$REG2, TESTNO, \$ERRPC, 0
741	001562	024726	DF20	: 0,0,0,0
742				
743			: ITEM 21	
744	001564	004755	EM21	: SUMMARY OF DUAL ADDRESSING ERRORS
745	001566	017757	DH21	: ADDROR ADDRAND ADDROR ADDRAND
746				: LOADED LOADED ENABLED ENABLED TESTNO #ERRORS
747	001570	023716	DT21	: ADDROR, ADDRAND, DATAOR, DATAND, TESTNO, ERRCNT, 0
748	001572	024732	DF21	: 0,0,0,0,0,1
749				
750			: ITEM 22	
751	001574	005017	EM22	: SUMMARY OF COUNT PATTERN FAILURES
752	001576	020077	DH22	: ADDROR ADDRAND PATRNOR PATRNAD DATAOR DATAAND TESTNO #ERRORS
753	001600	023734	DT22	: ADDROR ADDRAND, PATTOR, PATAMD, DATAOR, DATAND, TESTNO, ERRCNT, 0
754	001602	024740	DF22	: 0,0,0,0,0,0,0,1
755				
756			: ITEM 23	
757	001604	005061	EM23	: ERROR IN BYTE ADDRESSING OF PAR/PDR
758	001606	020176	DH23	: ADDRESS EXPECTD RECEIVD TESTNO ERRORPC
759	001610	023756	DT23	: \$REG0, \$REG2, \$REG1, TESTNO, \$ERRPC, 0
760	001612	024750	DF23	: 0,0,0,0,0
761				
762			: ITEM 24	
763	001614	005125	EM24	: ONE OF THE REGISTERS TIMED OUT
764	001616	020246	DH24	: REGADDR TESTNO ERRORPC
765	001620	023772	DT24	: \$REG0, TESTNO, \$ERRPC, 0
766	001622	024755	DF24	: 0,0,0
767				
768			: ITEM 25	
769	001624	005164	EM25	: LOW BYTE OF LOW SIZE REGISTER IS NOT ALL ONES
770	001626	020276	DH25	: REGADDR DATA TESTNO ERRORPC
771	001630	024002	DT25	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
772	001632	024760	DF25	: 0,0,0,0
773				
774			: ITEM 26	
775	001634	005242	EM26	: COULD WRITE ONE OF THE SIZE REGISTERS
776	001636	020276	DH25	: REGADDR DATA TESTNO ERRORPC
777	001640	024002	DT25	: \$REG0, \$REG2, TESTNO, \$ERRPC, 0
778	001642	024760	DF25	: 0,0,0,0
779				
780			: ITEM 27	
781	001644	005310	EM27	: HIGH SIZE REGISTER IS NOT ZERO
782	001646	020276	DH25	: REGADDR DATA TESTNO ERRORPC
783	001650	024002	DT25	: \$REG0, \$REG1, TESTNO, \$ERRPC, 0
784	001652	024760	DF25	: 0,0,0,0
785				
786			: ITEM 30	
787	001654	005347	EM30	: CPU ERROR REGISTER NOT ZERO AFTER LOADING NEGATIVE ONE.
788	001656	020276	DH25	: REGADDR DATA TESTNO ERRORPC
789	001660	024002	DT25	: \$REG0, \$REG1, TESTNO, \$ERRPC
790	001662	024760	DF25	: 0,0,0,0
791				


```

792      ;ITEM 31
793 001664 005436      EM31      :LOWER BYTE OF P.S.W. NOT CORRECT
794 001666 020336      DH31      :REGADDR DATAREC DATAEXP TESTNO ERRORPC
795 001670 024014      DT31      :$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
796 001672 024764      DF31      :0,0,0,0,0
797
798      ;ITEM 32
799 001674 005477      EM32      :MICRO BREAK REG LOADED INCORECTLY
800 001676 020336      DH31      :REGADDR DATAREC DATAEXP TESTNO ERRORPC
801 001700 024014      DT31      :$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
802 001702 024764      DF31      :0,0,0,0,0
803
804      ;ITEM 33
805 001704 005711      EM33      :DIDN'T LOAD PROGRAM INTERRUPT REQUEST REGISTER
806 001706 020336      DH31      :REGADDR DATAREC DATAEXP TESTNO ERRORPC
807 001710 024014      DT31      :$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
808 001712 024764      DF31      :0,0,0,0,0
809
810      ;IJWM 34
811 001714 005770      EM34      :LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER
812 001716 020336      DH31      :REGADDR DATAREC DATAEXP TESTNO ERRORPC
813 001720 024014      DT31      :$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
814 001722 024764      DF31      :0,0,0,0,0
815
816      ;ITEM 35
817 001724 006054      EM35      :KERNEL STACK POINTER NOT 1100 AFTER LOADING SSP AND USP
818 001726 020406      DH35      :KSP TESTNO ERRORPC
819 001730 024030      DT35      :$REG0,TESTNO,$ERRPC,0
820 001732 024771      DF35      :0,0,0
821
822      ;ITEM 36
823 001734 006142      EM36      :DUAL ADDRESSING BETWEEN PAR/PDR GROUPS
824 001736 020436      DH36      :INDEX INDEX PAR/PDR
825      :EXPECTD RECEIVD ADDRREAD TESTNO ERRORPC
826 001740 024040      DT36      :$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
827 001742 024774      DF36      :0,0,0,0,0
828
829      ;ITEM 37
830 001744 006211      EM37      :BAD RELOCATION, ON STORING DATA 18-BIT MAPPING
831 001746 020536      DH37      :ADDRESS GDDATA BADDATA TESTNO ERRORPC
832 001750 024054      DT37      :$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
833 001752 025001      DF37      :3,0,0,0,0
834
835      ;ITEM 40
836 001754 006270      EM40      :18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
837 001756 020606      DH40      :STARTADR FINISHADR TESTNO ERRORPC
838 001760 024070      DT40      :$TMP1,$TMP2,TESTNO,$ERRPC,0
839 001762 025006      DF40      :4,4,0,0
840
841      ;ITEM 41
842 001764 006351      EM41      :18-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY
843 001766 020652      DH41      :STARTADR TESTNO ERRORPC
844 001770 024102      DT41      :$TMP1,TESTNO,$ERRPC,0
845 001772 025012      DF41      :4,0,0
846
847      ;ITEM 42

```

848	001774	006433	EM42	: FAULTY CARRY PROPAGATION 18-BIT MAPPING.
849	001776	020704	DH42	: PATTERN DATA ADDRESS
850				: LOADED FETCHED INTENDED TESTNO ERRORPC
851	002000	024112	DT42	: \$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
852	002002	025015	DF42	: 0,0,3,0,0
853				
854			: ITEM 43	
855	002004	006504	EM43	: NO TRAP THRU ERRVEC, AT 18-BIT ADDRESS 760000
856	002006	021006	DH43	: TESTNO ERRORPC
857	002010	024126	DT43	: TESTNO,\$ERRPC,0
858	002012	025022	DF43	: 0,0
859				
860			: ITEM 44	
861	002014	006562	EM44	: DIDN'T GET WRAP AROUND TO ADDRESS ZERO
862	002016	021026	DH44	: DATA TESTNO ERRORPC
863	002020	024134	DT44	: \$REG1,TESTNO,\$ERRPC,0
864	002022	025024	DF44	: 0,0,0
865				
866			: ITEM 45	
867	002024	006631	EM45	: NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDRESS
868	002026	021056	DH45	: NON-EXADDR TESTNO ERRORPC
869	002030	024144	DT45	: \$REG0,TESTNO,\$ERRPC,0
870	002032	025027	DF45	: 3,0,0
871				
872			: ITEM 46	
873	002034	006706	EM46	: BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING
874	002036	020704	DH42	: PATTERN DATA ADDRESS
875				: LOADED FETCHED INTENDED TESTNO ERRORPC
876	002040	024112	DT42	: \$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC
877	002042	025015	DF42	: 0,0,3,0,0
878				
879			: ITEM 47	
880	002044	006767	EM47	: DID NOT GET UNIBUS ADDRESS
881	002046	020704	DH42	: PATTERN DATA ADDRESS
882				: LOADED FETCHED INTENDED TESTNO ERRORPC
883	002050	024112	DT42	: \$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0
884	002052	025015	DF42	: 0,0,3,0,0
885				
886			: ITEM 50	
887	002054	007022	EM50	: COMPARE CIRCUIT FOR TOP OF MEMORY BAD
888	002056	021111	DH50	: KIPAR4 SIZELO TESTNO ERRORPC
889	002060	024154	DT50	: KIPAR4,SIZELO,TESTNO,\$ERRPC,0
890	002062	025032	DF50	: 4,4,0,0
891				
892			: ITEM 51	
893	002064	007070	EM51	: PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS
894	002066	021155	DH51	: PGLNFD VABLKNO TESTNO ERRORPC
895	002070	024166	DT51	: \$REG1,\$REG3,TESTNO,\$ERRPC,0
896	002072	025036	DF51	: 0,0,0,0
897				
898			: ITEM 52	
899	002074	007143	EM52	: NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT
900	002076	021155	DH51	: PGLNFD VABLKNO TESTNO ERRORPC
901	002100	024166	DT51	: \$REG1,\$REG3,TESTNO,\$ERRPC,0
902	002102	025036	DF51	: 0,0,0,0
903				

904			: ITEM 53	
905	002104	007220	EM53	: DID NOT ABORT ON NON-RESIDENT PAGE KIPDR5
906	002106	021215	DH53	: TESTNO ERRORPC
907	002110	024200	DT53	: TESTNO,\$ERRPC,0
908	002112	025042	DF53	: 0,0
909				
910			: ITEM 54	
911	002114	007272	EM54	: DID NOT ABORT ON READ ONLY PAGE KIPDR4
912	002116	021215	DH53	: TESTNO ERRORPC
913	002120	024200	DT53	: TESTNO,\$ERRPC,0
914	002122	025042	DF53	: 0,0
915				
916			: ITEM 55	
917	002124	007341	EM55	: INCORRECT READ FROM PAGE KIPDR4 BIT09 (MMRO) CLEAR
918	002126	021235	DH55	: EXPDATA RECDATA TESTNO ERRORPC
919	002130	024206	DT55	: \$REG0,\$REG1,TESTNO,\$ERRPC,0
920	002132	025044	DF55	: 0,0,0,0
921				
922			: ITEM 56	
923	002134	007424	EM56	: NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR4
924	002136	021275	DH56	: (MMRO) KIPDR4 TESTNO ERRORPC
925	002140	024220	DT56	: MMRO,KIPDR4,TESTNO,\$ERRPC,0
926	002142	025050	DF56	: 0,0,0,0
927				
928			: ITEM 57	
929	002144	007503	EM57	: INCORRECT READ FROM PAGE KIPDR4, BIT09 (MMRO) SET
930	002146	021235	DH55	: EXPDATA RECDATA TESTNO ERRORPC
931	002150	024206	DT55	: \$REG0,\$REG1,TESTNO,\$ERRPC,0
932	002152	025044	DF55	: 0,0,0,0
933				
934			: ITEM 60	
935	002154	007565	EM60	: INCORRECT WRITE TO PAGE KIPDR4, BIT09 (MMRO) SET
936	002156	021215	DH53	: TESTNO ERRORPC
937	002160	024200	DT53	: TESTNO,\$ERRPC,0
938	002162	025042	DF53	: 0,0
939				
940			: ITEM 61	
941	002164	007646	EM61	: NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR7
942	002166	021275	DH56	: (MMRO) KIPDR4 TESTNO ERRORPC
943	002170	024220	DT56	: MMRO,KIPDR4,TESTNO,\$ERRPC,0
944	002172	025050	DF56	: 0,0,0,0
945				
946			: ITEM 62	
947	002174	007725	EM62	: INCORRECT READ FROM PAGE KIPDR7, BIT09 (MMRO) SET
948	002176	021235	DH55	: EXPDATA RECDATA TESTNO ERRORPC
949	002200	024206	DT55	: \$REG0,\$REG1,TESTNO,\$ERRPC,0
950	002202	025044	DF55	: 0,0,0,0
951				
952			: ITEM 63	
953	002204	010007	EM63	: TRAP WHEN NO RELOCATION
954	002206	021215	DH53	: TESTNO ERRORPC
955	002210	024200	DT53	: TESTNO,\$ERRPC,0

956	002212	025042	DF53	:0,0
957				
958			:ITEM 64	
959	002214	010037	EM64	:TOOK TWO VECTORS WITH TRAP AND ABORT ON SAME INSTRUCTION
960				:EXPECTING '1074' FOR KERNEL STACK POINTER
961	002216	021335	DH64	:RECEIVED TESTNO ERRORPC
962	002220	024232	DT64	:\$TMP0,TESTNO,\$ERRPC,0
963	002222	025054	DF64	:0,0,0

```

964
965      :ITEM 65
966 002224 010202      EM65      :MEMORY MANAGEMENT ABORT CONDITION WRONG
967 002226 021365      DH65      :EXPCOND ABRTCOND TESTNO ERRORPC
968 002230 024242      DT65      :MMEXP,PMR0,TESTNO,$ERRPC,0
969 002232 025057      DF65      :0,0,0,0
970
971      :ITEM 66
972 002234 010252      EM66      :BIT 12 OF MMR0 WAS NOT SET WHEN A.C.F. SATISFIED
973 002236 021425      DH66      :(MMR0) TESTNO ERRORPC
974 002240 024254      DT66      :PMR0,TESTNO,$ERRPC,0
975 002242 025063      DF66      :0,0,0
976
977      :ITEM 67
978 002244 010333      EM67      :NO TRAP WHEN INSTRUCTION CLEARING BIT09 (MMR0) CAUSES TRAP COND
979 002246 021215      DH53      :TESTNO ERRORPC
980 002250 024200      DT53      :TESTNO,$ERRPC,0
981 002252 025042      DF53      :0,0
982
983      :ITEM 70
984 002254 010434      EM70      :ERROR DURING M.M. ABORT IN TRAP SFQUENCE
985      :EXPECTED:
986 002256 021455      DH70      :XXX017 040241 173366 000004
987      :PROSTAT (MMR0) (MMR1) (MMR2) TESTNO ERRORPC
988      :RECEIVED:
989 002260 024264      DT70      :$REG1,PMR0,PMR1,PMR2,TESTNO,$ERRPC,0
990 002262 025066      DF70      :0,0,0,0,0,0
991
992      :ITEM 71
993 002264 010517      EM71      :INSTRUCTION FETCH DID NOT ABORT IN ILLEGAL MODE (10)
994 002266 021215      DH53      :TESTNO ERRORPC
995 002270 024200      DT53      :TESTNO,$ERRPC,0
996 002272 025042      DF53      :0,0
997
998      :ITEM 72
999 002274 010604      EM72      :ERROR CONDITION NOT CORRECT IN ILLEGAL MODE (10)
1000 002276 021606      DH72      :EXPSTAT (MMR0) TESTNO ERRORPC
1001 002300 024302      DT72      :$REG1,PMR0,TESTNO,$ERRPC,0
1002 002302 025074      DF72      :0,0,0,0
1003
1004      :ITEM 73
1005 002304 010665      EM73      :AT LEAST ONE M.M. STATUS REGISTERS WAS CLOKED AFTER BEING LOCK
1006 002306 021646      DH73      :ORIGINAL DATA      NEW DATA
1007      :(MMR0) (MMR1) (MMR2) (MMR0) (MMR1) (MMR2) TESTNO ERRORPC
1008 002310 024314      DT73      :PMR0,PMR1,PMR2,$TMP0,$TMP1,$TMP2,TESTNO,$ERRPC,0
1009 002312 025100      DF73      :0,0,0,0,0,0,0,0
1010
1011      :ITEM 74
1012 002314 010767      EM74      :DID NOT CHANGE MAPPING TO SUPERVISOR MODE
1013 002316 021215      DH53      :TESTNO ERRURPC
1014 002320 024200      DT53      :TESTNO,$ERRPC,0
1015 002322 025042      DF53      :0,0
1016
1017      :ITEM 75
1018 002324 011041      EM75      :ABORT CONDITION INCORRECT EXPECTING 100051
1019 002326 021777      DH75      :RECEIVD (MMR1) (MMR2) TESTNO ERRORPC
    
```


1020	002330	024336	DT75	:PMMR0,PMMR1,PMMR2,TESTNO,\$ERRPC,0
1021	002332	025110	DF75	:0,0,0,0,0
1022				
1023			:ITEM 76	
1024	002334	011114	EM76	:DID NOT CHANGE MAPPING TO USER MODE
1025	002336	021215	DH53	:TESTNO ERRORPC
1026	002340	024200	DT53	:TESTNO,\$ERRPC,0
1027	002342	025042	DF53	:0,0
1028				
1029			:ITEM 77	
1030	002344	011160	EM77	:ABORT CONDITION INCORRECT EXPECTING 100153
1031	002346	021777	DH75	:RECEIVD (MMR1) (MMR2) TESTNO ERRORPC
1032	002350	024336	DT75	:PMMR0,PMMR1,PMMR2,TESTNO,\$ERRPC,0
1033	002352	025110	DF75	:0,0,0,0,0
1034				
1035			:ITEM 100	
1036	002354	011233	EM100	:MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS
1037	002356	022047	DH100	:VIRTADR (MMR2) TESTNO ERRORPC
1038	002360	024352	DT100	:\$REG1,PMMR2,TESTNO,\$ERRPC,0
1039	002362	025115	DF100	:0,0,0,0
1040				
1041			:ITEM 101	
1042	002364	011304	EM101	:MMR0 LOCKED UP THE WRONG PAGE NUMBER
1043	002366	022107	DH101	:EXPECTD (MMR0) TESTNO ERRORPC
1044	002370	024364	DT101	:\$REG2,PMMR0,TESTNO,\$ERRPC,0
1045	002372	025121	DF101	:0,0,0,0
1046				
1047			:ITEM 102	
1048	002374	011351	EM102	:W-BIT NOT SET ON WRITE TO PAGE 4
1049	002376	022147	DH102	:KIPDR4 TESTNO ERRORPC
1050	002400	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1051	002402	025125	DF102	:0,0,0
1052				
1053			:ITEM 103	
1054	002404	011412	EM103	:W-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1055	002406	022147	DH102	:KIPDR4 TESTNO ERRORPC
1056	002410	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1057	002412	025125	DF102	:0,0,0
1058				
1059			:ITEM 104	
1060	002414	011473	EM104	:W-BIT DID NOT CLEAR ON WRITE TO KIPDR4
1061	002416	022147	DH102	:KIPDR4 TESTNO ERRORPC
1062	002420	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1063	002422	025125	DF102	:0,0,0
1064				
1065			:ITEM 105	
1066	002424	011542	EM105	:A-BIT DID NOT SET ON READ TO PAGE 4 A.C.F.-4
1067	002426	022147	DH102	:KIPDR4 TESTNO ERRORPC
1068	002430	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1069	002432	025125	DF102	:0,0,0
1070				
1071			:ITEM 106	
1072	002434	011617	EM106	:A-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4
1073	002436	022147	DH102	:KIPDR4 TESTNO ERRORPC
1074	002440	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1075	002442	025125	DF102	:0,0,0

1076				
1077			:ITEM 107	
1078	002444	011700	EM107	:A-BIT DID NOT CLEAR ON WRITE TO KIPAR4
1079	002446	022147	DH102	:KIPDR4 TESTNO ERRORPC
1080	002450	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1081	002452	025125	DF102	:0,0,0
1082				
1083			:ITEM 110	
1084	002454	011747	EM110	:W-BIT DID NOT SET ON WRITE TO I/O PAGE
1085	002456	022177	DH110	:KIPDR7 TESTNO ERRORPC
1086	002460	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1087	002462	025125	DF102	:0,0,0
1088				
1089			:ITEM 111	
1090	002464	012016	EM111	:W-BIT DID NOT REMAIN SET AFTER INTERNAL REGISTER WRITE
1091	002466	022177	DH110	:KIPDR7 TESTNO ERRORPC
1092	002470	024376	DT102	:\$TMP0,TESTNO,\$ERRPC,0
1093	002472	025125	DF102	:0,0,0
1094				
1095			:ITEM 112	
1096	002474	012105	EM112	:DUAL MAPPING BETWEEN PAGES
1097	002476	022227	DH112	:GDPAGE BDPAGE TESTNO ERRORPC
1098	002500	024406	DT112	:\$REG3,\$REG1,TESTNO,\$ERRPC,0
1099	002502	025130	DF112	:0,0,0,0
1100				
1101			:ITEM 113	
1102	002504	012140	EM113	:NO PAGE HAD BOTH ITS A & W BITS SET
1103	002506	022267	DH113	:TSTPAGE CONTENT TESTNO ERRORPC
1104	002510	024420	DT113	:\$REG3,\$REG0,TESTNO,\$ERRPC,0
1105	002512	025134	DF113	:0,0,0,0
1106				
1107			:ITEM 114	
1108	002514	012204	EM114	:DID NOT PICK UP CORRECT STACK POINTER
1109	002516	022327	DH114	:EXPECTD RECEIVD TESTNO ERRORPC
1110	002520	024432	DT114	:\$REG2,\$REG1,TESTNO,\$ERRPC,0
1111	002522	025140	DF114	:0,0,0,0
1112				
1113			:ITEM 115	
1114	002524	012252	EM115	:CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION
1115	002526	021215	DH53	:TESTNO ERRORPC
1116	002530	024200	DT53	:TESTNO,\$ERRPC,0
1117	002532	025042	DF53	:0,0
1118				
1119			:ITEM 116	
1120	002534	012333	EM116	:WRONG DATA FETCHED BY MFP INSTRUCTION
1121	002536	022367	DH116	:EXPECTD RECEIVD TESTNO ERRORPC
1122	002540	024444	DT116	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1123	002542	025144	DF116	:0,0,0,0
1124				
1125			:ITEM 117	
1126	002544	012401	EM117	:TRIED TO REFERENCE NON-RESIDENT PAGE
1127	002546	022427	DH117	:(MMR0) (MMR1) (MMR2) TESTNO ERRORPC
1128	002550	024456	DT117	:PMR0,PMR1,PMR2,TESTNO,\$ERRPC,0
1129	002552	025150	DF117	:0,0,0,0,0
1130				
1131			:ITEM 120	

1132	002554	012446	EM120	:STACK POINTER NOT CHANGED BY MTP INSTRUCTION
1133	002556	022477	DH120	:STKPTR TESTNO ERRORPC
1134	002560	024472	DT120	:\$REG1,TESTNO,\$ERRPC,0
1135	002562	025155	DF120	:0,0,0
1136				
1137			:ITEM 121	
1138	002564	012523	EM121	:INCORRECT STORE BY MTP INSTRUCTION
1139	002566	022527	DH121	:GDDATA STORED TESTNO ERRORPC
1140	002570	024444	DT116	:\$REG0,\$REG1,TESTNO,\$ERRPC,0
1141	002572	025144	DF116	:0,0,0,0
1142				
1143			:ITEM 122	
1144	002574	012566	EM122	:ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW
1145	002576	022567	DH122	:(PSW) TESTNO ERRORPC EXPECTING XXX340
1146	002600	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1147	002602	025160	DF122	:0,0,0
1148				
1149			:ITEM 123	
1150	002604	012650	EM123	:ABORTED THRU SUPERVISOR SPACE, SUPERVISOR PSW IS XXX140
1151	002606	022640	DH123	:(PSW) TESTNO ERRORPC
1152	002610	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1153	002612	025160	DF122	:0,0,0
1154				
1155			:ITEM 124	
1156	002614	012740	EM124	:ABORTED THRU USER SPACE, USER PSW IS XXX000
1157	002616	022640	DH123	:(PSW) TESTNO ERRORPC
1158	002620	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1159	002622	025160	DF122	:0,0,0
1160				
1161			:ITEM 125	
1162	002624	013014	EM125	:22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM
1163	002626	020606	DH40	:STARTADR FINISHADR TESTNO ERRORPC
1164	002630	024070	DT40	:\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
1165	002632	025006	DF40	:4,4,0,0
1166				
1167			:ITEM 126	
1168	002634	013075	EM126	:22-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY
1169	002636	020652	DH41	:STARTADR TESTNO ERRORPC
1170	002640	024102	DT41	:\$TMP1,TESTNO,\$ERRPC,0
1171	002642	025012	DF41	:4,0,0
1172				
1173			:ITEM 127	
1174	002644	013157	EM127	:DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER
1175	002646	022670	DH127	:TESTNO ERRORPC
1176	002650	024512	DT127	:TESTNO,\$ERRPC,0
1177	002652	025163	DF127	:0,0
1178				
1179			:ITEM 130	
1180	002654	013247	EM130	:ABORT IN KERNEL D-SPACE PICKED UP VECTOR FROM I-SPACE
1181	002656	022567	DH122	:(PSW) TESTNO ERRORPC EXPECTING XXX340
1182	002660	024502	DT122	:\$REG0,TESTNO,\$ERRPC,0
1183	002662	025160	DF122	:0,0,0
1184				
1185			:ITEM 131	
1186	002664	013335	EM131	:M.M. ABORT IN KERNEL D-SPACE HAD WRONG CONDITION
1187	002666	022707	DH131	:(MMR0) (MMR1) (MMR2) TESTNO ERRORPC EXPECTING 020031

1188	002670	024520	DT131	:PMMR0,PMMR1,PMMR2,TESTNO,\$ERRPC,0
1189	002672	025165	DF131	:0,0,0,0,0
1190				
1191			:ITEM 132	
1192	002674	013416	EM132	:D SPACE ENABLE CIRCUITRY HAS FAILED
1193	002676	017177	DH5	:ERROR AUTOI/D VIRTUAL
1194				:REGISTR REGISTR ADDRESS TESTNO PC AT ABORT
1195	002700	023562	DT5	:PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0
1196	002702	024662	DF5	:0,0,0,0,0
1197			:ITEM 133	
1198	002704	013462	EM133	:BYP BIT IN KIPDR COULD NOT BE CLEARED
1199	002706	023000	DH133	: PC KIPDR (KIPDR)
1200	002710	024534	DT133	: \$ERRPC,\$REG0,\$REG1,0
1201	002712	024651	DF2	:0,0,0
1202			:ITEM 134	
1203	002714	013530	EM134	:BYP BIT IN KIPDR COULD NOT BE SET
1204	002716	023000	DH133	
1205	002720	024534	DT133	
1206	002722	024651	DF2	
1207				
1208			:ITEM 135	
1209	002724	013572	EM135	:TEST DATA COULD NOT BE MADE HIT
1210	002726	023027	DH135	: PC CCR PARADR PAR PDR TST-DATA-ADR
1211	002730	024544	DT135	: \$ERRPC,\$REG0,\$REG1,\$REG2,\$REG3,\$REG4,0
1212	002732	024667	DF6	
1213				
1214			:ITEM 136	
1215	002734	013632	EM136	:TEST DATA REFERENCE NOT A MISS
1216				:CACHED DATA WAS NOT FORCED A MISS ON VIRTUAL PAGE BYPASS
1217	002736	023027	DH135	
1218	002740	024544	DT135	
1219	002742	024667	DF6	
1220				
1221			:ITEM 137	
1222	002744	013763	EM137	:TEST DATA REFERENCE NOT A MISS
1223				:CACHED DATA WAS NOT INVALIDATED ON VIRTUAL BYPASS
1224	002746	023027	DH135	
1225	002750	024544	DT135	
1226	002752	024667	DF6	
1227			:ITEM 140	
1228	002754	014112	EM140	:BYP BIT IN SIPDR COULD NOT BE CLEARED
1229	002756	023116	DH140	: PC SIPDR (SIPDR)
1230	002760	024534	DT133	
1231	002762	024651	DF2	
1232				
1233			:ITEM 141	
1234	002764	014160	EM141	:BYP IN SIPDR COULD NOT BE SET
1235	002766	023116	DH140	
1236	002770	024534	DT133	
1237	002772	024651	DF2	
1238				
1239			:ITEM 142	
1240	002774	014222	EM142	:BYP BIT IN UIPDR COULD NOT BE CLEARED
1241	002776	023145	DH142	: PC UIPDR (UIPDR)
1242	003000	024534	DT133	
1243	003002	024651	DF2	

```

1244
1245      :ITEM 143
1246 003004 014270      EM143      ;BYP BIT IN UIPDR COULD NOT BE SET
1247 003006 023145      DH142
1248 003010 024534      DT133
1249 003012 024651      DF2
1250
1251      :ITEM 144
1252 003014 005606      EM32M      ;MICRO BREAK REG LOADED INCORECTLY. 16 BITS WRITABLE
1253 003016 020336      DH31      ;REGADDR DATAREC DATAEXP TESTNO ERRORPC
1254 003020 024014      DT31      ;$REG0,$REG1,$REG2,TESTNO,$ERRPC,0
1255 003022 024764      DF31      ;0,0,0,0,0
1256
1257      :ITEM 145
1258 003024 014332      EM145      ;DPAR READ BACK INCORRECTLY
1259 003026 023174      DH145      ;ADDR EXP REC TEST ERR
1260 003030 024562      DT145      ;$TMP0,$TMP1,$TMP2,TESTNO,ERRORPC,0
1261 003032 024764      DF31      ;0,0,0,0,0
1262
1263      :ITEM 146
1264 003034 014461      EM146      ;NO KT ABORT
1265 003036 023243      DH146
1266 003040 024576      DT146
1267 003042 025163      DF127
1268
1269      :ITEM 147
1270 003044 014637      EM147      ;PS<08> FAILED TO SET
1271 003046 023243      DH146
1272 003050 024576      DT146
1273 003052 025163      DF127
1274
1275      :ITEM 150
1276 003054 014671      EM150      ;KT ABORT TRAPPED TO 4
1277 003056 023243      DH146
1278 003060 024576      DT146
1279 003062 025163      DF127
1280
1281      :ITEM 151
1282 003064 014775      EM151      ;KT ABORT TRAPPED TO 10
1283 003066 023243      DH146
1284 003070 024576      DT146
1285 003072 025163      DF127
1286
1287      :ITEM 152
1288 003074 015044      EM152      ;KT ABORT TRAPPED TO 240
1289 003076 023243      DH146
1290 003100 024576      DT146
1291 003102 025163      DF127
1292
1293      :ITEM 153
1294 003104 015110      EM153      ;KT TRAP DID NOT WORK
1295 003106 023243      DH146
1296 003110 024576      DT146
1297 003112 025163      DF127
1298
1299      :ITEM 154
    
```

1300	003114	000000	0	;DELETED
1301	003116	000000	0	
1302	003120	000000	0	
1303	003122	000000	0	
1304				
1305			; ITEM 155	
1306	003124	015217	EM155	;NO KT TRAP ON SOURCE OPERAND
1307	003126	023243	DH146	
1308	003130	024576	DT146	
1309	003132	025163	DF127	
1310				
1311			; ITEM 156	
1312	003134	015337	EM156	;NO ABORT ON NEXM
1313	003136	023243	DH146	
1314	003140	024576	DT146	
1315	003142	025163	DF127	
1316				
1317			; ITEM 157	
1318	003144	000000	0	;DELETED
1319	003146	000000	0	
1320	003150	000000	0	
1321	003152	000000	0	
1322				
1323			; ITEM 160	
1324	003154	015472	EM160	;NEXM BIT DID NOT SET IN CPUERR REG
1325	003156	023267	DH147	
1326	003160	024604	DT147	
1327	003162	025130	DF112	
1328				
1329			; ITEM 161	
1330	003164	015600	EM161	;NEXM BIT DID NOT CLEAR IN CPUERR REG
1331	003166	023243	DH146	
1332	003170	024576	DT146	
1333	003172	025163	DF127	
1334				
1335			; ITEM 162	
1336	003174	015651	EM162	;KT ABORT ON NEXM (KB11-B/C)
1337	003176	023243	DH146	
1338	003200	024576	DT146	
1339	003202	025163	DF127	
1340				
1341			; ITEM 163	
1342	003204	015730	EM163	;KT ABORT ON SL RED
1343	003206	023243	DH146	
1344	003210	024576	DT146	
1345	003212	025163	DF127	
1346				
1347			; ITEM 164	
1348	003214	016005	EM164	;KT ABORT ON ODD ADDRESS
1349	003216	023243	DH146	
1350	003220	024576	DT146	
1351	003222	025163	DF127	
1352				
1353			; ITEM 165	
1354	003224	016070	EM165	;KT ABORT FAILED TO OVER-RIDE NEXM (KB11-E/EM)
1355	003226	023243	DH146	

```

1356 003230 024576 DT146
1357 003232 025163 DF127
1358
1359 ; ITEM 166
1360 003234 016154 EM166 ; TMCE CACHE BEND DID NOT GO
1361 003236 023243 DH146
1362 003240 024576 DT146
1363 003242 025163 DF127
1364
1365 ; ITEM 167
1366 003244 016231 EM167 ;
1367 003246 023342 DH150
1368 003250 000000 0
1369 003252 000000 0
1370
1371 ; ITEM 170
1372 003254 016425 EM170 ; BEN 6 FAILED ON PS RESTORE
1373 003256 023243 DH146
1374 003260 024576 DT146
1375 003262 025163 DF127
1376
1377 ; ITEM 171
1378 003264 016521 EM171 ; GOING TO NEXT TEST
1379 003266 000000 0
1380 003270 000000 0
1381 003272 000000 0
1382 003274 ER200: ; STARTING ADDRESS FOR ITEM 201-377
1383
1384 ; ITEM 201
1385 003274 016546 EM201 ; THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS
1386 003276 023370 DH201 ; ADDRESS TESTNO
1387
1388 ; ITEM 301
1389 003300 024616 DT201 ; $REG0,,TESTNO,0
1390 003302 025172 DF201 ; 0,0
1391
1392 ; ITEM 202
1393 003304 016623 EM202 ; THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S
1394 003306 023407 DH202 ; ADDRESS ADDRESS
1395 ; LOADED JUST READ TESTNO
1396
1397 ; ITEM 302
1398 003310 024624 DT202 ; $REG0,$REG1,TESTNO,0
1399 003312 025174 DF202 ; 0,0,0
1400
1401 ; ITEM 203
1402 003314 016714 EM203 ; THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S
1403 003316 023457 DH203 ; ADDRESS DATA RED PATTERN COUNT TESTNO
1404
1405 ; ITEM 303
1406 003320 024634 DT203 ; $REG0,$REG2,$REG4,$REG1,TESTNO,0
1407 003322 025177 DF203 ; 0,0,0,0,0
1408
1409 ; MESSAGES
1410
1411
    
```


1412	003324	041600	052520	052440	MSG1:	.ASCIZ<CRLF>	'CPU UNDER TEST FOUND TO BE A ''
1413	003332	042116	051105	052040			
1414	003340	051505	020124	047506			
1415	003346	047125	020104	047524			
1416	003354	041040	020105	020101			
1417	003362	000					
1418	003363	113	030502	026461	MSG2:	.ASCIZ	'XB11-EM'<CRLF>
1419	003370	046505	000200				
1420	003374	041113	030461	041055	MSG3:	.ASCIZ	'XB11-B/C'<CRLF>
1421	003402	041457	000200				
1422	003406	041113	030461	041455	MSG4:	.ASCIZ	'XB11-CM
1423	003414	020115	020040	020040			'<CRLF>
1424	003422	020040	020040	020040			
1425	003430	020040	020040	100040			
1426	003436	000					
1427	003437	113	030502	026461	MSG5:	.ASCIZ	'XB11-E'<CRLF>
1428	003444	100105	000				
1429							
1430					.SBTTL	ERROR TABLE MESSAGES AND DATA POINTERS	
1431	003447	116	052117	052040	EM1:	.ASCIZ	?NOT THE CORRECT CPU ABORT CONDITION THRU 'ERRVEC' (004)?
1432	003454	042510	041440	051117			
1433	003462	042522	052103	041440			
1434	003470	052520	040440	047502			
1435	003476	052122	041440	047117			
1436	003504	044504	044524	047117			
1437	003512	052040	051110	020125			
1438	003520	042447	051122	042526			
1439	003526	023503	024040	030060			
1440	003534	024464	000				
1441	003537	125	042516	050130	EM2:	.ASCIZ	?UNEXPECTED CPU TRAP OR ABORT THRU 'ERRVEC' (004)?
1442	003544	041505	042524	020104			
1443	003552	050103	020125	051124			
1444	003560	050101	047440	020122			
1445	003566	041101	051117	020124			
1446	003574	044124	052522	023440			
1447	003602	051105	053122	041505			
1448	003610	020047	030050	032060			
1449	003616	000051					
1450	003620	047125	054105	042520	EM3:	.ASCII	?UNEXPECTED CACHE PARITY FRROR THRU 'CACHVEC' (114)?<CRLF>
1451	003626	052103	042105	041440			
1452	003634	041501	042510	050040			
1453	003642	051101	052111	020131			
1454	003650	051105	047522	020122			
1455	003656	044124	052522	023440			
1456	003664	040503	044103	042526			
1457	003672	023503	024040	030461			
1458	003700	024464	200				
1459	003703	127	046111	020114		.ASCIZ	?W! L RETRY THIS TEST ONCE.?
1460	003710	042522	051124	020131			
1461	003716	044124	051511	052040			
1462	003724	051505	020124	047117			
1463	003732	042503	000056				
1464	003736	047125	054105	042520	EM4:	.ASCII	?UNEXPECTED MAIN MEMOP' PARITY ERROR THRU 'CACHVEC' (114)?<CRLF>
1465	003744	052103	042105	046440			
1466	003752	044501	020116	042515			
1467	003760	047515	054522	050040			

1468	003766	051101	052111	020131	
1469	003774	051105	047522	020122	
1470	004002	044124	052522	023440	
1471	004010	040503	044103	042526	
1472	004016	023503	024040	030461	
1473	004024	024464	200		
1474	004027	127	046111	020114	.ASCIZ ?WILL RETRY THIS TEST ONCE?
1475	004034	042522	051124	020131	
1476	004042	044124	051511	052040	
1477	004050	051505	020124	047117	
1478	004056	042503	000		
1479					
1480	004061	125	042516	050130	EM5: .ASCIZ ?UNEXPECTED MEMORY MANAGEMENT TRAP OR ABORT?
1481	004066	041505	042524	020104	
1482	004074	042515	047515	054522	
1483	004102	046440	047101	043501	
1484	004110	046505	047105	020124	
1485	004116	051124	050101	047440	
1486	004124	020122	041101	051117	
1487	004132	000124			
1488	004134	042515	047515	054522	EM6: .ASCIZ ?MEMORY MANAGEMENT TRAP OR ABORT HAD INCORRECT CONDITION?
1489	004142	046440	047101	043501	
1490	004150	046505	047105	020124	
1491	004156	051124	050101	047440	
1492	004164	020122	041101	051117	
1493	004172	020124	040510	020104	
1494	004200	047111	047503	051122	
1495	004206	041505	020124	047503	
1496	004214	042116	052111	047511	
1497	004222	000116			
1498	004224	042515	047515	054522	EM7: .ASCIZ ?MEMORY MANAGEMENT REGISTER 0 WILL NOT CLEAR?
1499	004232	046440	047101	043501	
1500	004240	046505	047105	020124	
1501	004246	042522	044507	052123	
1502	004254	051105	030040	053440	
1503	004262	046111	020114	047516	
1504	004270	020124	046103	040505	
1505	004276	000122			
1506	004300	040503	023516	020124	EM10: .ASCIZ ?CAN'T SET 171000 IN MMR0?
1507	004306	042523	020124	033461	
1508	004314	030061	030060	044440	
1509	004322	020116	046515	030122	
1510	004330	000			
1511	004331	107	052117	052040	EM11: .ASCIZ ?GOT THE WRONG DATA BACK FROM MMR0?
1512	004336	042510	053440	047522	
1513	004344	043516	042040	052101	
1514	004352	020101	040502	045503	
1515	004360	043040	047522	020115	
1516	004366	046515	030122	000	
1517	004373	115	046505	051117	EM12: .ASCIZ ?MEMORY MANAGEMENT REGISTER 1 WILL NOT CLEAR?
1518	004400	020131	040515	040516	
1519	004406	042507	042515	052116	
1520	004414	051040	043505	051511	
1521	004422	042524	020122	020061	
1522	004430	044527	046114	047040	
1523	004436	052117	041440	042514	

1524	004444	051101	000		
1525	004447	115	051115	020061	EM13: .ASCIZ ?MMR1 DID NOT TRACK PROPERLY?
1526	004454	044504	020104	047516	
1527	004462	020124	051124	041501	
1528	004470	020113	051120	050117	
1529	004476	051105	054514	000	
1530	004503	115	051115	020062	EM14: .ASCIZ ?MMR2 DID NOT TRACK PROPERLY?
1531	004510	044504	020104	047516	
1532	004516	020124	051124	041501	
1533	004524	020113	051120	050117	
1534	004532	051105	054514	000	
1535	004537	115	046505	051117	EM15: .ASCIZ ?MEMORY MANAGEMENT REGISTER 3 WILL NOT CLEAR?
1536	004544	020131	040515	040516	
1537	004552	042507	042515	052116	
1538	004560	051040	043505	051511	
1539	004566	042524	020122	020063	
1540	004574	044527	046114	047040	
1541	004602	052117	041440	042514	
1542	004610	051101	000		
1543	004613	115	051115	020063	EM16: .ASCIZ ?MMR3 IS HOLDING WRONG DATA?
1544	004620	051511	044040	046117	
1545	004626	044504	043516	053440	
1546	004634	047522	043516	042040	
1547	004642	052101	000101		
1548	004646	052523	046515	051101	EM17: .ASCIZ ?SUMMARY OF PAR/PDR REFERENCE TIMEOUTS?
1549	004654	020131	043117	050040	
1550	004662	051101	050057	051104	
1551	004670	051040	043105	051105	
1552	004676	047105	042503	052040	
1553	004704	046511	047505	052125	
1554	004712	000123			
1555	004714	047506	046114	053517	EM20: .ASCIZ ?FOLLOWING PAR/PDR WILL NOT CLEAR?
1556	004722	047111	020107	040520	
1557	004730	027522	042120	020122	
1558	004736	044527	046114	047040	
1559	004744	052117	041440	042514	
1560	004752	051101	000		
1561	004755	123	046525	040515	EM21: .ASCIZ ?SUMMARY OF DUAL ADDRESSING ERRORS?
1562	004762	054522	047440	020106	
1563	004770	052504	046101	040440	
1564	004776	042104	042522	051523	
1565	005004	047111	020107	051105	
1566	005012	047522	051522	000	
1567	005017	123	046525	040515	EM22: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES?
1568	005024	054522	047440	020106	
1569	005032	047503	047125	020124	
1570	005040	040520	052124	051105	
1571	005046	020116	040506	046111	
1572	005054	051125	051505	000	
1573	005061	105	051122	051117	EM23: .ASCIZ ?ERRGR IN BYTE ADDRESSING OF PAR/PDR?
1574	005066	044440	020116	054502	
1575	005074	042524	040440	042104	
1576	005102	042522	051523	047111	
1577	005110	020107	043117	050040	
1578	005116	051101	050057	051104	
1579	005124	000			

1580	005125	117	042516	047440	EM24:	.ASCIZ ?ONE OF THE REGISTERS TIMED OUT?
1581	005132	020106	044124	020105		
1582	005140	042522	044507	052123		
1583	005146	051105	020123	044524		
1584	005154	042515	020104	052517		
1585	005162	000124				
1586	005164	047514	020127	054502	EM25:	.ASCIZ ?LOW BYTE OF LOW SIZE REGISTER IS NOT ALL ONES?
1587	005172	042524	047440	020106		
1588	005200	047514	020127	044523		
1589	005206	042532	051040	043505		
1590	005214	051511	042524	020122		
1591	005222	051511	047040	052117		
1592	005230	040440	046114	047440		
1593	005236	042516	000123			
1594	005242	047503	046125	020104	EM26:	.ASCIZ ?COULD WRITE ONE OF THE SIZE REGISTERS?
1595	005250	051127	052111	020105		
1596	005256	047117	020105	043117		
1597	005264	052040	042510	051440		
1598	005272	055111	020105	042522		
1599	005300	044507	052123	051105		
1600	005306	000123				
1601	005310	044510	044107	051440	EM27:	.ASCIZ ?HIGH SIZE REGISTER IS NOT ZERO?
1602	005316	055111	020105	042522		
1603	005324	044507	052123	051105		
1604	005332	044440	020123	047516		
1605	005340	020124	047532	047522		
1606	005346	000				
1607	005347	103	052520	042440	EM30:	.ASCIZ ?CPU ERROR REGISTER NOT ZERO AFTER LOADING NEGATIVE ONE?
1608	005354	051122	051117	051040		
1609	005362	043505	051511	042524		
1610	005370	020122	047516	020124		
1611	005376	042532	047522	040440		
1612	005404	052106	051105	046040		
1613	005412	040517	044504	043516		
1614	005420	047040	043505	052101		
1615	005426	053111	020105	047117		
1616	005434	000105				
1617	005436	047514	042527	020122	EM31:	.ASCIZ ?LOWER BYTE OF P.S.W. NOT CORRECT?
1618	005444	054502	042524	047440		
1619	005452	020106	027120	027123		
1620	005460	027127	047040	052117		
1621	005466	041440	051117	042522		
1622	005474	052103	000			
1623	005477	115	041511	030122	EM32:	.ASCIZ ?MICRO BREAK REG LOADED INCORRECTLY. ONLY LOWER BYTE SHOULD BE WRITABLE?
1624	005504	041040	042522	045501		
1625	005512	051040	043505	046040		
1626	005520	040517	042504	020104		
1627	005526	047111	047503	051122		
1628	005534	041505	046124	027131		
1629	005542	047440	046116	020131		
1630	005550	047514	042527	020122		
1631	005556	054502	042524	051440		
1632	005564	047510	046125	020104		
1633	005572	042502	053440	044522		
1634	005600	040524	046102	000105		
1635	005606	044515	051103	020060	EM32M:	.ASCIZ ?MICRO BREAK REG LOADED INCORRECTLY. ALL 16 BITS SHOULD BE WRITABLE?

1636	005614	051102	040505	020113	
1637	005622	042522	020107	047514	
1638	005630	042101	042105	044440	
1639	005636	041516	051117	042522	
1640	005644	052103	054514	020056	
1641	005652	046101	020114	033061	
1642	005660	041040	052111	020123	
1643	005666	044123	052517	042114	
1644	005674	041040	020105	051127	
1645	005702	052111	041101	042514	
1646	005710	000			
1647	005711	104	042111	023516	EM33: .ASCIZ ?DIDN'T LOAD PROGRAM INTERRUPT REQUEST REGISTER?
1648	005716	020124	047514	042101	
1649	005724	050040	047522	051107	
1650	005732	046501	044440	052116	
1651	005740	051105	052522	052120	
1652	005746	051040	050505	042525	
1653	005754	052123	051040	043505	
1654	005762	051511	042524	000122	
1655	005770	047514	042101	042105	EM34: .ASCIZ ?LOADED MORE THAN UPPER BYTE OF STACK LIMIT REGISTER?
1656	005776	046440	051117	020105	
1657	006004	044124	047101	052440	
1658	006012	050120	051105	041040	
1659	006020	052131	020105	043117	
1660	006026	051440	040524	045503	
1661	006034	046040	046511	052111	
1662	006042	051040	043505	051511	
1663	006050	042524	000122		
1664	006054	042513	047122	046105	EM35: .ASCIZ ?KERNEL STACK POINTER NOT 1100 AFTER LOADING SSP & USP?
1665	006062	051440	040524	045503	
1666	006070	050040	044517	052116	
1667	006076	051105	047040	052117	
1668	006104	030440	030061	020060	
1669	006112	043101	042524	020122	
1670	006120	047514	042101	047111	
1671	006126	020107	051523	020120	
1672	006134	020046	051525	000120	
1673	006142	052504	046101	040440	EM36: .ASCIZ ?DUAL ADDRESSING BETWEEN PAR/PDR GROUPS?
1674	006150	042104	042522	051523	
1675	006156	047111	020107	042502	
1676	006164	053524	042505	020116	
1677	006172	040520	027522	042120	
1678	006200	020122	051107	052517	
1679	006206	051520	000		
1680	006211	102	042101	051040	EM37: .ASCIZ ?BAD RELOCATION, ON STORING DATA 18-BIT MAPPING?
1681	006216	046105	041517	052101	
1682	006224	047511	026116	047440	
1683	006232	020116	052123	051117	
1684	006240	047111	020107	040504	
1685	006246	040524	030440	026470	
1686	006254	044502	020124	040515	
1687	006262	050120	047111	000107	
1688	006270	034061	041055	052111	EM40: .ASCIZ ?18-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
1689	006276	046440	050101	044520	
1690	006304	043516	050040	051517	
1691	006312	044523	046102	020105	

1692	006320	047510	042514	044440	
1693	006326	020116	040515	047111	
1694	006334	046440	046505	051117	
1695	006342	020131	051106	046517	
1696	006350	000			
1697	006351	061	026470	044502	EM41: .ASCIZ ?18-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY?
1698	006356	020124	040515	050120	
1699	006364	047111	020107	047520	
1700	006372	051523	041111	042514	
1701	006400	044040	046117	020105	
1702	006406	052101	052040	042510	
1703	006414	052040	050117	047440	
1704	006422	020106	042515	047515	
1705	006430	054522	000		
1706	006433	106	052501	052114	EM42: .ASCIZ ?FAULTY CARRY PROPAGATION 18-BIT MAPPING.?
1707	006440	020131	040503	051122	
1708	006446	020131	051120	050117	
1709	006454	043501	052101	047511	
1710	006462	020116	034061	041055	
1711	006470	052111	046440	050101	
1712	006476	044520	043516	000056	
1713	006504	047516	052040	040522	EM43: .ASCIZ ?NO TRAP THRU ERRVEC, AT 18-BIT ADDRESS 760000?
1714	006512	020120	044124	052522	
1715	006520	042440	051122	042526	
1716	006526	026103	040440	020124	
1717	006534	034061	041055	052111	
1718	006542	040440	042104	042522	
1719	006550	051523	033440	030066	
1720	006556	030060	000060		
1721	006562	044504	047104	052047	EM44: .ASCIZ ?DIDN'T GET WRAP AROUND TO ADDRESS ZERO?
1722	006570	043440	052105	053440	
1723	006576	040522	020120	051101	
1724	006604	052517	042116	052040	
1725	006612	020117	042101	051104	
1726	006620	051505	020123	042532	
1727	006626	047522	000		
1728	006631	116	020117	051124	EM45: .ASCIZ ?NO TRAP THRU ERRVEC, ON NON-EXISTANT ADDRESS?
1729	006636	050101	052040	051110	
1730	006644	020125	051105	053122	
1731	006652	041505	020054	047117	
1732	006660	047040	047117	042455	
1733	006666	044530	052123	047101	
1734	006674	020124	042101	051104	
1735	006702	051505	000123		
1736	006706	040502	020104	042522	EM46: .ASCIZ ?BAD RELOCATION, CARRY PROPAGATION 22-BIT MAPPING?
1737	006714	047514	040503	044524	
1738	006722	047117	020054	040503	
1739	006730	051122	020131	051120	
1740	006736	050117	043501	052101	
1741	006744	047511	020116	031062	
1742	006752	041055	052111	046440	
1743	006760	050101	044520	043516	
1744	006766	000			
1745	006767	104	042111	047040	EM47: .ASCIZ ?DID NOT GET UNIBUS ADDRESS?
1746	006774	052117	043440	052105	
1747	007002	052440	044516	052502	

1748	007010	020123	042101	051104	
1749	007016	051505	000123		
1750	007022	047503	050115	051101	EM50: .ASCIZ ?COMPARE CIRCUIT FOR TOP OF MEMORY BAD?
1751	007030	020105	044503	041522	
1752	007036	044525	020124	047506	
1753	007044	020122	047524	020120	
1754	007052	043117	046440	046505	
1755	007060	051117	020131	040502	
1756	007066	000104			
1757	007070	040520	042507	046040	EM51: .ASCIZ ?PAGE LENGTH ABORT AT WRONG VIRTUAL ADDRESS?
1758	007076	047105	052107	020110	
1759	007104	041101	051117	020124	
1760	007112	052101	053440	047522	
1761	007120	043516	053040	051111	
1762	007126	052524	046101	040440	
1763	007134	042104	042522	051523	
1764	007142	000			
1765	007143	116	020117	040520	EM52: .ASCIZ ?NO PAGE LENGTH ABORT, WHEN CONDITION CORRECT?
1766	007150	042507	046040	047105	
1767	007156	052107	020110	041101	
1768	007164	051117	026124	053440	
1769	007172	042510	020116	047503	
1770	007200	042116	052111	047511	
1771	007206	020116	047503	051122	
1772	007214	041505	000124		
1773	007220	044504	020104	047516	EM53: .ASCIZ ?DID NOT ABORT ON NON-RESIDENT PAGE KIPDR5?
1774	007226	020124	041101	051117	
1775	007234	020124	047117	047040	
1776	007242	047117	051055	051505	
1777	007250	042111	047105	020124	
1778	007256	040520	042507	045440	
1779	007264	050111	051104	000065	
1780	007272	044504	020104	047516	EM54: .ASCIZ ?DID NOT ABORT ON READ ONLY PAGE KIPDR4?
1781	007300	020124	041101	051117	
1782	007306	020124	047117	051040	
1783	007314	040505	020104	047117	
1784	007322	054514	050040	043501	
1785	007330	020105	044513	042120	
1786	007336	032122	000		
1787	007341	111	041516	051117	EM55: .ASCIZ ?INCORRECT READ FROM PAGE KIPDR4 BIT09 (MMRO) CLEAR?
1788	007346	042522	052103	051040	
1789	007354	040505	020104	051106	
1790	007362	046517	050040	043501	
1791	007370	020105	044513	042120	
1792	007376	032122	041040	052111	
1793	007404	034460	024040	046515	
1794	007412	030122	020051	046103	
1795	007420	040505	000122		
1796	007424	047516	046440	046456	EM56: .ASCIZ ?NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR4?
1797	007432	020056	051124	050101	
1798	007440	053440	042510	020116	
1799	007446	044502	030124	020071	
1800	007454	046450	051115	024460	
1801	007462	051440	052105	050040	
1802	007470	043501	020105	044513	
1803	007476	042120	032122	000	

1804	007503	111	041516	051117	EM57:	.ASCIZ ?INCORRECT READ FROM PAGE KIPDR4, BIT09 (MMRO) SET?
1805	007510	042522	052103	051040		
1806	007516	040505	020104	051106		
1807	007524	046517	050040	043501		
1808	007532	020105	044513	042120		
1809	007540	032122	020054	044502		
1810	007546	030124	020071	046450		
1811	007554	051115	024460	051440		
1812	007562	052105	000			
1813	007565	111	041516	051117	EM60:	.ASCIZ ?INCORRECT WRITE TO PAGE KIPDR4, BIT09 (MMRO) SET?
1814	007572	042522	052103	053440		
1815	007600	044522	042524	052040		
1816	007606	020117	040520	042507		
1817	007614	045440	050111	051104		
1818	007622	026064	041040	052111		
1819	007630	034460	024040	046515		
1820	007636	030122	020051	042523		
1821	007644	000124				
1822	007646	047516	046440	046456	EM61:	.ASCIZ ?NO M.M. TRAP WHEN BIT09 (MMRO) SET PAGE KIPDR7?
1823	007654	020056	051124	050101		
1824	007662	053440	042510	020116		
1825	007670	044502	030124	020071		
1826	007676	046450	051115	024460		
1827	007704	051440	052105	050040		
1828	007712	043501	020105	044513		
1829	007720	042120	033522	000		
1830	007725	111	041516	051117	EM62:	.ASCIZ ?INCORRECT READ FROM PAGE KIPDR7, BIT09 (MMRO) SET?
1831	007732	042522	052103	051040		
1832	007740	040505	020104	051106		
1833	007746	046517	050040	043501		
1834	007754	020105	044513	042120		
1835	007762	033522	020054	044502		
1836	007770	030124	020071	046450		
1837	007776	051115	024460	051440		
1838	010004	052105	000			
1839	010007	124	040522	020120	EM63:	.ASCIZ ?TRAP WHEN NO RELOCATION?
1840	010014	044127	047105	047040		
1841	010022	020117	042522	047514		
1842	010030	040503	044524	047117		
1843	010036	000				
1844	010037	124	047517	020113	EM64:	.ASCII ?TOOK TWO VECTORS WITH TRAP AND ABORT ON SAME INSTRUCTION?<CRLF>
1845	010044	053524	020117	042526		
1846	010052	052103	051117	020123		
1847	010060	044527	044124	052040		
1848	010066	040522	020120	047101		
1849	010074	020104	041101	051117		
1850	010102	020124	047117	051440		
1851	010110	046501	020105	047111		
1852	010116	052123	052522	052103		
1853	010124	047511	100116			
1854	010130	054105	042520	052103		.ASCIZ ?EXPECTING '1074' FOR KERNEL STACK POINTER?
1855	010136	047111	020107	030442		
1856	010144	033460	021064	043040		
1857	010152	051117	045440	051105		
1858	010160	042516	020114	052123		
1859	010166	041501	020113	047520		

1860	010174	047111	042524	000122		
1861	010202	042515	047515	054522	EM65:	.ASCIZ ?MEMORY MANAGEMENT ABORT CONDITION WRONG?
1862	010210	046440	047101	043501		
1863	010216	046505	047105	020124		
1864	010224	041101	051117	020124		
1865	010232	047503	042116	052111		
1866	010240	047511	020116	051127		
1867	010246	047117	000107			
1868	010252	044502	020124	031061	EM66:	.ASCIZ ?BIT 12 OF MMRO WAS NOT SET WHEN A.C.F. SATISFIED?
1869	010260	047440	020106	046515		
1870	010266	030122	053440	051501		
1871	010274	047040	052117	051440		
1872	010302	052105	053440	042510		
1873	010310	020116	027101	027103		
1874	010316	027106	051440	052101		
1875	010324	051511	044506	042105		
1876	010332	000				
1877	010333	116	020117	051124	EM67:	.ASCIZ ?NO TRAP WHEN INSTRUCTION CLEARING BIT09 (MMRO) CAUSES TRAP COND.?
1878	010340	050101	053440	042510		
1879	010346	020116	047111	052123		
1880	010354	052522	052103	047511		
1881	010362	020116	046103	040505		
1882	010370	044522	043516	041040		
1883	010376	052111	034460	024040		
1884	010404	046515	030122	020051		
1885	010412	040503	051525	051505		
1886	010420	052040	040522	020120		
1887	010426	047503	042116	000056		
1888	010434	051105	047522	020122	EM70:	.ASCII ?ERROR DURING M.M. ABORT IN TRAP SEQUENCE?<CRLF>
1889	010442	052504	044522	043516		
1890	010450	046440	046456	020056		
1891	010456	041101	051117	020124		
1892	010464	047111	052040	040522		
1893	010472	020120	042523	052521		
1894	010500	047105	042503	200		
1895	010505	105	050130	041505		.ASCIZ ?EXPECTED:?
1896	010512	042524	035104	000		
1897	010517	111	051516	051124	EM71:	.ASCIZ ?INSTRUCTION FETCH DID NOT ABORT IN ILLEGAL MODE (10)?
1898	010524	041525	044524	047117		
1899	010532	043040	052105	044103		
1900	010540	042040	042111	047040		
1901	010546	052117	040440	047502		
1902	010554	052122	044440	020116		
1903	010562	046111	042514	040507		
1904	010570	020114	047515	042504		
1905	010576	024040	030061	000051		
1906	010604	051105	047522	020122	EM72:	.ASCIZ ?ERROR CONDITION NOT CORRECT IN ILLEGAL MODE (10)?
1907	010612	047503	042116	052111		
1908	010620	047511	020116	047516		
1909	010626	020124	047503	051122		
1910	010634	041505	020124	047111		
1911	010642	044440	046114	043505		
1912	010650	046101	046440	042117		
1913	010656	020105	030450	024460		
1914	010664	000				
1915	010665	101	020124	042514	EM73:	.ASCIZ ?AT LEAST ONE M.M. STATUS REGISTERS WAS CLOCKED AFTER BEING LOCKED?

1916	010672	051501	020124	047117	
1917	010700	020105	027115	027115	
1918	010706	051440	040524	052524	
1919	010714	020123	042522	044507	
1920	010722	052123	051105	020123	
1921	010730	040527	020123	046103	
1922	010736	041517	042513	020104	
1923	010744	043101	042524	020122	
1924	010752	042502	047111	020107	
1925	010760	047514	045503	042105	
1926	010766	000			
1927	010767	104	042111	047040	EM74: .ASCIZ ?DID NOT CHANGE MAPPING TO SUPERVISOR MODE?
1928	010774	052117	041440	040510	
1929	011002	043516	020105	040515	
1930	011010	050120	047111	020107	
1931	011016	047524	051440	050125	
1932	011024	051105	044526	047523	
1933	011032	020122	047515	042504	
1934	011040	000			
1935	011041	101	047502	052122	EM75: .ASCIZ ?ABORT CONDITION INCORRECT EXPECTING 100051?
1936	011046	041440	047117	044504	
1937	011054	044524	047117	044440	
1938	011062	041516	051117	042522	
1939	011070	052103	042440	050130	
1940	011076	041505	044524	043516	
1941	011104	030440	030060	032460	
1942	011112	000061			
1943	011114	044504	020104	047516	EM76: .ASCIZ ?DID NOT CHANGE MAPPING TO USER MODE?
1944	011122	020124	044103	047101	
1945	011130	042507	046440	050101	
1946	011136	044520	043516	052040	
1947	011144	020117	051525	051105	
1948	011152	046440	042117	000105	
1949	011160	041101	051117	020124	EM77: .ASCIZ ?ABORT CONDITION INCORRECT EXPECTING 100153?
1950	011166	047503	042116	052111	
1951	011174	047511	020116	047111	
1952	011202	047503	051122	041505	
1953	011210	020124	054105	042520	
1954	011216	052103	047111	020107	
1955	011224	030061	030460	031465	
1956	011232	000			
1957	011233	115	051115	020062	EM100: .ASCIZ ?MMR2 LOCKED UP THE WRONG VIRTUAL ADDRESS?
1958	011240	047514	045503	042105	
1959	011246	052440	020120	044124	
1960	011254	020105	051127	047117	
1961	011262	020107	044526	052122	
1962	011270	040525	020114	042101	
1963	011276	051104	051505	000123	
1964	011304	046515	030122	046040	EM101: .ASCIZ ?MMR0 LOCKED UP THE WRONG PAGE NUMBER?
1965	011312	041517	042513	020104	
1966	011320	050125	052040	042510	
1967	011326	053440	047522	043516	
1968	011334	050040	043501	020105	
1969	011342	052516	041115	051105	
1970	011350	000			
1971	011351	127	041055	052111	EM102: .ASCIZ ?W-BIT NOT SET ON WRITE TO PAGE 4?

1972	011356	047040	052117	051440	
1973	011364	052105	047440	020116	
1974	011372	051127	052111	020105	
1975	011400	047524	050040	043501	
1976	011406	020105	000064		
1977	011412	026527	044502	020124	EM103: .ASCIZ ?W-BIT DID NOT REMAIN SET ON M.M. ABOR* AT PAGE 4?
1978	011420	044504	020104	047516	
1979	011426	020124	042522	040515	
1980	011434	047111	051440	052105	
1981	011442	047440	020116	027115	
1982	011450	027115	040440	047502	
1983	011456	052122	040440	020124	
1984	011464	040520	042507	032040	
1985	011472	000			
1986	011473	127	041055	052111	EM104: .ASCIZ ?W-BIT DID NOT CLEAR ON WRITE TO KIPDR4?
1987	011500	042040	042111	047040	
1988	011506	052117	041440	042514	
1989	011514	051101	047440	020116	
1990	011522	051127	052111	020105	
1991	011530	047524	045440	050111	
1992	011536	051104	000064		
1993	011542	026501	044502	020124	EM105: .ASCIZ ?A-BIT DID NOT SET ON READ TO PAGE 4 A.C.F.=4?
1994	011550	044504	020104	047516	
1995	011556	020124	042523	020124	
1996	011564	047117	051040	040505	
1997	011572	020104	047524	050040	
1998	011600	043501	020105	020064	
1999	011606	027101	027103	027106	
2000	011614	032075	000		
2001	011617	101	041055	052111	EM106: .ASCIZ ?A-BIT DID NOT REMAIN SET ON M.M. ABORT AT PAGE 4?
2002	011624	042040	042111	047040	
2003	011632	052117	051040	046505	
2004	011640	044501	020116	042523	
2005	011646	020124	047117	046440	
2006	011654	046456	020056	041101	
2007	011662	051117	020124	052101	
2008	011670	050040	043501	020105	
2009	011676	000064			
2010	011700	026501	044502	020124	EM107: .ASCIZ ?A-BIT DID NOT CLEAR ON WRITE TO KIPAR4?
2011	011706	044504	020104	047516	
2012	011714	020124	046103	040505	
2013	011722	020122	047117	053440	
2014	011730	044522	042524	052040	
2015	011736	020117	044513	040520	
2016	011744	032122	000		
2017	011747	127	041055	052111	EM110: .ASCIZ ?W-BIT DID NOT SET ON WRITE TO I/O PAGE?
2018	011754	042040	042111	047040	
2019	011762	052117	051440	052105	
2020	011770	047440	020116	051127	
2021	011776	052111	020105	047524	
2022	012004	044440	047457	050040	
2023	012012	043501	000105		
2024	012016	026527	044502	020124	EM111: .ASCIZ ?W-BIT DID NOT REMAIN SET AFTER INTERNAL REGISTER WRITE?
2025	012024	044504	020104	047516	
2026	012032	020124	042522	040515	
2027	012040	047111	051440	052105	

2028	012046	040440	052106	051105	
2029	012054	044440	052116	051105	
2030	012062	040516	020114	042522	
2031	012070	044507	052123	051105	
2032	012076	053440	044522	042524	
2033	012104	000			
2034	012105	104	040525	020114	EM112: .ASCIZ ?DUAL MAPPING BETWEEN PAGES?
2035	012112	040515	050120	047111	
2036	012120	020107	042502	053524	
2037	012126	042505	020116	040520	
2038	012134	042507	000123		
2039	012140	047516	050040	043501	EM113: .ASCIZ ?NO PAGE HAD BOTH ITS A & W BITS SET?
2040	012146	020105	040510	020104	
2041	012154	047502	044124	044440	
2042	012162	051524	040440	023040	
2043	012170	053440	041040	052111	
2044	012176	020123	042523	000124	
2045	012204	044504	020104	047516	EM114: .ASCIZ ?DID NOT PICK UP CORRECT STACK POINTER?
2046	012212	020124	044520	045503	
2047	012220	052440	020120	047503	
2048	012226	051122	041505	020124	
2049	012234	052123	041501	020113	
2050	012242	047520	047111	042524	
2051	012250	000122			
2052	012252	052503	051122	047105	EM115: .ASCIZ ?CURRENT MODE STACK NOT PUSHED IN MFP INSTRUCTION?
2053	012260	020124	047515	042504	
2054	012266	051440	040524	045503	
2055	012274	047040	052117	050040	
2056	012302	051525	042510	020104	
2057	012310	047111	046440	050106	
2058	012316	044440	051516	051124	
2059	012324	041525	044524	047117	
2060	012332	000			
2061	012333	127	047522	043516	EM116: .ASCIZ ?WRONG DATA FETCHED BY MFP INSTRUCTION?
2062	012340	042040	052101	020101	
2063	012346	042506	041524	042510	
2064	012354	020104	054502	046440	
2065	012362	050106	044440	051516	
2066	012370	051124	041525	044524	
2067	012376	047117	000		
2068	012401	124	044522	042105	EM117: .ASCIZ ?TRIED TO REFERENCE NON-RESIDENT PAGE?
2069	012406	052040	020117	042522	
2070	012414	042506	042522	041516	
2071	012422	020105	047516	026516	
2072	012430	042522	044523	042504	
2073	012436	052116	050040	043501	
2074	012444	000105			
2075	012446	052123	041501	020113	EM120: .ASCIZ ?STACK POINTER NOT CHANGED BY MTP INSTRUCTION?
2076	012454	047520	047111	042524	
2077	012462	020122	047516	020124	
2078	012470	044103	047101	042507	
2079	012476	020104	054502	046440	
2080	012504	050124	044440	051516	
2081	012512	051124	041525	044524	
2082	012520	047117	000		
2083	012523	111	041516	051117	EM121: .ASCIZ ?INCORRECT STORE BY MTP INSTRUCTION?

2084	012530	042522	052103	051440	
2085	012536	047524	042522	041040	
2086	012544	020131	052115	020120	
2087	012552	047111	052123	052522	
2088	012560	052103	047511	000116	
2089	012566	041101	051117	042524	EM122: .ASCIZ ?ABORTED THRU RIGHT VECTOR BUT PICKED UP WRONG PSW?
2090	012574	020104	044124	052522	
2091	012602	051040	043511	052110	
2092	012610	053040	041505	047524	
2093	012616	020122	052502	020124	
2094	012624	044520	045503	042105	
2095	012632	052440	020120	051127	
2096	012640	047117	020107	051520	
2097	012646	000127			
2098	012650	041101	051117	042524	EM123: .ASCIZ ?ABORTED THRU SUPERVISOR SPACE, SUPERVISOR PSW IS XXX340?
2099	012656	020104	044124	052522	
2100	012664	051440	050125	051105	
2101	012672	044526	047523	020122	
2102	012700	050123	041501	026105	
2103	012706	051440	050125	051105	
2104	012714	044526	047523	020122	
2105	012722	051520	020127	051511	
2106	012730	054040	054130	032063	
2107	012736	000060			
2108	012740	041101	051117	042524	EM124: .ASCIZ ?ABORTED THRU USER SPACE, USER PSW IS XXX000?
2109	012746	020104	044124	052522	
2110	012754	052440	042523	020122	
2111	012762	050123	041501	026105	
2112	012770	052440	042523	020122	
2113	012776	051520	020127	051511	
2114	013004	054040	054130	030060	
2115	013012	000060			
2116	013014	031062	041055	052111	EM125: .ASCIZ ?22-BIT MAPPING POSSIBLE HOLE IN MAIN MEMORY FROM?
2117	013022	046440	050101	044520	
2118	013030	043516	050040	051517	
2119	013036	044523	046102	020105	
2120	013044	047510	042514	044440	
2121	013052	020116	040515	047111	
2122	013060	046440	046505	051117	
2123	013066	020131	051106	046517	
2124	013074	000			
2125	013075	062	026462	044502	EM126: .ASCIZ ?22-BIT MAPPING POSSIBLE HOLE AT THE TOP OF MEMORY?
2126	013102	020124	040515	050120	
2127	013110	047111	020107	047520	
2128	013116	051523	041111	042514	
2129	013124	044040	046117	020105	
2130	013132	052101	052040	042510	
2131	013140	052040	050117	047440	
2132	013146	020106	042515	047515	
2133	013154	054522	000		
2134	013157	104	042111	047040	EM127: .ASCIZ ?DID NOT ABORT REFERENCE TO A MEMORY MANAGEMENT REGISTER?
2135	013164	052117	040440	047502	
2136	013172	052122	051040	043105	
2137	013200	051105	047105	042503	
2138	013206	052040	020117	020101	
2139	013214	042515	047515	054522	

2140	013222	046440	046501	043501	
2141	013230	046505	047105	020124	
2142	013236	042522	044507	052123	
2143	013244	051105	000		
2144	013247	101	047502	052122	EM130: .ASCIZ ?ABORT IN KERNAL D-SPACE PICKED UP VECTOR FROM I-SPACE?
2145	013254	044440	020116	042513	
2146	013262	047122	046101	042040	
2147	013270	051455	040520	042503	
2148	013276	050040	041511	042513	
2149	013304	020104	050125	053040	
2150	013312	041505	047524	020122	
2151	013320	051106	046517	044440	
2152	013326	051455	040520	042503	
2153	013334	000			
2154					
2155	013335	115	046456	020056	EM131: .ASCIZ ?M.M. ABORT IN KERNEL D-SPACE HAD WRONG CONDITION?
2156	013342	041101	051117	020124	
2157	013350	047111	045440	051105	
2158	013356	042516	020114	026504	
2159	013364	050123	041501	020105	
2160	013372	040510	020104	051127	
2161	013400	047117	020107	047503	
2162	013406	042116	052111	047511	
2163	013414	000116			
2164	013416	026504	050123	041501	EM132: .ASCIZ ?D-SPACE ENABLE CIRCUITRY HAS FAILED?
2165	013424	020105	047105	041101	
2166	013432	042514	041440	051111	
2167	013440	052503	052111	054522	
2168	013446	044040	051501	043040	
2169	013454	044501	042514	000104	
2170	013462	054502	020120	044502	EM133: .ASCIZ ?BYP BIT IN KIPDR COULD NOT BE CLEARED?
2171	013470	020124	047111	045440	
2172	013476	050111	051104	041440	
2173	013504	052517	042114	047040	
2174	013512	052117	041040	020105	
2175	013520	046103	040505	042522	
2176	013526	000104			
2177	013530	054502	020120	044502	EM134: .ASCIZ ?BYP BIT IN KIPDR COULD NOT BE SET?
2178	013536	020124	047111	045440	
2179	013544	050111	051104	041440	
2180	013552	052517	042114	047040	
2181	013560	052117	041040	020105	
2182	013566	042523	000124		
2183	013572	042524	052123	042055	EM135: .ASCIZ /TEST-DATA COULD NOT BE MADE HIT/
2184	013600	052101	020101	047503	
2185	013606	046125	020104	047516	
2186	013614	020124	042502	046440	
2187	013622	042101	020105	044510	
2188	013630	000124			
2189	013632	042524	052123	042055	EM136: .ASCII /TEST-DATA REFERENCE NOT A MISS/
2190	013640	052101	020101	042522	
2191	013646	042506	042522	041516	
2192	013654	020105	047516	020124	
2193	013662	020101	044515	051523	
2194	013670	005015	040503	044103	.ASCIZ <15><12>/CACHED DATA WAS NOT FORCED A MISS ON VIRTUAL PAGE BYPASS/
2195	013676	042105	042040	052101	

2196	013704	020101	040527	020123	
2197	013712	047516	020124	047506	
2198	013720	041522	042105	040440	
2199	013726	046440	051511	020123	
2200	013734	047117	053040	051111	
2201	013742	052524	046101	050040	
2202	013750	043501	020105	054502	
2203	013756	040520	051523	000	
2204	013763	124	051505	020124	EM137: .ASCII /TEST DATA REFERENCE NOT A MISS/
2205	013770	040504	040524	051040	
2206	013776	043105	051105	047105	
2207	014004	042503	047040	052117	
2208	014012	040440	046440	051511	
2209	014020	123			
2210	014021	015	041412	041501	.ASCIIZ <15><12>/CACHED DATA WAS NOT INVALIDATED ON VIRTUAL PAGE BYPASS/
2211	014026	042510	020104	040504	
2212	014034	040524	053440	051501	
2213	014042	047040	052117	044440	
2214	014050	053116	046101	042111	
2215	014056	052101	042105	047440	
2216	014064	020116	044526	052122	
2217	014072	040525	020114	040520	
2218	014100	042507	041040	050131	
2219	014106	051501	000123		
2220	014112	054502	020120	044502	EM140: .ASCIIZ ?BYP BIT IN SIPDR COULD NOT BE CLEARED?
2221	014120	020124	047111	051440	
2222	014126	050111	051104	041440	
2223	014134	052517	042114	047040	
2224	014142	052117	041040	020105	
2225	014150	046103	040505	042522	
2226	014156	000104			
2227	014160	054502	020120	044502	EM141: .ASCIIZ ?BYP BIT IN SIPDR COULD NOT BE SET?
2228	014166	020124	047111	051440	
2229	014174	050111	051104	041440	
2230	014202	052517	042114	047040	
2231	014210	052117	041040	020105	
2232	014216	042523	000124		
2233	014222	054502	020120	044502	EM142: .ASCIIZ ?BYP BIT IN UIPDR COULD NOT BE CLEARED?
2234	014230	020124	047111	052440	
2235	014236	050111	051104	041440	
2236	014244	052517	042114	047040	
2237	014252	052117	041040	020105	
2238	014260	046103	040505	042522	
2239	014266	000104			
2240	014270	054502	020120	044502	EM143: .ASCIIZ ?BYP BIT IN UIPDR COULD NOT BE SET?
2241	014276	020124	047111	052440	
2242	014304	050111	051104	041440	
2243	014312	052517	042114	047040	
2244	014320	052117	041040	020105	
2245	014326	042523	000124		
2246	014332	050104	051101	051040	EM145: .ASCIIZ /DPAR READ BACK INCORRECTLY/
2247	014340	040505	020104	040502	
2248	014346	045503	044440	041516	
2249	014354	051117	042522	052103	
2250	014362	054514	000		
2251	014365	012	042015	040520	ECO: .ASCIIZ <12><15>/DPAR READ BACK PROBLEM CORRECTED BY ECO NO. M8140-00002/<12><15

2252	014372	020122	042522	042101	
2253	014400	041040	041501	020113	
2254	014406	051120	041117	042514	
2255	014414	020115	047503	051122	
2256	014422	041505	042524	020104	
2257	014430	054502	042440	047503	
2258	014436	047040	027117	046440	
2259	014444	030470	030064	030055	
2260	014452	030060	031060	006412	
2261	014460	000			
2262	014461	015	051412	051123	EM146: .ASCII <CR><LF>/SSRC KT ABORT FLG DOESN'T GET TO TMCC E34(10) OR/<CR><LF>
2263	014466	020103	052113	040440	
2264	014474	047502	052122	043040	
2265	014502	043514	042040	042517	
2266	014510	047123	052047	043440	
2267	014516	052105	052040	020117	
2268	014524	046524	041503	042440	
2269	014532	032063	030450	024460	
2270	014540	047440	006522	012	
2271	014545	105	032063	030450	.ASCIIZ /E34(10) BAD OR TMCC AERF(1) DOESN'T GO HIGH ON A KT ABORT/
2272	014552	024460	041040	042101	
2273	014560	047440	020122	046524	
2274	014566	041503	040440	051105	
2275	014574	024106	024461	042040	
2276	014602	042517	047123	052047	
2277	014610	043440	020117	044510	
2278	014616	044107	047440	020116	
2279	014624	020101	052113	040440	
2280	014632	047502	052122	000	
2281	014637	015	050012	053523	EM147: .ASCIIZ<CR><LF>/PSW BIT 8 FAILED TO SET/
2282	014644	041040	052111	034040	
2283	014652	043040	044501	042514	
2284	014660	020104	047524	051440	
2285	014666	052105	000		
2286	014671	015	052012	041515	EM150: .ASCII<CR><LF>/TMCB SEGT DOESN'T GO LOW ON A KT ABORT OR/<CR><LF>
2287	014676	020102	042523	052107	
2288	014704	042040	042517	047123	
2289	014712	052047	043440	020117	
2290	014720	047514	020127	047117	
2291	014726	040440	045440	020124	
2292	014734	041101	051117	020124	
2293	014742	051117	005015		
2294	014746	052111	042040	042517	.ASCIIZ /IT DOESN'T GET TO DAPE/
2295	014754	047123	052047	043440	
2296	014762	052105	052040	020117	
2297	014770	040504	042520	000	
2298	014775	015	042012	050101	EM151: .ASCIIZ<CR><LF>/DAPE TV05*07 DOESN'T GO HIGH ON SEGT/
2299	015002	020105	053124	032460	
2300	015010	030052	020067	047504	
2301	015016	051505	023516	020124	
2302	015024	047507	044040	043511	
2303	015032	020110	047117	051440	
2304	015040	043505	000124		
2305	015044	005015	040504	042520	EM152: .ASCIIZ<CR><LF>/DAPE TV03 DOESN'T GO HIGH ON SEGT/
2306	015052	052040	030126	020063	
2307	015060	047504	051505	023516	

2308	015066	020124	047507	044040	
2309	015074	043511	020110	047117	
2310	015102	051440	043505	000124	
2311	015110	005015	046524	040503	EM153: .ASCII<CR><LF>/TMCA SEG+CON+PAR DOESN'T GO LOW OR IT DOES/<CR><LF>
2312	015116	051440	043505	041453	
2313	015124	047117	050053	051101	
2314	015132	042040	042517	047123	
2315	015140	052047	043440	020117	
2316	015146	047514	020127	051117	
2317	015154	044440	020124	047504	
2318	015162	051505	005015		
2319	015166	047516	020124	042507	.ASCIZ /NOT GET THRU TMCB E70(2)/
2320	015174	020124	044124	052522	
2321	015202	052040	041515	020102	
2322	015210	033505	024060	024462	
2323	015216	000			
2324	015217	015	052012	041515	EM155: .ASCII<CR><LF>/TMCA SEGTF DOESN'T GET TO E44 OR TMCE PAUSES H/<CR><LF>
2325	015224	020101	042523	052107	
2326	015232	020106	047504	051505	
2327	015240	023516	020124	042507	
2328	015246	020124	047524	042440	
2329	015254	032064	047440	020122	
2330	015262	046524	042503	050040	
2331	015270	052501	042523	020123	
2332	015276	006510	012		
2333	015301	104	042517	047123	.ASCIZ /DOESN'T GET TO E44 OR E44 BAD/
2334	015306	052047	043440	052105	
2335	015314	052040	020117	032105	
2336	015322	020064	051117	042440	
2337	015330	032064	041040	042101	
2338	015336	000			
2339	015337	015	052012	041515	EM156: .ASCII<CR><LF>/TMCC NEXM DOESN'T GO LOW OR IT DOESN'T GET THRU E34/<CR><LF>
2340	015344	020103	042516	046530	
2341	015352	042040	042517	047123	
2342	015360	052047	043440	020117	
2343	015366	047514	020127	051117	
2344	015374	044440	020124	047504	
2345	015402	051505	023516	020124	
2346	015410	042507	020124	044124	
2347	015416	052522	042440	032063	
2348	015424	005015			
2349	015426	051117	044440	020124	.ASCIZ /OR IT DOESN'T GET TO E14 OR E40 BAD/
2350	015434	047504	051505	023516	
2351	015442	020124	042507	020124	
2352	015450	047524	042440	032061	
2353	015456	047440	020122	032105	
2354	015464	020060	040502	000104	
2355	015472	005015	042516	046530	EM160: .ASCII<CR><LF>/NEXM BIT DIDN'T SET IN CPU ERROR REG/<CR><LF>
2356	015500	041040	052111	042040	
2357	015506	042111	023516	020124	
2358	015514	042523	020124	047111	
2359	015522	041440	052520	042440	
2360	015530	051122	051117	051040	
2361	015536	043505	005015		
2362	015542	051117	052040	041515	.ASCIZ /OR TMCC ABORT DOESN'T GO HIGH/
2363	015550	020103	041101	051117	

2364	015556	020124	047504	051505	
2365	015564	023516	020124	047507	
2366	015572	044040	043511	000110	
2367	015600	005015	042516	046530	EM161: .ASCIZ <CR><LF>/NEXM BIT DIDN'T CLEAR IN CPU ERROR REG/
2368	015606	041040	052111	042040	
2369	015614	042111	023516	020124	
2370	015622	046103	040505	020122	
2371	015630	047111	041440	052520	
2372	015636	042440	051122	051117	
2373	015644	051040	043505	000	
2374	015651	015	052012	041515	EM162: .ASCIZ<CR><LF>/TMCE KT BEND DOESN'T GO LOW ON TMCC NEXM LOW/
2375	015656	020105	052113	041040	
2376	015664	047105	020104	047504	
2377	015672	051505	023516	020124	
2378	015700	047507	046040	053517	
2379	015706	047440	020116	046524	
2380	015714	041503	047040	054105	
2381	015722	020115	047514	000127	
2382	015730	005015	046524	042503	EM163: .ASCIZ<CR><LF>/TMCE KT BEND DOESN'T GO LOW ON TMCD SL RED/
2383	015736	045440	020124	042502	
2384	015744	042116	042040	042517	
2385	015752	047123	052047	043440	
2386	015760	020117	047514	020127	
2387	015766	047117	052040	041515	
2388	015774	020104	046123	051040	
2389	016002	042105	000		
2390	016005	015	052012	041515	EM164: .ASCIZ<CR><LF>/TMCE KT BEND DOESN'T GO LOW ON TMCC ODD ADRS ERR/
2391	016012	020105	052113	041040	
2392	016020	047105	020104	047504	
2393	016026	051505	023516	020124	
2394	016034	047507	046040	053517	
2395	016042	047440	020116	046524	
2396	016050	041503	047440	042104	
2397	016056	040440	051104	020123	
2398	016064	051105	000122		
2399	016070	005015	052113	040440	EM165: .ASCIZ<CR><LF>?KT ABORT FAILED TO OVER-RIDE NEXM TRAP (KB11-E/EM?)
2400	016076	047502	052122	043040	
2401	016104	044501	042514	020104	
2402	016112	047524	047440	042526	
2403	016120	026522	044522	042504	
2404	016126	047040	054105	020115	
2405	016134	051124	050101	024040	
2406	016142	041113	030461	042455	
2407	016150	042457	000115		
2408	016154	005015	046524	042503	EM166: .ASCIZ<CR><LF>/TMCE CACHE BEND DIDN'T GO HIGH ON KT ABORT/
2409	016162	041440	041501	042510	
2410	016170	041040	047105	020104	
2411	016176	044504	047104	052047	
2412	016204	043440	020117	044510	
2413	016212	044107	047440	020116	
2414	016220	052113	040440	047502	
2415	016226	052122	000		
2416					
2417	016231	015	043012	046117	EM167: .ASCII<CR><LF>/FOLLOWING IS A LIST OF THE STACK LIMIT REG/<CR><LF>
2418	016236	047514	044527	043516	
2419	016244	044440	020123	020101	

2420	016252	044514	052123	047440	
2421	016260	020106	044124	020105	
2422	016266	052123	041501	020113	
2423	016274	044514	044515	020124	
2424	016302	042522	006507	012	
2425	016307	046	051440	020120	.ASCII/B SP VALUES THAT CAUSED AN ERROR. THEY ARE/<CR><LF>
2426	016314	040526	052514	051505	
2427	016322	052040	040510	020124	
2428	016330	040503	051525	042105	
2429	016336	040440	020116	051105	
2430	016344	047522	027122	020040	
2431	016352	044124	054505	040440	
2432	016360	042522	005015		
2433	016364	051107	052517	042520	.ASCIZ /GROUPED ACCORDING TO ERROR TYPES/
2434	016372	020104	041501	047503	
2435	016400	042122	047111	020107	
2436	016406	047524	042440	051122	
2437	016414	051117	052040	050131	
2438	016422	051505	000		
2439					
2440	016425	015	051412	051123	EM170: .ASCII<CR><LF>/SSRA PS RESTORE(1) DOESN'T GET TO RACK E63/<CR><LF>
2441	016432	020101	051520	051040	
2442	016440	051505	047524	042522	
2443	016446	030450	020051	047504	
2444	016454	051505	023516	020124	
2445	016462	042507	020124	047524	
2446	016470	051040	041501	020113	
2447	016476	033105	006463	012	
2448	016503	117	020122	033105	.ASCIZ /OR E63(5) BAD/
2449	016510	024063	024465	041040	
2450	016516	042101	000		
2451	016521	015	043412	044517	EM171: .ASCIZ<CR><LF> /GOING TO NEXT TEST/
2452	016526	043516	052040	020117	
2453	016534	042516	052130	052040	
2454	016542	051505	000124		
2455	016546	044124	020105	047506	EM201: .ASCIZ ?THE FOLLOWING ARE PAR/PDR REFERENCE TIMEOUTS?
2456	016554	046114	053517	047111	
2457	016562	020107	051101	020105	
2458	016570	040520	027522	042120	
2459	016576	020122	042522	042506	
2460	016604	042522	041516	020105	
2461	016612	044524	042515	052517	
2462	016620	051524	000		
2463	016623	124	042510	043040	EM202: .ASCIZ ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS FOR PAR'S/PDR'S?
2464	016630	046117	047514	044527	
2465	016636	043516	040440	042522	
2466	016644	042040	040525	020114	
2467	016652	042101	051104	051505	
2468	016660	044523	043516	042440	
2469	016666	051122	051117	020123	
2470	016674	047506	020122	040520	
2471	016702	023522	027523	042120	
2472	016710	023522	000123		
2473	016714	044124	020105	047506	EM203: .ASCIZ ?THE FOLLOWING ARE COUNT PATTERN ERRORS FOR PAR'S/PDR'S?
2474	016722	046114	053517	047111	
2475	016730	020107	051101	020105	

2476	016736	047503	047125	020124		
2477	016744	040520	052124	051105		
2478	016752	020116	051105	047522		
2479	016760	051522	043040	051117		
2480	016766	050040	051101	051447		
2481	016774	050057	051104	051447		
2482	017002	000				
2483						
2484						
2485	017003	105	050130	041505	DH1:	.ASCII ?EXPECTD ?
2486	017010	042124	040			
2487	017013	122	041505	044505	DH2:	.ASCIZ ?RECEIVD TESTNO PC AT ABORT?
2488	017020	042126	052040	051505		
2489	017026	047124	020117	050040		
2490	017034	020103	052101	040440		
2491	017042	047502	052122	000		
2492	017047	120	051101	05211:	DH3:	.ASCII ?PARITY ADDRESS CONTROL MAINTEN?<CRLF>
2493	017054	020131	040440	042104		
2494	017062	042522	051523	020040		
2495	017070	041440	047117	051124		
2496	017076	046117	046440	044501		
2497	017104	052116	047105	200		
2498	017111	103	047117	044504		.ASCIZ ?CONDITN REFERENC D REGISTR REGISTR TESTNO PC AT ABORT?
2499	017116	047124	051040	043105		
2500	017124	051105	047105	042103		
2501	017132	051040	043505	051511		
2502	017140	051124	051040	043505		
2503	017146	051511	051124	052040		
2504	017154	051505	047124	020117		
2505	017162	050040	020103	052101		
2506	017170	040440	047502	052122		
2507	017176	000				
2508	017177	105	051122	051117	DH5:	.ASCII ?ERROR AUTOI/D VIRTUAL?<CRLF>
2509	017204	020040	040440	052125		
2510	017212	044517	042057	053040		
2511	017220	051111	052524	046101		
2512	017226	200				
2513	017227	122	043505	051511		.ASCIZ ?REGISTR REGISTR ADDRESS TESTNO PC AT ABORT?
2514	017234	051124	051040	043505		
2515	017242	051511	051124	040440		
2516	017250	042104	042522	051523		
2517	017256	052040	051505	047124		
2518	017264	020117	050040	020103		
2519	017272	052101	040440	047502		
2520	017300	052122	000			
2521	017303	105	050130	041505	DH6:	.ASCII ?EXPECTD ERROR AUTOI/D VIRTUAL?<CRLF>
2522	017310	042124	042440	051122		
2523	017316	051117	020040	040440		
2524	017324	052125	044517	042057		
2525	017332	053040	051111	052524		
2526	017340	046101	200			
2527	017343	103	047117	044504		.ASCIZ ?CONDITN REGISTR REGISTR ADDRESS TESTNO PC AT ABORT?
2528	017350	047124	051040	043505		
2529	017356	051511	051124	051040		
2530	017364	043505	051511	051124		
2531	017372	040440	042104	042522		

2532	017400	051523	052040	051505						
2533	017406	047124	020117	050040						
2534	017414	020103	052101	040440						
2535	017422	047502	052122	000						
2536	017427	050	046515	030122	DH7:	.ASCIZ	?(MMR0)	TESTNO	ERRORPC?	
2537	017434	020051	052040	051505						
2538	017442	047124	020117	042440						
2539	017450	051122	051117	041520						
2540	017456	000								
2541	017457	114	040517	042504	DH11:	.ASCIZ	?LOADED	RECEIVD	TESTNO	ERRORPC?
2542	017464	020104	051040	041505						
2543	017472	044505	042126	052040						
2544	017500	051505	047124	020117						
2545	017506	042440	051122	051117						
2546	017514	041520	000							
2547	017517	105	050130	041505	DH13:	.ASCII	?EXPECTD	?		
2548	017524	042124	040							
2549	017527	050	046515	030522	DH12:	.ASCIZ	?(MMR1)	TESTNO	ERRORPC?	
2550	017534	020051	052040	051505						
2551	017542	047124	020117	042440						
2552	017550	051122	051117	041520						
2553	017556	000								
2554	017557	105	050130	041505	DH14:	.ASCIZ	?EXPECTD	(MMR2)	TESTNO	ERRORPC?
2555	017564	042124	024040	046515						
2556	017572	031122	020051	052040						
2557	017600	051505	047124	020117						
2558	017606	042440	051122	051117						
2559	017614	041520	000							
2560	017617	114	040517	042504	DH16:	.ASCII	?LOADED	?		
2561	017624	020104	040							
2562	017627	050	046515	031522	DH15:	.ASCIZ	?(MMR3)	TESTNO	ERRORPC?	
2563	017634	020051	052040	051505						
2564	017642	047124	020117	042440						
2565	017650	051122	051117	041520						
2566	017656	000								
2567	017657	101	042104	047522	DH17:	.ASCIZ	?ADDROR	ADDRAND	TESTNO	#ERRORS?
2568	017664	020122	040440	042104						
2569	017672	040522	042116	052040						
2570	017700	051505	047124	020117						
2571	017706	021440	051105	047522						
2572	017714	051522	000							
2573	017717	101	042104	042522	DH20:	.ASCIZ	?ADDRESS	DATA	TESTNO	ERRORPC?
2574	017724	051523	042040	052101						
2575	017732	020101	020040	052040						
2576	017740	051505	047124	020117						
2577	017746	042440	051122	051117						
2578	017754	041520	000							
2579	017757	101	042104	047522	DH21:	.ASCII	?ADDROR	ADDRAND	ADDROR	ADDRAND?<CRLF>
2580	017764	020122	040440	042104						
2581	017772	040522	042116	040440						
2582	020000	042104	047522	020122						
2583	020006	040440	042104	040522						
2584	020014	042116	200							
2585	020017	114	040517	042504		.ASCIZ	?LOADED	LOADED	ENABLED	ENABLED TESTNO #ERRORS?
2586	020024	020104	046040	040517						
2587	020032	042504	020104	042440						

2644	020536	042101	051104	051505	DH37:	.ASCIZ	?ADDRESS	GDDATA	BADDATA	TESTNO	ERRORPC?
2645	020544	020123	042107	040504							
2646	020552	040524	020040	040502							
2647	020560	042104	052101	020101							
2648	020566	042524	052123	047516							
2649	020574	020040	051105	047522							
2650	020602	050122	000103								
2651	020606	052123	051101	040524	DH40:	.ASCIZ	?STARTADR	FINISHADR	TESTNO	ERRORPC?	
2652	020614	051104	020040	044506							
2653	020622	044516	044123	042101							
2654	020630	020122	042524	052123							
2655	020636	047516	020040	051105							
2656	020644	047522	050122	000103							
2657	020652	052123	051101	040524	DH41:	.ASCIZ	?STARTADR	TESTNO	ERRORPC?		
2658	020660	051104	020040	042524							
2659	020666	052123	047516	020040							
2660	020674	051105	047522	050122							
2661	020702	000103									
2662	020704	040520	052124	051105	DH42:	.ASCII	?PATTERN	DATA	ADDRESS?	<CRLF>	
2663	020712	020116	040504	040524							
2664	020720	020040	020040	042101							
2665	020726	051104	051505	100123							
2666	020734	047514	042101	042105		.ASCIZ	?LOADED	FETCHED	INTENDED	TESTNO	ERRORPC?
2667	020742	020040	042506	041524							
2668	020750	042510	020104	047111							
2669	020756	042524	042116	042105							
2670	020764	020040	042524	052123							
2671	020772	047516	020040	051105							
2672	021000	047522	050122	000103							
2673	021006	042524	052123	047516	DH43:	.ASCIZ	?TESTNO	ERRORPC?			
2674	021014	020040	051105	047522							
2675	021022	050122	000103								
2676	021026	040504	040524	020040	DH44:	.ASCIZ	?DATA	TESTNO	ERRORPC?		
2677	021034	020040	042524	052123							
2678	021042	047516	020040	051105							
2679	021050	047522	050122	000103							
2680	021056	047516	026516	054105	DH45:	.ASCIZ	?NON-EXADDR	TESTNO	ERRORPC?		
2681	021064	042101	051104	052040							
2682	021072	051505	047124	020117							
2683	021100	042440	051122	051117							
2684	021106	041520	000								
2685	021111	113	050111	051101	DH50:	.ASCIZ	?KIPAR4	SIZELO	TESTNO	ERRORPC?	
2686	021116	020064	020040	051440							
2687	021124	055111	046105	020117							
2688	021132	020040	052040	051505							
2689	021140	047124	020117	042440							
2690	021146	051122	051117	041520							
2691	021154	000									
2692	021155	120	046107	047105	DH51:	.ASCIZ	?PGLENFD	VABLKNO	TESTNO	ERRORPC?	
2693	021162	042106	053040	041101							
2694	021170	045514	047516	052040							
2695	021176	051505	047124	020117							
2696	021204	042440	051122	051117							
2697	021212	041520	000								
2698	021215	124	051505	047124	DH53:	.ASCIZ	?TESTNO	ERRORPC?			
2699	021222	020117	042440	051122							

2812	022356	042440	051122	051117	
2813	022364	041520	000		
2814	022367	105	050130	041505	DH116: .ASCIZ ?EXPECTED RECEIVD TESTNO ERRORPC?
2815	022374	042124	051040	041505	
2816	022402	044505	042126	052040	
2817	022410	051505	047124	020117	
2818	022416	042440	051122	051117	
2819	022424	041520	000		
2820	022427	050	046515	030122	DH117: .ASCIZ ?(MMR0) (MMR1) (MMR2) TESTNO ERRORPC?
2821	022434	020051	024040	046515	
2822	022442	030522	020051	024040	
2823	022450	046515	031122	020051	
2824	022456	052040	051505	047124	
2825	022464	020117	042440	051122	
2826	022472	051117	041520	000	
2827	022477	123	045524	052120	DH120: .ASCIZ ?STKPTR TESTNO ERRORPC?
2828	022504	020122	052040	051505	
2829	022512	047124	020117	042440	
2830	022520	051122	051117	041520	
2831	022526	000			
2832	022527	107	042104	052101	DH121: .ASCIZ ?GDDATA STORED TESTNO ERRORPC?
2833	022534	020101	051440	047524	
2834	022542	042522	020104	052040	
2835	022550	051505	047124	020117	
2836	022556	042440	051122	051117	
2837	022564	041520	000		
2838	022567	050	051520	024527	DH122: .ASCIZ ?(PSW) TESTNO ERRORPC EXPECTING XXX340?
2839	022574	020040	052040	051505	
2840	022602	047124	020117	042440	
2841	022610	051122	051117	041520	
2842	022616	042440	050130	041505	
2843	022624	044524	043516	054040	
2844	022632	054130	032063	000060	
2845	022640	050050	053523	020051	DH123: .ASCIZ ?(PSW) TESTNO ERRORPC?
2846	022646	020040	042524	052123	
2847	022654	047516	020040	051105	
2848	022662	047522	050122	000103	
2849	022670	042524	052123	047516	DH127: .ASCIZ ?TESTNO ERRORPC?
2850	022676	042440	051122	051117	
2851	022704	041520	000		
2852	022707	050	046515	030122	DH131: .ASCIZ ?(MMR0) (MMR1) (MMR2) TESTNO ERRORPC EXPECTING 020031?
2853	022714	020051	024040	046515	
2854	022722	030522	020051	024040	
2855	022730	046515	031122	020051	
2856	022736	052040	051505	047124	
2857	022744	020117	042440	051122	
2858	022752	051117	041520	042440	
2859	022760	050130	041505	044524	
2860	022766	043516	030040	030062	
2861	022774	031460	000061		
2862	023000	020040	041520	020040	DH133: .ASCIZ ? PC KIPDR (KIPDR)?
2863	023006	020040	044513	042120	
2864	023014	020122	024040	044513	
2865	023022	042120	024522	000	
2866	023027	040	050040	020103	DH135: .ASCIZ / PC (CR PAR-ADR (PAR) (PDR) TST-DATA-ADRS(VA)/
2867	023034	020040	041440	051103	

2868	023042	020040	040520	026522					
2869	023050	042101	020122	020040					
2870	023056	050050	051101	020051					
2871	023064	024040	042120	024522					
2872	023072	020040	051524	026524					
2873	023100	040504	040524	040455					
2874	023106	051104	024123	040526					
2875	023114	000051							
2876	023116	020040	041520	020040	DH140:	.ASCIZ	? PC	SIPDR	(SIPDR)?
2877	023124	020040	044523	042120					
2878	023132	020122	024040	044523					
2879	023140	042120	024522	000					
2880	023145	040	050040	020103	DH142:	.ASCIZ	? PC	UIPDR	(UIPDR)?
2881	023152	020040	052440	050111					
2882	023160	051104	020040	052450					
2883	023166	050111	051104	000051					
2884	023174	042101	051104	051505	DH145:	.ASCIZ	/ADDRESS EXPECT RECEIVE TESTNO	ERRORPC/	
2885	023202	020123	054105	042520					
2886	023210	052103	051040	041505					
2887	023216	044505	042526	052040					
2888	023224	051505	047124	020117					
2889	023232	042440	051122	051117					
2890	023240	041520	000						
2891	023243	105	051122	051117	DH146:	.ASCIZ	/ERRORPC TEST NUMBER/		
2892	023250	041520	052040	051505					
2893	023256	020124	052516	041115					
2894	023264	051105	000						
2895	023267	105	051122	051117	DH147:	.ASCII	/ERRORPC	(PUERR REG	TST NUM/<CR><LF>
2896	023274	041520	041411	052520					
2897	023302	051105	020122	042522					
2898	023310	004507	051524	020124					
2899	023316	052516	006515	012					
2900	023323	011	054105	042520		.ASCIZ	/	EXPECT	ACTUAL/
2901	023330	052103	040411	052103					
2902	023336	040525	000114						
2903	023342	051105	047522	050122	DH150:	.ASCIZ	/ERRORPC TEST NUMBER/<CR><LF>		
2904	023350	020103	042524	052123					
2905	023356	047040	046525	042502					
2906	023364	006522	000012						
2907	023370	042101	051104	051505	DH201:	.ASCIZ	?ADDRESS TESTNO?		
2908	023376	020123	042524	052123					
2909	023404	047516	000						
2910	023407	101	042104	042522	DH202:	.ASCII	?ADDRESS ADDRESS?<CRLF>		
2911	023414	051523	040440	042104					
2912	023422	042522	051523	200					
2913	023427	114	040517	042504		.ASCIZ	?LOADED	JUSTREAD	TESTNO?
2914	023434	020104	045040	051525					
2915	023442	051124	040505	020104					
2916	023450	042524	052123	047516					
2917	023456	000							
2918	023457	101	042104	042522	DH203:	.ASCIZ	?ADDRESS DATARED PATTERN COUNT	TESTNO?	
2919	023464	051523	042040	052101					
2920	023472	051101	042105	050040					
2921	023500	052101	042524	047122					
2922	023506	041440	052517	052116					
2923	023514	020040	052040	051505					

2924	023522	047124	000117						
2925	023526	020040	000	TWOSP:	.ASCIZ	/ /			
2926		023532			.EVEN				
2927									
2928									
2929	023532	001224		DT1:	.WORD	CPUEXP			
2930	023534	001260	001264	DT2:	.WORD	PCPUER,TESTNO,BADPC,0			
2931	023542	000000							
2932	023544	001240	001234	DT3:	.WORD	PPARER,PLOADR,PCONTR,PMMAINT,TESTNO,BADPC,0			
2933	023552	001244	001264						
2934	023560	000000							
2935	023562	001250	001252	DT5:	.WORD	PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0			
2936	023570	001264	001262						
2937	023576	001226	001250	DT6:	.WORD	MMEXP,PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0			
2938	023604	001254	001264						
2939	023612	000000							
2940	023614	001160	001264	DT7:	.WORD	\$REG1,TESTNO,\$ERRPC,0			
2941	023622	000000							
2942	023624	001162	001160	DT11:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0			
2943	023632	001120	000000						
2944	023636	001162	001264	DT12:	.WORD	\$REG2,TESTNO,\$ERRPC,0			
2945	023644	000000							
2946	023646	001164	001162	DT13:	.WORD	\$REG3,\$REG2,TESTNO,\$ERRPC,0			
2947	023654	001120	000000						
2948	023660	001160		DT16:	.WORD	\$REG1			
2949	023662	001162	001264	DT15:	.WORD	\$REG2,TESTNO,\$ERRPC,0			
2950	023670	000000							
2951	023672	001276	001300	DT17:	.WORD	ADDROR,ADRAND,TESTNO,ERRCNT,0			
2952	023700	001302	000000						
2953	023704	001156	001162	DT20:	.WORD	\$REG0,\$REG2,TESTNO,\$ERRPC,0			
2954	023712	001120	000000						
2955	023716	001276	001300	DT21:	.WORD	ADDROR,ADRAND,DATAOR,DATAND,TESTNO,ERRCNT,0			
2956	023724	001270	001264						
2957	023732	000000							
2958	023734	001276	001300	DT22:	.WORD	ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,TESTNO,ERRCNT,0			
2959	023742	001274	001266						
2960	023750	001264	001302						
2961	023756	001156	001162	DT23:	.WORD	\$REG0,\$REG2,\$REG1,TESTNO,\$ERRPC,0			
2962	023764	001264	001120						
2963	023772	001156	001264	DT24:	.WORD	\$REG0,TESTNO,\$ERRPC,0			
2964	024000	000000							
2965	024002	001156	001160	DT25:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0			
2966	024010	001120	000000						
2967	024014	001156	001160	DT31:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0			
2968	024022	001264	001120						
2969	024030	001156	001264	DT35:	.WORD	\$REG0,TESTNO,\$ERRPC,0			
2970	024036	000000							
2971	024040	001156	001160	DT36:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0			
2972	024046	001264	001120						
2973	024054	001156	001160	DT37:	.WORD	\$REG0,\$REG1,\$REG2,TESTNO,\$ERRPC,0			
2974	024062	001264	001120						
2975	024070	001174	001176	DT40:	.WORD	\$TMP1,\$TMP2,TESTNO,\$ERRPC,0			
2976	024076	001120	000000						
2977	024102	001174	001264	DT41:	.WORD	\$TMP1,TESTNO,\$ERRPC,0			
2978	024110	000000							
2979	024112	001162	001164	DT42:	.WORD	\$REG2,\$REG3,\$REG0,TESTNO,\$ERRPC,0			

2980	024120	001264	001120	000000					
2981	024126	001264	001120	000000	DT43:	.WORD	TESTNO,\$ERRPC,0		
2982	024134	001160	001264	001120	DT44:	.WORD	\$REG1,TESTNO,\$ERRPC,0		
2983	024142	000000							
2984	024144	001156	001264	001120	DT45:	.WORD	\$REG0,TESTNO,\$ERRPC,0		
2985	024152	000000							
2986	024154	172350	177760	001264	DT50:	.WORD	KIPAR4,SIZE0,TESTNO,\$ERRPC,0		
2987	024162	001120	000000						
2988	024166	001160	001164	001264	DT51:	.WORD	\$REG1,\$REG3,TESTNO,\$ERRPC,0		
2989	024174	001120	000000						
2990	024200	001264	001120	000000	DT53:	.WORD	TESTNO,\$ERRPC,0		
2991	024206	001156	001160	001264	DT55:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0		
2992	024214	001120	000000						
2993	024220	177572	172310	001264	DT56:	.WORD	MMR0,KIPDR4,TESTNO,\$ERRPC,0		
2994	024226	001120	000000						
2995	024232	001172	001264	001120	DT64:	.WORD	\$TMP0,TESTNO,\$ERRPC,0		
2996	024240	000000							
2997	024242	001226	001250	001264	DT65:	.WORD	MMEXP,MMR0,TESTNO,\$ERRPC,0		
2998	024250	001120	000000						
2999	024254	001250	001264	001120	DT66:	.WORD	PMMR0,TESTNO,\$ERRPC,0		
3000	024262	000000							
3001	024264	001160	001250	001252	DT70:	.WORD	\$REG1,PMMR0,PMMR1,PMMR2,TESTNO,\$ERRPC,0		
3002	024272	001254	001264	001120					
3003	024300	000000							
3004	024302	001160	001250	001264	DT72:	.WORD	\$REG1,PMMR0,TESTNO,\$ERRPC,0		
3005	024310	001120	000000						
3006	024314	001250	001252	001254	DT73:	.WORD	PMMR0,PMMR1,PMMR2,\$TMP0,\$TMP1,\$TMP2,TESTNO,\$ERRPC,0		
3007	024322	001172	001174	001176					
3008	024330	001264	001120	000000					
3009	024336	001250	001252	001254	DT75:	.WORD	PMMR0,PMMR1,PMMR2,TESTNO,\$ERRPC,0		
3010	024344	001264	001120	000000					
3011	024352	001160	001254	001264	DT100:	.WORD	\$REG1,PMMR2,TESTNO,\$ERRPC,0		
3012	024360	001120	000000						
3013	024364	001162	001250	001264	DT101:	.WORD	\$REG2,PMMR0,TESTNO,\$ERRPC,0		
3014	024372	001120	000000						
3015	024376	001172	001264	001120	DT102:	.WORD	\$TMP0,TESTNO,\$ERRPC,0		
3016	024404	000000							
3017	024406	001164	001160	001264	DT112:	.WORD	\$REG3,\$REG1,TESTNO,\$ERRPC,0		
3018	024414	001120	000000						
3019	024420	001164	001156	001264	DT113:	.WORD	\$REG5,\$REG0,TESTNO,\$ERRPC,0		
3020	024426	001120	000000						
3021	024432	001162	001160	001264	DT114:	.WORD	\$REG2,\$REG1,TESTNO,\$ERRPC,0		
3022	024440	025760	000000						
3023	024444	001156	001160	001264	DT116:	.WORD	\$REG0,\$REG1,TESTNO,\$ERRPC,0		
3024	024452	001120	000000						
3025	024456	001250	001252	001254	DT117:	.WORD	PMMR0,PMMR1,PMMR2,TESTNO,\$ERRPC,0		
3026	024464	001264	001120	000000					
3027	024472	001160	001264	001120	DT120:	.WORD	\$REG1,TESTNO,\$ERRPC,0		
3028	024500	000000							
3029	024502	001156	001264	001120	DT122:	.WORD	\$REG0,TESTNO,\$ERRPC,0		
3030	024510	000000							
3031	024512	001264	001120	000000	DT127:	.WORD	TESTNO,\$ERRPC,0		
3032	024520	001250	001252	001254	DT131:	.WORD	PMMR0,PMMR1,PMMR2,TESTNO,\$ERRPC,0		
3033	024526	001264	001120	000000					
3034	024534	001120	001156	001160	DT133:	.WORD	\$ERRPC,\$REG0,\$REG1,0		
3035	024542	000000							

3036	024544	001120	001156	001160	DT135:	.WORD	\$ERRPC,\$REG0,\$REG1,\$REG2,\$REG3,\$REG4,0
3037	024552	001162	001164	001166			
3038	024560	000000					
3039	024562	001172	001174	001176	DT145:	.WORD	\$TMP0,\$TMP1,\$TMP2,TESTNO,\$ERRPC,0
3040	024570	001264	001120	000000			
3041	024576	001120	001264	000000	DT146:	.WORD	\$ERRPC,TESTNO,0
3042	024604	001120	001172	001174	DT147:	.WORD	\$ERRPC,\$TMP0,\$TMP1,TESTNO,0
3043	024612	001264	000000				
3044	024616	001156	001264	000000	DT201:	.WORD	\$REG0,TESTNO,0
3045	024624	001156	001160	001264	DT202:	.WORD	\$REG0,\$REG1,TESTNO,0
3046	024632	000000					
3047	024634	001156	001162	001166	DT203:	.WORD	\$REG0,\$REG2,\$REG4,\$REG1,TESTNO,0
3048	024642	001160	001264	000000			
3049							
3050							
3051	024650	000			DF1:	.BYTE	0
3052	024651	000	000	000	DF2:	.BYTE	0,0,0
3053	024654	000	002	000	DF3:	.BYTE	0,2,0,0,0,0
3054	024657	000	000	000			
3055	024662	000	000	000	DF5:	.BYTE	0,0,0,0,0
3056	024665	000	000				
3057	024667	000	000	000	DF6:	.BYTE	0,0,0,0,0,0
3058	024672	000	000	000			
3059	024675	000	000	000	DF7:	.BYTE	0,0,0
3060	024700	000	000	000	DF11:	.BYTE	0,0,0,0
3061	024703	000					
3062	024704	000	000	000	DF12:	.BYTE	0,0,0
3063	024707	000	000	000	DF13:	.BYTE	0,0,0,0
3064	024712	000					
3065	024713	000	000	000	DF16:	.BYTE	0,0,0
3066	024716	000	000	000	DF15:	.BYTE	0,0,0,0
3067	024721	000					
3068	024722	000	000	000	DF17:	.BYTE	0,0,0,1
3069	024725	001					
3070	024726	000	000	000	DF20:	.BYTE	0,0,0,0
3071	024731	000					
3072	024732	000	000	000	DF21:	.BYTE	0,0,0,0,0,1
3073	024735	000	000	001			
3074	024740	000	000	000	DF22:	.BYTE	0,0,0,0,0,0,0,1
3075	024743	000	000	000			
3076	024746	000	001				
3077	024750	000	000	000	DF23:	.BYTE	0,0,0,0,0
3078	024753	000	000				
3079	024755	000	000	000	DF24:	.BYTE	0,0,0
3080	024760	000	000	000	DF25:	.BYTE	0,0,0,0
3081	024763	000					
3082	024764	000	000	000	DF31:	.BYTE	0,0,0,0,0
3083	024767	000	000				
3084	024771	000	000	000	DF35:	.BYTE	0,0,0
3085	024774	000	000	000	DF36:	.BYTE	0,0,0,0,0
3086	024777	000	000				
3087	025001	003	000	000	DF37:	.BYTE	3,0,0,0,0
3088	025004	000	000				
3089	025006	004	004	000	DF40:	.BYTE	4,4,0,0
3090	025011	000					
3091	025012	004	000	000	DF41:	.BYTE	4,0,0

3092	025015	000	000	003	DF42:	.BYTE	0,0,3,0,0
3093	025020	000	000				
3094	025022	000	000		DF43:	.BYTE	0,0
3095	025024	000	000	000	DF44:	.BYTE	0,0,0
3096	025027	003	000	000	DF45:	.BYTE	3,0,0
3097	025032	004	004	000	DF50:	.BYTE	4,4,0,0
3098	025035	000					
3099	025036	000	000	000	DF51:	.BYTE	0,0,0,0
3100	025041	000					
3101	025042	000	000		DF53:	.BYTE	0,0
3102	025044	000	000	000	DF55:	.BYTE	0,0,0,0
3103	025047	000					
3104	025050	000	000	000	DF56:	.BYTE	0,0,0,0
3105	025053	000					
3106	025054	000	000	000	DF64:	.BYTE	0,0,0
3107	025057	000	000	000	DF65:	.BYTE	0,0,0,0
3108	025062	000					
3109	025063	000	000	000	DF66:	.BYTE	0,0,0
3110	025066	000	000	000	DF70:	.BYTE	0,0,0,0,0,0
3111	025071	000	000	000			
3112	025074	000	000	000	DF72:	.BYTE	0,0,0,0
3113	025077	000					
3114	025100	000	000	000	DF73:	.BYTE	0,0,0,0,0,0,0,0
3115	025103	000	000	000			
3116	025106	000	000				
3117	025110	000	000	000	DF75:	.BYTE	0,0,0,0,0
3118	025113	000	000				
3119	025115	000	000	000	DF100:	.BYTE	0,0,0,0
3120	025120	000					
3121	025121	000	000	000	DF101:	.BYTE	0,0,0,0
3122	025124	000					
3123	025125	000	000	000	DF102:	.BYTE	0,0,0
3124	025130	000	000	000	DF112:	.BYTE	0,0,0,0
3125	025133	000					
3126	025134	000	000	000	DF113:	.BYTE	0,0,0,0
3127	025137	000					
3128	025140	000	000	000	DF114:	.BYTE	0,0,0,0
3129	025143	000					
3130	025144	000	000	000	DF116:	.BYTE	0,0,0,0
3131	025147	000					
3132	025150	000	000	000	DF117:	.BYTE	0,0,0,0,0
3133	025153	000	000				
3134	025155	000	000	000	DF120:	.BYTE	0,0,0
3135	025160	000	000	000	DF122:	.BYTE	0,0,0
3136	025163	000	000		DF127:	.BYTE	0,0
3137	025165	000	000	000	DF131:	.BYTE	0,0,0,0,0
3138	025170	000	000				
3139							
3140							
3141							
3142	025172	000	000		DF201:	.BYTE	0,0
3143	025174	000	000	000	DF202:	.BYTE	0,0,0
3144	025177	000	000	000	DF203:	.BYTE	0,0,0,0,0
3145	025202	000	000				
3146						.EVEN	
3147							

```

3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161 025204
3162 025204 000240
3163 025206 005037 001102
3164 025212 005037 001206
3165 025216 005237 001100
3166 025222 042737 100000 001100
3167 025230 005327
3168 025232 000001
3169 025234 003072
3170 025236 012737
3171 025240 000001
3172 025242 025232
3173 025244 104400 025252
3174 025250 000407
3175
3176 025270
3177 025270 013746 001100
3178
3179 025274 104410
3180 025276 104400 025304
3181 025302 000421
3182
3183 025346
3184 025346 013746 001114
3185
3186 025352 104410
3187 025354 104400 001217
3188 025360 005037 001114
3189 025364 013700 000042
3190 025370 001414
3191 025372 005046
3192 025374 012746 025402
3193 02540C 000427
3194
3195 025402
3196 025402 013700 000042
3197 025406 001405
3198 025410 000005
3199 025412 004710
3200 025414 000240
3201 025416 000240
3202 025420 000240
3203 025422

```

```

;*****
.SBTTL END OF PASS ROUTINE

;*INCREMENT THE PASS NUMBER ($PASS)
;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
;*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
;*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
;*IF SW12=1 INHIBIT TRACE TRAP
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO LOOP

$EOP:
    NOP
    CLR $STNM          ;;ZERO THE TEST NUMBER
    CLR $TIMES        ;;ZERO THE NUMBER OF ITERATIONS
    INC $PASS         ;;INCREMENT THE PASS NUMBER
    BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
    DEC (PC)+         ;;LOOP?
$EOPCT: .WORD 1
    BGT $DOAGN        ;;YES
    MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD 1
    TYPE ,65$         ;;TYPE ASCIZ STRING
    BR 64$            ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <12><15>/END PASS #/
64$:
    MOV $PASS,-(SP)  ;;SAVE $PASS FOR TYPEOUT
                    ;;TYPE PASS NUMBER
                    ;;GO TYPE--DECIMAL ASCII WITH SIGN
    TYPDS
    TYPE ,67$        ;;TYPE ASCIZ STRING
    BR 66$          ;;GET OVER THE ASCIZ
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:
    MOV $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
                    ;;TOTAL NUMBER OF ERRORS
                    ;;GO TYPE--DECIMAL ASCII WITH SIGN
    TYPDS
    TYPE , $CRLF     ;;TYPE CARRIAGE RETURN, LINE FEED
    CLR $ERTTL      ;;CLEAR ERROR TOTAL
$GET42: MOV @#42,R0  ;;GET MONITOR ADDRESS
    BEQ $DOAGN      ;;BRANCH IF NO MONITOR
    CLR -(SP)       ;;INSURE THE 'T' BIT IS CLEAR
    MOV # $CLR.T,-(SP) ;;SETUP FOR AN RTI OR RTT
    BR $RTRN        ;;GO DO AN RTI OR RTT TO LOAD THE PSW
                    ;;WITH A CLEARED 'T' BIT
$CLR.T:
    MOV @#42,R0     ;;INSURE R0 CONTAINS THE MONITORS
    BEQ $DOAGN      ;;RETURN ADDRESS
    RESET          ;;CLEAR THE WORLD
$ENDAD: JSR PC,(R0) ;;GO TO MONITOR
    NOP            ;;SAVE ROOM
    NOP            ;;FOR
    NOP            ;;ACT11
$DOAGN:

```

```

3204 025422 013746 177776      MOV    @#PS,-(SP)      ;;PUT THE PS ON THE STACK AND
3205 025426 042716 000020      BIC    #20,(SP)      ;;CLEAR THE 'T' BIT
3206 025432 032737 010000 177570  BIT    #BIT12,@#SWR  ;;RUN WITH TRACE TRAP?
3207 025440 001005              BNE    1$            ;;BR IF NO
3208 025442 005137 025466      COM    $TBIT         ;;IS IT TIME FOR TRACE TRAP
3209 025446 100402              BMI    1$            ;;BR IF NO
3210 025450 052716 000020      BIS    #20,(SP)      ;;SET TRACE TRAP
3211 025454 012746 025462      1$:   MOV    #SLOOP,-(SP)  ;;JUMP TO START OF TEST
3212 025460 000002      $RTRN: RTI          ;;RETURN--THIS IS CHANGED TO
3213                                     ;;AN 'RTT' IF 'RTT' IS A LEGAL
3214                                     ;;INSTRUCTION
3215 025462              $LOOP:              ;;
3216 025462 000137 040272      JMP    @#LOOP        ;;RETURN
3217 025466 000000              $TBIT: .WORD    0      ;;'T' BIT STATE INDICATOR
3218 025470 377 377 000      $ENULL: .BYTE  -1,-1,0  ;;NULL CHARACTER STRING
3219                                     .EVEN
3220                                     ;;*****
3221                                     .SBTTL SCOPE HANDLER ROUTINE
3222                                     ;;
3223                                     ;;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3224                                     ;;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3225                                     ;;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
3226                                     ;;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3227                                     ;;*SW14=1 LOOP ON TEST
3228                                     ;;*SW11=1 INHIBIT ITERATIONS
3229                                     ;;*SW09=1 LOOP ON ERROR
3230                                     ;;*SW08=1 LOOP ON TEST IN SWR<6:0>
3231                                     ;;*CALL
3232                                     ;;* SCOPE ;;SCOPE=IOT
3233                                     ;;
3234                                     $SCOPE:
3235 025474 005037 001314      CLR    RETRY         ;;CLEAR THE RETRY FLAG BEFORE THIS TEST
3236 025474 005037 001302      CLR    ERRCNT        ;;CLEAR THE MULTIPLE ERROR COUNTER
3237 025500 006137 177570      ROL    @#SWR         ;;LOOP ON PRESENT TEST?
3238 025504 006137 177570      BMI    $OVER         ;;YES IF SW14=1
3239 025510 100514              ;#####START OF CODE FOR THE XOR TESTER#####
3240      $XTSTR: BR    6$      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
3241 025512 000416              ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
3242                                     ;;
3243 025514 013746 000004      MOV    @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
3244 025520 012737 025540 000004  MOV    #5$,@#ERRVEC  ;;SET FOR TIMEOUT
3245 025526 005737 177060      TST    @#177060      ;;TIME OUT ON XOR?
3246 025532 012637 000004      MCV    (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3247 025536 000466      BR    $SVLAD         ;;GO TO THE NEXT TEST
3248 025540 022626      5$:   CMP    (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
3249 025542 012637 000004      MOV    (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3250 025546 000426      BR    7$            ;;LOOP ON THE PRESENT TEST
3251 025550      6$:;#####END OF CODE FOR THE XOR TESTER#####
3252 025550 032737 000400 177570  BIT    #BIT08,@#SWR  ;;LOOP ON SPEC. TEST?
3253 025556 001407              BEQ    2$            ;;BR IF NO
3254 025560 013746 177570      MOV    @#SWR,-(SP)    ;;SET DESIRED TEST NUM. FROM SWR
3255 025564 042716 000200      BIC    #S$WRMK,(SP)   ;;STRIP AWAY UNDESIRED BITS
3256 025570 122637 001102      CMPB  (SP)+,$TSTNM   ;;ON THE RIGHT TEST?
3257 025574 001462              BEQ    $OVER         ;;BR IF YES
3258 025576 105737 001103      2$:   TSTB  $ERFLG     ;;HAS AN ERROR OCCURRED?
3259 025602 001421              BEQ    3$            ;;BR IF NO
    
```

```
3260 025604 123737 001117 001103      CMPB   $ERMAX,$ERFLG  ;;MAX. ERRORS FOR THIS TEST OCCURRED?
3261 025612 101015                    BHI    3$             ;;BR IF NO
3262 025614 032737 001000 177570      BIT    #BIT09,@#SWR  ;;LOOP ON ERROR?
3263 025622 001404                    BEQ    4$             ;;BR IF NO
3264 025624 013737 001112 001110 7$:  MOV    $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
3265 025632 000443                    BR     $OVER
3266 025634 105037 001103          4$:  CLRB  $ERFLG        ;;ZERO THE ERROR FLAG
3267 025640 005037 001206          CLR   $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
3268 025644 000415                    BR     1$             ;;ESCAPE TO THE NEXT TEST
3269 025646 032737 004000 177570 3$:  BIT    #BIT11,@#SWR  ;;INHIBIT ITERATIONS?
3270 025654 001011                    BNE   1$             ;;BR IF YES
3271 025656 005737 001100          TST   $PASS        ;;IF FIRST PASS OF PROGRAM
3272 025662 001406                    BEQ    1$             ;;INHIBIT ITERATIONS
3273 025664 005237 001106          INC   $ICNT        ;;INCREMENT ITERATION COUNT
3274 025670 023737 001206 001106      CMP   $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
3275 025676 002021                    BGE   $OVER        ;;BR IF MORE ITERATION REQUIRED
3276 025700 012737 000001 001106 1$:  MOV    #1,$ICNT     ;;REINITIALIZE THE ITERATION COUNTER
3277 025706 013737 025756 001206      MOV    $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
3278 025714 105237 001102          $SVLAD: INCB $STNM    ;;COUNT TEST NUMBERS
3279 025720 011637 001110          MOV    (SP),$LPADR  ;;SAVE SCOPE LOOP ADDRESS
3280 025724 011637 001112          MOV    (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
3281 025730 005037 001210          CLR   $ESCAPE     ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
3282 025734 112737 000001 001117      MOVB  #1,$ERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
3283 025742 013737 001102 177570 $OVER: MOV  $STNM,@#DISPLAY ;;DISPLAY TEST NUMBER
3284 025750 013716 001110          MOV    $LPADR,(SP) ;;FUDGE RETURN ADDRESS
3285 025754 000002                    RTI
3286 025756 000310          $MXCNT: 200.      ;;MAX. NUMBER OF ITERATIONS
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
```

.SBTTL ERROR HANDLER ROUTINE

```
;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;*AND GO TO ERTYPE ON ERROR
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW15=1      HALT ON ERROR
;*           HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
;*SW13=1      INHIBIT ERROR TYPEOUTS
;*SW10=1      BELL ON ERROR
;*SW09=1      LOOP ON ERROR
;*CALL
;*          ERROR  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
```

```
3303 025760 113737 001102 001264 $ERROR:
3304 025760 005237 001302          MOVB  $STNM,TESTNO  ;;SAVE TEST NUMBER FOR ERROR TYPE OUT
3305 025766 010037 001156          INC   ERRCNT        ;;KEEP COUNT OF MULTIPLE ERRORS
3306 025772 010037 001156          MOV   R0,$REG0     ;;SAVE R0
3307 025776 010137 001160          MOV   R1,$REG1     ;;SAVE R1
3308 026002 010237 001162          MOV   R2,$REG2     ;;SAVE R2
3309 026006 010337 001164          MOV   R3,$REG3     ;;SAVE R3
3310 026012 010437 001166          MOV   R4,$REG4     ;;SAVE R4
3311 026016 010537 001170          MOV   R5,$REG5     ;;SAVE R5
3312 026022 105237 001103          7$:  INCB  $ERFLG        ;;SET THE ERROR FLAG
3313 026026 001775                    BEQ    7$           ;;DON'T LET THE FLAG GO TO ZERO
3314 026030 013737 001102 177570      MOV   $STNM,@#DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
3315 026036 005737 177570          TST   @#SWR        ;;HALT ON ERROR = 1?
```

```

3316 026042 100001          BPL      8$          ;;BRANCH IF NO
3317 026044 000000          HALT                    ;;YES--HALT
3318 026046 032737 002000 177570 8$:      BIT      #BIT10,@#SWR  ;;BELL ON ERROR?
3319 026054 001402          BEQ      1$          ;;NO - SKIP
3320 026056 104400 001212          TYPE    ,SBELL        ;;RING BELL
3321 026062 005237 001114          INC     $ERTTL        ;;COUNT THE NUMBER OF ERRORS
3322 026066 011637 001120          MOV     (SP), $ERRPC  ;;GET ADDRESS OF ERROR INSTRUCTION
3323 026072 162737 000002 001120          SUB     #2, $ERRPC
3324 026100 117737 153014 001116          MOVNB  @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
3325 026106 032737 020000 177570          BIT      #BIT13,@#SWR  ;;SKIP TYPEOUT IF SET
3326 026114 001004          BNE     2$          ;;SKIP TYPEOUTS
3327 026116 004737 026666          JSR     PC, ERTYPE    ;;GO TO USER ERROR ROUTINE
3328 026122 104400 001217          TYPE    , $CRLF
3329 026126 005737 177570          2$:      TST      @#SWR    ;;HALT ON ERROR
3330 026132 100001          BPL      9$          ;;SKIP IF CONTINUE
3331 026134 000000          HALT                    ;;HALT ON ERROR!
3332 026136 022737 025412 000042 9$:      CMP      # $ENDAD, 42  ;;ACT-11?
3333 026144 001001          BNE     3$          ;;BRANCH IF NO
3334 026146 000000          HALT                    ;;YES
3335 026150 032737 001000 177570 3$:      BIT      #BIT09,@#SWR  ;;LOOP ON ERROR SWITCH SET?
3336 026156 001402          BEQ     4$          ;;BR IF NO
3337 026160 013716 001112          MOV     $LPERR, (SP)  ;;FUDGE RETURN FOR LOOPING
3338 026164 005737 001210          4$:      TST      $ESCAPE  ;;CHECK FOR AN ESCAPE ADDRESS
3339 026170 001402          BEQ     5$          ;;BR IF NONE
3340 026172 013716 001210          MOV     $ESCAPE, (SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
3341 026176          5$:
3342 026176 032737 001000 177570          BIT      #SW9, SWR    ;;IS THE LOOP ON ERROR SWITCH UP?
3343 026204 001421          BEQ     EREXIT        ;;BRANCH IF NOT LOOPING ON ERROR
3344 026206 005037 177766          CLR     CPUERR        ;;CLEAR CPU ERROR REGISTER
3345 026212 012737 177777 177744          MOV     #-1, MEMERR   ;;CLEAR MEMORY ERROR REGISTER
3346 026220 042737 176776 177572          BIC     #176776, MMRO ;;CLEAR MEMORY MANAGEMENT REG 0
3347          ;;LEAVE BIT0 AND BIT9 UNCHANGED
3348 026226 012737 177777 031672          MOV     #-1, CPFLAG  ;;RE-INITIALIZE CP TRAP FLAG
3349 026234 012737 177777 031776          MOV     #-1, PAFLAG  ;;RE-INITIALIZE PARITY TRAP FLAG
3350 026242 012737 177777 032230          MOV     #-1, MMFLAG  ;;RE-INITIALIZE MEMORY MANAGE. TRAP FLAG
3351 026250 000002          EREXIT: RTI          ;;RETURN TO TEST AFTER ERROR
3352          .SBTTL SPURIOUS ERROR HANDLER
3353          ;* THIS ROUTINE IS ENTERED BY AN UNEXPECTED TRAP TO 4 OR 114.
3354          ;* IF SWITCH 13 IS OFF, AN ERROR MESSAGE, THE ERROR REGISTER,
3355          ;* THE ERROR PC, AND THE TEST NUMBER ARE TYPED OUT.
3356          ;* IF SWITCH 13 IS ON, ONLY THE ERROR MESSAGE WILL BE TYPED.
3357          ;*****
3358 026252 011637 001120          CPUSPUR:MOV (SP), $ERRPC ;SAVE THE ERROR PC
3359 026256 012706 001100          MOV     #STACK, SP   ;RESTORE THE SP
3360 026262 104400 026270          TYPE    , 65$        ;;TYPE ASCIZ STRING
3361 026266 000414          BR      64$          ;;GET OVER THE ASCIZ
3362          ;;65$: .ASCIZ /UNEXPECTED TRAP TO 4/<15><12>
3363 026320          64$:
3364 026320 032737 020000 177570          BIT      #BIT13,@#SWR  ;;IS INHIBIT ERROR TYPEOUT ON?
3365 026326 001045          BNE     1$          ;;BRANCH IF YES
3366 026330 104400 026336          TYPE    , 67$        ;;TYPE ASCIZ STRING
3367 026334 000417          BR      66$          ;;GET OVER THE ASCIZ
3368          ;;67$: .ASCIZ /ERRORPC TSTNUM CPUERR REG/<15><12>
3369 026374          66$:
3370 026374 013737 177766 001172          MOV     @#CPUERR, $TMP0 ;GET CPU ERROR REG
3371 026402 113737 001102 001104          MOVNB  $TSTNM, $TSTNM ;GET TEST NUMBER
    
```

```

3372 026410 013746 001120      MOV      $ERRPC,-(SP)      ;;SAVE $ERRPC FOR TYPEOUT
3373                                ;;TYPE ERROR PC
3374 026414 104402              TYPOC                      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3375 026416 104400 023526      TYPE      ,TWOSP
3376 026422 013746 001104      MOV      $$TSTNM,-(SP)   ;;SAVE $$TSTNM FOR TYPEOUT
3377                                ;;TYPE TEST NUMBER
3378 026426 104402              TYPOC                      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3379 026430 104400 023526      TYPE      ,TWOSP
3380 026434 013746 001172      MOV      $TMP0,-(SP)    ;;SAVE $TMP0 FOR TYPEOUT
3381                                ;;TYPE ERROR REGISTER
3382 026440 104402              TYPOC                      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3383 026442 005037 177766      1$: CLR      @#CPUERR      ;;CLEAR CPU ERROR REG
3384 026446 013737 001316 001210  MOV      NXTTST,$ESCAPE  ;;SETUP ESCAPE ADDRESS
3385 026454 104171              ERROR      171          ;;MAKE THE ERROR CALL TO SYSMAC
3386 026456 011637 001120      CACHSPU:MOV      (SP),$ERRPC ;;SAVE ERROR PC
3387 026462 012706 001100      MOV      #STACK,SP      ;;RESTORE STACK
3388 026466 104400 026474      TYPE      ,65$          ;;TYPE ASCIZ STRING
3389 026472 000415              BR        64$          ;;GET OVER THE ASCIZ
3390                                ;;65$: .ASCIZ /UNEXPECTED TRAP TO 114/<15><12>
3391                                64$:
3392 026526 032737 020000 177570      BIT      #BIT13,@#SWR    ;;IS SWITCH 13 ON?
3393 026534 001045              BNE      1$           ;;BRANCH IF YES
3394 026536 104400 026544      TYPE      ,67$          ;;TYPE ASCIZ STRING
3395 026542 000417              BR        66$          ;;GET OVER THE ASCIZ
3396                                ;;67$: .ASCIZ /ERRORPC TSTNUM MEMERR REG/<15><12>
3397                                66$:
3398 026602 013737 177744 001172      MOV      @#MEMERR,$TMP0  ;;SAVE MEMORY ERROR REG
3399 026610 113737 001102 001104      MOV      $TSTNM,$$TSTNM ;;SAVE TEST NUMBER
3400 026616 013746 001120      MOV      $ERRPC,-(SP)   ;;SAVE $ERRPC FOR TYPEOUT
3401                                ;;TYPE ERROR PC
3402 026622 104402              TYPOC                      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3403 026624 104400 023526      TYPE      ,TWOSP
3404 026630 013746 001104      MOV      $$TSTNM,-(SP)   ;;SAVE $$TSTNM FOR TYPEOUT
3405                                ;;TYPE TEST NUMBER
3406 026634 104402              TYPOC                      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3407 026636 104400 023526      TYPE      ,TWOSP
3408 026642 013746 001172      MOV      $TMP0,-(SP)    ;;SAVE $TMP0 FOR TYPEOUT
3409                                ;;TYPE MEM ERROR REG
3410 026646 104402              TYPOC                      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3411 026650 013737 177744 177744 1$: MOV      @#MEMERR,@#MEMERR ;;CLEAR MEMERR REG.
3412 026656 013737 001316 001210  MOV      NXTTST,$ESCAPE  ;;SETUP ESCAPE ADDRESS
3413 026664 104171              ERROR      171
3414
3415
3416      .SBTTL  ERROR MESSAGE TYPE OUT ROUTINE
3417
3418
3419      :*
3420      :* THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
3421      :* THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
3422      :* AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
3423      :* TABLE STARTS AT '$ERRTB' AND HAS FOUR (4) POINTERS FOR EACH
3424      :* ENTRY, 'EM', 'DH', 'DT', 'DF'. THE 'EM' POINTS TO THE ERROR
3425      :* MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
3426      :* HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
3427      :* DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
3428      :* OF THE DATA TO BY TYPED. THE FORMAT OF THIS DATA IS
    
```

```

3428      : *      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
3429      : *      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
3430      : *      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
3431      : *      0      -16 BIT OCTAL FORMAT
3432      : *      1      -DECIMAL FORMAT
3433      : *      2      -22 BIT OCTAL FORMAT. DATA IS LOWER 16 BITS OF THE
3434      : *      PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
3435      : *      3      -22 BIT OCTAL FORMAT. DATA IS THE 16 BIT VIRTUAL
3436      : *      ADDRESS IN KERNEL I-SPACE.
3437      : *      4      -22 BIT OCTAL FORMAT. DATA IS A P.A.R. AND THE
3438      : *      OUTPUT IS THE BASE ADDRESS THAT THE P.A.R. POINTS TO.
3439 026666 010046      ERTYPE: MOV    R0,-(KSP)      ;SAVE R0 ON STACK
3440 026670 005000      CLR    R0          ;CLEAR R0
3441 026672 113700 001116  MOVB   @#ITEMB,R0    ;PUT ITEM NUMBER IN R0
3442 026676 001004      BNE   1$          ;BRANCH IF IT IS NON-ZERO
3443 026700 013746 001120  MOV    $ERRPC,-(KSP) ;PUT ERROR PC ON STACK FOR TYPING
3444 026704 104402      TYPOC ;TYPE FAILING PC
3445 026706 000551      BR    13$        ;GO TO RETURN
3446 026710 005300      1$: DEC   R0      ;ADJUST ITEM NUMBER TO BE A POINTER
3447 026712 072027 000003  ASH   #3,R0      ;LEFT SHIFT ITEM NO. 3 PLACES
3448 026716 100024      BPL   22$        ;BRANCH IF ITEM #LESS THAN 200
3449 026720 032700 001000  BIT   #BIT9,R0    ;SEE IF ITEM # WAS 3XX
3450 026724 001415      BEQ   21$        ;BRANCH IF ITEM # WAS 2XX
3451      ;AND TYPE ERROR MESSAGE
3452 026726 032737 000200 177570  BIT   #SW7,@#SWR  ;SEE IF SWITCH 7 IS UP
3453 026734 001404      BEQ   20$        ;BRANCH IF SWITCH IS NOT UP
3454      ;AND TYPE DATA, ON MULTIPLE ERRORS
3455 026736 062766 000004 000002  ADD   #4,2(KSP)   ;SKIP 'TYPE $CRLF' IF SW 7 IS UP
3456      ;INHIBIT MULTIPLE ERROR TYPEOUTS
3457 026744 000532      BR    13$        ;BRANCH TO EXIT
3458 026746 042700 177000 20$: BIC   #177000,R0  ;CLEAR UPPER BYTE OF R0
3459 026752 062700 003300  ADD   #ER200+4,R0 ;POINT TO DATA TABLE ENTRY
3460 026756 000424      BR    5$         ;GO TYPE DATA TABLE
3461 026760 042700 177000 21$: BIC   #177000,R0  ;CLEAR UPPER BYTE OF R0
3462 026764 062700 001710  ADD   #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
3463      ;ITEM 1 AND ITEM 201
3464      ::          GET POINTER TO ERROR MESSAGE AND TYPE IT
3465      ::          IF THE POINTER IS NOT ZERO
3466 026770 062700 001364 22$: ADD   #$ERRTB,R0 ;ADD BASE OF ERROR TABLE
3467 026774 012037 027004  MOV    (R0)+,2$   ;PUT MESSAGE POINTER IN TYPE STATEMENT
3468 027000 001404      BEQ   3$         ;BRANCH IF NO ERROR MESSAGE
3469 027002 104400      TYPE ;TYPE ERROR MESSAGE
3470 027004 000000      .WORD 0          ;POINTER TO ERROR MESSAGE
3471 027006 104400 001217  TYPE   ,$CRLF    ;TYPE CRLF
3472      ::          GET THE POINTER TO THE DATA HEADER AND
3473      ::          TYPE IT IF THE POINTER IS NOT ZERO
3474 027012 012037 027022 3$: MOV    (R0)+,4$   ;PUT HEADER POINTER IN TYPE STATEMENT
3475 027016 001404      BEQ   5$         ;BRANCH IF NO DATA HEADER
3476 027020 104400      TYPE ;TYPE THE DATA HEADER
3477 027022 000000      .WORD 0          ;POINTER TO DATA HEADER
3478 027024 104400 001217  TYPE   ,$CRLF    ;TYPE CRLF
3479      ::          THIS IS THE START OF THE DATA OUTPUT IF THE
3480      ::          DATA POINTER IS NOT ZERO. R0 POINTS TO THE
3481      ::          DATA FORMAT, R1 POINTS TO THE ADDRESS OF
3482      ::          THE DATA WORDS.
3483 027030 010146      5$: MOV    R1,-(KSP) ;SAVE R1 ON THE STACK

```



```

3484 027032 012001      MOV      (R0)+,R1      ;PUT DATA TABLE POINTER IN R1
3485 027034 001475      BEQ      12$          ;BRANCH IF NO DATA TABLE
3486 027036 012000      MOV      (R0)+,R0      ;PICK UP DATA FORMAT POINTER
3487 027040 105710      6$:     TSTB      (R0)      ;IS THIS WORD OCTAL
3488 027042 001003      BNE      7$          ;BRANCH IF NOT 16-BIT OCTAL
3489          ::          THIS IS 16 BIT OCTAL FORMAT (DF = 0)
3490 027044 013146      MOV      @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
3491 027046 104402      TYPOC          ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
3492 027050 000461      BR      11$         ;GET READY FOR NEXT WORD
3493 027052 122710 000001 7$:     CMPB      #1,(R0)      ;IS THE WORD DECIMAL
3494 027056 001003      BNF      8$          ;BRANCH IF NOT DECIMAL
3495          ::          THIS IS DECIMAL FORMAT (DF = 1)
3496 027060 013146      MOV      @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
3497 027062 104410      TYPDS          ;TYPE THE WORD ON STACK AS DECIMAL
3498 027064 000453      BR      11$         ;GET READY FOR NEXT WORD
3499 027066 122710 000002 8$:     CMPB      #2,(R0)      ;IS WORD 22-BIT PHYSICAL ADDRESS
3500 027072 001012      BNE      9$          ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
3501          ::          THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
3502 027074 012146      MOV      (R1)+,-(KSP)  ;PUT ADDR OF LOW WORD ON STACK
3503 027076 004737 031144  JSR      PC,$DB20     ;CONVERT NUMBER TO OCTAL ASCIZ
3504 027102 062716 000003  ADD      #3,(KSP)      ;ONLY WANT 8 DIGITS
3505 027106 012637 027114  MOV      (KSP)+,30$    ;PUT POINTER AFTER 'TYPE' CALL
3506 027112 104400      TYPE          ;TYPE ASCIZ STRING
3507 027114 000000      .WORD      0          ;WORD HOLDS POINTER TO ASCIZ STRING
3508 027116 000436      BR      11$         ;GET READY FOR NEXT WORD
3509 027120 122710 000003 9$:     CMPB      #3,(R0)      ;IS THIS A 16-BIT VIRTUAL ADDRESS
3510 027124 001004      BNE      10$         ;BRANCH IF NOT 16-BIT VIRT. ADDR.
3511          ::          THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
3512          ::          KERNEL I-SPACE ASSUMED. (DF = 3)
3513 027126 013146      MOV      @ (R1)+,-(KSP) ;PUT 16-BIT VIRTUAL ADDR ON STACK
3514 027130 004737 027242  JSR      PC,TYPVAD     ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
3515 027134 000427      BR      11$         ;GET READY FOR NEXT WORD
3516          ::          THIS IS FORMAT 4. DATA WORD IS A P.A.R.
3517          ::          OUTPUT WILL BE 22-BIT. PAR LEFT SHIFTED 6.
3518 027136 010246      10$:    MOV      R2,-(KSP)     ;SAVE R2 ON THE STACK
3519 027140 010346      MOV      R3,-(KSP)     ;SAVE R3 ON THE STACK
3520 027142 013103      MOV      @ (R1)+,R3     ;LOAD DATA WORD INTO R3
3521 027144 005002      CLR      R2            ;R2 WILL HOLD UPPER SIX BITS OF NUMBER
3522 027146 073227 000006  ASHC      #6,R2         ;LEFT SHIFT <R2:R3> 6 PLACES
3523 027152 010237 001232  MOV      R2,PADRSR      ;STORE UPPER BITS OF ADDRESS
3524 027156 010337 001230  MOV      R3,PADRSL      ;STORE LOWER 16 BITS OF ADDRESS
3525 027162 012746 001230  MOV      #PADRSL,-(KSP) ;PUT ADDRESS OF LOWER 16 BITS ON STACK
3526 027166 004737 031144  JSR      PC,$DB20     ;CONVERT TWO WORDS TO ASCIZ
3527 027172 062716 000003  ADD      #3,(KSP)      ;I ONLY WANT 8 DIGITS
3528 027176 012637 027204  MOV      (KSP)+,31$    ;PUT POINTER AFTER TYPE CALL
3529 027202 104400      TYPE          ;POINTER TO ASCIZ STRING FOLLOWS
3530 027204 000000      .WORD      0          ;POINTER TO ASCIZ STRING
3531 027206 011603      MOV      (KSP),+R3     ;RESTORE R3 FROM STACK
3532 027210 012602      MOV      (KSP)+,R2     ;RESTORE R2 FROM STACK
3533 027212 000400      BR      11$         ;GET READY FOR NEXT WORD
3534 027214 005200      11$:    INC      R0            ;POINT TO NEXT FORMAT BYTE
3535 027216 104400 027236  TYPE      ,32$         ;TYPE TWO SPACES
3536 027222 005711      TST      (R1)         ;IS THERE ANOTHER WORD?
3537 027224 001401      BEQ      12$         ;BRANCH IF ALL DONE
3538 027226 000704      BR      6$           ;GO BACK FOR NEXT NUMBER
3539 027230 012601      12$:    MOV      (KSP)+,R1     ;RESTORE R1

```

3540 027232 012600
 3541 027234 000207
 3542 027236 020040 000
 3543 027242
 3544
 3545
 3546
 3547
 3548
 3549
 3550
 3551
 3552
 3553
 3554
 3555
 3556 027242 104412
 3557 027244 016601 000002
 3558 027250 005000
 3559 027252 073027 000003
 3560 027256 006300
 3561 027260 006001
 3562 027262 006001
 3563 027264 006001
 3564 027266 062700 172340
 3565 027272 011003
 3566 027274 005002
 3567 027276 073227 000006

```

13$: MOV (KSP)+,R0 ;RESTORE R0
RTS PC ;RETURN TO ERROR ROUTINE
32$: .ASCIZ ? ? ;TWO SPACES
.EVEN

.SBTTL CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
:*
:* THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
:* IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO
:* R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
:* CORRECT KERNEL I-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL
:* ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
:* AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
:* TO ASCIZ AND TYPED.
:*
TYPVAD: SAVREG ;SAVE ALL REGISTERS
MOV 2(KSP),R1 ;PUT VIRTUAL ADDR IN R1
CLR R0 ;CLEAR R0 FOR CALCULATIONS
ASHC #3,R0 ;LEFT SHIFT R0,R1 3 PLACES
ASL R0 ;LEFT SHIFT R0 ONE MORE PLACE
ROR R1 ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
ROR R1 ;RIGHT SHIFT R1
ROR R1 ;RIGHT SHIFT R1
ADD #KIPAR0,R0 ;FORM DESIRED PAR ADDR IN R0
MOV (R0),R3 ;PUT CONTENTS OF PAR IN R3
CLR R2 ;CLEAR R2 FOR PHYSICAL ADDR CALCULATIONS
ASHC #6,R2 ;LEFT SHIFT <R2,R3> 6 PLACES
    
```

```
3568 027302 060103      ADD      R1,R3      ;ADD OFFSET IN R1 TO BASE IN R3
3569 027304 005502      ADC      R2          ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
3570 027306 010237 001232  MOV      R2,PADRSR  ;PUT UPPER 6 BITS OF ADDR IN CORE
3571 027312 010337 001230  MOV      R3,PADRSL  ;PUT LOWER 16 BITS OF ADDR IN CORE
3572 027316 012746 001230  MOV      #PADRSL,-(KSP) ;PUT POINTER TO LOWER 16 BITS ON STACK
3573 027322 004737 031144  JSR      PC,$DB20   ;CONVERT NUMBER TO OCTAL ASCIZ
3574 027326 062716 000003  ADD      #3,(KSP)   ;ONLY TYPE 8 DIGITS
3575 027332 012637 027340  MOV      (KSP)+,3$  ;PUT POINTER AFTER TYPE INST
3576 027336 1044C0          TYPE          ;TYPE THE 22-BIT VIRTUAL ADDRESS
3577 027340 000000          3$: .WORD 0      ;THIS WORD HOLDS THE POINTER TO
3578                                ;THE ASCIZ STRING
3579 027342 104414          RESREG       ;RESTORE ALL THE REGISTERS
3580 027344 012616          MOV      (KSP)+,(KSP) ;LEAVE ONLY RETURN ADDR ON STACK
3581 027346 000207          RTS      PC        ;RETURN TO ERROR HANDLER
```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```
;*SAVE R0-R5
;*CALL:
;* SAVREG
;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;*
;*TOP---(+16)
;* +2---(+18)
;* +4---R5
;* +6---R4
;* +8---R3
;*+10---R2
;*+12---R1
;*+14---R0
```

\$SAVREG:

```
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
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
MOV      22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
MOV      22(SP),-(SP)  ;;SAVE PS OF CALL
MOV      22(SP),-(SP)  ;;SAVE PC OF CALL
RTI
```

;*RESTORE R0-R5

;*CALL:

;* RESREG

\$RESREG:

```
MOV      (SP)+,22(SP)  ;;RESTORE PC OF CALL
MOV      (SP)+,22(SP)  ;;RESTORE PS OF CALL
MOV      (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
MOV      (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
MOV      (SP)+,R5      ;;POP STACK INTO R5
```

```
3602 027350
3603 027350 010046
3604 027352 010146
3605 027354 010246
3606 027356 010346
3607 027360 010446
3608 027362 010546
3609 027364 016646 000022
3610 027370 016646 000022
3611 027374 016646 000022
3612 027400 016646 000022
3613 027404 000002
3614
3615
3616
3617
3618 027406
3619 027406 012666 000022
3620 027412 012666 000022
3621 027416 012666 000022
3622 027422 012666 000022
3623 027426 012605
```

```

3624 027430 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
3625 027432 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
3626 027434 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
3627 027436 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
3628 027440 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
3629 027442 000002      RTI
3630
3631
3632      .SBTTL  TYPE ROUTINE
3633
3634      ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
3635      ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3636      ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3637      ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3638      ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3639
3640      ;*CALL:
3641      ;*1) USING A TRAP INSTRUCTION
3642      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3643      ;*OR
3644      ;*      TYPE
3645      ;*      MESADR
3646
3647      ;*2) USING A JSR INSTRUCTION
3648      ;*      MOV      PS,-(SP)      ;;PUSH PROCESSOR STATUS WORD ON THE STACK
3649      ;*      JSR      PC,$TYPE      ;;CALL TYPE ROUTINE
3650      ;*      MESADDR      ;;FIRST ADDRESS OF MESSAGE
3651
3652 027444 105737 001153      $TYPE:  TSTB      $TFPLG      ;;IS THERE A TERMINAL?
3653 027450 100002      BPL      1$      ;;BR IF YES
3654 027452 000000      HALT      ;;HALT HERE IF NO TERMINAL
3655 027454 000407      BR      3$      ;;LEAVE
3656 027456 010046      1$:      MOV      R0,-(SP)      ;;SAVE R0
3657 027460 017600 000002      MOV      @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
3658 027464 112046      2$:      MOVVB     (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
3659 027466 001005      BNE      4$      ;;BR IF IT ISN'T THE TERMINATOR
3660 027470 005726      TST      (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
3661 027472 012600      MOV      (SP)+,R0      ;;RESTORE R0
3662 027474 062716 000002      3$:      ADD      #2,(SP)      ;;ADJUST RETURN PC
3663 027500 000002      RTI      ;;RETURN
3664 027502 122716 000011      4$:      CMPB     #HT,(SP)      ;;BRANCH IF <HT>
3665 027506 001424      BEQ      8$
3666 027510 122716 000200      CMPB     #CRLF,(SP)      ;;BRANCH IF NOT
3667 027514 001004      BNE      5$
3668 027516 005726      TST      (SP)+      ;;POP <CR><LF> EQUIV
3669 027520 104400 001217      TYPE     ,$CRIF
3670 027524 000757      BR      2$      ;;GET NEXT CHARACTER
3671 027526 004737 027604      5$:      JSR      PC,$TYPEC      ;;GO TYPE THIS CHARACTER
3672 027532 123726 001152      6$:      CMPB     $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
3673 027536 001352      BNE      2$      ;;IF NO GO GET NEXT CHAR.
3674 027540 013746 001150      MOV      $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
3675
3676 027544 105366 000001      7$:      DECB     1(SP)      ;;AND THE NULL CHAR.
3677 027550 002770      BLT      6$      ;;DOES A NULL NEED TO BE TYPED?
3678 027552 004737 027604      JSR      PC,$TYPEC      ;;BR IF NO--GO POP THE NULL OFF OF STACK
3679 027556 000772      BR      7$      ;;GO TYPE A NULL
                          ;;LOOP

```

```

3680
3681      ;;HORIZONTAL TAB PROCESSOR
3682
3683 027560 112716 000040      8$:   MOVB   #' (SP)      ;;REPLACE TAB WITH SPACE
3684 027564 004737 027604      9$:   JSR    PC,$TYPEC      ;;TYPE A SPACE
3685 027570 132737 000007 027650      BITB   #7,$SCHARCNT      ;;BRANCH IF NOT AT
3686 027576 001372           BNE    9$                ;;TAB STOP
3687 027600 005726           TST    (SP)+            ;;POP SPACE OFF STACK
3688 027602 000730           BR     2$                ;;GET NEXT CHARACTER
3689 027604 105777 151334      $TYPEC: TSTB   @ $TPS      ;;WAIT UNTIL PRINTER IS READY
3690 027610 100375           BPL    $TYPEC
3691 027612 116677 000002 151326      MOVB   2(SP),@ $TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3692 027620 122766 000015 000002      CMPB   #CR,2(SP)        ;;BRANCH IF
3693 027626 001003           BNE    1$                ;;NOT <CR>
3694 027630 105037 027650      CLRB   $SCHARCNT
3695 027634 000406           BR     $TYPEX            ;;EXIT
3696 027636 122766 000012 000002 1$:   (MPB   #LF,2(SP)        ;;BRANCH IF
3697 027644 001402           BEQ   $TYPEX            ;;<LF>
3698 027646 105227           INCB   (PC)+            ;;INC SPACE
3699 027650 000000      $SCHARCNT: .WORD 0      ;;COUNT
3700 027652 000207      $TYPEX: RTS    PC
3701
3702      ;;*****
3703
3704      .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
3705
3706      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3707      ;*OCTAL (ASCII) NUMBER AND TYPE IT.
3708      ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3709      ;*CALL:
3710      ;*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
3711      ;*   TYPOS      ;;CALL FOR TYPEOUT
3712      ;*   .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3713      ;*   .BYTE   M              ;;M=1 OR 0
3714      ;*                                     ;;1=TYPE LEADING ZEROS
3715      ;*                                     ;;0=SUPPRESS LEADING ZEROS
3716
3717      ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3718      ;*$TYPOS OR $TYPOC
3719      ;*CALL:
3720      ;*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
3721      ;*   TYPON      ;;CALL FOR TYPEOUT
3722
3723      ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3724      ;*CALL:
3725      ;*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
3726      ;*   TYPOC      ;;CALL FOR TYPEOUT
3727
3728 027654 017646 000000      $TYPOS: MOV    @ (SP),-(SP)      ;;PICKUP THE MODE
3729 027660 116637 000001 030077      MOVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
3730 027666 112637 030101      MOVB   (SP)+,$SOMODE+1    ;;NUMBER OF DIGITS TO TYPE
3731 027672 062716 000002      ADD    #2,(SP)           ;;ADJUST RETURN ADDRESS
3732 027676 000406           BR     $TYPON
3733 027700 112737 000001 030077      $TYPOC: MOVB   #1,$OFILL      ;;SET THE ZERO FILL SWITCH
3734 027706 112737 000006 030101      MOVB   #6,$SOMODE+1      ;;SET FOR SIX(6) DIGITS
3735 027714 112737 000005 030076      $TYPON: MOVB   #5,$OCNT      ;;SET THE ITERATION COUNT
    
```

```

3736 027722 010346      MOV      R3,-(SP)      ;;SAVE R3
3737 027724 010446      MOV      R4,-(SP)      ;;SAVE R4
3738 027726 010546      MOV      R5,-(SP)      ;;SAVE R5
3739 027730 113704 030101  MOVB    $OMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
3740 027734 005404      NFG      R4
3741 027736 062704 000006  ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
3742 027742 110437 030100  MOVB    R4,$OMODE      ;;SAVE IT FOR USE
3743 027746 113704 030077  MOVB    $OFILL,R4      ;;GET THE ZERO FILL SWITCH
3744 027752 016605 000012  MOV     12(SP),R5      ;;PICKUP THE INPUT NUMBER
3745 027756 005003      CLR     R3            ;;CLEAR THE OUTPUT WORD
3746 027760 006105      1$:    ROL     R5            ;;ROTATE MSB INTO 'C'
3747 027762 000404      BR      3$           ;;GO DO MSB
3748 027764 006105      2$:    ROL     R5            ;;FORM THIS DIGIT
3749 027766 006105      ROL     R5
3750 027770 006105      ROL     R5
3751 027772 010503      MOV     R5,R3
3752 027774 006103      3$:    ROL     R3            ;;GET LSB OF THIS DIGIT
3753 027776 105337 030100  DECB    $OMODE        ;;TYPE THIS DIGIT?
3754 030002 100016      BPL     7$           ;;BR IF NO
3755 030004 042703 177770  BIC     #177770,R3     ;;GET RID OF JUNK
3756 030010 001002      BNE     4$           ;;TEST FOR 0
3757 030012 005704      TST     R4            ;;SUPPRESS THIS 0?
3758 030014 001403      BEQ     5$           ;;BR IF YES
3759 030016 005204      4$:    INC     R4            ;;DON'T SUPPRESS ANYMORE 0'S
3760 030020 052703 000060  BIS     #'0,R3        ;;MAKE THIS DIGIT ASCII
3761 030024 052703 000040  5$:    BIS     #' ,R3      ;;MAKE ASCII IF NOT ALREADY
3762 030030 110337 030074  MOVB    R3,8$         ;;SAVE FOR TYPING
3763 030034 104400 030074  TYPE    ,8$          ;;GO TYPE THIS DIGIT
3764 030040 105337 030076  7$:    DECB    $OCNT      ;;COUNT BY 1
3765 030044 003347      BGT     2$           ;;BR IF MORE TO DO
3766 030046 002402      BLT     6$           ;;BR IF DONE
3767 030050 005204      INC     R4            ;;INSURE LAST DIGIT ISN'T A BLANK
3768 030052 000744      BR      2$           ;;GO DO THE LAST DIGIT
3769 030054 012605      6$:    MOV     (SP)+,R5     ;;RESTORE R5
3770 030056 012604      MOV     (SP)+,R4     ;;RESTORE R4
3771 030060 012603      MOV     (SP)+,R3     ;;RESTORE R3
3772 030062 016666 000002 000004  MOV     2(SP),4(SP)   ;;SET THE STACK FOR RETURNING
3773 030070 012616      MOV     (SP)+,(SP)
3774 030072 000002      RTI
3775 030074 000      8$:    .BYTE 0            ;;RETURN
3776 030075 000      .BYTE 0            ;;STORAGE FOR ASCII DIGIT
3777 030076 000      $OCNT: .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
3778 030077 000      $OFILL: .BYTE 0     ;;OCTAL DIGIT COUNTER
3779 030100 000000  $OMODE: .WORD 0      ;;ZERO FILL SWITCH
;;*****
;;*****
.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;*REPLACED WITH SPACES.
;*(ALL:
;*      MOV     NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS      ;;GO TO THE ROUTINE
    
```

```

3792
3793 030102          $TYPDS:
3794 030102 010046  MOV      R0,-(SP)      ;;PUSH R0 ON STACK
3795 030104 010146  MOV      R1,-(SP)      ;;PUSH R1 ON STACK
3796 030106 010246  MOV      R2,-(SP)      ;;PUSH R2 ON STACK
3797 030110 010346  MOV      R3,-(SP)      ;;PUSH R3 ON STACK
3798 030112 010546  MOV      R5,-(SP)      ;;PUSH R5 ON STACK
3799 030114 012746 020200  MOV      #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
3800 030120 016605 000020  MOV      20(SP),R5     ;;GET THE INPUT NUMBER
3801 030124 100004          BPL      1$           ;;BR IF INPUT IS POS.
3802 030126 005405          NEG      R5           ;;MAKE THE BINARY NUMBER POS.
3803 030130 112766 000055 00000'  MOVB    #'-',1(SP)    ;;MAKE THE ASCII NUMBER NEG.
3804 030136 005000          1$:  CLR      R0           ;;ZERO THE CONSTANTS INDEX
3805 030140 012703 030316  MOV      #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
3806 030144 112723 000040  MOVB    #'',(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
3807 030150 005002          2$:  CLR      R2           ;;CLEAR THE BCD NUMBER
3808 030152 016001 030306  MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
3809 030156 160105          3$:  SUB      R1,R5       ;;FORM THIS BCD DIGIT
3810 030160 002402          BLT     4$           ;;BR IF DONE
3811 030162 005202          INC     R2           ;;INCREASE THE BCD DIGIT BY 1
3812 030164 000774          BR      3$
3813 030166 060105          4$:  ADD      R1,R5       ;;ADD BACK THE CONSTANT
3814 030170 005702          TST     R2           ;;CHECK IF BCD DIGIT=0
3815 030172 001002          BNE     5$           ;;FALL THROUGH IF 0
3816 030174 105716          TSTB   (SP)          ;;STILL DOING LEADING 0'S?
3817 030176 100407          BMI     7$           ;;BR IF YES
3818 030200 106316          5$:  ASLB   (SP)          ;;MSD?
3819 030202 103003          BCC     6$           ;;BR IF NO
3820 030204 116663 000001 177777  MOVB    1(SP),-1(R3)   ;;YES--SET THE SIGN
3821 030212 052702 000060  6$:  BIS     #'0,R2       ;;MAKE THE BCD DIGIT ASCII
3822 030216 052702 000040  7$:  BIS     #' ,R2       ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
3823 030222 110223          MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3824 030224 005720          TST    (R0)+        ;;JUST INCREMENTING
3825 030226 020027 000010  CMP     R0,#10       ;;CHECK THE TABLE INDEX
3826 030232 002746          BLT    2$           ;;GO DO THE NEXT DIGIT
3827 030234 003002          BGT    8$           ;;GO TO EXIT
3828 030236 010502          MOV    R5,R2        ;;GET THE LSD
3829 030240 000764          BR     6$           ;;GO CHANGE TO ASCII
3830 030242 105726          8$:  TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
3831 030244 100003          BPL    9$           ;;BR IF NO
3832 030246 116663 177777 177776  MOVB   -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
3833 030254 105013          9$:  CLRB   (R3)        ;;SET THE TERMINATOR
3834 030256 012605          MOV    (SP)+,R5     ;;POP STACK INTO R5
3835 030260 012603          MOV    (SP)+,R3     ;;POP STACK INTO R3
3836 030262 012602          MOV    (SP)+,R2     ;;POP STACK INTO R2
3837 030264 012601          MOV    (SP)+,R1     ;;POP STACK INTO R1
3838 030266 012600          MOV    (SP)+,R0     ;;POP STACK INTO R0
3839 030270 104400 030316  TYPE    $DBLK        ;;NOW TYPE THE NUMBER
3840 030274 016666 000002 000004  MOV    2(SP),4(SP)   ;;ADJUST THE STACK
3841 030302 012616          MOV    (SP)+,(SP)
3842 030304 000002          RTI                    ;;RETURN TO USER
3843 030306 023420          $DTBL: 10000.
3844 030310 001750          1000.
3845 030312 000144          100.
3846 030314 000012          10.
3847 030316 000004          $DBLK: .BLKW 4
    
```

```
3848 ;:*****
3849
3850 .SBTTL TRAP DECODER
3851
3852 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
3853 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3854 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3855 ;*GO TO THAT ROUTINE.
3856
3857 030326 010046          $TRAP: MOV     R0,-(SP)          ;;SAVE R0
3858 030330 016600 000002  MOV     2(SP),R0          ;;GET TRAP ADDRESS
3859 030334 005740          TST     -(R0)             ;;BACKUP BY 2
3860 030336 111000          MOVB   (R0),R0           ;;GET RIGHT BYTE OF TRAP
3861 030340 016000 030346  MOV     $TRPAD(R0),R0    ;;INDEX TO TABLE
3862 030344 000200          RTS      R0              ;;GO TO ROUTINE
3863
3864
3865 .SBTTL TRAP TABLE
3866
3867 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3868 ;*BY THE 'TRAP' INSTRUCTION.
3869
3870 ;          ROUTINE
3871 ;          -----
3872 030346          $TRPAD:
3873 030346 027444          $TYPE  ;;CALL=TYPE      TRAP+0(104400) TTY TYPEOUT ROUTINE
3874 030350 027700          $TYPOC ;;CALL=TYPOC    TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3875 030352 027654          $TYPOS ;;CALL=TYPOS    TRAP+4(104404) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3876 030354 027714          $TYPON ;;CALL=TYPON     TRAP+6(104406) TYPE OCTAL NUMBER (AS PER LAST CALL)
3877 030356 030102          $TYPDS ;;CALL=TYPDS   TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
3878 030360 027350          $$SAVREG ;;CALL=SAVREG  TRAP+12(104412) SAVE R0-R5 ROUTINE
3879 030362 027406          $RESREG ;;CALL=RESREG  TRAP+14(104414) RESTORE R0-R5 ROUTINE
3880 030364 031264          TBITOFF ;;CALL=TBITO   TRAP+16(104416) THIS WILL TURN OFF T BIT TRAPPING
3881 030366 031312          TBITRESTORE ;;CALL=TBITR  TRAP+20(104420) THIS WILL RETURN THE T BIT TO PR
3882
3883 ;:*****
3884
3885 .SBTTL POWER DOWN AND UP ROUTINES
3886
3887 ;POWER DOWN ROUTINE
3888 030370 012737 030516 000024 $PWRDN: MOV     #$ILLUP,@PWRVEC ;;SET FOR FAST UP
3889 030376 012737 000340 000026  MOV     #340,@PWRVEC+2 ;;PRIO:7
3890 030404 010046          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3891 030406 010146          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3892 030410 010246          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3893 030412 010346          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3894 030414 010446          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3895 030416 010546          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
3896 030420 010637 030522          MOV     SP,$SAVR6      ;;SAVE SP
3897 030424 012737 030436 000024  MOV     #$PWRUP,@PWRVEC ;;SET UP VECTOR
3898 030432 000000          HALT
3899 030434 000776          BR      -2             ;;HANG UP
3900
3901 ;POWER UP ROUTINE
3902 030436 013706 030522          $PWRUP: MOV     $SAVR6,SP ;;GET SP
3903 030442 005037 030522          CLR     $SAVR6        ;;WAIT LOOP FOR THE TTY
```


K 7

```
3904 030446 005237 030522 1$: INC $SAVR6 ;;WAIT FOR THE INC
3905 030452 001375 BNE 1$ ;;OF WORD
3906 030454 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
3907 030456 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
3908 030460 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
3909 030462 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
3910 030464 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
3911 030466 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
3912 030470 012737 030370 000024 MOV #SPWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
3913 030476 012737 000340 000026 MOV #3%0,@PWRVEC+2 ;;PRIO:7
3914 030504 104400 TYPE ;;REPORT THE POWER FAILURE
3915 030506 030524 $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
3916 030510 012716 MOV (PC)+,(SP) ;;RESTART AT START
3917 030512 040076 $PWRAD: .WORD START ;;RESTART ADDRESS
3918 030514 000002 RTI
3919 030516 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
3920 030520 000776 BR .-2 ;; BEFORE THE POWER DOWN WAS COMPLETE
3921 030522 000000 $SAVR6: 0 ;;PUT THE SP HERE
3922 030524 006412 047520 042527 PWRMSG: .ASCIZ <12><15>?POWER FAILURE, RESTARTING PROGRAM?
3923 030532 020122 040506 046111
3924 030540 051125 026105 051040
3925 030546 051505 040524 052122
3926 030554 047111 020107 051120
3927 030562 043517 040522 000115
3928
3929
3930 .SBTTL ROUTINE TO SIZE MEMORY
3931
3932 ;*CALL:
3933 ;* JSR PC,$SIZE
3934 ;* RETURN
3935 ;*$LSTAD WILL CONTAIN:
3936 ;* WITH KT11 OPTION -- LAST VIRTUAL ADDRESS OF THE LAST BANK
3937 ;* WITHOUT KT11 OPTION -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
3938 ;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
3939 ;*$KT11 IS THE MEMORY MANAGEMENT KEY
3940 ;*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
3941 ;* MUST BE SETUP BEFORE THE CALL
3942 ;*BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
3943 ;* DETERMINED BY ROUTINE
3944 ;* --NOTE--
3945 ;*THIS ROUTINE SUPPORTS PDP 11/74.
3946 ;*IF ACTUAL MEMORY IS LESS THAN THAT INDICATED BY SIZE REGISTER
3947 ;*AND A REFERENCE IS MADE TO A MEMORY ADDRESS THAT IS GREATER THAN
3948 ;*ACTUAL MEMORY BUT LESS THAN SIZE REGISTER (INDICATED), THEN A
3949 ;*MEMORY REFERENCE TIMEOUT TO VECTOR 114 WILL OCCUR.
3950
3951 030570 010046 $SIZE: MOV R0,-(SP) ;;SAVE R0 ON THE STACK
3952 030572 010146 MOV R1,-(SP) ;;SAVE R1 ON THE STACK
3953 030574 010246 MOV R2,-(SP) ;;SAVE R2 ON THE STACK
3954 030576 010346 MOV R3,-(SP) ;;SAVE R3 ON THE STACK
3955 030600 013746 000004 MOV @ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
3956 030604 013746 000006 MOV @ERRVEC+2,-(SP)
3957 030610 013746 000114 MOV @114,-(SP) ;;SAVE PRESENT PARITY VECTOR PS &PC
3958 030614 013746 000116 MOV @116,-(SP)
3959 030620 010600 MOV SP,R0 ;;SAVE THE STACK POINTER
```

```

3960 030622 013737 177776 000006      MOV      @#PS,@#ERRVEC+2  ;;SET ERRVEC PS TO PRESENT PS
3961 030630 012701 003776              MOV      #3776,R1        ;;SETUP ADDRESS
3962 030634 105727              TSTB     (PC)+           ;;USE MEMORY MANAGEMENT?
3963 030636 000200      $K111: .WORD    200      ;;SET TO USE MEMORY MANAGEMENT
3964 030640 100060              BPL      $SCORE         ;;BR IF NO
3965 030642 012737 030774 000004      MOV      #S$KTNEX,@#ERRVEC ;;SET FOR TIMEOUT
3966 030650 005737 177572              TST      @#SRO          ;;KT11 ARE YOU THERE?
3967 030654 052737 100000 030636      BIS      #100000,$K111   ;;YES--SET KT11 KEY
3968 030662 005046              CLR      -(SP)          ;;INITIALIZE FOR 'PAR' LOADING
3969 030664 012702 172340      MOV      #KIPAR0,R2     ;;ADDRESS OF FIRST 'PAR'
3970 030670 012703 000010      MOV      #^D8,R3        ;;LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'
3971 030674 012762 077406 177740 1$:   MOV      #77406,-40(R2)  ;;PDR = 4K, UP, READ/WRITE
3972 030702 011622              MOV      (SP),(R2)+     ;;LOAD 'PAR'
3973 030704 062716 000200      ADD      #200,(SP)      ;;UPDATE FOR NEXT 'PAR'
3974 030710 077307              SOB      R3,1$         ;;LOOP UNTIL ALL EIGHT ARE LOADED
3975 030712 012742 177600      MOV      #177600,-(R2)  ;;SETUP KIPAR7 FOR I/O
3976 030716 005042              CLR      -(R2)         ;;SETUP KIPAR6 FOR TESTING
3977 030720 012737 000020 172516      MOV      #20,@#SR3     ;;ENABLE 22-BIT ADDRESSING
3978 030726 005237 177572              INC      @#SRO          ;;TURN ON MEMORY MANAGEMENT
3979 030732 012737 030764 000004      MOV      #S$KTOUT,@#ERRVEC ;;SET FOR TIME OUT
3980 030740 012737 031106 000114      MOV      #S$MTMOUT,@#114 ;;SET FOR MEM REF TIMEOUT
3981 030746 005737 143776      2$:   TST      @#143776     ;;TRAP ON NON-EX-MEM
3982 030752 062712 000040      ADD      #40,(R2)      ;;MAKE A 1K STEP
3983 030756 023712 172356      CMP      @#KIPAR7,(R2)  ;;LAST ONE?
3984 030762 101371              BHI      2$            ;;NO--TRY IT
3985 030764 011202      $KTOUT: MOV      (R2),R2       ;;GET LAST BANK+1
3986 030766 005037 177572              CLR      @#SRO          ;;TURN OFF MEMORY MANAGEMENT
3987 030772 000421              BR       $SIZEX
3988 030774 042737 100000 030636 $KTNEX: BIC      #100000,$K111 ;;KT11 NON-EXISTENT
3989 031002 012737 031032 000004 $SCORE: MOV      #S$CROUT,@#ERRVEC ;;SET FOR TIMEOUT
3990 031010 005002              CLR      R2            ;;SET UP BANK
3991 031012 062701 004000      1$:   ADD      #4000,R1     ;;INCREMENT BY 1K
3992 031016 062702 000040      ADD      #40,R2        ;;1K STEP
3993 031022 005711              TST      (R1)          ;;TRAP ON TIME OUT
3994 031024 022701 177776      CMP      #177776,R1    ;;LAST ONE
3995 031030 001370              BNE      1$           ;;NO--TRY AGAIN
3996 031032 162701 004000      $CROUT: SUB     #4000,R1
3997 031036 162702 000040      $SIZEX: SUB     #40,R2  ;;DROP BACK
3998 031042 010006              MOV      R0,SP         ;;RESTORE THE STACK
3999 031044 012637 000116      MOV      (SP)+,@#116   ;;RESTORE PARITY VECTOR
4000 031050 012637 000114      MOV      (SP)+,@#114
4001 031054 012637 000006      MOV      (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
4002 031060 012637 000004      MOV      (SP)+,@#ERRVEC
4003 031064 010137 031140      MOV      R1,$LSTAD     ;;LAST ADDRESS
4004 031070 010237 031142      MOV      R2,$LSTBK     ;;LAST BANK
4005 031074 012603              MOV      (SP)+,R3     ;;RESTORE R3
4006 031076 012602              MOV      (SP)+,R2     ;;RESTORE R2
4007 031100 012601              MOV      (SP)+,R1     ;;RESTORE R1
4008 031102 012600              MOV      (SP)+,R0     ;;RESTORE R0
4009 031104 000207              RTS      PC
4010 031106 032737 000001 177744 $MTMOUT: BIT     #BIT0,@#MEMERR ;;MAKE SURE TRAP TO 114 IS
4011 031114 001005              BNE      1$           ;;DUE TO MEM REF TIMEOUT
4012                                ;;IF NOT, IS IT AN ABORT?
4013 031116 032737 100000 177744      BIT     #BIT15,@#MEMERR ;;CPU ABORT?
4014 031124 001001              BNE      1$           ;;IF YES EXIT OUT
4015 031126 000002              RTI                    ;;IF NOT, CONTINUE

```

```

4016 031130 012737 177777 177744 1$:  MOV    #-1,@MEMERR    ;;CLEAR MEM ERROR REG
4017 031136 000712          BR      $KTOUT
4018 031140 000000          $LSTAD: .WORD 0          ;;CONTAINS THE LAST ADDRESS
4019 031142 000000          $LSTBK: .WORD 0          ;;CONTAINS THE LAST BANK
4020          .EVEN
4021
4022          ;;*****
4023
4024          .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
4025
4026          ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
4027          ;*UNSIGNED OCTAL ASCII NUMBER.
4028          ;*CALL
4029          ;*   MOV    #PNTR,-(SP)    ;;POINTER TO LOW WORD OF BINARY NUMBER
4030          ;*   JSR    PC,@#$DB20    ;;CALL THE ROUTINE
4031          ;*   RETURN                ;;THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
4032
4033
4034 031144 104412          $DB20: SAVREG                ;;SAVE ALL REGISTERS
4035 031146 016601 000002  MOV    2(SP),R1            ;;PICKUP THE POINTER TO LOW WORD
4036 031152 012705 031263  MOV    #SOCTVL+13.,R5    ;;POINTER TO DATA TABLE
4037 031156 012704 000014  MOV    #12.,R4           ;;DO ELEVEN CHARACTERS
4038 031162 012703 177770  MOV    #^C7,R3           ;;MASK
4039 031166 012100          MOV    (R1)+,R0          ;;LOWER WORD
4040 031170 012101          MOV    (R1)+,R1          ;;HIGH WORD
4041 031172 005002          CLR    R2                ;;TERMINATOR
4042 031174 110245          1$:  MOVB   R2,-(R5)        ;;PUT CHARACTER IN DATA TABLE
4043 031176 010002          MOV    R0,R2             ;;GET THIS DIGIT
4044 031200 005304          DEC    R4                ;;COUNT THIS CHARACTER
4045 031202 003007          BGT    3$                ;;BR IF NOT THE LAST DIGIT
4046 031204 001405          BEQ    2$                ;;BR IF IT IS THE LAST DIGIT
4047 031206 005205          INC    R5                ;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
4048 031210 010566 000002  MOV    R5,2(SP)          ;;ASCII CHAR. & PUT IT ON THE STACK
4049 031214 104414          RESREG                ;;RESTORE ALL REGISTERS
4050 031216 000207          RTS    PC                ;;RETURN TO USER
4051 031220 006203          2$:  ASR    R3                ;;POSITION THE MASK FOR THE LAST DIGIT
4052 031222 006001          3$:  ROR    R1                ;;POSITION THE BINARY NUMBER FOR
4053 031224 006000          ROR    R0                ;;THE NEXT OCTAL DIGIT
4054 031226 006001          ROR    R1
4055 031230 006000          ROR    R0
4056 031232 006001          ROR    R1
4057 031234 006000          ROR    R0
4058 031236 040302          BIC    R3,R2             ;;MASK OUT ALL JUNK
4059 031240 062702 000060  ADD    #'0,R2            ;;MAKE THIS CHAR. ASCII
4060 031244 000753          BR     1$                ;;GO PUT IT IN THE DATA TABLE
4061 031246 000016          $OCTVL: .BLKB 14.        ;;RESERVE DATA TABLE
4062
4063
4064          .SBTTL  ***** SUBROUTINES UNIQUE TO THIS PROGRAM *****
4065
4066
4067
4068          .SBTTL  TURN OFF AND SAVE T-BIT
4069          ;*
4070          ;*   THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO', IT IS
4071          ;*   USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
    
```

```
4072      :*      IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS
4073      :*      PREVIOUS STATUS WHEN CONDITIONS WARRANT.
4074      :*
4075      .EQUIV BIT4,TBIT      ;T-BIT IS BIT04 IN PROC. STATUS
4076 031264 TBITOFF:
4077 031264 032766 000020 000002 BIT #TBIT,2(KSP) ;IS THE T-BIT ON?
4078 031272 001406 BEQ 1$ ;BRANCH TO EXIT IF IT IS NOT ON
4079 031274 016637 000002 001312 MOV 2(KSP),OLDPSW ;SAVE OLD PSW FOR RESTORING T BIT
4080 031302 042766 000020 000002 BIC #TBIT,2(KSP) ;CLEAR T BIT IF IT IS ON
4081 031310 000006 1$: RTT ;RETURN TO PROGRAM
4082
```

4083
4084
4085
4086
4087
4088
4089
4090
4091
4092
4093 031312
4094 031312 013766 001312 000002
4095 031320 042737 000020 001312
4096
4097 031326 000006
4098
4099
4100
4101
4102
4103
4104
4105
4106
4107
4108
4109
4110
4111
4112
4113
4114
4115
4116
4117 031330 010046
4118 031332 012700 000020
4119 031336 005025
4120 031340 077002
4121 031342 012600
4122 031344 000207
4123
4124
4125
4126

.SBTTL RESTORE T-BIT TO ITS PREVIOUS CONDITION

* THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
* USED TO RESTORE THE T-BIT AFTER A PARTICULAR TEST THAT CANNOT
* BE RUN WITH THE T-BIT ON. IT USES THE PROCESSOR STATUS STORED
* IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
* AND DOES AN 'RTT'.

TBITRESTORE:
MOV OLDPSW,2(KSP) ;PUT OLD PSW ON STACK
BIC #TBIT,OLDPSW ;CLEAR T-BIT IN 'OLDPSW' SO THAT
;IT WON'T BE TURNED ON BY ACCIDENT
RTT ;RETURN TO PROGRAM AND INHIBIT
;T BIT TRAP AFTER THIS INSTRUCTION.

.SBTTL CLEAR 16 PARS OR PDRS STARTING FROM ADDRESS IN R5

* SUBROUTINE CLRREG
* THIS SUBROUTINE CLEARS 16 CONSECUTIVE MEMORY MANAGEMENT
* REGISTERS STARTING WITH THE REGISTER POINTED TO BY R5.
* IT SAVES R0 ON THE STACK AND LOADS A COUNT IN R0,
* CLEARS THE PAR'S OR PDR'S, AND THEN RESTORES R0 BEFORE
* RETURNING.
* THE CALLING SEQUENCE IS:
* MOV #KIPAR0,R5 ;PUT ADDRESS OF FIRST KERNEL PAR IN R5
* JSR PC,CLRREG ;GO CLEAR ALL KERNEL PAR'S

CLRREG: MOV R0,-(KSP) ;SAVE R0 ON STACK
MOV #20,R0 ;PUT COUNT IN R0
1\$: CLR (R5)+ ;CLEAR PAR ORPDR POINTED TO BY R5
SOB R0,1\$;BRANCH BACK 15 DECIMAL TIMES.
MOV (KSP)+,R0 ;RESTORE R0 FROM STACK
RTS PC ;RETURN TO TEST

.SBTTL CLEANUP LOCATIONS THAT HOLD LOGICAL 'AND' AND 'OR'

* SUBROUTINE CLEANUP

4127
4128
4129
4130
4131
4132
4133
4134 031346
4135 031346 005037 001266
4136 031352 005037 001276
4137 031356 005037 001272
4138 031362 012700 177777
4139 031366 010037 001270
4140 031372 010037 001300
4141 031376 010037 001274
4142 031402 000207
4143
4144
4145
4146
4147
4148
4149
4150
4151
4152 031404
4153 031404 005227
4154 031406 177777
4155 031410 001401
4156 031412 000000
4157
4158
4159
4160
4161 031414 012637 001306
4162 031420 012637 001310
4163 031424 013737 177766 001260
4164 031432 013737 001260 177766
4165 031440 023737 001260 001224
4166 031446 001402
4167 031450 104001
4168 031452 000414
4169 031454 050037 001276
4170 031460 005100
4171 031462 040037 001300
4172 031466 005100
4173 031470 105737 001103
4174 031474 001002
4175 031476 104201
4176 031500 000401
4177 031502 104301
4178 031504 012737 177777 031406
4179 031512 013746 001310
4180 031516 013746 001306
4181 031522 000006
4182

.*
.* THIS SUBROUTINE IS USED TO INITIALIZE ALL THE LOCATIONS THAT
.* HOLD THE 'LOGICAL AND' AND 'LOGICAL OR' OF THE DATA AND ADDRESSES
.* THAT FAILED DURING THE EXECUTION OF A TEST.
.*

CLEANUP: ;STARTING ADDRESS OF SUBROUTINE.
CLR DATAOR ;LOCATION FOR LOGICAL OR OF BAD DATA
CLR ADDROR ;LOCATION FOR LOGICAL OR OF ADDRESS
CLR PATTOR ;LOCATION FOR LOGICAL OR OF PATTERN LOADED
MOV #-1,R0 ;LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND LOCS
MOV R0,DATAND ;LOCATION FOR LOGICAL AND OF BAD DATA
MOV R0,ADRAND ;LOCATION FOR LOGICAL AND OF ADDRESS
MOV R0,PATAND ;LOCATION FOR LOGICAL AND OF PATTERN LOADED
RTS PC ;RETURN TO TEST

.SBTTL P.A.R. OR P.D.R. ADDRESS TIMED OUT WHEN REFERENCED

.*
.* THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
.* REFERENCE TO A P.A.R. OR A P.D.R. TIMED OUT ON THE UNIBUS. IT
.* KEEPS A LOGICAL AND AND A LOGICAL OR OF EACH ADDRESS THAT TIMES
.* OUT.
.*

TIMEOUT: ;STARTING ADDRESS OF SUBROUTINE
TOFLAG: INC (PC)+ ;INCREMENT ONE TIME GATE
 .WORD -1 ;ONE TIME ENTANCE FLAG
 BEQ 10\$;BRANCH IF FLAG IS NOW ZERO
 HALT ;I HAVE ENTERED THIS ROUTINE BEFORE
 ;I FINISHED REPORTING THE FIRST ERROR
 ;THE SECOND ENTRY ADDRESS IS ON THE
 ;STACK AND THE FIRST ERROR CONDITION
 ;IS PROBABLY STILL LOCKED UP .
10\$: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS
 MOV (KSP)+,OLDPS ;SAVE OLD PSW
 MOV @#CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
 MOV PCPUER,@#CPUERR ;CLEAR CPU ERROR REGISTER
 CMP PCPUER,CPUEXP ;SEE IF EXPECTEED CONDITION CAME UP.
 BEQ 1\$;BRANCH IF IT WAS A TIMEOUT
 ERROR 1 ;NOT RIGHT CONDITION
 BR 3\$;BRANCH TO EXIT
1\$: BIS R0,ADDROR ;PERFORM LOGIAL OR OF FAILING ADDRESS
 COM R0 ;GET R0 READY FOR AND
 BIC R0,ADRAND ;PERFORM LOGICAL AND
 COM R0 ;PUT R0 BACK AS IT WAS
 TSTB \$ERFLG ;IS HIS THE FIRST ERROR
 BNE 2\$;BRANCH IF NOT FIRST ERROR
 ERROR 201 ;NO REGISTER RESPONSE.
 BR 3\$;BRANCH TO EXIT
2\$: ERROR 301 ;CONTINUE NO RESPONSE TABLE
3\$: MOV #-1,TOFLAG ;RESET ONE TIME GATE
 MOV OLDPS,-(KSP) ;RESTORE OLD PSW
 MOV OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON THE STACK
 RTT ;RETURN TO THE TEST

4183
4184
4185
4186
4187
4188
4189
4190
4191
4192
4193 031524
4194 031524 012637 001306
4195 031530 050037 001276
4196 031534 005100
4197 031536 040037 001300
4198 031542 005100
4199 031544 050137 001266
4200 031550 005101
4201 031552 040137 001270
4202 031556 005101
4203 031560 105737 001103
4204 031564 001002
4205 031566 104202
4206 031570 000401
4207 031572 104302
4208 031574 000177 147506
4209
4210
4211
4212
4213
4214
4215
4216
4217
4218
4219
4220
4221
4222
4223 031600
4224 031600 012637 001306
4225 031604 050037 001276
4226 031610 005100
4227 031612 040037 001300
4228 031616 005100
4229 031620 050137 001272
4230 031624 005101
4231 031626 040137 001274
4232 031632 005101
4233 031634 050237 001266
4234 031640 005102
4235 031642 040237 001270
4236 031646 005102

```

.SBTTL DUAL ADDRESSING WHEN LOADING A P.A.R. OR P.D.R.
:*****
:
: THIS SUBROUTINE WILL LOG AND REPORT ALL DUAL ADDRESSING ERRORS
: FOUND IN BOTH P.A.R.'S AND P.D.R.'S. A 'LOGICAL OR' AND A
: 'LOGICAL AND' OF THE WRITTEN ADDRESSES WILL BE MAINTAINED IN
: 'ADDROR' AND 'ADRAND'. THE LOG ON THE ADDITIONAL OR FAILING,
: ADDRESSES WILL BE MAINTAINED IN 'DATAOR' AND 'DATAND'.
:*****
DUALADR:
MOV      (KSP)+,OLDPC      ;STARTING LOCATION OF SUBROUTINE
BIS      R0,ADDROR        ;SAVE RETURN ADDRESS IN CASE OF LOOP
COM      R0                ;LOGICAL OR OF WRITTEN ADDRESS
BIC      R0,ADRAND        ;GET R0 READY FOR AND
COM      R0                ;PERFORM LOGICAL AND
BIS      R1,DATAOR        ;PUT R0 BACK AS IT WAS
COM      R1                ;LOGICAL OR OF DUALED ADDRESS
BIC      R1,DATAND        ;GET R1 READY FOR AND
COM      R1                ;PERFORM LOGICAL AND
TSTB     $ERFLG           ;PUT R1 BACK AS IT WAS
BNE      1$              ;SEE IF THIS IS FIRST ERROR
ERROR    202             ;BRANCH IF NOT FIRST ERROR
BR       2$              ;BRANCH TO EXIT
1$:      ERROR           302
2$:      JMP             @OLDPC ;RETURN TO TEST

```

```

.SBTTL COUNT PATTERN ERRORS IN P.A.R.'S OR P.D.R.'S
:*****
:
: THIS SUBROUTINE IS USED TO LOG AND REPORT THE COUNT PATTERN
: ERRORS OCCURRING WHEN TESTING THE P.A.R.'S AND P.D.R.'S.
: THE 'LOGICAL OR' AND 'LOGICAL AND' OF VARIOUS DATA WILL BE
: MAINTAINED AS FOLLOWS:
: ADDRESSES OF FAILING REGISTERS IN 'ADDROR' AND 'ADRAND'
: DATA FETCHED FROM REGISTERS IN 'DATAOR' AND 'DATAND'
: PATTERN LOADED INTO THE REGISTERS IN 'PATTOR' AND 'PATAND'.
:*****
PARCOUNT:
MOV      (KSP)+,OLDPC      ;STARTING ADDRESS OF THIS SUBROUTINE
BIS      R0,ADDROR        ;SAVE RETURN ADDRESS IN CASE OF LOOP
COM      R0                ;LOGICAL OR OF FAILING ADDRESS
BIC      R0,ADRAND        ;GET R0 READY FOR AND
COM      R0                ;PERFORM LOGICAL AND
BIS      R1,PATTOR        ;PUT R0 BACK AS IT WAS
COM      R1                ;LOGICAL OR OF PATTERN LOADED
BIC      R1,PATAND        ;GET R1 READY FOR AND
COM      R1                ;PERFORM LOGICAL AND
BIS      R2,DATAOR        ;PUT R1 BACK AS IT WAS
COM      R2                ;LOGICAL OR OF DATA FETCHED
BIC      R2,DATAND        ;GET R2 READY FOR AND
COM      R2                ;PERFORM LOGICAL AND

```

```
4237 031650 105737 001103          TSTB  $ERFLG      ;SEE IF THIS IS THE FIRST ERROR
4238 031654 001002                   BNE   1$          ;BRANCH IF NOT FIRST ERROR
4239 031656 104203                   ERROR 203
4240 031660 000401                   BR    2$          ;BRANCH TO EXIT
4241 031662 104303          1$:   ERROR 303
4242 031664 000177 147416          2$:   JMP   @OLDPC      ;RETURN TO TEST
4243
4244
```

4245 .SBTTL ***** TRAP HANDLING ROUTINES *****

4246 .SBTTL CPU TRAP HANDLER ROUTINE
4247 :*****
4248 :*
4249 :* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
4250 :* 'ERRVEC' (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
4251 :*


```

4252      ;*      TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
4253      ;*      IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
4254      ;*      UNEXPECTED ERROR MESSAGE IS GIVEN. IF THE WORD 'CPUEXP' IS
4255      ;*      NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
4256      ;*      'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED. 'PCPUER' CAN
4257      ;*      BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
4258      ;*      IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
4259      ;*
4260      ;*****
4261      031670 005227      CPUER:  INC      (PC)+      ;MAKE FLAG ZERO IF FIRST TIME
4262      031672 177777      CPFLAG: .WORD  -1          ;NEGATIVE ONE FOR A FLAG
4263      031674 001401      BEQ      10$        ;BRANCH IF FIRST TIME IN
4264      031676 000000      HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
4265      ;I FINISHED REPORTING THE FIRST ERROR
4266      ;THE SECOND ENTRY ADDRESS IS ON THE
4267      ;STACK AND THE FIRST ERROR CONDITION
4268      ;IS PROBABLY STILL LOCKED UP .
4269      031700 012637 001306 10$:  MOV      (KSP)+,OLDPC  ;SAVE RETURN ADDRESS IN CASE OF LOOP
4270      031704 012637 001310      MOV      (KSP)+,OLDPS  ;SAVE OLD PSW IN CASE OF LOOP
4271      031710 013737 177766 001260  MOV      CPUERR,PCPUER ;SAVE CPU ERROR REGISTER
4272      031716 013737 001306 001262  MOV      OLDPC,BADPC  ;SAVE PC+2 AT TIME OF ABORT
4273      031724 005737 001224      TST      CPUEXP      ;SEE IF ANY CONDITION WAS EXPECTED
4274      031730 001406      BEQ      2$          ;BRANCH IF NO TRAP WAS EXPECTED
4275      031732 023737 001260 001224  CMP      PCPUER,CPUEXP ;SEE IF EXPECTED ERROR OCCURED
4276      031740 001403      BEQ      1$          ;BRANCH IF ERROR CODES MATCH
4277      031742 104001      ERROR  1          ;ERROR TYPE OUT ITEM 1
4278      031744 000401      BR       1$          ;SKIP NEXT INSTRUCTION
4279      031746 104002      2$:  ERROR  2          ;ERROR ITEM 2 NO CPU TRAP EXPECTED
4280      031750 005037 177766 1$:  CLR      CPUERR      ;CLEAR CPU ERROR REGISTER
4281      031754 012737 177777 031672  MOV      #-1,CPFLAG  ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
4282      031762 013746 001310      MOV      OLDPS,-(KSP) ;PUSH OLD PSW BACK ON STACK
4283      031766 013746 001306      MOV      OLDPC,-(KSP) ;PUSH RETURN ADDRESS BACK ON STACK
4284      031772 000006      RTT                    ;RETURN FROM INTERRUPT OR ABORT
4285
4286
4287
4288      .SBTTL  CACHE TRAPS AND ABORTS HANDLER ROUTINE
4289      ;*****
4290      ;*
4291      ;*      THIS SUBROUTINE WILL HANDLE ALL UNEXPECTED PARITY ERRORS. IF
4292      ;*      THE PARITY ERROR IS AN ABORT THE TEST THAT WAS RUNNING WILL
4293      ;*      BE RESTARTED ONCE AFTER THE ABORT CONDITION IS REPORTED. ON THE
4294      ;*      SECOND ABORT IN A SINGLE TEST THE NEXT TEST IS ATTEMPTED
4295      ;*      AFTER THE ABORT IS REPORTED.
4296      ;*      IF THE PARITY ERROR IS A TRAP THE CACHE WILL BE CLEANED UP BY
4297      ;*      REMOVING THE BAD WORD THAT MAY BE IN THE CACHE, AND THE TEST
4298      ;*      WILL BE CONTINUED AFTER THE PARITY ERROR IS REPORTED.
4299      ;*
4300      ;*****
4301      031774 005227      MEMER:  INC      (PC)+      ;MAKE FLAG ZERO IF FIRST TIME
4302      031776 177777      PAFLAG: .WORD  -1          ;NEGATIVE ONE FOR A FLAG
4303      032000 001401      BEQ      10$        ;BRANCH IF FIRST TIME IN
4304      032002 000000      HALT                    ;I HAVE ENTERED THIS ROUTINE BEFORE
4305      ;I FINISHED REPORTING THE FIRST ERROR
4306      ;THE SECOND ENTRY ADDRESS IS ON THE
4307      ;STACK AND THE FIRST ERROR CONDITION
    
```

```

4308
4309 032004 012737 032224 000114 10$: MOV #5$,CACHVEC ;IS PROBABLY STILL LOCKED UP.
4310 ;SET CACHE VECTOR TO RTI UNTIL THE
4311 032012 012637 001306 MOV (KSP)+,OLDPC ;THE CACHE HAS BEEN FIXED.
4312 032016 012637 001310 MOV (KSP)+,OLDPS ;SAVE RETURN ADDRESS IN CASE OF LOOP
4313 032022 013737 177740 001234 MOV @#LOADRS,PLOADR ;SAVE OLD PSW IN CASE OF LOOP
4314 032030 013737 177742 001236 MOV @#HIADRS,PHIADR ;SAVE LOWER CACHE ADDR REG
4315 032036 013737 177744 001240 MOV @#MEMERR,PPARER ;SAVE HIGH BITS OF FAILING ADDR
4316 032044 013737 177746 001242 MOV @#CONTRL,PCONTR ;SAVE MEMORY ERROR REGISTER
4317 032052 013737 177750 001244 MOV @#MAINT,PMAINT ;SAVE CONTROL REGISTER FOR TYPE OUT
4318 032060 013737 177752 001246 MOV @#HITMIS,PHITMI ;SAVE MAINTENANCE REGISTER
4319 032066 013737 001306 001262 MOV OLDPC,BADPC ;SAVE HIT/MISS REGISTER
4320 032074 032737 000014 001240 BIT #14,PPARER ;SAVE PC+2 AT TIME OF ABORT
4321 032102 001005 BNE 1$ ;WAS THIS A MAIN MEMORY REFERENCE
4322 032104 012737 031774 000114 MOV #MEMER,CACHVEC ;BRANCH IF IT WAS A MAIN MEMORY ERROR
4323 032112 104003 ERROR 3 ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
4324 032114 000415 BR 2$ ;UNEXPECTED CACHE PARITY ERROR
4325 032116 010046 1$: MOV RO,-(KSP) ;BRANCH TO EXIT POINT
4326 032120 013700 001234 MOV PLOADR,RO ;SAVE RO ON STACK
4327 032124 042700 176000 BIC #176000,RO ;PUT LOW 16 BITS OF BAD ADDR IN RO
4328 032130 074027 000002 XOR RO,#BIT1 ;CLEAR UPPER 6 BITS OF ADDRESS
4329 032134 005710 TST (RO) ;REFERENCE OTHER WORD OF PAIR
4330 032136 012600 MOV (KSP)+,RO ;READ AT SAME INDEX AS BAD WORD
4331 032140 012737 031774 000114 MOV #MEMER,CACHVEC ;RESTORE RO FROM STACK
4332 032146 104004 ERROR 4 ;LOAD ADDRESS OF THIS ROUTINE IN VECTOR
4333 032150 005737 001314 2$: TST RETRY ;UNEXPECTED MAIN MEMORY PARITY ERROR
4334 032154 001006 BNE 3$ ;ARE YOU RETRYING THIS TEST?
4335 032156 005237 001314 INC RETRY ;BRANCH IF THIS IS SECOND TRY
4336 032162 013737 001110 001306 MOV $LPADR,OLDPC ;SET RETRY FLAG
4337 032170 000403 BR 4$ ;RETURN TO START OF THIS TEST
4338 032172 013737 001316 001306 3$: MOV NXTTST,OLDPC ;BRANCH TO EXIT
4339 ;RETURN TO START OF NEXT TEST
4340 032200 013737 001240 177744 4$: MOV PPARER,MEMERR ;SINCE THIS IS THE SECOND ABORT
4341 032206 012737 177777 031776 MOV #-1,PAFLAG ;CLEAR MEMORY ERROR REGISTER
4342 032214 013746 001310 MOV OLDPS,-(KSP) ;RESTORE A NEGATIVE ONE FOR NEXT TIME
4343 032220 013746 001306 MOV OLDPC,-(KSP) ;PUSH OLD PSW BACK ON STACK
4344 032224 000006 5$: RTT ;PUSH RETURN ADDRESS BACK ON STACK
4345 ;RETURN FROM INTERRUPT.

```

.SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE
:*****

```

:
: THIS SUBROUTINE WILL HANDLE MOST OF THE MEMORY MANAGEMENT TRAPS
: AND ABORTS THAT ARE GENERATED DURING THIS PROGRAM. ANY M.M.
: ABORTS OR TRAPS THAT OCCUR WHILE 'MMEXP' IS ZERO ARE UNEXPECTED
: AND WILL BE REPORTED AS SUCH. THIS MAY OCCUR BECAUSE SOME LOGIC
: THAT IS TO INHIBIT A TRAP HAS FAILED.
: THERE ARE ALSO TIMES WHEN I AM EXPECTING A CERTAIN CONDITION TO
: OCCUR, AND WHEN THIS CONDITION IN 'MMEXP' DOES NOT MATCH THE
: CONTENTS OF 'PMMRO' (SAME AS 'MMRO', AN ERROR IS REPORTED.
:
:*****

```

```

4361 032226 005227 MMTRAP: INC (PC)+ ;MAKE FLAG ZERO IF FIRST TIME
4362 032230 177777 MMFLAG. .WORD -1 ;FLAG SHOULD BE NEG ONE
4363 032232 001401 BEQ 10$ ;BRANCH IF FIRST TIME INTO ROUTINE

```

```
4364 032234 000000          HALT          ;I HAVE ENTERED THIS ROUTINE BEFORE
4365                                     ;I FINISHED REPORTING THE FIRST ERROR
4366                                     ;THE SECOND ENTRY ADDRESS IS ON THE
4367                                     ;STACK AND THE FIRST ERROR CONDITION
4368                                     ;IS PROBABLY STILL LOCKED UP
4369 032236 011637 001262    10$:  MOV      (KSP),BADPC      ;SAVE PC AT TIME OF ABORT OR TRAP
4370 032242 012637 001306      MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF LOOP
4371 032246 012637 001310      MOV      (KSP)+,OLDPS      ;SAVE OLD PSW IN CASE OF LOOP
4372 032252 013737 177572 001250  MOV      MMR0,PMR0        ;SAVE STATUS REGISTER
4373 032260 013737 177574 001252  MOV      MMR1,PMR1        ;SAVE AUTO INC/DEC REGISTER
4374 032266 013737 177576 001254  MOV      MMR2,PMR2        ;SAVE VIRTUAL ADDRESS REGISTER
4375 032274 005737 001226      TST      MMEXP            ;SEE IF ANY M.M. TRAPS WERE EXPECTED
4376 032300 001002          BNE      5$              ;BRANCH IF ONE WAS EXPECTED
4377 032302 104005          ERROR      5              ;UNEXPECTED M.M. ABORT OR TRAP
4378 032304 000405          BR        1$              ;BRANCH TO EXIT
4379 032306 023737 001226 001250  5$:  CMP      MMEXP,PMR0        ;SEE IF ABORT OR TRAP WAS EXPECTED
4380 032314 001401          BEQ      1$              ;BRANCH IF CONDITION CORRECT
4381 032316 104006          ERROR      6              ;ERROR TYPE OUT ITEM 6
4382 032320 042737 177376 177572  1$:  BIC      #177376,@MMR0    ;CLEAR ALL BITS EXCEPT 0 AND 8
4383 032326 012737 177777 032230  MOV      #-1,MMFLAG      ;RESTORE A NEGATIVE ONE TO FLAG
4384 032334 013746 001310      MOV      OLDPS,-(KSP)     ;PUSH OLD PSW ONTO STACK
4385 032340 013746 001306      MOV      OLDPC,-(KSP)     ;PUSH RETURN ADDRESS ON STACK
4386 032344 000006          RTT          ;RETURN TO MAIN PROGRAM
```

```
4387
4388
4389 .SBTTL D-SPACE TESTS MEMORY MANAGEMENT ABORT SERVICE ROUTINE
4390 :*****
4391 :
4392 : THIS ROUTINE WILL BE ENTERED IF A MEMORY MANAGEMENT ABORT
4393 : OCCURS DURING THE D-SPACE ENABLE TESTS. IF THE ABORT IS A
4394 : NON-RESIDENT ABORT THE PROBLEM IS PROBABLY IN THE D-SPACE
4395 : ENABLE LOGIC ON PAGES 'SAPK' AND 'SSRB' IN THE PRINTS.
4396 : IN ALL OF THE D-SPACE ENABLE TESTS, D-SPACE PAGES 2 & 3
4397 : ARE MAPPED NON-RESIDENT AND I-SPACE PAGE 4 IS MAPPED NON-
4398 : RESIDENT. ALL OTHER PAGES ARE MAPPED RESIDENT, 4K, READ/WRITE.
4399 : THEREFORE IF THE N.R. PAGE IS 2 OR 3 YOU ARE PROBABLY NOT
4400 : FORCING I-SPACE WHEN YOU SHOULD. BUT IF THE N.R. PAGE IS 4
4401 : YOU ARE PROBABLY FORCING I-SPACE WHEN YOU SHOULD ALLOW D-SPACE
4402 :
4403 :*****
```

```
4404 032346          NODSPAC:          ;STARTING ADDRESS FOR ABORT SERVICE
4405 032346 005227      INC      (PC)+          ;MAKE FLAG ZERO IF FIRST TIME
4406 032350 177777      NDFLAG: .WORD      -1          ;FLAG SHOULD BE NEG ONE
4407 032352 001401      BEQ      10$          ;BRANCH IF FIRST TIME INTO ROUTINE
4408 032354 000000      HALT          ;I HAVE ENTERED THIS ROUTINE BEFORE
4409                                     ;I FINISHED REPORTING THE FIRST ERROR
4410                                     ;THE SECOND ENTRY ADDRESS IS ON THE
4411                                     ;STACK AND THE FIRST ERROR CONDITION
4412                                     ;IS PROBABLY STILL LOCKED UP
4413 032356 011637 001262    10$:  MOV      (KSP),BADPC      ;SAVE PC AT TIME OF ABORT OR TRAP
4414 032362 012637 001306      MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF LOOP
4415 032366 012637 001310      MOV      (KSP)+,OLDPS      ;SAVE OLD PSW IN CASE OF LOOP
4416 032372 013737 177572 001250  MOV      MMR0,PMR0        ;SAVE STATUS REGISTER
4417 032400 013737 177574 001252  MOV      MMR1,PMR1        ;SAVE AUTO INC/DEC REGISTER
4418 032406 013737 177576 001254  MOV      MMR2,PMR2        ;SAVE VIRTUAL ADDRESS REGISTER
4419 032414 005737 001250      TST      PMR0            ;WAS ABORT NON-RESIDENT?
```

```
4420 032420 100002          BPL      1$          ;BRANCH IF NOT, IT IS UNEXPECTED.
4421 032422 104132          ERROR    132         ;D-SPACE ENABLE FAULTY
4422 032424 000401          BR       2$          ;BRANCH TO EXIT
4423 032426 104005          1$:      ERROR      5          ;UNEXPECTED M.M. ABORT
4424 032430 042737 177376 177572 2$:      BIC      #177376,MMR0 ;CLEAR ALL BITS EXCEPT 0 AND 8
4425 032436 012737 177777 032350      MOV      #-1,NDFLAG ;RESTORE A NEGATIVE ONE TO FLAG
4426 032444 013746 001310      MOV      OLDPS,-(KSP) ;PUSH OLD PSW ONTO STACK
4427 032450 013746 001306      MOV      OLDPC,-(KSP) ;PUSH RETURN ADDRESS ON STACK
4428 032454 000006          RTT          ;RETURN TO MAIN PROGRAM
4429
4430
4431
```

.SBTTL TRAP ROUTINES FOR ABORT IN SUPERVISOR OR USER MODE

THESE NEXT THREE SUBROUTINES ARE USED FOR THE TESTS THAT VERIFY
THE VECTOR IS PICKED UP FROM KERNEL SPACE DURING AN ABORT.
'KERVEC' IS WHERE I SHOULD GO SINCE IT IS AT 250 WHICH IS
KERNEL SPACE VIRTUAL 250.
'SUPVEC' IS AT 350 WHICH IS SUPERVISOR VIRTUAL 250
'USEVEC' IS AT 450 WHICH IS USER SPACE VIRTUAL 250

THEY ALL READ THE PROCESSOR STATUS WHICH IS DIFFERENT FOR EACH
VECTOR AND THEN JUMP TO 'RDMMRO'. 'RDMMRO' READS MMR0, MMR1,
AND MMR2 AND RETURNS TO THE TEST WHERE THEY ARE TESTED.

```
4446 032456 013700 177776      KERVEC: MOV      PSW,R0          ;PUT PROCESSOR STATUS INTO R0
4447 032462 012637 001306      MOV      (KSP)+,OLDPC        ;SAVE RETURN ADDRESS
4448 032466 012637 001310      MOV      (KSP)+,OLDPS        ;SAVE OLD PROCESSOR STATUS
4449 032472 122700 000340      CMPB    #340,R0             ;SEE IF CORRECT PSW WAS PICKED UP
4450 032476 001401          BEQ     1$                  ;BRANCH IF PSW IS CORRECT
4451 032500 104122          ERROR    122               ;WRONG PSW PICKED UP
4452 032502 000137 032546      1$:      JMP     RDMMRO            ;JUMP TO READ MMR0
4453
4454 032506 012637 001306      SUPVEC: MOV      (KSP)+,OLDPC  ;SAVE RETURN ADDRESS
4455 032512 012637 001310      MOV      (KSP)+,OLDPS        ;SAVE OLD PROCESSOR STATUS
4456 032516 013700 177776      MOV      PSW,R0             ;READ PSW FOR ERROR PRINT OUT
4457 032522 104123          ERROR    123               ;WRONG VECTOR, POSSIBLE WRONG PSW
4458 032524 000137 032546      JMP     RDMMRO            ;GO READ MMR0
4459
4460 032530 012637 001306      USEVEC: MOV      (KSP)+,OLDPC  ;SAVE RETURN ADDRESS
4461 032534 012637 001310      MOV      (KSP)+,OLDPS        ;SAVE OLD PROCESSOR STATUS
4462 032540 013700 177776      MOV      PSW,R0             ;READ PSW FOR ERROR PRINT OUT
4463 032544 104124          ERROR    124               ;WRONG VECTOR, POSSIBLE WRONG PSW
4464
4465
4466 032546 013737 177572 001250      RDMMRO: MOV      MMR0,PMMR0    ;READ MMR0 TO BE CHECKED LATER
4467 032554 013737 177574 001252      MOV      MMR1,PMMR1          ;READ MMR1 FOR POSSIBLE ERROR REPORT
4468 032562 013737 177576 001254      MOV      MMR2,PMMR2          ;READ MMR2 FOR POSSIBLE ERROR REPORT
4469 032570 013746 001310      MOV      OLDPS,-(KSP)        ;PUSH OLD PROCESSOR STATUS ON STACK
4470 032574 013746 001306      MOV      OLDPC,-(KSP)        ;PUSH RETURN ADDRESS ON STACK
4471 032600 000006          RTT          ;RETURN TO TEST TO CHECK PMMR0
4472
4473
4474
4475
```

*** TEST FOR VARIOUS KB11 PROCESSORS ***

```

4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487 032602 105037 001362
4488 032606 005037 001360
4489 032612 012737 033050 000010
4490 032620 000007
4491
4492 032622 012737 000001 001360
4493 032630 005037 177750
4494 032634 005005
4495 032636 012700 177746
4496 032642 012701 177750
4497 032646 012702 170202
4498 032652 052710 040000
4499 032656 032710 040000
4500 032662 001403
4501 032664 042710 040000
4502 032670 005205
4503 032672 052711 000001
4504 032676 032711 000001
4505 032702 001410
4506 032704 052710 004000
4507 032710 032710 004000
4508 032714 001403
4509 032716 042710 004000
4510 032722 005205
4511 032724 042711 000001
4512 032730 052737 100000 172300
4513 032736 032737 100000 172300
4514 032744 001404
4515 032746 042737 100000 172300
4516 032754 005205
4517 032756 052712 100000
4518 032762 032712 100000
4519 032766 001403
4520 032770 042712 100000
4521 032774 005205
4522 032776 022705 000002
4523 033002 101021
4524 033004 005000
4525 033006 005037 177746
4526 033012 013701 177746
4527 033016 001402
4528 033020 005200
4529 033022 001373
4530 033024
4531 033024 005737 001360

```

```

:::
::: *THIS ROUTINE POLES THE RESULTS OF ATTEMPTS TO SET TO ONE
::: *CERTAIN CRITICAL BITS THAT ARE KNOWN TO BE OPERATIVE ON A KB11CM,
::: *OR KB11EM PROCESSOR. IF TWO OUT OF FOUR OF THE TESTS ARE
::: *POSITIVE THEN THE KB11CM OR KB11EM FLAG IS SET, IF LESS THAN TWO OF THE
::: *TESTS ARE POSITIVE THEN THE KB11E FLAG OR NO FLAG IS SET. THE DETERMINATION
::: *OF WHICH PAIR IS VALID IS BASED ON THE RESULTS OF EXECUTING AN MFPT OPCODE
::: *(OPCODE 7). IF THIS INSTRUCTION TRAPS THIS IS AN KB11CM OR
::: *A PLAIN 1170 (KB11-B OR KB11-C). IF THE INSTRUCTION DOES NOT TRAP THEN
::: *THIS IS A KB11-E OR KB11-EM.
:::
KBTST: CLR      @#KB11CM      ;RESET THE MP FLAG
        CLR      @#KB11E      ;CLEAR KB11E AND KB11EM FLAGS
        MOV      @#MFPTTR,@#RESVEC ;SET UP TRAP ADDRESS FOR MFPT AT RESERV VECTOR
        MFPT      ;EXECUTE MFPT. WILL TRAP ON 1170 (KB11B/C) OR
                   ;KB11CM (11/74)
T1:    MOV      #1,@#KB11E      ;HERE IF KB11E OR KB11EM. SET FLAG
        CLR      @#MAINT      ;CLEAR THE MAINTENANCE REGISTER
        CLR      R5           ;RESET THE TEST COUNTER
        MOV      @#CONTRL,R0    ;GET THE ADDRESS OF...
        MOV      @#MAINT,R1     ;CCR,MAINT,AND MAPH00...
        MOV      @#MAPH00,R2    ;AND PLACE IN R0-R2
        BIS      @#BIT14,(R0)   ;TRY TO SET IVSS BIT
        BIT      @#BIT14,(R0)   ;DID IT SET?
        BEQ      T2           ;NO,GO TO NEXT TEST
        BIC      @#BIT14,(R0)   ;CLEAR IT.
        INC      R5           ;TEST IS POSITIVE
T2:    BIS      @#BIT0,(R1)     ;SET EDMA IN MAINT REGISTER
        BIT      @#BIT0,(R1)
        BEQ      T3
        BIS      @#BIT11,(R0)   ;TRY TO SET DMMA IN CCR
        BIT      @#BIT11,(R0)
        BEQ      T3
        BIC      @#BIT11,(R0)
        INC      R5
T3:    BIC      @#BIT0,(R1)     ;MAKE SURE EDMA IS CLEAR
        BIS      @#BIT15,KIPDR0 ;TRY TO SET BYP ON A PDR
        BIT      @#BIT15,KIPDR0
        BEQ      T4
        BIC      @#BIT15,KIPDR0
        INC      R5
T4:    BIS      @#BIT15,(R2)    ;TRY TO SET BYP ON UNIBUS MAP
        BIT      @#BIT15,(R2)
        BEQ      T.END
        BIC      @#BIT15,(R2)
        INC      R5
T.END: CMP      #2,R5         ;IS THE RESULT OF THE TEST >=2
        BHI      2$           ;NO,THIS IS A KB11E OR KB11-B/C (11/70)
        CLR      R0
        CLR      @#CONTRL
        MOV      @#CONTRL,R1
        BEQ      4$
        INC      R0
        BNE      3$
4$:    TST      @#KB11E      ;IS IT A KB11-E OR KB11-EM?

```

```

4532 033030 001404          BEQ      1$          ;BR IF NEITHER. MUST BE KB11CM
4533 033032 012737 000400 001360  MOV     #BIT8,@#KB11E ;SET UPPER BYTE (KB11-EM)
4534 033040 000402          BR       2$          ;DONE
4535 033042 105237 001362    1$:     INCB     @#KB11CM ;YES, FLAG THIS AS A MODIFIED PROCESSOR
4536 033046 000207          2$:     RTS      PC      ;DONE
4537
4538 033050          MFPTTR:          ;HERE IF MFPT TRAPPED. SEE IF 1170 OR KB11CM
4539 033050 012716 032630    MOV     #T1,(SP)     ;SET UP RETURN ADDRESS FOR RTI
4540 033054 000002          RTI          ;RETURN
4541 033056          ENDKB:
4542          ;:*****
4543          ;*
4544          ;SIZE MEMORY AND COMPARE IT WITH THE SYSTEM SIZE REGISTER
4545          ;PRINT A WARNING IF THEY DISAGREE.
4546          ;*
4547          ;:*****
4548
4549 033056 052737 000200 030636  SIZMEM: BIS     #BIT07,$KT11
4550 033064 004737 030570          JSR     PC,$SIZE
4551 033070 062737 000037 031142  ADD     #37,$LSTBK
4552 033076 023737 177760 031142  CMP     @#SIZELO,$LSTBK ;EQUAL?
4553 033104 001550          BEQ     OKSIZ
4554 033106 104400 033114          TYPE   ,65$          ;;TYPE ASCIZ STRING
4555 033112 000433          BR     ,64$          ;;GET OVER THE ASCIZ
4556          ;;65$: .ASCIZ <15><12>/WARNING- THE SIZE OF MEMORY IS DIFFERENT FROM THAT/
4557 033202          64$:
4558 033202 104400 033210          TYPE   ,67$          ;;TYPE ASCIZ STRING
4559 033206 000425          BR     ,66$          ;;GET OVER THE ASCIZ
4560          ;;67$: .ASCIZ <15><12>/!INDICATED BY THE SYSTEM SIZE REGISTER./
4561 033262          66$:
4562 033262 104400 033270          TYPE   ,69$          ;;TYPE ASCIZ STRING
4563 033266 000421          BR     ,68$          ;;GET OVER THE ASCIZ
4564          ;;69$: .ASCIZ <15><12>/ SIZEHI SIZELO ACTUAL/
4565 033332          68$:
4566 033332 104400 001217          TYPE   , $SCRLF
4567 033336 013746 177762          MOV     @#SIZEHI,-(SP) ;;SAVE @#SIZEHI FOR TYPEOUT
4568 033342 104404          TYPOS          ;;GO TYPE--OCTAL ASCII
4569 033344 006          .BYTE 6          ;;TYPE 6 DIGIT(S)
4570 033345 000          .BYTE 0          ;;SUPPRESS LEADING ZEROS
4571 033346 104400 033354          TYPE   ,71$          ;;TYPE ASCIZ STRING
4572 033352 000404          BR     ,70$          ;;GET OVER THE ASCIZ
4573          ;;71$: .ASCIZ / /
4574 033364          70$:
4575 033364 013746 177760          MOV     @#SIZELO,-(SP) ;;SAVE @#SIZELO FOR TYPEOUT
4576 033370 104404          TYPOS          ;;GO TYPE--OCTAL ASCII
4577 033372 006          .BYTE 6          ;;TYPE 6 DIGIT(S)
4578 033373 000          .BYTE 0          ;;SUPPRESS LEADING ZEROS
4579 033374 104400 033402          TYPE   ,73$          ;;TYPE ASCIZ STRING
4580 033400 000404          BR     ,72$          ;;GET OVER THE ASCIZ
4581          ;;73$: .ASCIZ / /
4582 033412          72$:
4583 033412 013746 031142          MOV     $LSTBK,-(SP) ;;SAVE $LSTBK FOR TYPEOUT
4584 033416 104404          TYPOS          ;;GO TYPE--OCTAL ASCII
4585 033420 006          .BYTE 6          ;;TYPE 6 DIGIT(S)
4586 033421 000          .BYTE 0          ;;SUPPRESS LEADING ZEROS
4587 033422 104400 001217          TYPE   , $SCRLF ;DO CARRIAGE RETURN,LINE FEED
    
```

4588 033426 000207
4589
4590
4591
4592

OKSIZ: RTS PC ;RFTURN

```

4593          .SBTTL
4594          .SBTTL ***** ENTRY POINT 1 --- STARTING ADDRESS 200 *****
4595          .SBTTL ***** TEST CODE STARTS AT ADDRESS 40000 *****
4596          .=40000
4597          040000
4598 040000 012737 000000 001222 STRT1: MOV #0,FSTTST ;LOAD INDEX TO START TABLE
4599 040006 000433          BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4600 040010 012737 000002 001222 STRT2: MOV #2,FSTTST ;LOAD INDEX TO START TABLE
4601 040016 000427          BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4602 040020 012737 000004 001222 STRT3: MOV #4,FSTTST ;LOAD INDEX TO START TABLE
4603 040026 000423          BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4604 040030 012737 000006 001222 STRT4: MOV #6,FSTTST ;LOAD INDEX TO START TABLE
4605 040036 000417          BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4606 040040 012737 000010 001222 STRT5: MOV #10,FSTTST ;LOAD INDEX TO START TABLE
4607 040046 000413          BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4608 040050 012737 000012 001222 STRT6: MOV #12,FSTTST ;LOAD INDEX TO START TABLE
4609 040056 000407          BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4610 040060 012737 000014 001222 STRT7: MOV #14,FSTTST ;LOAD INDEX TO START TABLE
4611 040066 000403          BR START ;BRANCH TO STARTING ADDRESS OF PROGRAM
4612 040070 012737 000016 001222 STRT8: MOV #16,FSTTST ;LOAD INDEX TO START TABLE
4613
4614
4615
4616
4617 040076 012706 001100          START: MOV #KERSTK,KSP ;SET UP KERNAL STACK
4618 040102 012737 001014 177746 MOV #1014,CONTRL ;TURN OFF CACHE
4619 040110 012706 001100          MOV #SCMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
4620 040114 005026          2$: CLR (R6)+ ;:CLEAR MEMORY LOCATION
4621 040116 022706 001140          CMP #STKS,R6 ;:DONE?
4622 040122 001374          BNE 2$ ;:LOOP BACK IF NO
4623 040124 012737 025474 000020 MOV #SSCOPE,IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
4624 040132 012737 000340 000022 MOV #340,IOTVEC+2 ;:PRIORITY LEVEL 7
4625 040140 012737 025760 000030 MOV #SEERROR,EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
4626 040146 012737 000340 000032 MOV #340,EMTVEC+2 ;:PRIORITY LEVEL 7
4627 040154 012737 030326 000034 MOV #STRAP,TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
4628 040162 012737 000340 000036 MOV #340,TRAPVEC+2 ;:PRIORITY LEVEL 7
4629 040170 012737 030370 000024 MOV #SPWRDN,PWRVEC ;:POWER FAILURE VECTOR
4630 040176 012737 000340 000026 MOV #340,PWRVEC+2 ;:PRIORITY LEVEL 7
4631 040204 013737 025240 025232 MOV SENDCT,SEOPCT ;:SET UP END-OF-PROGRAM COUNTER
4632 040212 005037 001206          CLR $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
4633 040216 005037 001210          CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
4634 040222 112737 000001 001117 MOVB #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
4635 040230 012737 025460 000014 MOV #SRTRN,TBITVEC ;:SET 'T' BIT VECTOR TO SRTRN
4636 040236 012737 000340 000016 MOV #340,TBITVEC+2 ;:PRIORITY LEVEL 7
4637 040244 012737 000006 025460 MOV #RTT,$RTRN ;:SET $RTRN TO A RTT
4638 040252 005037 025466          CLR $TBIT ;:CLEAR 'T' BIT SWITCH
4639 040256 012737 040256 001110 MOV #,$LPADR ;:INITIALIZE LOOP ADDRESS
4640 040264 012737 040264 001112 MOV #,$LPERR ;:INITIALIZE LOOP ON ERROR
4641
4642 040272 005037 177572          LOOP: CLR MMRO ;GET INTO 16 BIT MODE ON SECOND PASS
4643
4644 040276 004737 032602          JSR PC,KBTST ;SEE IF KB11-E PROCESSOR
4645 040302 005037 172516          CLR MMR3 ;TURN OFF EVERY THING POSSIBLE
4646 040306 012700 177777          MOV #-1,R0 ;PUT NEG ONE IN R0 TO INITIALIZE FLAGS
4647 040312 010037 031672          MOV R0,CPFLAG ;INITIALIZE CPU ERROR FLAG
4648 040316 010037 031776          MOV R0,PAFLAG ;INITIALIZE PARITY ERROR FLAG

```



```

4649 040322 010037 032230      MOV      R0,MMFLAG      ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
4650 040326 010037 032350      MOV      R0,NDFLAG      ;INITIALIZE NO D-SPACE TRAP FLAG
4651 040332 010037 177744      MOV      R0,MEMERR      ;CLEAR PARITY ERROR REGISTER (177744)
4652 040336 010037 177766      MOV      R0,CPUERR      ;CLEAR CPU ERROR REGISTER (177766)
4653 040342 005037 001224      CLR      CPUEXP         ;NOT EXPECTING ANY CPU TRAPS
4654 040346 005037 001226      CLR      MMEXP         ;NOT EXPECTING ANY MEMORY MANAGEMENT TRAPS
4655 040352 012737 031670 000004  MOV      #CPUER,ERRVEC   ;LOAD ADDRESS OF CPU TRAP ROUTINE
4656 040360 012737 000340 000006  MOV      #340,ERRVEC+2   ;SET PRIORITY LEVEL 7
4657 040366 012737 031774 000114  MOV      #MEMER,CACHVEC  ;LOAD ADDRESS OF PARITY TRAP ROUTINE
4658 040374 012737 000340 000116  MOV      #340,CACHVEC+2  ;SET PRIORITY LEVEL 7
4659 040402 012737 032226 000250  MOV      #MMTRAP,MMVEC   ;LOAD ADDRESS OF MEMORY MANAGEMENT TRAP
4660 040410 012737 000340 000252  MOV      #340,MMVEC+2    ;SET PRIORITY LEVEL 7
4661 040416 004737 031346      JSR      PC,CLEANUP      ;INITIALIZE ALL ERROR LOCATIONS
4662 040422 012706 001100      MOV      #KERSTK,KSP     ;SET UP KERNEL STACK POINTER
4663 040426 005227 177777      INC      #-1             ;:FIRST TIME?
4664 040432 001036      BNE      64$            ;:BRANCH IF NO
4665 040434 022737 025412 000042  CMP      #SENDAD,@#42    ;:ACT-11?
4666 040442 001432      BEQ      64$            ;:BRANCH IF YES
4667 040444 104400 040452      TYPE     ,65$           ;:TYPE ASCIZ STRING
4668 040450 000427      BR       64$            ;:GET OVER THE ASCIZ
4669          ;:65$:          .ASCIZ <CRLF>?CEKBE-D  PDP 11/70-74MP MEMORY MANAGEMENT ?<CRLF>
4670 040530      64$:
4671
4672 040530 005227 177777      INC      #-1             ;:FIRST TIME?
4673 040534 001026      BNE      100$           ;:BR IF NO
4674 040536 104400 003324      TYPE     ,MSG1           ;:<15><12>CPU UNDER TEST FOUND TO BE A
4675 040542 005737 001360      TST      @#KB11E         ;:IS THIS A KB11-E OR KB11-EM?
4676 040546 001011      BNE      101$           ;:BR IF EITHER ONE
4677 040550 105737 001362      TSTB     @#KB11CM        ;:IS IT A 11/74 (KB11CM)
4678 040554 001003      BNE      1$             ;:BR IF IT IS
4679 040556 104400 003374      TYPE     ,MSG3           ;:KB11-B/C<15><12>
4680 040562 000413      BR       100$           ;:SKIP OTHER MESSAGE
4681 040564 104400 003406      TYPE     ,MSG4           ;:11/74 (KB11CM)<15><12>
4682 040570 000410      BR       100$           ;:SKIP CISP MESSAGE
4683 040572 105737 001360      TSTB     @#KB11E         ;:IS IT A KB11-E?
4684 040576 001403      BEQ      102$           ;:BR IF NOT. MUST BE KB11-EM
4685 040600 104400 003437      TYPE     ,MSG5           ;:KB11-E<15><12>
4686 040604 000402      BR       100$           ;:SKIP KB11-EM MESSAGE
4687 040606 104400 003363      TYPE     ,MSG2           ;:KB11-EM<15><12>
4688 040612      100$:
4689 040612 005227 177777      INC      #-1             ;:FIRST TIME?
4690 040616 001002      BNE      110$           ;:BR IF NOT
4691 040620 004737 033056      JSR      PC,SIZMEM       ;:SIZE MEM AND COMPARE WITH SIZE REG
4692 040624 013700 001222      MOV      FSTTST,R0      ;:LOAD START TABLE INDEX
4693 040630 001417      BEQ      TST1           ;:START WITH TEST ONE IF ZERO
4694 040632 012737 040000 177776  MOV      #40000,PSW      ;:GO TO SUPERVISOR MODE
4695 040640 012706 000700      MOV      #SUPSTK,SSP     ;:SET UP SUPERVISOR STACK POINTER
4696 040644 012737 140000 177776  MOV      #140000,PSW     ;:GO TO USER MODE
4697 040652 012706 000600      MOV      #USESTK,USP     ;:SET UP USER STACK POINTER
4698 040656 012737 000340 177776  MOV      #340,PSW        ;:GO TO KERNEL MODE PRIORITY LEVEL 7
4699 040664 000170 001336      JMP      @STRTAB(R0)     ;:JUMP TO CORRECT ENTRY POINT
4700
4701
4702          ;*          THIS FIRST GROUP OF TESTS IS FOR TESTING THE ADDRESS DECODE
4703          ;*          LOGIC FOR THE INTERNAL REGISTERS ON PAGE 'SCCE' AND TO MAKE
4704          ;*          SOME DETERMINATION ABOUT THE RESPONDING ADDRESSES. IF AN

```

4705 : * ADDRESS DOES NOT FUNCTION AS THE CORRESPONDING REGISTER SHOULD
4706 : * AN ERROR WILL BE FLAGGED. THE MULTIPLEXERS AND INTERNAL BUS
4707 : * DRIVERS ON PAGES 'SCCM' AND 'SCCN' ARE ALSO UTILIZED IN
4708 : * THESE TESTS AND COULD BE THE SOURCE OF ANY PROBLEMS ENCOUNTERED.
4709
4710
4711

4712 : *****
4713 : *TEST 1 TRY TO READ ALL CPU REGISTERS
4714 : *
4715 : * THIS TEST ENSURES THAT SOMETHING RESPONDS TO ADDRESSES 177760
4716 : * THRU 177776. IF A TRAP TO 'ERRVEC' (004) OCCURS IT IS ASSUMED
4717 : * THAT A REGISTER HAS TIMED OUT, AND AN ERROR IS REPORTED.
4718 : *
4719 : * ERRORS THAT OCCUR IN THIS TEST ARE REPORTED FROM AN AREA
4720 : * AT THE END OF THIS TEST, STARTING AT '50\$'.
4721 : *****

4722 040670
4723 040670 012737 040726 001110 TST1: MOV #20\$, \$LPADR ;SET LOOP ADDRESS POINTER TO 20\$
4724 040676 012737 040726 001112 MOV #20\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 20\$
4725 040704 012737 000001 001102 MOV #1, \$TSTNM ;LOAD TEST NUMBER INTO MEMORY
4726 040712 013737 001102 177570 MOV \$TSTNM, DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
4727 040720 012737 041012 001316 MOV #TST2, NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
4728 : *FOR ESCAPE ON PARITY ERRORS
4729 040726 012737 040752 001112 20\$: MOV #1\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
4730 040734 012737 041002 000004 MOV #50\$, ERRVEC ;SET TIME OUT VECTOR TO SPECIAL ROUTINE
4731 040742 012706 001100 MOV #1100, KSP ;SET KERNEL STACK POINTER TO 1100
4732 040746 012700 177760 MOV #177760, R0 ;PUT FIRST CPU REGISTER ADDRESS IN R0
4733 040752 000240 1\$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4734 040754 011001 MOV (R0), R1 ;TRY TO READ THE CPU REGISTERS
4735 040756 062700 000002 100\$: ADD #2, R0 ;POINT TO NEXT CPU REGISTER
4736 040762 001373 BNE 1\$;BRANCH IF NOT ALL READ YET
4737 040764 012737 040726 001112 MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST
4738 040772 012737 031670 000004 MOV #CPUER, ERRVEC ;SET UP C.P.U. ERROR VECTOR
4739 041000 000404 BR TST2 ;:TEST OVER BRANCH TO NEXT TEST
4740
4741
4742 041002 062706 000004 50\$: ADD #4, KSP ;RE ADJUST KERNEL STACK POINTER
4743 041006 104024 ERROR 24 ;CPU REGISTER TIMED OUT
4744 041010 000762 BR 100\$;GO BACK AND FINISH TEST
4745
4746
4747

4748 : *****
4749 : *TEST 2 SYSTEM SIZE REGISTERS
4750 : *
4751 : * BOTH THE LO SIZE AND THE HI SIZE REGISTERS ARE READ TO SEE THAT
4752 : * THEY HOLD SOMETHING THAT LOOKS CORRECT. (IE. LO SIZE SHOULD
4753 : * HAVE 377 IN ITS LOW BYTE, AND HI SIZE SHOULD BE ZERO) THEY
4754 : * ARE ALSO VERIFIED TO BE READ ONLY REGISTERS.
4755 : *****

4756 041012
4757 041012 000004 TST2: SCOPE
4758 041014 012737 041146 001316 MOV #TST3, NXTTST ;SAVE STARTING ADDRESS OF NEXT
4759 : *TEST FOR ESCAPE ON PARITY ERRORS
4760 041022 012737 041034 001112 20\$: MOV #21\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 21\$

```
4761 041030 012700 177760      MOV      #177760,R0      ;PUT ADDRESS OF SIZE LO REGISTER INTO R0
4762 041034 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4763 041036 011001      MOV      (R0),R1        ;READ SYSTEM SIZE LO REGISTER
4764 041040 122701 000377      CMPB    #377,R1        ;SEE IF LOW BYTE IS ALL ONES
4765 041044 001401      BEQ     1$             ;BRANCH IF LOW BYTE IS ALL ONES
4766 041046 104025      ERROR   25            ;LOW BYTE OF LO SIZE IS NOT -1
4767 041050 012737 041056 001112 1$:  MOV      #22$,$LPERR    ;SET LOOP ON ERROR POINTER TO 22$
4768 041056 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4769 041060 005010      CLR     (R0)          ;TRY TO CLEAR SIZE LO REGISTER
4770 041062 011002      MOV      (R0),R2        ;READ SIZE LO REGISTER INTO R2
4771 041064 001002      BNE     2$            ;BRANCH IF IT IS NOT ZERO
4772 041066 104026      ERROR   26            ;COULD WRITE SIZE LO REG
4773 041070 010110      MOV      R1,(R0)        ;RESTORE DATA TO ADDRESS
4774 041072 012737 041104 001112 2$:  MOV      #23$,$LPERR    ;SET LOOP ON ERROR POINTER TO 23$
4775 041100 012700 177762      MOV      #177762,R0     ;PUT ADDRESS OF SIZE HIGH REG INTO R0
4776 041104 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4777 041106 011001      MOV      (R0),R1        ;READ SYSTEM SIZE HIGH REGISTER
4778 041110 001401      BEQ     3$            ;BRANCH IF IT IS ZERO
4779 041112 104027      ERROR   27            ;(177762) IS NOT ZERO
4780 041114 012737 041122 001112 3$:  MOV      #24$,$LPERR    ;SET LOOP ON ERROR POINTER TO 24$
4781 041122 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4782 041124 012710 177777      MOV      #-1,(R0)       ;TRY TO LOAD SIZE HIGH REGISTER
4783 041130 011002      MOV      (R0),R2        ;READ SIZE HIGH REGISTER
4784 041132 001402      BEQ     4$            ;BRANCH IF REGISTER IS STILL ZERO
4785 041134 104026      ERROR   26            ;COULD WRITE SIZE REGISTER
4786 041136 010110      MOV      R1,(R0)        ;RESTORE DATA TO ADDRESS
4787 041140 012737 041022 001112 4$:  MOV      #20$,$LPERR    ;SET LOOP POINTER TO START OF TEST
```

```
4788
4789
4790      ;*****
4791      ;*TEST 3      CPU ERROR REGISTER
4792      ;*
4793      ;*      THE CPU ERROR REGISTER IS TESTED TO SEE THAT IT IS CLEAR
4794      ;*      AFTER A NEGATIVE ONE HAS BEEN LOADED INTO IT.  THERE IS NO
4795      ;*      WAY TO SET ANY BITS UNDER PROGRAM CONTROL.
4796      ;*
4797      ;*****
```

```
4798 041146      TST3:
4799 041146 000004      SCOPE
4800 041150 012737 041212 001316      MOV      #TST4,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
4801      ;TEST FOR ESCAPE ON PARITY ERRORS
4802 041156 012737 041170 001112 20$:  MOV      #1$,$LPERR    ;SET LOOP ON ERROR POINTER TO 1$
4803 041164 012700 177766      MOV      #177766,R0     ;LOAD ADDRESS OF C.P.U. ERROR REG
4804 041170 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
4805 041172 012710 177777      MOV      #-1,(R0)       ;WRITE A NEGATIVE ONE INTO CPU ERROR REG
4806 041176 011001      MOV      (R0),R1        ;READ C.P.U. ERROR REGISTER
4807 041200 001401      BEQ     2$            ;CPU ERROR REGISTER SHOULD BE ZERO
4808 041202 104030      ERROR   30            ;C.P.U. ERROR REGISTER NOT ZERO
4809 041204 012737 041156 001112 2$:  MOV      #20$,$LPERR    ;SET LOOP POINTER TO START OF TEST
```

```
4810
4811
4812      ;*****
4813      ;*TEST 4      MICRO PROGRAM BREAK REGISTER
4814      ;*
4815      ;*      IN A STANDARD KB11-C PROCESSOR
4816
```

```
4817 :* THE MICRO BREAK REGISTER IS A LOW BYTE ONLY REGISTER, WITH
4818 :* THE UPPER BYTE ALWAYS ZERO. THIS TEST LOADS A FULL WORD INTO
4819 :* ADDRESS 177770 AND CHECKS TO SEE THAT ONLY THE LOW BYTE IS
4820 :* STORED IN THE REGISTER. IF THIS IS A (KB11-E)
4821 :* THEN ALL 16 BITS ARE WRITABLE AND THE REG IS TESTED AS SUCH.
4822 :* A ONE IS SHIFTED ACROSS THE REG IN BOTH CASES IN THIS TEST.
4823 :*
4824 :*
4825 :*****
4826 TST4:
4827 041212 000004 SCOPE
4828 041214 012737 041334 001316 MOV #TST5,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4829 :TEST FOR ESCAPE ON PARITY ERRORS
4830 041222 012737 041234 001112 20$: MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
4831 041230 012700 177770 MOV #177770,R0 ;LOAD ADDRESS OF MICRO BREAK REGISTER
4832 041234 000240 1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
4833 041236 005010 CLR (R0) ;START WITH ALL ZERO DATA
4834 041240 011001 MOV (R0),R1 ;READ MICRO BREAK REGISTER
4835 041242 005002 CLR R2 ;CHECK FOR ZERO
4836 041244 020102 CMP R1,R2 ;DID YOU REFERENCE THE MICRO BREAK REG
4837 041246 001020 BNE 3$ ;BRANCH IF DATA WRONG
4838 041250 012703 000001 MOV #1,R3 ;START SHIFTING A ONE
4839 041254 010302 6$: MOV R3,R2 ;SET UP EXPECTED DATA IN R2
4840 041256 010310 MOV R3,(R0) ;WRITE MICRO BREAK REG
4841 041260 011001 MOV (R0),R1 ;READ IT BACK
4842 041262 005737 001360 TST KB11E ;IS THIS A KB11E OR KB11EM PROCESSOR?
4843 041266 001002 BNE 5$ ;BR IF KB11-E
4844 041270 042702 177400 BIC #177400,R2 ;ONLY LOWER BYTE SHOULD BE WRITTEN
4845 041274 020102 5$: CMP R1,R2 ;MATCH?
4846 041276 001004 BNE 3$ ;BR IF NOT
4847 041300 000241 CLC ;CLEAR CARRY FOR ROTATE
4848 041302 006103 ROL R3 ;SHIFT A 1 IN DATA TO WRITE
4849 041304 103407 BCS 2$ ;DONE IF CARRY SETS
4850 041306 000762 BR 6$ ;REPEAT IF NOT DONE
4851 041310 005737 001360 3$: TST KB11E ;ERROR COMES HERE. IS THIS A KB11-E OR KB11-EM?
4852 041314 001402 BEQ 4$ ;BR TO ERROR EM32 IF NOT KB11-E OR KB11-EM
4853 041316 104144 ERROR 144 ;ERROR FOR KB11-E PROCESSOR
4854 041320 000401 BR 2$ ;DONE
4855 041322 104032 4$: ERROR 32 ;DIDN'T LOAD MICRO BREAK REGISTER
4856 041324 005010 2$: CLR (R0) ;LEAVE MICRO BREAK REGISTER ZERO
4857 041326 012737 041222 001112 MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
```

```
4858
4859
4860 :*****
4861 :*TEST 5 PROGRAM INTERRUPT REQUEST REGISTER
4862 :*
4863 :* THE PROGRAM INTERRUPT REQUEST REGISTER'S UPPER BYTE IS LOADED
4864 :* DIRECTLY AND THE LOWER BYTE IS AN ENCODE OF THE UPPER BYTE.
4865 :* THIS TEST LOADS A FULL WORD INTO ADDRESS 177772 AND CHECKS
4866 :* THAT THE UPPER BYTE IS LOADED DIRECTLY AND THE LOWER BYTE
4867 :* IS ENCODED CORRECTLY.
4868 :*
4869 :*****
4870 TST5:
4871 041334 000004 SCOPE
4872 041336 012737 041412 001316 MOV #TST6,NXTTST ;SAVE STARTING ADDRESS OF NEXT
```

```
4873                                     ;TEST FOR ESCAPE ON PARITY ERRORS
4874 041344 012737 041360 001112 20$: MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
4875 041352 000237          SPL 7 ;SET PRIORITY LEVEL AT SEVEN
4876 041354 012700 177772          MOV #177772, R0 ;LOAD ADDRESS OF PROGRAM INTERRUPT REG
4877 041360 000240          NOP ;THIS IS A SYNC POINT FOR SCOPING
4878 041362 012710 004777          MOV #004777, (R0) ;ONLY UPPER BYTE (LESS BIT 8) IS LOADED
4879                                     ;DIRECTLY, LOWER BYTE IS ENCODED
4880 041366 011001          MOV (R0), R1 ;READ PROGRAM INTERRUPT REGISTER
4881 041370 012702 004146          MOV #004146, R2 ;LOAD EXPECTED DATA
4882 041374 020102          CMP R1, R2 ;DID YOU REFERENCE PROGRAM INTERRUPT REG
4883 041376 001401          BEQ 2$ ;BRANCH IF IT WAS THE PROG. INTERRUPT
4884 041400 104033          ERROR 33 ;NOT THE PROG. INT. REQUEST REGISTER
4885 041402 005010          CLR (R0) ;LEAVE THE P.I.R. REGISTER ZERO
4886 041404 012737 041344 001112 2$: MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
```

```
4887
4888
4889 *****
4890 *TEST 6 STACK LIMIT REGISTER
4891 *
4892 * THE STACK LIMIT REGISTER IS A HIGH BYTE ONLY REGISTER, SO THIS
4893 * TEST TRIES TO LOAD BOTH BYTES OF ADDRESS 177774 AND CHECKS
4894 * THAT ONLY THE HIGH BYTE IS LOADED. THE LOW BYTE WILL ALWAYS
4895 * BE READ AS ZERO.
4896 *
4897 *****
```

```
4898 041412 TST6:
4899 041412 000004          SCOPE
4900 041414 012737 041466 001316          MOV #TST7, NXTTST ;SAVE STARTING ADDRESS OF NEXT
4901                                     ;TEST FOR ESCAPE ON PARITY ERRORS
4902 041422 012737 041434 001112 20$: MOV #1$, $LPEPR ;SET LOOP ON ERROR POINTER TO 1$
4903 041430 012700 177774          MOV #177774, R0 ;LOAD ADDRESS OF STACK LIMIT REGISTER
4904 041434 000240          NOP ;THIS IS A SYNC POINT FOR SCOPING
4905 041436 012710 000777          MOV #777, (R0) ;ONLY HIGH BYTE SHOULD BE LOADED
4906 041442 011001          MOV (R0), R1 ;READ STACK LIMIT REGISTER
4907 041444 005010          CLR (R0) ;LEAVE STACK LIMIT REGISTER CLEAR
4908 041446 012702 000400          MOV #400, R2 ;LOAD EXPECTED DATA INTO R2
4909 041452 020102          CMP R1, R2 ;DID YOU REFERENCE THE STACK LIMIT REG
4910 041454 001401          BEQ 2$ ;BRANCH IF IT WAS STACK LIMIT
4911 041456 104034          ERROR 34 ;NOT STACK LIMIT REGISTER
4912 041460 012737 041422 001112 2$: MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
```

```
4913
4914
4915 *****
4916 *TEST 7 PROCESSOR STATUS WORD
4917 *
4918 * THIS TEST SETS THE PRIORITY LEVEL TO 7 AND CLEARS THE
4919 * CONDITION CODES AND CHECKS TO SEE THAT ADDRESS 177776
4920 * HAS 340 IN ITS LOW BYTE.
4921 *
4922 *****
```

```
4923 TST7:
4924 041466          SCOPE
4925 041466 000004          MOV #TST10, NXTTST ;SAVE STARTING ADDRESS OF NEXT
4926 041470 012737 041544 001316          MOV #1$, $LPERR ;TEST FOR ESCAPE ON PARITY ERRORS
4927                                     ;SET LOOP ON ERROR POINTER TO 1$
4928 041476 012737 041514 001112 20$: MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
```

4929	041504	012700	177776			MOV	#177776,R0	:LOAD ADDRESS OF P.S.W. INTO R0
4930	041510	012702	177740			MOV	#177740,R2	:LOAD EXPECTED DATA INTO R2
4931	041514	000237		18:		SPL	7	:SET PRIORITY TO LEVEL SEVEN
4932	041516	000257				CCC		:CLEAR ALL CONDITION CODES
4933	041520	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
4934	041522	111001				MOVB	(R0),R1	:LOAD LOWER BYTE OF P.S.W. INTO R1
4935	041524	042701	000020			BIC	#TBIT,R1	:CLEAR T-BIT IF IT IS ON
4936	041530	020201				CMF	R2,R1	:SIGN EXTEND OF LOWER BYTE OF PSW
4937								:LOWER BYTE SHOULD BE 340
4938	041532	001401				BEQ	2\$:BRANCH IF PSW IS CORRECT
4939	041534	104031				ERROR	31	:MUST HAVE READ SOME OTHER REGISTER
4940	041536	012737	041476	001112	2\$:	MOV	#20\$,\$LPERR	:SET LOOP POINTER TO START OF TEST

4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951

```
*****  
:TEST 10      SET UP THE STACK POINTERS FOR REST OF TESTS  
:  
:      THIS TEST IS USED TO SET THE KERNEL STACK POINTER AT  
:      1100, THE SUPERVISOR STACK POINTER AT 700, AND THE USER  
:      STACK POINTER AT 600.  IT THEN RETURNS TO KERNEL MODE AND  
:      VERIFIES THAT THE KERNEL STACK POINTER IS STILL AT 1100.  
:*****
```

4952
4953
4954
4955
4956
4957
4958
4959
4960
4961
4962
4963
4964
4965
4966
4967
4968

```
TST10:  
SCOPE  
MOV      #TST11,NXTTST  :SAVE STARTING ADDRESS OF NEXT  
:TEST FOR ESCAPE ON PARITY ERRORS  
CLR      PSW             :GET TO KERNEL MODE, PRIORITY 0  
MOV      #KERSTK,KSP    :SETUP KERNEL STACK POINTER  
MOV      #40000,PSW     :GET INTO SUPERVISOR MODE  
MOV      #SUPSTK,SSP    :SET UP SUPERVISOR STACK POINTER  
MOV      #140000,PSW    :GET INTO USER MODE  
MOV      #USESTK,USP    :SET UP USER STACK POINTER  
CLR      PSW             :GET BACK INTO KERNEL MODE  
MOV      KSP,R0         :READ KSRNEL STACK POINTER  
CMP      #KERSTK,R0     :SEE IF KERNEL STACK IS STILL 1100  
BEQ      TST11          :;BRANCH IF KERNEL STACK IS OKAY  
ERROR    35             :KERNEL STACK POINTER IS NOT RIGHT
```

4969
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979

```
.SBITL ***** ENTRY POINT 2 --- STARTING ADDRESS 204 *****  
.SBITL ** TEST READ/WRITE BITS IN MEMORY MANAGEMENT STATUS REGISTERS **  
:  
:      THIS GROUP OF TESTS EXERCISES ALL OF THE READ/WRITE BITS  
:      IN MMR0, TESTS THE PROPER FUNCTIONING OF MMR1, AND TRIES A  
:      FEW VIRTUAL ADDRESSES IN MMR2.  IT ALSO EXERCISES THE BITS  
:      IN MMR3.  THESE TESTS SHOW THAT ONCE THE 'SSRC NO ERROR' FLIP-  
:      FLOP IS SET MMR1 AND MMR2 STOP TRACKING THE C.P.U. OPERATIONS.  
:*****
```

4980
4981
4982
4983
4984

```
*****  
:TEST 11      BIT TEST OF MEMORY MANAGEMENT REGISTER 0  
:  
:      THIS TEST TRIES TO SET AND CLEAR BITS <15:12> AND <09> OF  
:      MMR0.  'INIT' IS ISSUED TO SEE THAT IT WILL CLEAR THESE BITS.  
:*****
```

```

4985      : * THE OTHER BITS OF MMRO ARE CLOCKED ONLY ON MEMORY
4986      : * MANAGEMENT ERROR CONDITIONS IF RELOCATION IS ENABLED.
4987      : * BIT <00> ENABLES FULL RELOCATION AND BIT <08> ENABLES
4988      : * RELOCATION ON THE DESTINATION CYCLE ONLY. THESE BITS WILL
4989      : * BE TESTED IN A LATER TEST.
4990      : *
4991      : *
4992      : * *****

```

```

4992 041626      TST11:
4993 041626 000004      SCOPE
4994 041630 012737 042032 001316      MOV #TST12,NXTTST ;SAVE STARTING ADDRESS OF NEXT
4995                                     ;TEST FOR ESCAPE ON PARITY ERRORS
4996 041636      ENTPT2:
4997 041636 012737 041666 001110      MOV #20$, $LPADR ;SET LOOP ADDRESS POINTER TO 20$
4998 041644 012737 041666 001112      MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
4999 041652 012737 000011 001102      MOV #11, $STNM ;LOAD TEST NUMBER INTO MEMORY
5000 041660 013737 001102 177570      MOV $STNM, DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
5001 041666 012700 177572      20$: MOV #MMRO, R0 ;PUT ADDRESS OF MMRO IN R0
5002 041672 012710 171000      MOV #171000, (R0) ;LOAD WRITABLE BITS IN MMRO
5003 041676 000240      NOP ;THIS IS A SYNC POINT FOR SCOPING
5004 041700 000005      RESET ;'INI1', SHOULD CLEAR ALL BITS OF MMRO
5005 041702 011001      MOV (R0), R1 ;READ MMRO
5006 041704 001401      BEQ 1$ ;BRANCH IF MMRO IS CLEAR
5007 041706 104007      ERROR 7 ;MMRO WON'T CLEAR
5008 041710 012737 041716 001112 1$: MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
5009 041716 000240 11$: NOP ;THIS IS SYNC POINT FOR SCOPING
5010 041720 012710 171000      MOV #171000, (R0) ;SET BITS <15:12><9> OF MMRO
5011 041724 011001      MOV (R0), R1 ;READ MMRO
5012 041726 022701 171000      CMP #171000, R1 ;SEE IF BITS <15:12><9> ARE SET
5013 041732 001401      BEQ 2$ ;BRANCH IF CORRECT BITS ARE SET
5014 041734 104010      ERROR 10 ;CAN'T SET 171000 IN MMRO
5015 041736 012737 041744 001112 2$: MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
5016 041744 000240 12$: NOP ;THIS IS SYNC POINT FOR SCOPING
5017 041746 005010      CLR (R0) ;CLEAR UPPER 7 BITS OF MMRO
5018 041750 011001      MOV (R0), R1 ;READ MMRO
5019 041752 001401      BEQ 3$ ;BRANCH IF MMRO IS CLEAR
5020 041754 104007      ERROR 7 ;MMRO WON'T CLEAR
5021 041756 012737 041774 001112 3$: MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
5022 041764 012702 100000      MOV #BIT15, R2 ;SET BIT IN R2 TO FLOAT THRU MMRO
5023 041770 012703 000007      MOV #7, R3 ;PUT LOOP COUNT IN R3
5024 041774 000240 5$: NOP ;THIS A IS SYNC POINT FOR SCOPING
5025 041776 010210      MOV R2, (R0) ;LOAD R2 INTO MMRO
5026 042000 011001      MOV (R0), R1 ;READ MMRO IN R1
5027 042002 005010      CLR (R0) ;CLEAR MMRO
5028 042004 010204      MOV R2, R4 ;PUT PATTERN IN R4
5029 042006 042704 006777      BIC #006777, R4 ;MASK OFF BITS <11:10><8:0>
5030 042012 020401      CMP R4, R1 ;COMPARE PATTERN WITH MMRO
5031 042014 001401      BEQ 4$ ;BRANCH IF DATA MATCHES
5032 042016 104011      ERROR 11 ;GOT WRONG DATA FROM MMRO
5033 042020 006002 4$: ROR R2 ;SHIFT BIT TO RIGHT
5034 042022 077314      SOB R3, 5$ ;LOOP BACK 6 TIMES
5035 042024 012737 041666 001112      MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
5036
5037
5038      : * *****
5039      : * TEST 12 BIT TEST OF MEMORY MANAGEMENT REGISTER 1
5040      : *

```


5041
5042
5043
5044
5045
5046
5047
5048
5049
5050
5051
5052
5053
5054

THIS TEST WILL CAUSE BITS IN MMR1 TO BE LOCKED BY SETTING BITS <15:13> IN MMR0. THE REGISTER ONLY GETS CLOCKED ON AN INSTRUCTION THAT AUTO INCREMENTS OR AUTO DECREASES A REGISTER. THE LOWER BYTE IS ALWAYS CLOCKED FIRST AND THE UPPER BYTE IS CLOCKED ONLY IF BOTH SOURCE AND DESTINATION AUTO INCREMENT OR DECREMENT A REGISTER. ALL COUNTS (+1,-1,+2,-2) THAT CAN BE GENERATED WITH JUST THE C.P. ARE CLOCKED INTO BOTH HIGH AND LOW BYTES. ALL REGISTER NUMBERS ARE GENERATED, INCLUDING ALL 3 R6'S, AND ALL THE BITS THAT HOLD THE REGISTER NUMBER IN BOTH BYTES ARE TESTED. HOWEVER NOT ALL REGISTER NUMBERS ARE CLOCKED INTO BOTH HIGH AND LOW BYTES, BUT ENOUGH ARE USED TO TEST THE LOGIC.

5055 042032
5056 042032 000004
5057 042034 012737 042464 001316
5058
5059 042042 012701 177574 208:
5060 042046 012704 020000
5061 042052 012737 042060 001112
5062 042060 000240 118:
5063 042062 012737 100000 177572
5064
5065 042070 012703 013427
5066 042074 011102
5067 042076 020203
5068 042100 001401
5069 042102 104013
5070 042104 012737 042112 001112 108:
5071 042112 000240 128:
5072 042114 000005
5073 042116 011102
5074 042120 001401
5075 042122 104012
5076 042124 012737 042132 001112 18:
5077 042132 012700 177572 138:
5078 042136 000240
5079 042140 012720 040000
5080
5081 042144 011102
5082 042146 012703 010027
5083 042152 020203
5084 042154 001401
5085 042156 104013
5086 042160 005037 177572 28:
5087 042164 012737 042172 001112
5088 042172 012700 177574 148:
5089 042176 000240
5090 042200 010440
5091
5092 042202 011102
5093 042204 012703 000360
5094 042210 020203
5095 042212 001401
5096 042214 104013

ST12:
 SCOPE
 MOV #TST13,NXTTST ;SAVE STARTING ADDRESS OF NEXT
 ;TEST FOR ESCAPE ON PARITY ERRORS
 MOV MMR1,R1 ;PUT ADDRESS OF MMR1 IN R1
 MOV #BIT13,R4 ;SET READ ONLY BIT IN R4
 MOV #118,\$LPERR ;SET LOOP ON ERROR POINTER TO 118
 NOP ;THIS IS SYNC POINT FOR SCOPING
 MOV #BIT15,@MMR0 ;LOCK UP MMR1 LOWER BYTE 027
 ;UPPER BYTE 027-WORD 013427
 MOV #013427,R3 ;PUT EXPECTED DATA IN R3
 MOV (R1),R2 ;READ MMR1 INTO R2
 CMP R2,R3 ;SEE IF DATA MATCHES
 BEQ 108 ;BRANCH IF DATA MATCHES
 ERROR 13 ;MMR1 DID NOT TRACK PROPERLY
 MOV #128,\$LPERR ;SET LOOP ON ERROR POINTER TO 128
 NOP ;THIS IS SYNC POINT FOR SCOPING
 RESET ;ISSUE INIT
 MOV (R1),R2 ;SEE IF MMR1 IS CLEARED
 BEQ 18 ;BRANCH IF MMR1 IS ZERO
 ERROR 12 ;CAN'T CLEAR MMR1
 MOV #138,\$LPERR ;SET LOOP ON ERROR POINTER TO 138
 MOV MMR0,R0 ;PUT ADDRESS OF MMR0 INTO R0
 NOP ;THIS IS SYNC POINT FOR SCOPING
 MOV #BIT14,(R0)+ ;LOCK UP MMR1-LOWER BYTE 027
 ;UPPER BYTE 020-WORD 010027
 MOV (R1),R2 ;READ MMR1
 MOV #010027,R3 ;PUT EXPECTED DATA IN R3
 CMP R2,R3 ;SEE IF DATA MATCHES
 BEQ 28 ;BRANCH IF DATA MATCHES
 ERROR 13 ;MMR1 DID NOT TRACK PROPERLY
 CLR @MMR0 ;CLEAR MMR0
 MOV #148,\$LPERR ;SET LOOP ON ERROR POINTER TO 148
 MOV MMR0+2,R0 ;PUT ADDRESS OF MMR0 PLUS 2 IN R0
 NOP ;THIS IS SYNC POINT FOR SCOPING
 MOV R4,-(R0) ;LOCK UP MMR1-LOWER BYTE 360
 ;UPPER BYTE 000-WORD 000360
 MOV (R1),R2 ;READ MMR1
 MOV #000360,R3 ;PUT EXPECTED DATA IN R3
 CMP R2,R3 ;SEE IF DATA MATCHES
 BEQ 38 ;BRANCH IF DATA MATCHES
 ERROR 13 ;MMR1 DID NOT TRACK PROPERLY

5097	042216	005010			3\$:	CLR	(R0)	;CLEAR MMRO
5098	042220	012737	042226	001112		MOV	#15\$,SLPERR	;SET LOOP ON ERROR POINTER TO 15\$
5099	042226	012705	177573		15\$:	MOV	#MMR0+,R5	;PUT ADDRESS OF MMRO'S UPPER BYTE IN R5
5100	042232	012702	001321			MOV	#READON+1,R2	;PUT ADDRESS OF READ ONLY BIT <13> IN R2
5101	042236	000240				NOP		;THIS IS SYNC POINT FOR SCOPING
5102	042240	112225				MOVB	(R2)+,(R5)+	;LOCK UP MMR1-LOWER BYTE 012
5103								;UPPER BYTE 015-WORD 006412
5104	042242	011102				MOV	(R1),R2	;READ MMR1
5105	042244	012703	006412			MOV	#006412,R3	;PUT EXPECTED DATA IN R3
5106	042250	020203				CMP	R2,R3	;SEE IF DATA MATCHES
5107	042252	001401				BEQ	4\$;BRANCH IF DATA MATCHES
5108	042254	104013				ERROR	13	;MMR1 DID NOT TRACK PROPERLY
5109	042256	005010			4\$:	CLR	(R0)	;CLEAR MMRO
5110	042260	012737	042266	001112		MOV	#16\$,SLPERR	;SET LOOP ON ERROR POINTER TO 16\$
5111	042266	012702	177574		16\$:	MOV	#MMR0+2,R2	;PUT ADDRESS OF MMRO'S UPPER BYTE
5112								;PLUS 1 INTO R2
5113	042272	012705	001322			MOV	#READON+2,R5	;PUT ADDRESS OF READ ONLY BIT <13>
5114								;PLUS 2 IN R5
5115	042276	000240				NOP		;THIS IS SYNC POINT FOR SCOPING
5116	042300	114542				MOVB	-(R5),-(R2)	;LOCK UP MMR1-LOWER BYTE 375
5117								;UPPER BYTE 372-WORD 175375
5118	042302	011102				MOV	(R1),R2	;READ MMR1
5119	042304	012703	175375			MOV	#175375,R3	;PUT EXPECTED DATA IN R3
5120	042310	020203				CMP	R2,R3	;SEE IF DATA MATCHES
5121	042312	001401				BEQ	5\$;BRANCH IF DATA MATCHES
5122	042314	104013				ERROR	13	;MMR1 DID NOT TRACK PROPERLY
5123	042316	005010			5\$:	CLR	(R0)	;CLEAR MMRO
5124	042320	012737	042340	001112		MOV	#17\$,SLPERR	;SET LOOP ON ERROR POINTER TO 17\$
5125	042326	012737	040000	177776		MOV	#40000,@PSW	;SET SUPERVISOR MODE
5126	042334	010637	001172			MOV	SSP,@STMP0	;SAVE SUPERVISOR STACK POINTER
5127	042340	012706	177574		17\$:	MOV	#MMR0+2,SSP	;PUT ADDRESS OF MMRO+2 IN SSP
5128	042344	000240				NOP		;THIS IS SYNC POINT FOR SCOPING
5129	042346	012746	040000			MOV	#40000,-(SSP)	;LOCK UP MMR1-LOWER BYTE 027
5130								;UPPER BYTE 366-WORD 173027
5131	042352	011102				MOV	(R1),R2	;READ MMR1
5132	042354	012703	173027			MOV	#173027,R3	;PUT EXPECTED DATA IN R3
5133	042360	020203				CMP	R2,R3	;SEE IF DATA MATCHES
5134	042362	001401				BEQ	6\$;BRANCH IF DATA MATCHES
5135	042364	104013				ERROR	13	;MMR1 DID NOT TRACK PROPERLY
5136	042366	013706	001172		6\$:	MOV	STMP0,SSP	;RESTORE THE SUPERVISOR STACK POINTER
5137	042372	005010				CLR	(R0)	;CLEAR MMRO
5138	042374	012737	042414	001112		MOV	#18\$,SLPERR	;SET LOOP ON ERROR POINTER TO 18\$
5139	042402	012737	140000	177776		MOV	#140000,@PSW	;SET USER MODE
5140	042410	010637	001172			MOV	USP,@STMP0	;SAVE USER STACK POINTER
5141	042414	012706	177572		18\$:	MOV	#MMR0,USP	;PUT ADDRESS OF MMRO IN USP
5142	042420	000240				NOP		;THIS IS SYNC POINT FOR SCOPING
5143	042422	012726	100000			MOV	#100000,(USP)+	;LOCK UP MMR1-LOWER BYTE 027
5144								;UPPER BYTE 026-WORD 013027
5145	042426	011102				MOV	(R1),R2	;READ MMR1
5146	042430	012703	013027			MOV	#013027,R3	;PUT EXPECTED DATA IN R3
5147	042434	020203				CMP	R2,R3	;SEE IF DATA MATCHES
5148	042436	001401				BEQ	7\$;BRANCH IF DATA IS GOOD
5149	042440	104013				ERROR	13	;MMR1 DID NOT TRACK PROPERLY
5150	042442	013706	001172		7\$:	MOV	@STMP0,USP	;RESTORE USER STACK POINTER
5151	042446	005037	177776			CLR	@PSW	;GO BACK TO KERNEL MODE
5152	042452	005037	177572			CLR	@MMR0	;LET ALL M.M.R'S TRACK

5153 042456 012737 042042 001112 MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST

5154
5155
5156
5157
5158
5159
5160
5161
5162
5163
5164
5165
5166
5167
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188
5189
5190
5191
5192
5193
5194
5195
5196
5197
5198
5199
5200
5201
5202
5203
5204
5205
5206
5207
5208

```
*****  
*TEST 13 BIT TEST OF MEMORY MANAGEMENT REGISTER 2  
*  
* HERE MMR2 WILL BE READ SEVERAL DIFFERENT TIMES TO  
* SEE THAT IT IS TRACKING PROPERLY. THEN IT WILL BE  
* LOCKED UP AND READ IT TO SEE THAT IT DOES NOT CHANGE AFTER  
* IT IS ONCE LOCKED UP. NOT ALL BITS WILL BE TESTED IN THIS  
* TEST BUT A LATER TEST RUNS A COUNT PATTERN THROUGH MMR2. THIS  
* CANNOT BE DONE HERE SINCE IT REQUIRES THAT THE ABORT LOGIC IS  
* FUNCTIONING PROPERLY.  
*****
```

```
TST13:  
SCOPE  
MOV #TST14,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
;TEST FOR ESCAPE ON PARITY ERRORS  
20$: MOV #MMR0,R0 ;PUT ADDRESS OF MMR0 IN R0  
MOV #MMR2,R1 ;PUT ADDRESS OF MMR2 IN R1  
MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$  
CLR @MMR0 ;MAKE SURE THAT MMR0 IS CLEAR  
11$: NOP ;THIS A IS SYNC POINT FOR SCOPING  
21$: MOV (R1),R2 ;READ MMR2 TO R2  
MOV #21$,R3 ;PUT ADDRESS OF LAST INST IN R3  
CMP R2,R3 ;SEE IF MMR2 HELD CORRECT ADDRESS  
BEQ 1$ ;BRANCH IF DATA MATCHES  
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY  
1$: MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$  
12$: NOP ;THIS A IS SYNC POINT FOR SCOPING  
22$: MOV @MMR2,R2 ;READ MMR2 TO R2  
MOV #22$,R3 ;PUT ADDRESS OF LAST INST IN R3  
CMP R2,R3 ;SEE IF MMR2 HELD CORRECT ADDRESS  
BEQ 2$ ;BRANCH IF DATA MATCHES  
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY  
2$: MOV #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$  
13$: NOP ;THIS A IS SYNC POINT FOR SCOPING  
23$: MOV #40000,(R0) ;LOCK UP MMR1 AND MMR2  
MOV (R1),R2 ;READ MMR2 INTO R2  
MOV #23$,R3 ;PUT LOCKING INST'S ADDRESS IN R3  
CMP R2,R3 ;SEE IF MMR2 HOLDS RIGHT ADDRESS  
BEQ 3$ ;BRANCH IF DATA MATCHES  
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY  
3$: MOV #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$  
14$: RESET ;ISSUE INIT TO CLEAR MMR1 & 2  
15$: NOP ;THIS A IS SYNC POINT FOR SCOPING  
25$: MOV (R1),R2 ;READ MMR2 INTO R2  
MOV #25$,R3 ;PUT ADDRESS OF LAST INST IN R3  
CMP R2,R3 ;SEE IF MMR2 IS STILL TRACKING  
BEQ TST14 ;GO TO NEXT TEST IF IT IS  
ERROR 14 ;MMR2 DID NOT TRACK PROPERLY
```

```
*****  
*TEST 14 BIT TEST OF MEMORY MANAGEMENT REGISTER 3  
*  
*****
```

5209 : * THIS TEST SETS AND CLEARS BITS <05:04> AND <02:00> OF MMR3
5210 : * IT DOES NOT TEST THAT THE BITS FUNCTION PROPERLY SINCE THAT
5211 : * REQUIRES MORE LOGIC. BUT IT DOES TEST THE REGISTER AND THE
5212 : * DATA PATH TO AND FROM THE REGISTER. THE PROPER FUNCTIONING
5213 : * OF THESE BITS IS TESTED LATER IN SEVERAL TESTS.
5214 : *
5215 : *

```
*****  
TST14:  
5216 042640  
5217 042640 000004  
5218 042642 012737 043004 001316 SCOPE  
5219 MOV #TST15,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
5220 042650 012700 172516 20$: MOV #MMR3,R0 ;TEST FOR ESCAPE ON PARITY ERRORS  
5221 042654 012701 000001 MOV #1,R1 ;PUT ADDRESS OF MMR3 IN R0  
5222 042660 005010 CLR (R0) ;SET BIT TO FLOAT THRU MMR3  
5223 042662 011002 MOV (R0),R2 ;CLEAR MMR3  
5224 042664 001401 BEQ 5$ ;READ MMR3 INTO R2  
5225 042666 104015 ERROR 15 ;BRANCH IF ZERO  
5226 042670 012703 000006 5$: MOV #6,R3 ;CAN'T CLEAR MMR3  
5227 042674 012737 042702 001112 MOV #1$,$LPERR ;SET UP LOOP COUNT  
5228 042702 000240 1$: NOP ;SET LOOP ON ERROR POINTER TO 1$  
5229 042704 010110 MOV R1,(R0) ;THIS A IS SYNC POINT FOR SCOPING  
5230 042706 011002 MOV (R0),R2 ;LOAD MMR3  
5231 042710 020102 CMP R1,R2 ;READ MMR3 INTO R2  
5232 042712 001404 BEQ 2$ ;DOES DATA MATCH PATTERN  
5233 042714 032701 000010 BIT #BIT3,R1 ;BRANCH IF IT MATCHES  
5234 042720 001001 BNE 2$ ;WAS THIS BIT 3  
5235 042722 104016 ERROR 16 ;BRANCH IF IT WAS  
5236 042724 006301 2$: ASL R1 ;MMR3 HELD WRONG DATA  
5237 042726 077313 SOB R3,1$ ;LEFT SHIFT DATA PATTERN  
5238 042730 012701 000067 MOV #67,R1 ;BRANCH BACK IF R3 NOT 0  
5239 042734 012737 042742 001112 MOV #12$,$LPERR ;PUT DATA PATTERN IN R1  
5240 042742 000240 12$: NOP ;SET LOOP ON ERROR POINTER TO 12$  
5241 042744 010110 MOV R1,(R0) ;THIS A IS SYNC POINT FOR SCOPING  
5242 042746 011002 MOV (R0),R2 ;TRY TO SET ALL BITS IN MMR3  
5243 042750 020102 CMP R1,R2 ;READ MMR3 INTO R2  
5244 042752 001401 BEQ 3$ ;SEE IF ALL BITS GOT SET  
5245 042754 104016 ERROR 16 ;BRANCH IF ALL BITS WERE SET  
5246 ;MMR3 HELD WRONG DATA  
5247 042756 012737 042764 001112 3$: MOV #13$,$LPERR ;SET LOOP ON ERROR POINTER TO 13$  
5248 042764 000240 13$: NOP ;THIS A IS SYNC POINT FOR SCOPING  
5249 042766 000005 RESET ;ISSUE 'INIT'  
5250 042770 011001 MOV (R0),R1 ;READ MMR3 INTO R2  
5251 042772 001401 BEQ 4$ ;BRANCH IF MMR3 INTO R2  
5252 042774 104015 ERROR 15 ;CAN'T CLEAR MMR3  
5253 042776 012737 042650 001112 4$: MOV #20$,$LPERR ;RESTORE LOOP POINTER TO FIRST INST.
```

5254
5255 .SBTTL ***** ENTRY POINT 3 --- STARTING ADDRESS 210 *****
5256 .SBTTL ***** PAGE ADDRESS AND DESCRIPTOR REGISTER TESTS *****
5257 : *
5258 : * THIS GROUP OF TESTS IS USED TO CHECK ALL THE PAR'S AND
5259 : * PDR'S; THEIR ADDRESS DECODING ON DIRECT REFERENCES, THEIR
5260 : * READ/WRITE BITS, AND DUAL ADDRESSING WITHIN A GROUP OR BETWEEN
5261 : * TWO GROUPS.
5262 : *
5263 : *
5264

5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283 043004
5284 043004 000004
5285 043006 012737 043140 001316
5286
5287 043014
5288 043014 012737 043044 001110
5289 043022 012737 043044 001112
5290 043030 012737 000015 001102
5291 043036 013737 001102 177570
5292 043044 004737 031346
5293 043050 012737 031404 000004
5294 043056 012737 043070 001112
5295 043064 012700 172340
5296 043070 000240
5297 043072 011001
5298 043074 062700 000002
5299 043100 022700 172376
5300 043104 103371
5301 043106 012737 043044 001112
5302 043114 012737 031670 000004
5303 043122 005737 001302
5304 043126 001404
5305 043130 013737 001302 001204
5306 043136 104017
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320 043140

.SBTTL TEST THAT ALL P.A.R.'S & P.D.R.'S RESPOND
* THESE NEXT SIX (6) TESTS VERIFY THAT THERE ARE 6 GROUPS OF
* 16 CONTIGUOUS ADDRESSES IN THE I/O PAGE THAT WILL RESPOND
* WITHOUT TIMING OUT. AT THIS POINT THE TEST IS NOT CONCERNED
* WITH EXACTLY WHAT IS RESPONDING, JUST THAT SOMETHING RESPONDS.

*TEST 15 READ ALL KERNEL PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
*
* THIS TEST DOES A READ FROM ALL THE KERNEL PAGE ADDRESS REGISTERS
* 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE TEST AND
* IF ANY OF THE 16 KERNEL ADDRESS REGISTERS TIME OUT THEIR I/O
* PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END OF THE TEST A
* SUMMARY OF THE ERRORS IS GIVEN AND 'ERRVEC' IS RE-SET TO 'CPUER'.
*
* ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'

TST15:
SCOPE
MOV #TST16,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
ENTPT3:
MOV #20\$,SLPADR ;SET LOOP ADDRESS POINTER TO 20\$
MOV #20\$,SLPERR ;SET LOOP ON ERROR POINTER TO 20\$
MOV #15,\$STSTM ;LOAD TEST NUMBER INTO MEMORY
MOV \$STSTM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
20\$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
MOV #KIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
1\$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #KDPAR7,R0 ;SEE IF KDPAR7 HAS BEEN TRIED
BHS 1\$;BRANCH IF NOT DONE
MOV #20\$,SLPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
BEQ TST16 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,\$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 17 ;SUMMARY OF PAR ERRORS

*TEST 16 READ ALL SUPERVISOR PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT
*
* THIS TEST DOES A READ FROM ALL THE SUPERVISOR PAGE ADDRESS
* REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
* TEST, AND IF ANY OF THE 16 SUPERVISOR ADDRESS REGISTERS TIME OUT
* THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
* OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
* 'CPUER'.
* ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'

TST16:

5321	043140	000004				SCOPE		
5322	043142	012737	043244	001316		MOV	#TST17,NXTTST	:SAVE STARTING ADDRESS OF NEXT
5323								:TEST FOR ESCAPE ON PARITY ERRORS
5324	043150	004737	031346		20\$:	JSR	PC,CLEANUP	:INITIALIZE ERROR LOCATIONS
5325	043154	012737	031404	000004		MOV	#TIMEOUT,ERRVEC	:SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
5326	043162	012737	043174	001112		MOV	#1\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 1\$
5327	043170	012700	172240			MOV	#SIPARO,R0	:PUT ADDRESS OF FIRST PAR IN R0
5328	043174	000240			1\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
5329	043176	011001				MOV	(R0),R1	:THIS WILL TIMEOUT IF REGISTER DECODING BAD
5330	043200	062700	000002			ADD	#2,R0	:POINT TO NEXT REGISTER
5331	043204	022700	172276			CMP	#SDPAR7,R0	:SEE IF SDPAR7 HAS BEEN TRIED
5332	043210	103371				BHIS	1\$:BRANCH IF NOT DONE
5333	043212	012737	043150	001112		MOV	#20\$, \$LPERR	:PUT LOOP ON ERROR POINTER TO START OF TEST
5334	043220	012737	031670	000004		MOV	#CPUER,ERRVEC	:RE-SET NORMAL CPU TRAP SERVICE ROUTINE
5335	043226	005737	001302			TST	ERRCNT	:SEE IF ANY ERRORS OCCURRED ON THIS TEST
5336	043232	001404				BEQ	TST17	:BRANCH TO NEXT TEST IF NO ERRORS
5337	043234	013737	001302	001204		MOV	ERRCNT,\$TMP5	:SAVE NUMBER OF ERRORS FOR TYPEOUT
5338	043242	104017				ERROR	17	:SUMMARY OF PAR ERRORS

:TEST 17 READ ALL USER PAGE ADDRESS REGISTERS, CHECK FOR TIMEOUT

:
: THIS TEST DOES A READ FROM ALL THE USER PAGE ADDRESS
: REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
: TEST, AND IF ANY OF THE 16 USER ADDRESS REGISTERS TIME OUT
: THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
: OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
: 'CPUER'.
: ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'

TST17:

5352	043244					SCOPE		
5353	043244	000004				MOV	#TST20,NXTTST	:SAVE STARTING ADDRESS OF NEXT
5354	043246	012737	043350	001316				:TEST FOR ESCAPE ON PARITY ERRORS
5355								:INITIALIZE ERROR LOCATIONS
5356	043254	004737	031346		20\$:	JSR	PC,CLEANUP	:SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
5357	043260	012737	031404	000004		MOV	#TIMEOUT,ERRVEC	:SET LOOP ON ERROR POINTER TO 1\$
5358	043266	012737	043300	001112		MOV	#1\$, \$LPERR	:PUT ADDRESS OF FIRST PAR IN R0
5359	043274	012700	177640			MOV	#UIPARO,R0	:THIS IS A SYNC POINT FOR SCOPING
5360	043300	000240			1\$:	NOP		:THIS WILL TIMEOUT IF REGISTER DECODING BAD
5361	043302	011001				MOV	(R0),R1	:POINT TO NEXT REGISTER
5362	043304	062700	000002			ADD	#2,R0	:SEE IF UDPAR7 HAS BEEN TRIED
5363	043310	022700	177676			CMP	#UDPAR7,R0	:BRANCH IF NOT DONE
5364	043314	103371				BHIS	1\$:PUT LOOP ON ERROR POINTER TO START OF TEST
5365	043316	012737	043254	001112		MOV	#20\$, \$LPERR	:RE-SET NORMAL CPU TRAP SERVICE ROUTINE
5366	043324	012737	031670	000004		MOV	#CPUER,ERRVEC	:SEE IF ANY ERRORS OCCURRED ON THIS TEST
5367	043332	005737	001302			TST	ERRCNT	:BRANCH TO NEXT TEST IF NO ERRORS
5368	043336	001404				BEQ	TST20	:SAVE NUMBER OF ERRORS FOR TYPEOUT
5369	043340	013737	001302	001204		MOV	ERRCNT,\$TMP5	:SUMMARY OF PAR ERRORS
5370	043346	104017				ERROR	17	

:TEST 20 READ ALL KERNEL PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT

:
: THIS TEST DOES A READ FROM ALL THE KERNEL PAGE DESCRIPTOR
: REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE

5371
5372
5373
5374
5375
5376

5377 :: TEST, AND IF ANY OF THE 16 KERNEL DESCRIPTOR REGISTERS TIME OUT
5378 :: THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
5379 :: OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
5380 :: 'CPUER'.
5381 :: ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'
5382 ::
5383 ::

```
*****  
TST20:  
5384 043350 SCOPE  
5385 043350 000004 MOV #TST21,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
5386 043352 012737 043454 001316 JSR PC,CLEANUP ;TEST FOR ESCAPE ON PARITY ERRORS  
5387 ;INITIALIZE ERROR LOCATIONS  
5388 043360 004737 031346 20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS  
5389 043364 012737 031404 000004 MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE  
5390 043372 012737 043404 001112 MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$  
5391 043400 012700 172300 MOV #KIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0  
5392 043404 000240 1$: NOP ;THIS IS A SYNC POINT FOR SCOPING  
5393 043406 011001 MOV (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD  
5394 043410 062700 000002 ADD #2,R0 ;POINT TO NEXT REGISTER  
5395 043414 022700 172336 CMP #KDPDR7,R0 ;SEE IF KDPDR7 HAS BEEN TRIED  
5396 043420 103371 BHS 1$ ;BRANCH IF NOT DONE  
5397 043422 012737 043360 001112 MOV #20$, $LPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST  
5398 043430 012737 031670 000004 MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE  
5399 043436 005737 001302 TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST  
5400 043442 001404 BEQ TST21 ;BRANCH TO NEXT TEST IF NO ERRORS  
5401 043444 013737 001302 001204 MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT  
5402 043452 104017 ERROR 17 ;SUMMARY OF PDR ERRORS  
5403  
5404
```

```
*****  
*TEST 21 READ ALL SUPERVISOR PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT  
5405 ::  
5406 ::  
5407 :: THIS TEST DOES A READ FROM ALL THE SUPERVISOR PAGE DESCRIPTOR  
5408 :: REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE  
5409 :: TEST, AND IF ANY OF THE 16 SUPERVISOR DESCRIPTOR REGISTERS TIME OUT  
5410 :: THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END  
5411 :: OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO  
5412 :: 'CPUER'.  
5413 :: ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'  
5414 ::  
5415 ::
```

```
*****  
TST21:  
5416 043454 SCOPE  
5417 043454 000004 MOV #TST22,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
5418 043456 012737 043560 001316 JSR PC,CLEANUP ;TEST FOR ESCAPE ON PARITY ERRORS  
5419 ;INITIALIZE ERROR LOCATIONS  
5420 043464 004737 031346 20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS  
5421 043470 012737 031404 000004 MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE  
5422 043476 012737 043510 001112 MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$  
5423 043504 012700 172200 MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0  
5424 043510 000240 1$: NOP ;THIS IS A SYNC POINT FOR SCOPING  
5425 043512 011001 MOV (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD  
5426 043514 062700 000002 ADD #2,R0 ;POINT TO NEXT REGISTER  
5427 043520 022700 172236 CMP #SDPDR7,R0 ;SEE IF SDPDR7 HAS BEEN TRIED  
5428 043524 103371 BHS 1$ ;BRANCH IF NOT DONE  
5429 043526 012737 043464 001112 MOV #20$, $LPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST  
5430 043534 012737 031670 000004 MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE  
5431 043542 005737 001302 TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST  
5432 043546 001404 BEQ TST22 ;BRANCH TO NEXT TEST IF NO ERRORS
```

5433 043550 013737 001302 001204
5434 043556 104017
5435
5436
5437
5438
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448 043560
5449 043560 000004
5450 043562 012737 043664 001316
5451
5452 043570 004737 031346
5453 043574 012737 031404 000004 20\$:
5454 043602 012737 043614 001112
5455 043610 012700 177600
5456 043614 000240 1\$:
5457 043616 011001
5458 043620 062700 000002
5459 043624 022700 177636
5460 043630 103371
5461 043632 012737 043570 001112
5462 043640 012737 031670 000004
5463 043646 005737 001302
5464 043652 001404
5465 043654 013737 001302 001204
5466 043662 104017
5467
5468
5469
5470
5471
5472

MOV ERRCNT,\$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 17 ;SUMMARY OF PDR ERRORS

*TEST 22 READ ALL USER PAGE DESCRIPTOR REGISTERS, CHECK FOR TIMEOUT

THIS TEST DOES A READ FROM ALL THE USER PAGE DESCRIPTOR
REGISTERS. 'ERRVEC' IS SET TO 'TIMEOUT' AT THE BEGINNING OF THE
TEST, AND IF ANY OF THE 16 USER DESCRIPTOR REGISTERS TIME OUT
THEIR I/O PAGE ADDRESS IS REPORTED AND LOGGED. AT THE END
OF THE TEST A SUMMARY OF ERRORS IS GIVEN AND 'ERRVEC' IS SET TO
'CPUER'.
ALL ERRORS IN THIS TEST ARE REPORTED BY SUBROUTINE 'TIMEOUT'

TST22:
SCOPE
MOV #TST23,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #TIMEOUT,ERRVEC ;SET CPU TRAP VECTOR TO TIMEOUT ROUTINE
MOV #1\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST PDR IN R0
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R1 ;THIS WILL TIMEOUT IF REGISTER DECODING BAD
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #UDPDR7,R0 ;SEE IF UDPDR7 HAS BEEN TRIED
BHS 1\$;BRANCH IF NOT DONE
MOV #20\$,\$LPERR ;PUT LOOP ON ERROR POINTER TO START OF TEST
MOV #CPUER,ERRVEC ;RE-SET NORMAL CPU TRAP SERVICE ROUTINE
TST ERRCNT ;SEE IF ANY ERRORS OCCURRED ON THIS TEST
BEQ TST23 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,\$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 17 ;SUMMARY OF PDR ERRORS

.SBTTL TEST FOR DUAL ADDRESSING ON LOAD WITHIN GROUPS
* THESE NEXT SIX (6) TESTS WILL CHECK FOR DUAL ADDRESSING WITHIN A
* GROUP OF PAR'S OR PDR'S. FIRST ALL OF THE REGISTERS IN A GROUP
* ARE CLEARED AND READ TO SEE THAT THEY CAN EACH HOLD ZEROES, AND
* THAT THE DATA PATH DOES NOT HAVE A BIT STUCK AT ONE.

5473

.*

THEN ONE REGISTER AT A TIME, WITHIN THAT GROUP, IS LOADED

5474 :*
5475 :*
5476 :*

WITH A NEGATIVE ONE WHILE ALL REGISTERS ARE READ TO SEE
THAT ONLY THE REGISTER UNDER TEST WAS NOT ZERO.

5477
5478 *TEST 23 DUAL ADDRESS KERNEL PAGE ADDRESS REGISTERS, ON LOADING
5479 *
5480 * THIS TEST FIRST CLEARS ALL THE KERNEL PAGE ADDRESS REGISTERS,
5481 * AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING
5482 *
5483 * WITH I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME
5484 * IS LOADED WITH A NEGATIVE ONE. ALL KERNEL ADDRESS REGISTERS
5485 * ARE NOW READ TO SEE THAT ONLY THE REGISTER UNDER TEST IS NON-
5486 * ZERO. INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM
5487 * IS GIVEN AT THE END OF THIS TEST.
5488 *
5489 * ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
5490 * 'DUALADR'.
5491 *.....

```
TST23:
043664 000004 044074 001316 SCOPE
043664 012737 044074 001316 MOV #TST24,NXTTST ;SAVE STARTING ADDRESS OF NEXT
043666 012737 044074 001316 ;TEST FOR ESCAPE ON PARITY EFRORS
043674 004737 031346 20$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
043700 012737 043726 001112 MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
043706 012705 172340 MOV #KIPAR0,R5 ;PUT ADDRESS OF FIRST PAR IN R5
043712 004737 031330 JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
043716 012700 172340 MOV #KIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
043722 012701 000020 MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
043726 000240 1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
043730 011002 MOV (R0),R2 ;READ PAR TO R2
043732 001401 BEQ 2$ ;BRANCH IF PAR IS 0
043734 104020 ERROR 20 ;PAR NOT ZERO
043736 062700 000002 2$: ADD #2,R0 ;POINT TO NEXT REGISTER
043742 077107 SOB R1,1$ ;BRANCH BACK TO 1$ 15 TIMES

::
:: NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
:: REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
:: IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
::
043744 012737 043756 001112 MOV #3$, $LPERR ;SET LOOP ON ERROR POINTER TO 3$
043752 012700 172340 MOV #KIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IS R0
043756 012705 172340 3$: MOV #KIPAR0,R5 ;LOAD STARTING ADDRESS INTO R5
043762 004737 031330 JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
043766 012701 172376 MOV #KDPAR7,R1 ;PUT KDPAR7 ADDRESS IN R1
043772 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
043774 012710 177777 MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
044000 005037 001172 4$: CLR $TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
044004 011102 MOV (R1),R2 ;READ ALL REGISTERS
044006 001406 BEQ 6$ ;BRANCH IF REGISTER IS 0
044010 020001 CMP R0,R1 ;IS ADDRESS OF NON-ZERO REGISTER SAME
044012 001402 BEQ 5$ ;AS THAT OF REGISTER UNDER TEST
044014 004737 031524 JSR PC,DUALADR ;BRANCH IF ADDRESSES MATCH
044020 005237 001172 5$: INC $TMP0 ;LOG AND REPORT ERRORS
044024 162701 000002 6$: SUB #2,R1 ;SET FLAG WHEN ADDRESSES MATCH
044030 022701 172340 CMP #KIPAR0,R1 ;POINT TO NEXT REGISTER
044034 101761 BLOS 4$ ;SEE IF ALL REGISTERS HAVE BEEN READ
044036 062700 000002 ADD #2,R0 ;BRANCH IF MORE TO READ
044042 022700 172376 CMP #KDPAR7,R0 ;NOW LOAD NEXT REGISTER
;SEE IF THERE ARE MORE REGISTERS TO TEST
```

5533	044046	103343		
5534	044050	012737	043674	001112
5535	044056	005737	001302	
5536	044062	001404		
5537	044064	013737	001302	001204
5538	044072	104021		
5539				

BHIS	3\$:BRANCH IF MORE REGISTERS TO TEST
MOV	#20\$, \$LPERR	:SET LOOP ON ERROR POINTER TO START OF TEST
TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
BEQ	TST24	:BRANCH TO NEXT TEST IF NO ERRORS
MOV	ERRCNT, \$TMP5	:SAVE NUMBER OF ERRORS FOR PAR OUT
ERROR	21	:SUMMARY OF DUAL ADDRESSING ERRORS

5540 :*****~
5541 :*TEST 24 DUAL ADDRESS SUPERVISOR PAGE ADDRESS REGISTERS, ON LOADING
5542 :
5543 : THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE ADDRESS
5544 : AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH
5545 : I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
5546 : LOADED WITH A NEGATIVE ONE. ALL SUPERVISOR ADDRESS REGISTERS
5547 : ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
5548 : INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
5549 : THE END OF THIS TEST.
5550 :
5551 : ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
5552 : 'DUALADR'.
5553 :
5554 :*****~

5555 044074 TST24:
5556 044074 000004 SCOPE
5557 044076 012737 044304 001316 MOV #TST25,NXTTST ;SAVE STARTING ADDRESS OF NEXT
5558 : ;TEST FOR ESCAPE ON PARITY ERRORS
5559 044104 004737 031346 20\$: JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
5560 044110 012737 044136 001112 MOV #1\$,SLPERR ;SET LOOP ON ERROR POINTER TO 1\$
5561 044116 012705 172240 MOV #SIPAR0,R5 ;PUT ADDRESS OF FIRST PAR IN R5
5562 044122 004737 031330 JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
5563 044126 012700 172240 MOV #SIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R0
5564 044132 012701 000020 MOV #20,R1 ;BRANCH COUNT IS 16 DECIMAL
5565 044136 000240 1\$: NOP ;THIS IS A SYNC POINT FOR SCOPING
5566 044140 011002 MOV (R0),R2 ;READ PAR TO R2
5567 044142 001401 BEQ 2\$;BRANCH IF PAR IS 0
5568 044144 104020 ERROR 20 ;PAR NOT ZERO
5569 044146 062700 000002 2\$: ADD #2,R0 ;POINT TO NEXT REGISTER
5570 044152 077107 SOB R1,1\$;BRANCH BACK TO 1\$ 15 TIMES
5571 :
5572 : NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5573 : REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
5574 : IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
5575 :
5576 044154 012737 044166 001112 MOV #3\$,SLPERR ;SET LOOP ON ERROR POINTER TO 3\$
5577 044162 012700 172240 MOV #SIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IS R0
5578 044166 012705 172240 3\$: MOV #SIPAR0,R5 ;LOAD STARTING ADDRESS INTO R5
5579 044172 004737 031330 JSR PC,CLRREG ;CLEAR 16 REGISTERS POINTED TO BY R5
5580 044176 012701 172276 MOV #SDPAR7,R1 ;PUT SDPAR7 ADDRESS IN R1
5581 044202 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
5582 044204 012710 177777 MOV #-1,(R0) ;LOAD REGISTER UNDER TEST
5583 044210 005037 001172 4\$: CLR \$TMP0 ;FLAG TO INDICATE THERE WAS A MATCH
5584 044214 011102 MOV (R1),R2 ;READ ALL REGISTERS
5585 044216 001406 BEQ 6\$;BRANCH IF REGISTER IS 0
5586 044220 020001 CMP R0,R1 ;IS ADDRESS OF NON-ZERO REGISTER SAME
5587 : ;AS THAT OF REGISTER UNDER TEST
5588 044222 001402 BEQ 5\$;BRANCH IF ADDRESSES MATCH
5589 044224 004737 031524 JSR PC,DUALADR ;LOG AND REPORT ERRORS
5590 044230 005237 001172 5\$: INC \$TMP0 ;SET FLAG WHEN ADDRESSES MATCH
5591 044234 162701 000002 6\$: SUB #2,R1 ;POINT TO NEXT REGISTER
5592 044240 022701 172240 CMP #SIPAR0,R1 ;SEE IF ALL REGISTERS HAVE BEEN READ
5593 044244 101761 BLOS 4\$;BRANCH IF MORE TO READ
5594 044246 062700 000002 ADD #2,R0 ;NOW LOAD NEXT REGISTER
5595 044252 022700 172276 CMP #SDPAR7,R0 ;SEE IF THERE ARE MORE REGISTERS TO TEST

5596	044256	103343			BHIS	3\$:BRANCH IF MORE REGISTERS TO TEST
5597	044260	012737	044104	001112	MOV	#20\$, \$LPERR	:SET LOOP ON ERROR POINTER TO START OF TEST
5598	044266	005737	001302		TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
5599	044272	001404			BEQ	TST25	:BRANCH TO NEXT TEST IF NO ERRORS
5600	044274	013737	001302	001204	MOV	ERRCNT, \$TMP5	:SAVE NUMBER OF ERRORS FOR PAR OUT
5601	044302	104021			ERROR	21	:SUMMARY OF DUAL ADDRESSING ERRORS

5602
5603
5604 :*****
5605 :*TEST 25 DUAL ADDRESS USER PAGE ADDRESS REGISTERS, ON LOADING
5606 :
5607 : THIS TEST FIRST CLEARS ALL THE USER PAGE ADDRESS
5608 : AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH
5609 : I-SPACE ADDRESS REGISTER ZERO, ONE REGISTER AT A TIME IS
5610 : LOADED WITH A NEGATIVE ONE. ALL USER ADDRESS REGISTERS
5611 : ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
5612 : INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
5613 : THE END OF THIS TEST.

5614 : ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
5615 : 'DUALADR'.
5616 :*****

5617 044304
5618 044304 000004
5619 044306 012737 044514 001316
5620 :
5621 044314 004737 031346 001112 20\$: JSR PC,CLEANUP ;SAVE STARTING ADDRESS OF NEXT
5622 044320 012737 044346 001112 :MOV #1\$, \$LPERR ;TEST FOR ESCAPE ON PARITY ERRORS
5623 044326 012705 177640 :MOV #UIPAR0,R5 ;INITIALIZE ERROR LOCATIONS
5624 044332 004737 031330 :JSR PC,CLRREG ;SET LOOP ON ERROR POINTER TO 1\$
5625 044336 012700 177640 :MOV #UIPAR0,R0 ;PUT ADDRESS OF FIRST PAR IN R5
5626 044342 012701 000020 :MOV #20,R1 ;CLEAR 16 REGISTERS POINTED TO BY R5
5627 044346 000240 1\$: NOP ;PUT ADDRESS OF FIRST PAR IN R0
5628 044350 011002 :MOV (R0),R2 ;BRANCH COUNT IS 16 DECIMAL
5629 044352 001401 :BEQ 2\$;THIS IS A SYNC POINT FOR SCOPING
5630 044354 104020 :ERROR 20 ;READ PAR TO R2
5631 044356 062700 000002 2\$: ADD #2,R0 ;BRANCH IF PAR IS 0
5632 044362 077107 :SOB R1,1\$;PAR NOT ZERO
5633 : ;POINT TO NEXT REGISTER
5634 : ;BRANCH BACK TO 1\$ 15 TIMES

5635 :
5636 : NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5637 : REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
5638 : IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)

5638	044364	012737	044376	001112	MOV	#3\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 3\$
5639	044372	012700	177640		MOV	#UIPAR0,R0	:PUT ADDRESS OF FIRST PAR IN R0
5640	044376	012705	177640		3\$: MOV	#UIPAR0,R5	:LOAD STARTING ADDRESS INTO R5
5641	044402	004737	031330		JSR	PC,CLRREG	:CLEAR 16 REGISTERS POINTED TO BY R5
5642	044406	012701	177676		MOV	#UDPAR7,R1	:PUT UDPAR7 ADDRESS IN R1
5643	044412	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
5644	044414	012710	177777		MOV	#-1,(R0)	:LOAD REGISTER UNDER TEST
5645	044420	005037	001172		4\$: CLR	\$TMP0	:FLAG TO INDICATE THERE WAS A MATCH
5646	044424	011102			MOV	(R1),R2	:READ ALL REGISTERS
5647	044426	001406			BEQ	6\$:BRANCH IF REGISTER IS 0
5648	044430	020001			CMP	R0,R1	:IS ADDRESS OF NON-ZERO REGISTER SAME
5649							:AS THAT OF REGISTER UNDER TEST
5650	044432	001402			BEQ	5\$:BRANCH IF ADDRESSES MATCH
5651	044434	004737	031524		SR	PC,DUALADR	:LOG AND REPORT ERRORS

5652	044440	005237	001172	5\$:	INC	\$TMP0			;SET FLAG WHEN ADDRESSES MATCH
5653	044444	162701	000002	6\$:	SUB	#2,R1			;POINT TO NEXT REGISTER
5654	044450	022701	177640		CMP	#UIPAR0,R1			;SEE IF ALL REGISTERS HAVE BEEN READ
5655	044454	101761			BLOS	4\$;BRANCH IF MORE TO READ
5656	044456	062700	000002		ADD	#2,R0			;NOW LOAD NEXT REGISTER
5657	044462	022700	177676		CMP	#UDPAR7,R0			;SEE IF THERE ARE MORE REGISTERS TO TEST
5658	044466	103343			BHIS	3\$;BRANCH IF MORE REGISTERS TO TEST
5659	044470	012737	044314	001112	MOV	#20\$,\$LPERR			;SET LOOP ON ERROR POINTER TO START OF TEST
5660	044476	005737	001302		TST	ERRCNT			;SEE IF THERE WERE ANY ERRORS
5661	044502	001404			BEQ	TST26			;BRANCH TO NEXT TEST IF NO ERRORS
5662	044504	013737	001302	001204	MOV	ERRCNT,\$TMP5			;SAVE NUMBER OF ERRORS FOR PAR OUT
5663	044512	104021			ERROR	21			;SUMMARY OF DUAL ADDRESSING ERRORS

*TEST 26 DUAL ADDRESS KERNEL PAGE DESCRIPTOR REGISTERS, ON LOADING

THIS TEST FIRST CLEARS ALL THE KERNEL PAGE DESCRIPTOR AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH !-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS LOADED WITH A NEGATIVE ONE. ALL KERNEL DESCRIPTOR REGISTERS ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO. INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT THE END OF THIS TEST.

ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE 'DUALADR'.

TST26:

5679	044514				SCOPE				
5680	044514	000004			MOV	#TST27,NXTTST			;SAVE STARTING ADDRESS OF NEXT
5681	044516	012737	044724	001316					;TEST FOR ESCAPE ON PARITY ERRORS
5682									;INITIALIZE ERROR LOCATIONS
5683	044524	004737	031346		20\$:	JSR	PC,CLEANUP		;SET LOOP ON ERROR POINTER TO 1\$
5684	044530	012737	044556	001112	MOV	#1\$,\$LPERR			;PUT ADDRESS OF FIRST PDR IN R5
5685	044536	012705	172300		MOV	#KIPDR0,R5			;CLEAR 16 REGISTERS POINTED TO BY R5
5686	044542	004737	031330		JSR	PC,CLRREG			;PUT ADDRESS OF FIRST PDR IN R0
5687	044546	012700	172300		MOV	#KIPDR0,R0			;BRANCH COUNT IS 16 DECIMAL
5688	044552	012701	000020		MOV	#20,R1			;THIS IS A SYNC POINT FOR SCOPING
5689	044556	000240		1\$:	NOP				;READ PDR TO R2
5690	044560	011002			MOV	(R0),R2			;BRANCH IF PDR IS 0
5691	044562	001401			BEQ	2\$;PDR NOT ZERO
5692	044564	104020			ERROR	20			;POINT TO NEXT REGISTER
5693	044566	062700	000002	2\$:	ADD	#2,R0			;BRANCH BACK TO 1\$ 15 TIMES
5694	044572	077107			SOB	R1,1\$			
5695									
5696									
5697									
5698									
5699									
5700	044574	012737	044606	001112					
5701	044602	012700	172300		MOV	#3\$,\$LPERR			;SET LOOP ON ERROR POINTER TO 3\$
5702	044606	012705	172300		MOV	#KIPDR0,R0			;PUT ADDRESS OF FIRST PDR IS R0
5703	044612	004737	031330	3\$:	MOV	#KIPDR0,R5			;LOAD STARTING ADDRESS INTO R5
5704	044616	012701	172336		JSR	PC,CLRREG			;CLEAR 16 REGISTERS POINTED TO BY R5
5705	044622	000240			MOV	#KDPDR7,R1			;PUT KDPDR7 ADDRESS IN R1
5706	044624	012710	177777		NOP				;THIS IS A SYNC POINT FOR SCOPING
5707	044630	005037	001172	4\$:	MOV	#-1,(R0)			;LOAD REGISTER UNDER TEST
					CLR	\$TMP0			;FLAG TO INDICATE THERE WAS A MATCH

5708	044634	011102			MOV	(R1),R2		:READ ALL REGISTERS
5709	044636	001406			BEQ	6\$:BRANCH IF REGISTER IS 0
5710	044640	020001			CMP	RO,R1		:IS ADDRESS OF NON-ZERO REGISTER SAME
5711								:AS THAT OF REGISTER UNDER TEST
5712	044642	001402			BEQ	5\$:BRANCH IF ADDRESSES MATCH
5713	044644	004737	031524		JSR	PC,DUALADR		:LOG AND REPORT ERRORS
5714	044650	005237	001172	5\$:	INC	\$TMP0		:SET FLAG WHEN ADDRESSES MATCH
5715	044654	162701	000002	6\$:	SUB	#2,R1		:POINT TO NEXT REGISTER
5716	044660	022701	172300		CMP	#KIPDR0,R1		:SEE IF ALL REGISTERS HAVE BEEN READ
5717	044664	101761			BLOS	4\$:BRANCH IF MORE TO READ
5718	044666	062700	000002		ADD	#2,RO		:NOW LOAD NEXT REGISTER
5719	044672	022700	172336		CMP	#KDPDR7,RO		:SEE IF THERE ARE MORE REGISTERS TO TEST
5720	044676	103343			BHIS	3\$:BRANCH IF MORE REGISTERS TO TEST
5721	044700	012737	044524	001112	MOV	#20\$,\$LPERR		:SET LOOP ON ERROR POINTER TO START OF TEST
5722	044706	005737	001302		TST	ERRCNT		:SEE IF THERE WERE ANY ERRORS
5723	044712	001404			BEQ	TST27		:BRANCH TO NEXT TEST IF NO ERRORS
5724	044714	013737	001302	001204	MOV	ERRCNT,\$TMP5		:SAVE NUMBER OF ERRORS FOR PDR OUT
5725	044722	104021			ERROR	21		:SUMMARY OF DUAL ADDRESSING ERRORS

:TEST 27 DUAL ADDRESS SUPERVISOR PAGE DESCRIPTOR REGISTERS, ON LOADING

: THIS TEST FIRST CLEARS ALL THE SUPERVISOR PAGE DESCRIPTOR
: AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH
: I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
: LOADED WITH A NEGATIVE ONE. ALL SUPERVISOR DESCRIPTOR REGISTERS
: ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
: INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
: THE END OF THIS TEST.

: ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
: 'DUALADR'.
:*****

TST27:

5741	044724				SCOPE			
5742	044724	000004			MOV	#TST30,NXTTST		:SAVE STARTING ADDRESS OF NEXT
5743	044726	012737	045134	001316				:TEST FOR ESCAPE ON PARITY ERRORS
5744								:INITIALIZE ERROR LOCATIONS
5745	044734	004737	031346		20\$:	JSR	PC,CLEANUP	:SET LOOP ON ERROR POINTER TO 1\$
5746	044740	012737	044766	001112	MOV	#1\$,\$LPERR		:PUT ADDRESS OF FIRST PDR IN R5
5747	044746	012705	172200		MOV	#SIPDR0,R5		:CLEAR 16 REGISTERS POINTED TO BY R5
5748	044752	004737	031330		JSR	PC,CLRREG		:PUT ADDRESS OF FIRST PDR IN RO
5749	044756	012700	172200		MOV	#SIPDR0,RO		:BRANCH COUNT IS 16 DECIMAL
5750	044762	012701	000020		MOV	#20,R1		:THIS IS A SYNC POINT FOR SCOPING
5751	044766	000240			1\$:	NOP		:READ PDR TO R2
5752	044770	011002			MOV	(RO),R2		:BRANCH IF PDR IS 0
5753	044772	001401			BEQ	2\$:PDR NOT ZERO
5754	044774	104020			ERROR	20		:POINT TO NEXT REGISTER
5755	044776	062700	000002	2\$:	ADD	#2,RO		:BRANCH BACK TO 1\$ 15 TIMES
5756	045002	077107			SOB	R1,1\$		

: NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
: REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
: IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)

5762	045004	012737	045016	001112	MOV	#3\$,\$LPERR		:SET LOOP ON ERROR POINTER TO 3\$
5763	045012	012700	172200		MOV	#SIPDR0,RO		:PUT ADDRESS OF FIRST PDR IS RO

5764	045016	012705	172200	3\$:	MOV	#SIPDR0,R5	:LOAD STARTING ADDRESS INTO R5
5765	045022	004737	031330		JSR	PC,CLRREG	:CLEAR 16 REGISTERS POINTED TO BY R5
5766	045026	012701	172236		MOV	#SDPDR7,R1	:PUT SDPDR7 ADDRESS IN R1
5767	045032	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
5768	045034	012710	177777		MOV	#-1,(R0)	:LOAD REGISTER UNDER TEST
5769	045040	005037	001172	4\$:	CLR	\$TMP0	:FLAG TO INDICATE THERE WAS A MATCH
5770	045044	011102			MOV	(R1),R2	:READ ALL REGISTERS
5771	045046	001406			BEQ	6\$:BRANCH IF REGISTER IS 0
5772	045050	020001			CMP	R0,R1	:IS ADDRESS OF NON-ZERO REGISTER SAME
5773							:AS THAT OF REGISTER UNDER TEST
5774	045052	001402			BEQ	5\$:BRANCH IF ADDRESSES MATCH
5775	045054	004737	031524		JSR	PC,DUALADR	:LOG AND REPORT ERRORS
5776	045060	005237	001172	5\$:	INC	\$TMP0	:SET FLAG WHEN ADDRESSES MATCH
5777	045064	162701	000002	6\$:	SUB	#2,R1	:POINT TO NEXT REGISTER
5778	045070	022701	172200		CMP	#SIPDR0,R1	:SEE IF ALL REGISTERS HAVE BEEN READ
5779	045074	101761			BLOS	4\$:BRANCH IF MORE TO READ
5780	045076	062700	000002		ADD	#2,R0	:NOW LOAD NEXT REGISTER
5781	045102	022700	172236		CMP	#SDPDR7,R0	:SEE IF THERE ARE MORE REGISTERS TO TEST
5782	045106	103343			BHIS	3\$:BRANCH IF MORE REGISTERS TO TEST
5783	045110	012737	044734	001112	MOV	#20\$,\$LPERR	:SET LOOP ON ERROR POINTER TO START OF TEST
5784	045116	005737	001302		TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
5785	045122	001404			BEQ	TST30	:BRANCH TO NEXT TEST IF NO ERRORS
5786	045124	013737	001302	001204	MOV	ERRCNT,\$TMP5	:SAVE NUMBER OF ERRORS FOR PDR OUT
5787	045132	104021			ERROR	21	:SUMMARY OF DUAL ADDRESSING ERRORS
5788							
5789							

: *TEST 30 DUAL ADDRESS USER PAGE DESCRIPTOR REGISTERS, ON LOADING

: THIS TEST FIRST CLEARS ALL THE USER PAGE DESCRIPTOR
: AND CHECKS TO SEE THAT THEY EACH HOLD ZERO. THEN, STARTING WITH
: I-SPACE DESCRIPTOR REGISTER ZERO, ONE REGISTER AT A TIME IS
: LOADED WITH A NEGATIVE ONE. ALL USER DESCRIPTOR REGISTERS
: ARE NOW READ TO SEE THAT ONLY THE ONE UNDER TEST IS NON-ZERO.
: INDIVIDUAL ERRORS ARE REPORTED AND A SUMMARY OF THEM IS GIVEN AT
: THE END OF THIS TEST.

: ALL ERRORS IN THIS TEST ARE REPORTED AND ANALYZED BY SUBROUTINE
: 'DUALADR'.
: *****

TST30:

5803	045134				SCOPE		
5804	045134	000004			MOV	#TST31,NXTTST	:SAVE STARTING ADDRESS OF NEXT
5805	045136	012737	045344	001316			:TEST FOR ESCAPE ON PARITY ERRORS
5806							:INITIALIZE ERROR LOCATIONS
5807	045144	004737	031346		20\$:	JSR	PC,CLEANUP
5808	045150	012737	045176	001112	MOV	#1\$,\$LPERR	:SET LOOP ON ERROR POINTER TO 1\$
5809	045156	012705	177600		MOV	#UIPDR0,R5	:PUT ADDRESS OF FIRST PDR IN R5
5810	045162	004737	031330		JSR	PC,CLRREG	:CLEAR 16 REGISTERS POINTED TO BY R5
5811	045166	012700	177600		MOV	#UIPDR0,R0	:PUT ADDRESS OF FIRST PDR IN R0
5812	045172	012701	000020		MOV	#20,R1	:BRANCH COUNT IS 16 DECIMAL
5813	045176	000240		1\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
5814	045200	011002			MOV	(R0),R2	:READ PDR TO R2
5815	045202	001401			BEQ	2\$:BRANCH IF PDR IS 0
5816	045204	104020			ERROR	20	:PDR NOT ZERO
5817	045206	062700	000002	2\$:	ADD	#2,R0	:POINT TO NEXT REGISTER
5818	045212	077107			SOB	R1,1\$:BRANCH BACK TO 1\$ 15 TIMES
5819							


```
5820 :: NOW START DUAL ADDRESSING TEST BY LOADING -1 INTO ONE
5821 :: REGISTER AND READING THE REST TO SEE ONLY THAT REGISTER
5822 :: IS NON-ZERO. (DROPPED BITS WILL BE FOUND IN NEXT TEST.)
5823 ::
5824 045214 012737 045226 001112      MOV    #3$, $LPERR      ;SET LOOP ON ERROR POINTER TO 3$
5825 045222 012700 177600              MOV    #UIPDR0,R0      ;PUT ADDRESS OF FIRST PDR IS R0
5826 045226 012705 177600      3$:  MOV    #UIPDR0,R5      ;LOAD STARTING ADDRESS INTO R5
5827 045232 004737 031330      JSR    PC,CLRREG       ;CLEAR 16 REGISTERS POINTED TO BY R5
5828 045236 012701 177636      MOV    #UDPDR7,R1     ;PUT UDPDR7 ADDRESS IN R1
5829 045242 000240              NOP                    ;THIS IS A SYNC POINT FOR SCOPING
5830 045244 012710 177777      MOV    #-1,(R0)       ;LOAD REGISTER UNDER TEST
5831 045250 005037 001172      4$:  CLR    $TMP0         ;FLAG TO INDICATE THERE WAS A MATCH
5832 045254 011102              MOV    (R1),R2        ;READ ALL REGISTERS
5833 045256 001406              BEQ    6$             ;BRANCH IF REGISTER IS 0
5834 045260 020001              CMP    R0,R1          ;IS ADDRESS OF NON-ZERO REGISTER SAME
5835 ::                               ;AS THAT OF REGISTER UNDER TEST
5836 045262 001402              BEQ    5$             ;BRANCH IF ADDRESSES MATCH
5837 045264 004737 031524      JSR    PC,DUALADR     ;LOG AND REPORT ERRORS
5838 045270 005237 001172      5$:  INC    $TMP0         ;SET FLAG WHEN ADDRESSES MATCH
5839 045274 162701 000002      6$:  SUB    #2,R1         ;POINT TO NEXT REGISTER
5840 045300 022701 177600      CMP    #UIPDR0,R1     ;SEE IF ALL REGISTERS HAVE BEEN READ
5841 045304 101761              BLOS   4$             ;BRANCH IF MORE TO READ
5842 045306 062700 000002      ADD    #2,R0          ;NOW LOAD NEXT REGISTER
5843 045312 022700 177636      CMP    #UDPDR7,R0     ;SEE IF THERE ARE MORE REGISTERS TO TEST
5844 045316 103343              BHIS   3$             ;BRANCH IF MORE REGISTERS TO TEST
5845 045320 012737 045144 001112      MOV    #20$, $LPERR   ;SET LOOP ON ERROR POINTER TO START OF TEST
5846 045326 005737 001302      TST    ERRCNT         ;SEE IF THERE WERE ANY ERRORS
5847 045332 001404              BEQ    TST31         ;BRANCH TO NEXT TEST IF NO ERRORS
5848 045334 013737 001302 001204      MOV    ERRCNT,$TMP5   ;SAVE NUMBER OF ERRORS FOR PDR OUT
5849 045342 104021              ERROR  21            ;SUMMARY OF DUAL ADDRESSING ERRORS
5850 ::
```

```
5851 .SBTTL      TEST FOR BAD READ/WRITE BITS IN P.A.R.'S & P.D.R.'S
5852 :*          THESE NEXT SIX (6) TESTS CHECK FOR BAD BITS IN THE MEMORY CHIPS
5853 :*          THAT MAKE UP THE PAR'S AND PDR'S.  THE REGISTERS ARE LOADED WITH
5854 :*          ZERO AND MODIFIED BY '401' UNTIL 177777 IS REACHED IN EACH ONE
5855 :*          (THE NUMBER 401 WAS CHOSEN FOR FASTER RUN-TIME).  THE BITS THAT
5856 :*          ARE NOT IMPLEMENTED OR THAT ARE NOT READ/WRITE ARE MASKED OUT
5857 :*          OF THE DATA COMPARE.  A LOG OF MULTIPLE ERRORS IS KEPT FOR EACH
5858 :*          GROUP AND IT IS REPORTED AT THE CONCLUSION OF EACH TEST.
5859 :*
5860 ::
```

```
5861 *****
5862 :*          TEST 31      COUNT PATTERN IN KERNEL PAGE ADDRESS REGISTERS
5863 :*
5864 :*          THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE ADDRESS
5865 :*          REGISTERS.  SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
5866 :*          TO BE MASKED OUT OF THE COMPARE.  IF THE COUNT PATTERN DOES
5867 :*          NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
5868 :*          PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
5869 :*          SUMMARY OF THE ERRORS IS GIVEN.
5870 ::
5871 :*          ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
5872 :*          SUBROUTINE 'PARCOUNT'.
5873 ::
```

```
5874 045344
5875 045344 000004      TST31:  SCOPE
```

```
5876 045346 012737 045474 001316      MOV      #TST32,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
5877                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5878 045354 012737 000012 001206      MOV      #12,$TIMES        ;DO 12 ITERATIONS
5879 045362 004737 031346 001112 20$: JSR      PC,CLEANUP        ;INITIALIZE ERROR LOCATIONS
5880 045366 012737 045402 001112      MOV      #2$,$LPERR       ;SET LOOP ON ERROR POINTER TO 1$
5881 045374 005001                                     CLR      R1                ;CLEAR REGISTER TO HOLD COUNT PATTERN
5882 045376 012700 172340 1$: MOV      #KIPAR0,R0       ;PUT ADDRESS OF FIRST REGISTER IS R0
5883 045402 000240 2$: NOP                                     ;THIS IS A SYNC POINT FOR SCOPING
5884 045404 010110      MOV      R1,(R0)          ;LOAD COUNT INTO REGISTER
5885 045406 011002      MOV      (R0),R2         ;READ REGISTER BACK TO R2
5886 045410 010104      MOV      R1,R4          ;PUT PATTERN IS R4
5887 045412 020402      CMP      R4,R2          ;SEE IF DATA MATCHES PATTERN
5888 045414 001402      BEQ     3$              ;BRANCH IF DATA IS GOOD
5889 045416 004737 031600      JSR      PC,PARCOUNT     ;LOG AND REPORT COUNT ERROR
5890 045422 062700 000002 3$: ADD      #2,R0            ;POINT TO NEXT REGISTER
5891 045426 022700 172376      CMP      #KDPAR7,R0       ;SEE IF YOU PASSED THE KDPAR7 PAR
5892 045432 103363      BHIS    2$              ;BRANCH IF MORE PAR'S TO TEST
5893 045434 022701 177777      CMP      #177777,R1       ;SEE IF COUNT HAS REACHED 177777
5894 045440 001403      BEQ     4$              ;BRANCH IF COUNT IS 177777
5895 045442 062701 000401      ADD      #401,R1          ;INCREASE COUNT PATTERN
5896 045446 000753      BR      1$              ;BRANCH TO CONTINUE TEST
5897 045450 012737 045362 001112 4$: MOV      #20$,$LPERR      ;SET LOOP POINTER TO START OF TEST
5898 045456 005737 001302      TST     ERRCNT           ;SEE IF THERE WERE ANY ERRORS
5899 045462 001404      BEQ     TST32           ;BRANCH TO NEXT TEST IF NO ERRORS
5900 045464 013737 001302 001204      MOV      ERRCNT,$TMP5     ;SAVE NUMBER OF ERRORS FOR TYPEOUT
5901 045472 104022      ERROR    22              ;SUMMARY OF COUNT PATTERN FAILURES
```

```
5902
5903 :*****
5904 :*TEST 32      COUNT PATTERN IN SUPERVISOR PAGE ADDRESS REGISTERS
5905 :
5906 :   THIS TEST RUNS A COUNT PATTERN THRU THE SUPERVISOR PAGE ADDRESS
5907 :   REGISTERS.  SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED
5908 :   TO BE MASKED OUT OF THE COMPARE.  IF THE COUNT PATTERN DOES
5909 :   NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
5910 :   PATTERN, AND BAD DATA ARE REPORTED.  AT THE END OF THE TEST A
5911 :   SUMMARY OF THE ERRORS IS GIVEN.
5912 :
5913 :   ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY
5914 :   SUBROUTINE 'PARCOUNT'.
```

```
5915 :*****
5916 :TST32:
5917 045474
5918 045474 000004      SCOPE
5919 045476 012737 045624 001316      MOV      #TST33,NXTTST    ;SAVE STARTING ADDRESS OF NEXT
5920                                     ;TEST FOR ESCAPE ON PARITY ERRORS
5921 045504 012737 000012 001206      MOV      #12,$TIMES        ;DO 12 ITERATIONS
5922 045512 004737 031346 001112 20$: JSR      PC,CLEANUP        ;INITIALIZE ERROR LOCATIONS
5923 045516 012737 045532 001112      MOV      #2$,$LPERR       ;SET LOOP ON ERROR POINTER TO 1$
5924 045524 005001                                     CLR      R1                ;CLEAR REGISTER TO HOLD COUNT PATTERN
5925 045526 012700 172240 1$: MOV      #SIPAR0,R0       ;PUT ADDRESS OF FIRST REGISTER IS R0
5926 045532 000240 2$: NOP                                     ;THIS IS A SYNC POINT FOR SCOPING
5927 045534 010110      MOV      R1,(R0)          ;LOAD COUNT INTO REGISTER
5928 045536 011002      MOV      (R0),R2         ;READ REGISTER BACK TO R2
5929 045540 010104      MOV      R1,R4          ;PUT PATTERN IS R4
5930 045542 020402      CMP      R4,R2          ;SEE IF DATA MATCHES PATTERN
5931 045544 001402      BEQ     3$              ;BRANCH IF DATA IS GOOD
```

5932	045546	004737	031600			JSR	PC,PARCOUNT	:LOG AND REPORT COUNT ERROR
5933	045552	062700	000002	3\$:		ADD	#2,R0	:POINT TO NEXT REGISTER
5934	045556	022700	172276			CMP	#SDPAR7,R0	:SEE IF YOU PASSED THE SDPAR7 PAR
5935	045562	103363				BHIS	2\$:BRANCH IF MORE PAR''S TO TEST
5936	045564	022701	177777			CMP	#177777,R1	:SEE IF COUNT HAS REACHED 177777
5937	045570	001403				BEQ	4\$:BRANCH IF COUNT IS 177777
5938	045572	062701	000401			ADD	#401,R1	:INCREASE COUNT PATTERN
5939	045576	000753				BR	1\$:BRANCH TO CONTINUE TEST
5940	045600	012737	045512	001112	4\$:	MOV	#20\$,\$LPERR	:SET LOOP POINTER TO START OF TEST
5941	045606	005737	001302			TST	ERRCNT	:SEE IF THERE WERE ANY ERRORS
5942	045612	001404				BEQ	TST33	:BRANCH TO NEXT TEST IF NO ERRORS
5943	045614	013737	001302	001204		MOV	ERRCNT,\$TMP5	:SAVE NUMBER OF ERRORS FOR TYPEOUT
5944	045622	104022				ERROR	22	:SUMMARY OF COUNT PATTERN FAILURES

5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959

```
*****  
:TEST 33          COUNT PATTERN IN USER PAGE ADDRESS REGISTERS  
: :  
: THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE ADDRESS  
: REGISTERS. SINCE ALL THE BITS ARE IMPLEMENTED, NONE NEED  
: TO BE MASKED OUT OF THE COMPARE. IF THE COUNT PATTERN DOES  
: NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA  
: PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A  
: SUMMARY OF THE ERRORS IS GIVEN.  
: :  
: ALL ERRORS FOUND BY THIS TEST ARE REPORTED AND ANALYZED BY  
: SUBROUTINE 'PARCOUNT'.  
: :  
:*****
```

5960 045624
5961 045624 000004
5962 045626 012737 045754 001316
5963
5964 045634 012737 000012 001206
5965 045642 004737 031346
5966 045646 012737 045662 001112
5967 045654 005001
5968 045656 012700 177640
5969 045662 000240
5970 045664 010110
5971 045666 011002
5972 045670 010104
5973 045672 020402
5974 045674 001402
5975 045676 004737 031600
5976 045702 062700 000002
5977 045706 022700 177676
5978 045712 103363
5979 045714 022701 177777
5980 045720 001403
5981 045722 062701 000401
5982 045726 000753
5983 045730 012737 045642 001112
5984 045736 005737 001302
5985 045742 001404
5986 045744 013737 001302 001204
5987 045752 104022

```
TST33:  
SCOPE  
MOV #TST34,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
;TEST FOR ESCAPE ON PARITY ERRORS  
MOV #12,$TIMES ;DO 12 ITERATIONS  
JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS  
MOV #2$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$  
CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN  
MOV #UIPAR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0  
NOP ;THIS IS A SYNC POINT FOR SCOPING  
MOV R1,(R0) ;LOAD COUNT INTO REGISTER  
MOV (R0),R2 ;READ REGISTER BACK TO R2  
MOV R1,R4 ;PUT PATTERN IS R4  
CMP R4,R2 ;SEE IF DATA MATCHES PATTERN  
BEQ 3$ ;BRANCH IF DATA IS GOOD  
JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR  
ADD #2,R0 ;POINT TO NEXT REGISTER  
CMP #UDPAR7,R0 ;SEE IF YOU PASSED THE UDPAR7 PAR  
BHIS 2$ ;BRANCH IF MORE PAR''S TO TEST  
CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777  
BEQ 4$ ;BRANCH IF COUNT IS 177777  
ADD #401,R1 ;INCREASE COUNT PATTERN  
BR 1$ ;BRANCH TO CONTINUE TEST  
MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST  
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS  
BEQ TST34 ;BRANCH TO NEXT TEST IF NO ERRORS  
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT  
ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES
```

5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000
6001
6002 045754
6003 045754 000004
6004 045756 012737 046132 001316
6005
6006 045764 012737 000012 001206
6007 045772 004737 031346 20\$:
6008 045776 012737 046012 001112
6009 046004 005001
6010 046006 012700 172300 1\$:
6011 046012 000240 2\$:
6012 046014 010110
6013 046016 011002
6014 046020 010104
6015 046022 105737 001362
6016 046026 001003
6017 046030 005737 001360
6018 046034 001403
6019 046036 012737 000360 046046 6\$:
6020 046044 042704 100360 5\$:
6021 046050 020402
6022 046052 001402
6023 046054 004737 031600
6024 046060 062700 000002 3\$:
6025 046064 022700 172336
6026 046070 103350
6027 046072 022701 177777
6028 046076 001403
6029 046100 062701 000401
6030 046104 000740
6031 046106 012737 045772 001112 4\$:
6032 046114 005737 001302
6033 046120 001404
6034 046122 013737 001302 001204
6035 046130 104022
6036

*TEST 34 COUNT PATTERN IN KERNEL PAGE DESCRIPTOR REGISTERS
*
* THIS TEST RUNS A COUNT PATTERN THRU THE KERNEL PAGE DESCRIPTOR
* REGISTERS. SINCE BITS <15> AND <05:04> ARE NOT IMPLEMENTED
* AND BITS <07:06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED
* OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT TO THE
* DESCRIPTOR REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN
* DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
* PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
* SUMMARY OF THE ERRORS IS GIVEN.

TST34:
SCOPE
MOV #TST35,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
MOV #12,\$TIMES ;DO 12 ITERATIONS
JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #2,\$\$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
MOV #KIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV R1,(R0) ;LOAD COUNT INTO REGISTER
MOV (R0),R2 ;READ REGISTER BACK TO R2
MOV R1,R4 ;PUT PATTERN IS R4
TSTB KB11CM ;IS THIS A MODIFIED CPU?
BNE 6\$;YES
TST KB11E ;IS IT A KB11E OR KB11EM?
BEQ 5\$;BR IF NOT
MOV #360,5\$+2 ;DIDDLE 100360
BIC #100360,R4 ;CLEAR BITS NOT FOUND IN REGISTER.
CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
BEQ 3\$;BRANCH IF DATA IS GOOD
JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #KDPDR7,R0 ;SEE IF YOU PASSED THE KDPDR7 PDR
BHS 2\$;BRANCH IF MORE PDR'S TO TEST
CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
BEQ 4\$;BRANCH IF COUNT IS 177777
ADD #401,R1 ;INCREASE COUNT PATTERN
BR 1\$;BRANCH TO CONTINUE TEST
MOV #20,\$\$LPERR ;SET LOOP POINTER TO START OF TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST35 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,\$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES

6037
6038
6039
6040
6041
6042
6043
6044
6045
6046
6047
6048
6049
6050
6051
6052
6053
6054
6055
6056
6057
6058
6059
6060
6061
6062
6063
6064
6065
6066
6067
6068
6069
6070
6071
6072
6073
6074
6075
6076
6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6090
6091
6092

*TEST 35 COUNT PATTERN IN SUPERVISOR PAGE DESCRIPTOR REGISTERS

THIS TEST RUNS A COUNT PATTERN THRU THE SUPERVISOR PAGE DESCRIPTOR REGISTERS. SINCE BITS <15> AND <05:04> ARE NOT IMPLEMENTED AND BITS <07:06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT TO THE DESCRIPTOR REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A SUMMARY OF THE ERRORS IS GIVEN.

TST35:

```
SCOPE
MOV #TST36,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;DO 12 ITERATIONS
MOV #12,$TIMES
JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #2,$SLPERR ;SET LOOP ON ERROR POINTER TO 1$
CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
MOV #SIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0
2$: NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV R1,(R0) ;LOAD COUNT INTO REGISTER
MOV (R0),R2 ;READ REGISTER BACK TO R2
MOV R1,R4 ;PUT PATTERN IS R4
TSTB KB11CM ;IS THIS A MODIFIED CPU?
BNE 6$ ;YES
TST KB11E ;IS IT A KB11E OR KB11EM?
BEQ 5$ ;BR IF NOT
MOV #360,5$+2 ;DIDDLE 100360
BIC #100360,R4 ;CLEAR BITS NOT FOUND IN REGISTER.
CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
BEQ 3$ ;BRANCH IF DATA IS GOOD
JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #SDPDR7,R0 ;SEE IF YOU PASSED THE SDPDR7 PDR
BHS 2$ ;BRANCH IF MORE PDR'S TO TEST
CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
BEQ 4$ ;BRANCH IF COUNT IS 177777
ADD #401,R1 ;INCREASE COUNT PATTERN
BR 1$ ;BRANCH TO CONTINUE TEST
4$: MOV #20,$SLPERR ;SET LOOP POINTER TO START OF TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST36 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES
```

*TEST 36 COUNT PATTERN IN USER PAGE DESCRIPTOR REGISTERS

THIS TEST RUNS A COUNT PATTERN THRU THE USER PAGE DESCRIPTOR REGISTERS. SINCE BITS <15> AND <05:04> ARE NOT IMPLEMENTED AND BITS <07:06> CANNOT BE SET BY DIRECT LOAD, THEY ARE MASKED OUT OF THE DATA COMPARE. THESE BITS ARE STILL SENT TO THE

6093 :: DESCRIPTOR REGISTER TO FIND BAD ETCHES. IF THE COUNT PATTERN
6094 :: DOES NOT MATCH THE DATA RECEIVED, THE REGISTER ADDRESS, DATA
6095 :: PATTERN, AND BAD DATA ARE REPORTED. AT THE END OF THE TEST A
6096 :: SUMMARY OF THE ERRORS IS GIVEN.
6097 ::
6098 ::
6099 ::*****

```
TST36:
SCOPE
MOV #TST37,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;DO 12 ITERATIONS
MOV #12,$TIMES
JSR PC,CLEANUP ;INITIALIZE ERROR LOCATIONS
MOV #2,$$LPERR ;SET LOOP ON ERROR POINTER TO 1$
CLR R1 ;CLEAR REGISTER TO HOLD COUNT PATTERN
MOV #UIPDR0,R0 ;PUT ADDRESS OF FIRST REGISTER IS R0
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV R1,(R0) ;LOAD COUNT INTO REGISTER
MOV (R0),R2 ;READ REGISTER BACK TO R2
MOV R1,R4 ;PUT PATTERN IS R4
TSTB KB11CM ;IS THIS A MODIFIED CPU?
BNE 6$ ;YES
TST KB11E ;IS IT A KB11E OR KB11EM?
BEQ 5$ ;BR IF NOT
MOV #360,$$+2 ;DIDDLE 100360
BIC #100360,R4 ;CLEAR BITS NOT FOUND IN REGISTER.
CMP R4,R2 ;SEE IF DATA MATCHES PATTERN
BEQ 3$ ;BRANCH IF DATA IS GOOD
JSR PC,PARCOUNT ;LOG AND REPORT COUNT ERROR
ADD #2,R0 ;POINT TO NEXT REGISTER
CMP #UDPDR7,R0 ;SEE IF YOU PASSED THE UDPDR7 PDR
BHS 2$ ;BRANCH IF MORE PDR'S TO TEST
CMP #177777,R1 ;SEE IF COUNT HAS REACHED 177777
BEQ 4$ ;BRANCH IF COUNT IS 177777
ADD #401,R1 ;INCREASE COUNT PATTERN
BR 1$ ;BRANCH TO CONTINUE TEST
MOV #20,$$LPERR ;SET LOOP POINTER TO START OF TEST
TST ERRCNT ;SEE IF THERE WERE ANY ERRORS
BEQ TST37 ;BRANCH TO NEXT TEST IF NO ERRORS
MOV ERRCNT,$TMP5 ;SAVE NUMBER OF ERRORS FOR TYPEOUT
ERROR 22 ;SUMMARY OF COUNT PATTERN FAILURES
```

6133
6134 .SBTTL TEST FOR CORRECT BYTE ADDRESSING OF P.A.R.'S & P.D.R.'S
6135 :* THESE NEXT SIX (6) TESTS ARE USED TO TEST THE LOGIC THAT
6136 :* ALLOWS BYTE ADDRESSING OF THE PAR'S AND PDR'S. IN EACH
6137 :* CASE THE I-SPACE REGISTER 0 IS USED, SINCE IT IS REALLY
6138 :* THE WRITE PULSE TO THE REGISTER SET THAT IS BEING TESTED.
6139 :* IT IS ASSUMED THAT IF ONE REGISTER OF A GROUP FUNCTIONS
6140 :* PROPERLY THAT THEY ALL WILL, SINCE ALL REGISTERS HAVE BEEN
6141 :* BIT TESTED EARLIER.

6142 :*
6143 :*****
6144 :*TEST 37 BYTE ADDRESSING OF KERNEL PAGE ADDRESS REGISTERS
6145 :*
6146 :* THIS TEST CHECKS THE 'SAPA WRITE LOBYTE' AND 'SAPA WRITE HIBYTE'
6147 :* SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.
6148 :* BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT

6205	046676	112710	000124			MOV	#124,(R0)	;WRITE THE UPPER BYTE OF REGISTER
6206	046702	013701	172240			MOV	SIPAR0,R1	;READ REGISTER INTO R1
6207	046706	020102				CMP	R1,R2	;SEE IF REGISTER HOLDS CORRECT DATA
6208	046710	001401				BEQ	2\$;BRANCH TO EXIT IF DATA RIGHT
6209	046712	104023				ERROR	23	;DIDN'T LOAD CORRECT BYTE
6210	046714	012737	046614	001112	2\$:	MOV	#20\$,\$LPERR	;SET LOOP POINTER TO START OF TEST

6211
6212
6213
6214
6215
6216
6217
6218
6219
6220

```
*****  
*TEST 41      BYTE ADDRESSING OF USER PAGE ADDRESS REGISTERS  
*  
* THIS TEST CHECKS THE 'SAPC WRITE LOBYTE' AND 'SAPC WRITE HIBYTE'  
* SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.  
* BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT  
* GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.  
*****
```

6221
6222
6223
6224
6225
6226
6227
6228
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238
6239
6240
6241
6242
6243
6244
6245
6246
6247

```
TST41:  
SCOPE  
MOV #TST42,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
;TEST FOR ESCAPE ON PARITY ERRORS  
20$: MOV #11$,$LPERR ;SET LOOP ON ERROR POINTER TO 11$  
MOV #15,R2 ;PUT EXPECTED DATA IN R2  
MOV #UIPAR0,R0 ;PUT ADDRESS OF REGISTER IN R0  
11$: CLR UIPAR0 ;CLEAR THE REGISTER UNDER TEST  
NOP ;THIS IS A SYNC POINT FOR SCOPING  
MOV #172015,(R0) ;LOAD LOWER BYTE OF REGISTER  
MOV UIPAR0,R1 ;READ REGISTER INTO R1  
CMP R1,R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN  
BEQ 1$ ;BRANCH IF DATA MATCHES  
ERROR 23 ;DIDN'T LOAD CORRECT BYTE  
1$: MOV #12$,$LPERR ;SET LOOP ON ERROR POINTER TO 12$  
MOV #UIPAR0+1,R0 ;POINT TO UPPER BYTE OF REGISTER  
MOV #052015,R2 ;LOAD EXPECTED DATA IN R2  
12$: NOP ;THIS IS A SYNC POINT FOR SCOPING  
MOV #124,(R0) ;WRITE THE UPPER BYTE OF REGISTER  
MOV UIPAR0,R1 ;READ REGISTER INTO R1  
CMP R1,R2 ;SEE IF REGISTER HOLDS CORRECT DATA  
BEQ 2$ ;BRANCH TO EXIT IF DATA RIGHT  
ERROR 23 ;DIDN'T LOAD CORRECT BYTE  
2$: MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
```

6248
6249
6250
6251
6252
6253
6254
6255
6256

```
*****  
*TEST 42      BYTE ADDRESSING OF KERNEL PAGE DESCRIPTOR REGISTERS  
*  
* THIS TEST CHECKS THE 'SAPD WRITE LOBYTE' AND 'SAPD WRITE HIBYTE'  
* SIGNAL GENERATION FOR KERNEL PAGE ADDRESS REGISTERS.  
* BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT  
* GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.  
*****
```

6257
6258
6259
6260

```
TST42:  
SCOPE  
MOV #TST43,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
;TEST FOR ESCAPE ON PARITY ERRORS
```


6261	047050	012737	047066	001112	20\$:	MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
6262	047056	012702	000015			MOV	#15, R2	:PUT EXPECTED DATA IN R2
6263	047062	012700	172300			MOV	#KIPDR0, R0	:PUT ADDRESS OF REGISTER IN R0
6264	047066	005037	172300		11\$:	CLR	KIPDR0	:CLEAR THE REGISTER UNDER TEST
6265	047072	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6266	047074	112710	172015			MOVB	#172015, (R0)	:LOAD LOWER BYTE OF REGISTER
6267	047100	013701	172300			MOV	KIPDR0, R1	:READ REGISTER INTO R1
6268	047104	020102				CMP	R1, R2	:SEE IF ONLY LOWER BYTE WAS WRITTEN
6269	047106	001401				BEQ	1\$:BRANCH IF DATA MATCHES
6270	047110	104023				ERROR	23	:DIDN'T LOAD CORRECT BYTE
6271	047112	012737	047130	001112	1\$:	MOV	#12\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 12\$
6272	047120	012700	172301			MOV	#KIPDR0+1, R0	:POINT TO UPPER BYTE OF REGISTER
6273	047124	012702	052015			MOV	#052015, R2	:LOAD EXPECTED DATA IN R2
6274	047130	000240			12\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
6275	047132	112710	000124			MOVB	#124, (R0)	:WRITE THE UPPER BYTE OF REGISTER
6276	047136	013701	172300			MOV	KIPDR0, R1	:READ REGISTER INTO R1
6277	047142	020102				CMP	R1, R2	:SEE IF REGISTER HOLDS CORRECT DATA
6278	047144	001401				BEQ	2\$:BRANCH TO EXIT IF DATA RIGHT
6279	047146	104023				ERROR	23	:DIDN'T LOAD CORRECT BYTE
6280	047150	012737	047050	001112	2\$:	MOV	#20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST

 *TEST 43 BYTE ADDRESSING OF SUPERVISOR PAGE DESCRIPTOR REGISTERS

 THIS TEST CHECKS THE 'SAPE WRITE LOBYTE' AND 'SAPE WRITE HIBYTE'
 SIGNAL GENERATION FOR SUPERVISOR PAGE ADDRESS REGISTERS.
 BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
 GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.

6292	047156					TST43:	SCOPE	
6293	047156	000004					MOV	#TST44, NXTTST
6294	047160	012737	047274	001316				:SAVE STARTING ADDRESS OF NEXT
6295								:TEST FOR ESCAPE ON PARITY ERRORS
6296	047166	012737	047204	001112	20\$:	MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
6297	047174	012702	000015			MOV	#15, R2	:PUT EXPECTED DATA IN R2
6298	047200	012700	172200			MOV	#SIPDR0, R0	:PUT ADDRESS OF REGISTER IN R0
6299	047204	005037	172200		11\$:	CLR	SIPDR0	:CLEAR THE REGISTER UNDER TEST
6300	047210	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6301	047212	112710	172015			MOVB	#172015, (R0)	:LOAD LOWER BYTE OF REGISTER
6302	047216	013701	172200			MOV	SIPDR0, R1	:READ REGISTER INTO R1
6303	047222	020102				CMP	R1, R2	:SEE IF ONLY LOWER BYTE WAS WRITTEN
6304	047224	001401				BEQ	1\$:BRANCH IF DATA MATCHES
6305	047226	104023				ERROR	23	:DIDN'T LOAD CORRECT BYTE
6306	047230	012737	047246	001112	1\$:	MOV	#12\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 12\$
6307	047236	012700	172201			MOV	#SIPDR0+1, R0	:POINT TO UPPER BYTE OF REGISTER
6308	047242	012702	052015			MOV	#052015, R2	:LOAD EXPECTED DATA IN R2
6309	047246	000240			12\$:	NOP		:THIS IS A SYNC POINT FOR SCOPING
6310	047250	112710	000124			MOVB	#124, (R0)	:WRITE THE UPPER BYTE OF REGISTER
6311	047254	013701	172200			MOV	SIPDR0, R1	:READ REGISTER INTO R1
6312	047260	020102				CMP	R1, R2	:SEE IF REGISTER HOLDS CORRECT DATA
6313	047262	001401				BEQ	2\$:BRANCH TO EXIT IF DATA RIGHT
6314	047264	104023				ERROR	23	:DIDN'T LOAD CORRECT BYTE
6315	047266	012737	047166	001112	1\$:	MOV	#20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST
6316								

6317
6318
6319
6320
6321
6322
6323
6324
6325
6326
6327 047274
6328 047274 000004
6329 047276 012737 047412 001316
6330
6331 047304 012737 047322 001112 20\$:
6332 047312 012702 000015
6333 047316 012700 177600
6334 047322 005037 177600 11\$:
6335 047326 000240
6336 047330 112710 172015
6337 047334 013701 177600
6338 047340 020102
6339 047342 001401
6340 047344 104023
6341 047346 012737 047364 001112 1\$:
6342 047354 012700 177601
6343 047360 012702 052015
6344 047364 000240 12\$:
6345 047366 112710 000124
6346 047372 013701 177600
6347 047376 020102
6348 047400 001401
6349 047402 104023
6350 047404 012737 047304 001112 2\$:

*TEST 44 BYTE ADDRESSING OF USER PAGE DESCRIPTOR REGISTERS
*
* THIS TEST CHECKS THE 'SAPF WRITE LOBYTE' AND 'SAPF WRITE HIBYTE'
* SIGNAL GENERATION FOR USER PAGE ADDRESS REGISTERS.
* BOTH SIGNALS ARE CONTROLLED BY THE LOGIC ON 'SAPK' THAT
* GENERATES 'SAPK LO BYTE B H' AND 'SAPK HI BYTE B H'.

TST44:
SCOPE
MOV #TST45,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
MOV #11\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 11\$
MOV #15,R2 ;PUT EXPECTED DATA IN R2
MOV #UIPDRO,R0 ;PUT ADDRESS OF REGISTER IN R0
CLR UIPDRO ;CLEAR THE REGISTER UNDER TEST
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOVB #172015,(R0) ;LOAD LOWER BYTE OF REGISTER
MOV UIPDRO,R1 ;READ REGISTER INTO R1
CMP R1,R2 ;SEE IF ONLY LOWER BYTE WAS WRITTEN
BEQ 1\$;BRANCH IF DATA MATCHES
ERROR 23 ;DIDN'T LOAD CORRECT BYTE
MOV #12\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 12\$
MOV #UIPDRO+1,R0 ;POINT TO UPPER BYTE OF REGISTER
MOV #052015,R2 ;LOAD EXPECTED DATA IN R2
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOVB #124,(R0) ;WRITE THE UPPER BYTE OF REGISTER
MOV UIPDRO,R1 ;READ REGISTER INTO R1
CMP R1,R2 ;SEE IF REGISTER HOLDS CORRECT DATA
BEQ 2\$;BRANCH TO EXIT IF DATA RIGHT
ERROR 23 ;DIDN'T LOAD CORRECT BYTE
MOV #20\$,\$LPERR ;SET LOOP POINTER TO START OF TEST

.SBTTL DUAL ADDRESSING BETWEEN GROUPS OF REGISTERS

*TEST 45 DUAL ADDRESSING FOR ALL PAR'S AND PDR'S
*
* THIS TEST WILL ENSURE THAT THERE IS NO DUAL ADDRESSING BETWEEN
* GROUPS OF PAR'S AND PDR'S. THAT IS, WHEN YOU REFERENCE A KERNEL
* I-SPACE PAR YOU ARE REALLY REFERENCING THAT PAR. FIRST EACH
* I-SPACE PAR OR PDR IS LOADED WITH A UNIQUE NUMBER 0-12 AND
* THEN THEY ARE EACH CHECKED FOR THE CORRECT DATA.
* EACH GROUP HAS ALREADY BEEN CHECKED FOR DUAL ADDRESSING WITHIN
* ITS OWN GROUP, SO ONLY ONE REGISTER FROM EACH GROUP NEEDS TO
* BE TESTED HERE.

6351
6352
6353
6354
6355
6356
6357
6358
6359
6360
6361
6362
6363
6364
6365
6366
6367 047412
6368 047412 000004
6369 047414 012737 047514 001316
6370
6371 047422 012737 047436 001112 20\$:
6372 047430 005000

TST45:
SCOPE
MOV #TST46,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
MOV #1\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 1\$
CLR R0 ;THIS WILL HOLD THE INDEX OF EACH REGISTER

```
6373                                     ;AND THE COUNT LOADED
6374 047432 012704 000006                MOV #6,R4 ;THIS IS THE NUMBER OF TIMES TO DO THIS LOOP
6375 047436 010070 001322                MOV R0,@PARTAB(R0) ;LOAD PAR OR PDR WITH INDEX NUMBER
6376 047442 062700 000002                ADD #2,R0 ;CHANGE INDEX TO POINT TO NEXT REGISTER
6377 047446 077405                       SOB R4,1$ ;BRANCH BACK 5 TIMES
6378 047450 012704 000006                MOV #6,R4 ;DO NEXT LOOP 6 TIMES ALSO
6379 047454 012737 047466 001112        MOV #10$,$LPERR ;SET LOOP ON ERROR POINTER TO 10$
6380 047462 162700 000002                SUB #2,R0 ;ADJUST INDEX FOR REGISTER DESIRED
6381 047466 017001 001322                MOV @PARTAB(R0),R1 ;READ PAR OR PDR INTO R1
6382 047472 020001                       CMP R0,R1 ;SEE IF INDEX EQUALS DATA
6383 047474 001403                       BEQ 3$ ;BRANCH IFCORRECT REGISTER WAS READ.
6384 047476 016002 001322                MOV PARTAB(R0),R2 ;SAVE ADDRESS OF BAD REGISTER
6385 047502 104036                       ERROR 36 ;DUAL ADDRESSING
6386 047504 077412                       SOB R4,2$ ;BRANCH BACK 5 TIMES
6387 047506 012737 047422 001112        MOV #20$,$LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
```

```
.SBTTL ***** ENTRY POINT 4 --- STARTING ADDRESS 214 *****
.SBTTL ***** RELOCATION AND ADDER TESTS *****
```

```
THIS GROUP OF TESTS CHECKS MEMORY MANAGEMENT'S RELOCATION ADDER
FIRST USING DESTINATION ONLY RELOCATION, THEN WITH FULL 18-BIT
RELOCATION AND FINALLY WITH 22-BIT RELOCATION.
```

```
*****
:TEST 46 18-BIT MAPPING ADDER TESTING
```

```
THIS TEST USES 'DESTINATION ONLY' RELOCATION TO CHECK THE
FULL ADD PROPERTIES OF THE RELOCATION ADDER. TWELVE PAIRS
OF NUMBERS ARE ADDED, GENERATING PHYSICAL ADDRESSES ABOVE
16K.
```

```
*****
```

```
TST46:
SCOPE
MOV #TST47,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
```

```
THE FOLLOWING CODE WILL CLEAR ALL PAR'S AND PDR'S SO THAT
THE RELOCATION ADDERS CAN BE CHECKED, ONLY THE KERNEL I-SPACE
PDR'S AND PAR'S WILL BE MAPPED RESIDENT AND READ WRITE.
```

```
ENTPT4:
MOV #20$,$LPADR ;SET LOOP ADDRESS POINTER TO 20$
MOV #20$,$LPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #46,$STSTM ;LOAD TEST NUMBER INTO MEMORY
MOV $STSTM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
MOV #KIPDR0,R5 ;PUT ADDRESS OF KIPDR0 IN R5
JSR PC,CLRREG ;CLEAR KERNEL PDR'S
MOV #KIPAR0,R5 ;PUT ADDRESS OF KIPAR0 IN R5
JSR PC,CLRREG ;CLEAR KERNEL PAR'S
MOV #SIPDR0,R5 ;PUT ADDRESS OF SIPDR0 IN R5
JSR PC,CLRREG ;CLEAR SUPERVISOR PDR'S
MOV #SIPAR0,R5 ;PUT ADDRESS OF SIPAR0 IN R5
```

```
6408 047514
6409 047514 000004
6410 047516 012737 051130 001316
6411
6412
6413
6414
6415
6416
6417 047524
6418 047524 012737 050740 001110
6419 047532 012737 050740 001112
6420 047540 012737 000046 001102
6421 047546 013737 001102 177570
6422 047554 012705 172300
6423 047560 004737 031330
6424 047564 012705 172340
6425 047570 004737 031330
6426 047574 012705 172200
6427 047600 004737 031330
6428 047604 012705 172240
```

6429	047610	004737	031330			JSR	PC,CLRREG	:CLEAR SUPERVISOR PAR'S
6430	047614	012705	177600			MOV	#UIPDRO,R5	:PUT ADDRESS OF UIPDRO IN R5
6431	047620	004737	031330			JSR	PC,CLRREG	:CLEAR USER PDR'S
6432	047624	012705	177640			MOV	#UIPAR0,R5	:PUT ADDRESS OF UIPARO IN R5
6433	047630	004737	031330			JSR	PC,CLRREG	:CLEAR USER PAR'S
6434	047634	012700	077406	30\$:		MOV	#77406,R0	:MAKE KERNEL I PAGES 4K, R/W, EXPAND UP
6435	047640	012702	000010			MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
6436	047644	012701	172300			MOV	#KIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
6437	047650	010021		64\$:		MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
6438	047652	077202				SOB	R2,64\$:BRANCH BACK TO 64\$ IF R2 IS NOT ZERO
6439	047654	012737	000000	172340		MOV	#000,KIPAR0	:MAP PAGE 0 TO 0 - 4K
6440	047662	012737	000200	172342		MOV	#200,KIPAR1	:MAP PAGE 1 TO 4K - 8K
6441	047670	012737	000400	172344		MOV	#400,KIPAR2	:MAP PAGE 2 TO 8K - 12K
6442	047676	012737	000600	172346		MOV	#600,KIPAR3	:MAP PAGE 3 TO 12K - 16K
6443	047704	012737	177600	172356		MOV	#177600,KIPAR7	:MAP PAGE 7 TO I/O PAGE
6444	047712	012737	047744	001112		MOV	#1\$,SLPERR	:SET LOOP ON ERROR POINTER TO 1\$
6445	047720	013737	100000	001172		MOV	@#100000,\$TMP0	:SAVE DATA AT TEST LOCATION
6446	047726	012737	001000	172350		MOV	#1000,@#KIPAR4	:LOAD PAR4 WITH 1000
6447	047734	012700	100000			MOV	#100000,R0	:PUT VIRTUAL ADDRESS IN R0
6448	047740	012701	125200			MOV	#125200,R1	:PUT DATA PATTERN IN R1
6449	047744	052737	000400	177572	1\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6450	047752	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6451	047754	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 100000
6452	047756	013702	100000			MOV	@#100000,R2	:READ (100000) INTO R2
6453	047762	000005				RESET		:CLEAR MMR0
6454	047764	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6455	047766	001401				BEQ	2\$:BRANCH IF DATA MATCHES
6456	047770	104037				ERROR	37	:RELOCATION FAILED
6457	047772	013737	001172	100000	2\$:	MOV	\$TMP0,@#100000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6458	050000	012737	050032	001112		MOV	#3\$,SLPERR	:SET LOOP ON ERROR POINTER TO 3\$
6459	050006	013737	137776	001172		MOV	@#137776,\$TMP0	:SAVE DATA AT TEST LOCATION
6460	050014	012737	001252	172350		MOV	#1252,@#KIPAR4	:LOAD PAR4 WITH 1252
6461	050022	012700	112576			MOV	#112576,R0	:PUT VIRTUAL ADDRESS IN R0
6462	050026	012701	125201			MOV	#125201,R1	:PUT DATA PATTERN IN R1
6463	050032	052737	000400	177572	3\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6464	050040	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6465	050042	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 137776
6466	050044	013702	137776			MOV	@#137776,R2	:READ (137776) INTO R2
6467	050050	000005				RESET		:CLEAR MMR0
6468	050052	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6469	050054	001401				BEQ	4\$:BRANCH IF DATA MATCHES
6470	050056	104037				ERROR	37	:RELOCATION FAILED
6471	050060	013737	001172	137776	4\$:	MOV	\$TMP0,@#137776	:RESTORE ORIGINAL DATA TO TEST LOCATION
6472	050066	012737	050120	001112		MOV	#5\$,SLPERR	:SET LOOP ON ERROR POINTER TO 5\$
6473	050074	013737	117776	001172		MOV	@#117776,\$TMP0	:SAVE DATA AT TEST LOCATION
6474	050102	012737	001125	172350		MOV	#1125,@#KIPAR4	:LOAD PAR4 WITH 1125
6475	050110	012700	105276			MOV	#105276,R0	:PUT VIRTUAL ADDRESS IN R0
6476	050114	012701	125202			MOV	#125202,R1	:PUT DATA PATTERN IN R1
6477	050120	052737	000400	177572	5\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6478	050126	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6479	050130	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 117776
6480	050132	013702	117776			MOV	@#117776,R2	:READ (117776) INTO R2
6481	050136	000005				RESET		:CLEAR MMR0
6482	050140	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6483	050142	001401				BEQ	6\$:BRANCH IF DATA MATCHES
6484	050144	104037				ERROR	37	:RELOCATION FAILED

6485	050146	013737	001172	117776	6\$:	MOV	\$TMP0,@#117776	:RESTORE ORIGINAL DATA TO TEST LOCATION
6486	050154	012737	050206	001112		MOV	#7\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 7\$
6487	050162	013737	102000	001172		MOV	@#102000,\$TMP0	:SAVE DATA AT TEST LOCATION
6488	050170	012737	001010	172350		MOV	#1010,@#KIPAR4	:LOAD PAR4 WITH 1010
6489	050176	012700	101000			MOV	#101000,R0	:PUT VIRTUAL ADDRESS IN R0
6490	050202	012701	125203			MOV	#125203,R1	:PUT DATA PATTERN IN R1
6491	050206	052737	000400	177572	7\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6492	050214	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6493	050216	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 102000
6494	050220	013702	102000			MOV	@#102000,R2	:READ (102000) INTO R2
6495	050224	000005				RESET		:CLEAR MMR0
6496	050226	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6497	050230	001401				BEQ	8\$:BRANCH IF DATA MATCHES
6498	050232	104037				ERROR	37	:RELOCATION FAILED
6499	050234	013737	001172	102000	8\$:	MOV	\$TMP0,@#102000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6500	050242	012737	050274	001112		MOV	#9\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 9\$
6501	050250	013737	142000	001172		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
6502	050256	012737	001314	172350		MOV	#1314,@#KIPAR4	:LOAD PAR4 WITH 1314
6503	050264	012700	110400			MOV	#110400,R0	:PUT VIRTUAL ADDRESS IN R0
6504	050270	012701	125204			MOV	#125204,R1	:PUT DATA PATTERN IN R1
6505	050274	052737	000400	177572	9\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6506	050302	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6507	050304	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6508	050306	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
6509	050312	000005				RESET		:CLEAR MMR0
6510	050314	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6511	050316	001401				BEQ	10\$:BRANCH IF DATA MATCHES
6512	050320	104037				ERROR	37	:RELOCATION FAILED
6513	050322	013737	001172	142000	10\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6514	050330	012737	050362	001112		MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
6515	050336	013737	122000	001172		MOV	@#122000,\$TMP0	:SAVE DATA AT TEST LOCATION
6516	050344	012737	001104	172350		MOV	#1104,@#KIPAR4	:LOAD PAR4 WITH 1104
6517	050352	012700	111400			MOV	#111400,R0	:PUT VIRTUAL ADDRESS IN R0
6518	050356	012701	125205			MOV	#125205,R1	:PUT DATA PATTERN IN R1
6519	050362	052737	000400	177572	11\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6520	050370	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6521	050372	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 122000
6522	050374	013702	122000			MOV	@#122000,R2	:READ (122000) INTO R2
6523	050400	000005				RESET		:CLEAR MMR0
6524	050402	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6525	050404	001401				BEQ	12\$:BRANCH IF DATA MATCHES
6526	050406	104037				ERROR	37	:RELOCATION FAILED
6527	050410	013737	001172	122000	12\$:	MOV	\$TMP0,@#122000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6528	050416	012737	050450	001112		MOV	#13\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 13\$
6529	050424	013737	142000	001172		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
6530	050432	012737	001356	172350		MOV	#1356,@#KIPAR4	:LOAD PAR4 WITH 1356
6531	050440	012700	104200			MOV	#104200,R0	:PUT VIRTUAL ADDRESS IN R0
6532	050444	012701	125206			MOV	#125206,R1	:PUT DATA PATTERN IN R1
6533	050450	052737	000400	177572	13\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6534	050456	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6535	050460	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6536	050462	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
6537	050466	000005				RESET		:CLEAR MMR0
6538	050470	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6539	050472	001401				BEQ	14\$:BRANCH IF DATA MATCHES
6540	050474	104037				ERROR	37	:RELOCATION FAILED

6541	050476	013737	001172	142000	14\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6542	050504	012737	050536	001112		MOV	#15\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 15\$
6543	050512	013737	142000	001172		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
6544	050520	012737	001242	172350		MOV	#1242,@#KIPAR4	:LOAD PAR4 WITH 1242
6545	050526	012700	115600			MOV	#115600,R0	:PUT VIRTUAL ADDRESS IN R0
6546	050532	012701	125207			MOV	#125207,R1	:PUT DATA PATTERN IN R1
6547	050536	052737	000400	177572	15\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6548	050544	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6549	050546	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6550	050550	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
6551	050554	000005				RESET		:CLEAR MMR0
6552	050556	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6553	050560	001401				BEQ	16\$:BRANCH IF DATA MATCHES
6554	050562	104037				ERROR	37	:RELOCATION FAILED
6555	050564	013737	001172	142000	16\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6556	050572	012737	050624	001112		MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
6557	050600	013737	142000	001172		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
6558	050606	012737	001377	172350		MOV	#1377,@#KIPAR4	:LOAD PAR4 WITH 1377
6559	050614	012700	102100			MOV	#102100,R0	:PUT VIRTUAL ADDRESS IN R0
6560	050620	012701	125210			MOV	#125210,R1	:PUT DATA PATTERN IN R1
6561	050624	052737	000400	177572	17\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6562	050632	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6563	050634	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6564	050636	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
6565	050642	000005				RESET		:CLEAR MMR0
6566	050644	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6567	050646	001401				BEQ	18\$:BRANCH IF DATA MATCHES
6568	050650	104037				ERROR	37	:RELOCATION FAILED
6569	050652	013737	001172	142000	18\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6570	050660	012737	050712	001112		MOV	#19\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 19\$
6571	050666	013737	142000	001172		MOV	@#142000,\$TMP0	:SAVE DATA AT TEST LOCATION
6572	050674	012737	001221	172350		MOV	#1221,@#KIPAR4	:LOAD PAR4 WITH 1221
6573	050702	012700	117700			MOV	#117700,R0	:PUT VIRTUAL ADDRESS IN R0
6574	050706	012701	125211			MOV	#125211,R1	:PUT DATA PATTERN IN R1
6575	050712	052737	000400	177572	19\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6576	050720	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6577	050722	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 142000
6578	050724	013702	142000			MOV	@#142000,R2	:READ (142000) INTO R2
6579	050730	000005				RESET		:CLEAR MMR0
6580	050732	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6581	050734	001401				BEQ	20\$:BRANCH IF DATA MATCHES
6582	050736	104037				ERROR	37	:RELOCATION FAILED
6583	050740	013737	001172	142000	20\$:	MOV	\$TMP0,@#142000	:RESTORE ORIGINAL DATA TO TEST LOCATION
6584	050746	012737	051000	001112		MOV	#21\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 21\$
6585	050754	013737	140000	001172		MOV	@#140000,\$TMP0	:SAVE DATA AT TEST LOCATION
6586	050762	012737	001377	172350		MOV	#1377,@#KIPAR4	:LOAD PAR4 WITH 1377
6587	050770	012700	100100			MOV	#100100,R0	:PUT VIRTUAL ADDRESS IN R0
6588	050774	012701	125212			MOV	#125212,R1	:PUT DATA PATTERN IN R1
6589	051000	052737	000400	177572	21\$:	BIS	#BIT8,@#MMR0	:TURN ON DESTINATION ONLY RELOCATION
6590	051006	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
6591	051010	010110				MOV	R1,(R0)	:TRY TO LOAD DATA PATTERN INTO 140000
6592	051012	013702	140000			MOV	@#140000,R2	:READ (140000) INTO R2
6593	051016	000005				RESET		:CLEAR MMR0
6594	051020	020102				CMP	R1,R2	:SEE IF DATA MATCHES PATTERN
6595	051022	001401				BEQ	22\$:BRANCH IF DATA MATCHES
6596	051024	104037				ERROR	37	:RELOCATION FAILED


```
6597 051026 013737 001172 140000 22$: MOV $TMP0,@#140000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6598 051034 012737 051066 001112 MOV #23$, $LPERR ;SET LOOP ON ERROR POINTER TO 23$
6599 051042 013737 140000 001172 MOV @#140000,$TMP0 ;SAVE DATA AT TEST LOCATION
6600 051050 012737 001370 172350 MOV #1370,@#KIPAR4 ;LOAD PAR4 WITH 1370
6601 051056 012700 101000 MOV #101000,R0 ;PUT VIRTUAL ADDRESS IN R0
6602 051062 012701 125213 MOV #125213,R1 ;PUT DATA PATTERN IN R1
6603 051066 052737 000400 177572 23$: BIS #BIT8,@#MMRO ;TURN ON DESTINATION ONLY RELOCATION
6604 051074 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
6605 051076 010110 MOV R1,(R0) ;TRY TO LOAD DATA PATTERN INTO 140000
6606 051100 013702 140000 MOV @#140000,R2 ;READ (140000) INTO R2
6607 051104 000005 RESET ;CLEAR MMRO
6608 051106 020102 CMP R1,R2 ;SEE IF DATA MATCHES PATTERN
6609 051110 001401 BEQ 24$ ;BRANCH IF DATA MATCHES
6610 051112 104037 ERROR 37 ;RELOCATION FAILED
6611 051114 013737 001172 140000 24$: MOV $TMP0,@#140000 ;RESTORE ORIGINAL DATA TO TEST LOCATION
6612 051122 012737 047634 001112 MOV #30$, $LPERR ;SET LOOP ON ERROR POINTER TO START OF TEST
6613
6614 :*****
6615 :*TEST 47 18-BIT MAPPING CARRY PROPAGATION
6616 :*
6617 :* THIS TEST USES FULL 18-BIT RELOCATION TO CHECK THE CARRY
6618 :* PROPAGATION OF THE RELOCATION ADDER. SINCE THIS TEST SCANS
6619 :* MEMORY FROM 00100000 TO 00750000 ON 2K BOUNDARIES, IT WILL
6620 :* REPORT ANY HOLES THAT IT FINDS IN MEMORY UP TO THE SIZE
6621 :* JUMPERS. THE INFORMATION GIVEN WILL BE THE ADDRESS WHERE
6622 :* THE HOLE WAS DISCOVERED AND THE FIRST GOOD ADDRESS AFTER THE
6623 :* HOLE.
6624 :*****
6625 TST47:
6626 051130 SCOPE
6627 051130 000004 MOV #TST50,NXTTST ;SAVE STARTING ADDRESS OF NEXT
6628 051132 012737 051760 001316 ;TEST FOR ESCAPE ON PARITY ERRORS
6629 051140 005037 001304 20$: CLR HOLFLG ;MAKE SURE HOLE FLAG STARTS AT 0
6630 051144 012737 051574 000004 MOV #10$,ERRVEC ;SET ERRVEC POINTER TO 10$
6631 051152 012737 051740 000114 MOV #30$,CACHVEC ;SET UP CACHE VECTOR POINTER FOR SIZLO TO HIGH
6632 051160 012737 000777 172350 MOV #777,KIPAR4 ;LOAD PAR4 WITH STARTING BASE
6633 051166 012737 001000 172352 MOV #1000,KIPARS ;START ADDRESSING WITH 16K
6634 051174 012700 100100 MOV #100100,R0 ;LOAD VIRTUAL ADDR FOR PAR4 IN R0
6635 051200 012701 120000 MOV #120000,R1 ;LOAD VIRTUAL ADDR FOR PAR5 IN R1
6636 051204 012702 001000 MOV #1000,R2 ;LOAD DATA PATTERN INTO R2
6637 051210 012737 051300 001112 MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
6638 051216 012737 000001 177572 MOV #1,@#MMRO ;TURN ON 18-BIT MAPPING
6639
6640 ;FULL RELOCATION STARTS HERE AND CONTINUES FOR REST OF PROGRAM
6641
6642 051224 012737 051300 001112 1$: MOV #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
6643 051232 005037 001260 CLR PCPUER ;CLEAR CPU ERROR FLAG
6644 051236 011137 001172 MOV (R1),$TMP0 ;SAVE DATA AT TEST LOCATION
6645 051242 005737 001260 TST PCPUER ;SEE IF THERE WAS A CPU TRAP
6646 051246 001014 BNE 2$ ;BRANCH IF TRAP OCCURRED
6647 051250 005737 001304 TST HOLFLG ;SEE IF A HOLE WAS FOUND IN MEMORY
6648 051254 001411 BEQ 2$ ;BRANCH IF NO HOLE WAS FOUND
6649 051256 013737 172352 001176 MOV KIPAR5,$TMP2 ;SAVE PAR THAT POINTS TO END OF HOLE
6650 051264 012737 051140 001112 MOV #20$, $LPERR ;SET LOOP ON ERROR POINTER TO 20$
6651 051272 104040 ERROR 40 ;HOLE IN MEMORY FROM $TMP1 TO $TMP2
6652 051274 005037 001304 CLR HOLFLG ;CLEAR HOLE FLAG IN CASE THERE ARE MORE
```

```

6653 051300 000240      2$:  NOP                ;THIS A IS SYNC POINT FOR SCOPING
6654 051302 010210      MOV      R2,(R0)      ;LOAD TEST PATTERN INTO TEST LOCATION
6655 051304 011103      MOV      (R1),R3      ;READ TEST LOCATION VIA DIFFERENT V. A.
6656 051306 020203      CMP      R2,R3        ;SEE IF CORRECT LOCATION WAS REFERENCED
6657 051310 001401      BEQ      3$           ;BRANCH IF CORRECT DATA WAS OBTAINED
6658 051312 104042      ERROR   42           ;BAD RELOCATION
6659 051314 013711 001172 3$:  MOV      $TMP0,(R1)   ;RESTORE ORIGINAL DATA TO TEST LOCATION
6660 051320 062737 000100 172350 ADD      #100,KIPAR4  ;CHANGE BASE ADDRESS
6661 051326 062737 000100 172352 ADD      #100,KIPAR5  ;CHANGE BASE ADDRESS
6662 051334 005202      INC      R2           ;CHANGE DATA PATTERN
6663 051336 022737 007500 172352 CMP      #7500,KIPAR5 ;SEE IF PAST LAST ADDRESS
6664 051344 103327      BHS     1$           ;BRANCH IF NOT PAST LAST ADDRESS
6665 051346 005737 001304 TST      HOLFLG       ;SEE IF YOU ARE IN THE MIDDLE OF A HOLE
6666 051352 001401      BEQ      4$           ;BRANCH IF NOT IN MIDDLE OF A HOLE
6667 051354 104041      ERROR   41           ;IN MIDDLE OF A HOLE IN MEMORY
6668                                     ;HOLFLG HAS NO. OF TIME OUTS $TMP1
6669                                     ;HAS PAR OF FIRST TIMEOUT
6670                                     ;
6671                                     .SBTTL          TEST 'SAPN UNIBUS ADRS L' IN 18-BIT MAPPING
6672                                     :*
6673                                     :*
6674                                     :* ADDRESS 760000 IS GENERATED BY CARRY PROPAGATION WHILE THE
6675                                     :* PROGRAM IS EXPECTING A TIME OUT. (760000 IS THE FIRST I/O
6676                                     :* PAGE ADDRESS AND IS NOT IMPLEMENTED EXCEPT FOR DIAGNOSTICS)
6677                                     :* KIPAR4 WILL CONTAIN 007577 AND R0 HAS 100100 (WHICH
6678                                     :* SELECTS KIPAR4 AND GENERATES ADDRESS 760000).
6679                                     :* IF THE WRONG BIT IN THE CPU ERROR REGISTER IS SET OR IF
6680                                     :* NO TRAP TO 'ERRVEC' OCCURS AN ERROR IS REPORTED.
6681                                     :*
6682 051356 012737 031670 000004 4$:  MOV      #CPUER,ERRVEC ;RESTORE NORMAL ROUTINE FOR TRAPS THRU 4
6683 051364 012737 031774 000114      MOV      #MEMER,CACHVEC ;RESTORE NORMAL CACHE PARITY ERROR POINTER
6684 051372 012737 051424 001112      MOV      #40$, $LPERR   ;SET LOOP ON ERROR POINTER TO 40$
6685 051400 012737 007577 172350      MOV      #7577,KIPAR4  ;LOAD KIPAR4 WITH 007577
6686 051406 012702 007600      MOV      #7600,R2      ;LOAD DATA PATTERN INTO R2
6687 051412 012737 000020 001224      MOV      #20,CPUEXP    ;EXPECTING UNIBUS TIMEOUT
6688 051420 005037 001260      CLR      PCPUER        ;CLEAR CPU TRAP FLAG
6689 051424 000240      40$:  NOP                ;THIS A IS SYNC POINT FOR SCOPING
6690 051426 010210      MOV      R2,(R0)      ;LOAD 760000 THRU PAGE 4
6691                                     ;THIS INSTRUCTION SHOULD TIME OUT
6692                                     ;OVER THE UNIBUS.
6693 051430 005737 001260      TST      PCPUER        ;SEE IF TRAP OCCURRED
6694 051434 001001      BNE     6$           ;BRANCH IF TRAP
6695 051436 104043      ERROR   43           ;NO CPU TRAP
6696
6697                                     .SBTTL          TEST 'SAPN NOT CACHE ADRS H' 18-BIT MAPPING
6698                                     :*
6699                                     :*
6700                                     :* ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, THIS WILL
6701                                     :* CAUSE '18BIT PARAROUND' TO BE ASSERTED, KNOCKING DOWN
6702                                     :* 'NOT CACHE ADRS'. THEN, IF THERE IS LESS THAN 120K OF MEMORY
6703                                     :* ON THE SYSTEM, THE SIZE REGISTER IS USED AS A PAR AND A
6704                                     :* CARRY IS PROPAGATED TO CAUSE '18BIT OVERFLOW' TO BE ASSERTED
6705                                     :* WHICH SHOULD GENERATE 'NOT CACHE ADRS'.
6706                                     :*
6707 051440 012737 051464 001112 6$:  MOV      #16$, $LPERR  ;SET LOOP ON ERROR POINTER TO 16$
6708 051446 005037 001224      CLR      CPUEXP        ;NO TRAPS THRU ERRVEC EXPECTED HERE

```



```
6709 051452 012737 007777 172350 MOV #7777,KIPAR4 ;LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE
6710 051460 005037 000000 CLR @#000000 ;CLEAR ADDRESS ZERO
6711 051464 000240 16$: NOP ;THIS A IS SYNC POINT FOR SCOPING
6712 051466 013701 100100 MOV @#100'00,R1 ;THIS SHOULD READ ADDRESS ZERO INTO R1
6713 051472 005701 TST R1 ;SEE IF YOU REALLY READ ADDRESS ZERO
6714 051474 001401 BEQ 7$ ;BRANCH IF YOU READ ADDRESS ZERO
6715 051476 104044 ERROR 44 ;DIDN'T READ ADDRESS ZERO
6716 051500 012737 051536 001112 7$: MOV #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$
6717 051506 022737 000170 031142 CMP #170, $LSTBK ;IS THERE 120K ON THE SYSTEM?
6718 051514 101421 BLOS 8$ ;BRANCH IF MORE THAN 120K ON SYSTEM
6719 051516 012737 000040 001224 MOV #40, CPUEXP ;EXPECTING CACHE NON-EXISTANT MEMORY
6720 051524 005037 001260 CLR PCPUER ;CLEAR TRAP THRU ERRVEC FLAG
6721 051530 013737 177760 172350 MOV SIZELO, KIPAR4 ;GET READY TO GENERATE NON-EXIST ADDR.
6722 051536 000240 17$: NOP ;THIS A IS SYNC POINT FOR SCOPING
6723 051540 013701 100100 MOV @#100100, R1 ;READ FROM NON-EXISTANT ADDRESS
6724 051544 005737 001260 TST PCPUER ;SEE IF TRAP THRU ERRVEC OCCURRED
6725 051550 001003 BNE 8$ ;BRANCH TO EXIT IF TRAP HAPPENED
6726 051552 012700 100100 MOV #100100, R0 ;SAVE VIRTUAL ADDRESS FOR ERROR TYPE OUT
6727 051556 104045 ERROR 45 ;NO TRAP THRU ERRVEC
6728 051560 005037 001224 8$: CLR CPUEXP ;NO CPU TRAPS EXPECTED
6729 051564 012737 051140 001 12 MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
6730 051572 000472 BR TST50 ;BRANCH TO NEXT TEST
6731
6732 ;:***** TRAP TO HERE THRU ERRVEC *****
6733
6734 051574 012637 001306 10$: MOV (KSP)+, OLDPC ;SAVE RETURN ADDRESS
6735 051600 012637 001310 MOV (KSP)+, OLDPS ;SAVE OLD PROCESSOR STATUS
6736 051604 013737 177766 001260 MOV CPUERR, PCPUER ;SAVE CPU ERROR REGISTER
6737 051612 022737 000040 001260 CMP #40, PCPUER ;WAS TRAP NON-EXISTANT MEMORY
6738 051620 001012 BNE 12$ ;BRANCH IF MEMORY EXISTS
6739 051622 023737 172350 177760 CMP KIPAR4, SIZELO ;SEE IF PAR 4 MATCHES SIZE REGISTER
6740 051630 001004 BNE 11$ ;BRANCH IF NO MATCH, POSSIBLE ERROR
6741 ;IN COMPARE CIRCUIT
6742 051632 012737 051356 001306 MOV #4$, OLDPC ;CHANGE RETURN ADDRESS IF AT TOP OF MFM
6743 051640 000430 BR 14$ ;BRANCH TO EXIT
6744 051642 104050 11$: ERROR 50 ;COMPARE CIRCUIT FOR SIZE JUMPERS BAD
6745 051644 000426 BR 14$ ;BRANCH TO EXIT & CONTINUE
6746
6747 ;:***** COME HERE IF YOU DON'T GET CACHE NON-EXISTANT MEMORY ERROR
6748 ;:***** THERE MIGHT BE A HOLE IN MEMORY.
6749
6750 051646 022737 051140 001260 12$: CMP #20$, PCPUER ;SEE IF ADDRESS TIMED OUT
6751 051654 001011 BNE 13$ ;BRANCH IF NO TIME OUT, UNEXPECTED ERROR
6752 051656 005737 001304 TST HOLFLG ;HAS THIS HAPPENED BEFORE?
6753 051662 001003 BNE 15$ ;BRANCH IF NOT FIRST TIMEOUT
6754 051664 013737 172352 001174 MOV KIPAR5, $TMP1 ;SAVE PAR WHEN HOLE FIRST DISCOVERED
6755 051672 005237 001304 15$: INC HOLFLG ;KEEP COUNT OF CONSECUTIVE TIMEOUTS
6756 051676 000411 BR 14$ ;BRANCH TO EXIT AND CONTINUE TEST
6757 051700 013737 001306 001262 13$: MOV OLDPC, BADPC ;MOVE PC OF UNEXPECTED ERROR FOR TYPE OUT
6758 051706 104002 ERROR 2 ;UNEXPECTED TRAP THRU ERRVEC
6759 051710 013737 001110 001306 MOV $LPADR, OLDPC ;RETURN TO START OF TEST
6760 051716 005037 001304 CLR HOLFLG ;CLEAR HOLE FLAG IN CASE IT WAS SET
6761 051722 005037 177766 14$: CLR CPUERR ;CLEAR THE CPU ERROR REGISTER
6762 051726 013746 001310 MOV OLDPS, -(KSP) ;PUSH OLD PROCESSOR STATUS ON STACK
6763 051732 013746 001306 MOV OLDPC, -(KSP) ;PUSH RETURN ADDRESS ON STACK
6764 051736 000002 RTI ;RETURN TO TEST AND CONTINUE
```

```
6765  
6766  
6767  
6768 051740 012637 001306  
6769 051744 012637 001310  
6770 051750 012737 051356 001306  
6771 051756 000761  
6772  
6773  
6774  
6775  
6776  
6777  
6778  
6779  
6780  
6781  
6782  
6783  
6784  
6785  
6786 051760  
6787 051760 000004  
6788 051762 012737 052634 001316  
6789  
6790  
6791  
6792  
6793  
6794 051770 022737 007377 177760  
6795 051776 003105  
6796 052000 012737 000020 172516  
6797 052006 012737 052450 000004 20$:  
6798 052014 012737 052614 000114  
6799 052022 005037 001304  
6800 052026 012700 100100  
6801 052032 012701 120000  
6802 052036 012737 007377 172350  
6803 052044 012737 007400 172352  
6804 052052 012702 007400  
6805 052056 012737 052132 001112 1$:  
6806 052064 005037 001260  
6807 052070 011137 001172  
6808 052074 005737 001260  
6809 052100 001014  
6810 052102 005737 001304  
6811 052106 001411  
6812 052110 013737 172352 001176  
6813 052116 012737 052006 001112  
6814 052124 104125  
6815 052126 005037 001304  
6816 052132 000240 2$:  
6817 052134 010210  
6818 052136 011103  
6819 052140 020203  
6820 052142 001401
```

***** COME HERE IF TRAP TO 114 (SIZELO HIGHER THAN ACTUAL MEMORY)
30\$: MOV (KSP)+,OLDPC ;
MOV (KSP)+,OLDPS ;
MOV #4\$,OLDPC ;
BR 14\$;BRANCH TO EXIT

*TEST 50 22-BIT MAPPING CARRY PROPAGATION
*
* THIS TEST USES FULL 22-BIT RELOCATION TO CHECK THE CARRY
* PROPAGATION THAT PERTAINS TO 22 BIT ADDRESSES. THIS TEST
* ALSO SCANS MEMORY THIS TIME FROM ADDRESS 00740000
* TO 16740000 OR THE SIZE JUMPERS, ON 8K BOUNDARIES. AGAIN
* IF ANY HOLES ARE FOUND THE ADDRESS WHERE THEY ARE DISCOVERED
* AND THE FIRST GOOD ADDRESS AFTER THE HOLE WILL BE REPORTED

TS*50:
SCOPE
MOV #TST51,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS

22-BIT MAPPING STARTS HERE AND CONTINUES FOR THE REST OF
THE PROGRAM.

20\$: CMP #7377,SIZELO ;IS THERE AT LEAST 120K ON SYSTEM?
BGT 4\$;BRANCH IF LESS THAN 120K
MOV #BIT4,MMR3 ;ENABLE 22-BIT MAPPING
MOV #10\$,ERRVEC ;SET ERRVEC POINTER TO 10\$
MOV #30\$,CACHVEC ;SET UP CACHE VECTOR POINTER FOR SIZELO TO HIGH (KB11-EM)
CLR HOLFLG ;START HOLE FLAG AT ZERO
MOV #100100,R0 ;LOAD VIRTUAL ADDRESS FOR PAGE 4
MOV #120000,R1 ;LOAD VIRTUAL ADDRESS FOR PAGE 5
MOV #7377,KIPAR4 ;LOAD BASE ADDRESS INTO PAR 4
MOV #7400,KIPAR5 ;LOAD BASE ADDRESS +100 INTO PAR5
MOV #7400,R2 ;LOAD DATA PATTERN INTO R2
1\$: MOV #2\$,SLPERR ;SET LOOP ON ERROR POINTER TO 2\$
CLR PCPUER ;CLEAR CPU TRAP FLAG
MOV (R1),\$TMP0 ;SAVE DATA AT TEST LOCATION USING PAGE 5
TST PCPUER ;SEE IF CPU TRAP OCCURRED
BNE 2\$;BRANCH IF TRAP OCCURED
TST HOLFLG ;SEE IF A HOLE IN MEMORY WAS FOUND
BEQ 2\$;BRANCH IF NO HOLE WAS FOUND
MOV KIPAR5,\$TMP2 ;SAVE PAR THAT POINTS TO END OF HOLE
MOV #20\$,SLPERR ;SET LOOP ON ERROR POINTER TO 20\$
ERROR 125 ;HOLE IN MEMORY FROM \$TMP1 TO \$TMP2
CLR HOLFLG ;CLEAR FLAG IN CASE OF MORE HOLES
2\$: NOP ;THIS A IS SYNC POINT FOR SCOPING
MOV R2,(R0) ;WRITE DATA PATTERN INTO TEST LOCATION
MOV (R1),R3 ;READ TEST LOCATION VIA DIFFERENT VIRT.ADDR
CMP R2,R3 ;SEE IF DATA MATCHES
BEQ 3\$;BRANCH IF DATA IS GOOD

6821	052144	104046			ERROR	46	:BAD RELOCATION 22-BIT MAPPING	
6822	052146	013711	001172	3\$:	MOV	\$TMP0,(R1)	:RESTORE ORIGINAL DATA USING PAGE 5	
6823	052152	062702	000400		ADD	#400,R2	:CHANGE DATA PATTERN	
6824	052156	062737	000400	172350	ADD	#400,KIPAR4	:GET READY TO TEST NEXT BIT	
6825	052164	062737	000400	172352	ADD	#400,KIPAR5	:NEW PHYSICAL ADDRESS	
6826	052172	022737	167400	172352	CMP	#167400,KIPAR5	:MAKE SURE YOU DON'T GET ON THE UNIBUS	
6827	052200	103326			BHIS	1\$:BRANCH IF PAR 5 IS NOT PAST LIMIT	
6828	052202	005737	001304		TST	HOLFLG	:SEE IF MEMORY ENDS WITH A HOLE	
6829	052206	001401			BEQ	4\$:BRANCH IF NO HOLE AT END OF MEMORY	
6830	052210	104126			ERROR	126	:HOLE AT END OF MEMORY	
6831								
6832					.SBTTL	TEST 'SAPN UNIBUS ADRS L' IN 22-BIT MAPPING		
6833					:*			
6834					:*	UNIBUS ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION		
6835					:*	SINCE THE MAP IS DISABLED THIS SHOULD REFERENCE PHYSICAL		
6836					:*	ADDRESS 000000.		
6837					:*			
6838					:*			
6839	052212	012737	000020	172516	4\$:	MOV	#BIT4,MMR3	:ENABLE 22-BIT MAPPING
6840	052220	012737	031670	000004	MOV	#CPUER,ERRVEC	:RESTORE NORMAL CPU TRAP ROUTINE	
6841	052226	012737	031774	000114	MOV	#MEMER,CACHVEC	:RESTORE NORMAL CACHE PARITY ERROR POINTER	
6842	052234	012737	000020	001224	MOV	#20,CPUEXP	:POSSIBLE U.B. TIME OUT	
6843	052242	012737	167777	172350	MOV	#167777,KIPAR4	:GET READY TO TEST U.B. ADDRESS 0	
6844	052250	012737	000000	172352	MOV	#0,KIPAR5	:SHOULD GO TO PHYSICAL ADDRESS 0	
6845	052256	012702	017000		MOV	#17000,R2	:LOAD DATA PATTERN INTO R2	
6846	052262	012737	052274	001112	MOV	#5\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 6\$	
6847	052270	011037	001172		MOV	(R0), \$TMP0	:SAVE DATA IN LOCATION 0 USING PAGE 4	
6848	052274	000240			5\$:	NOP	:THIS A IS SYNC POINT FOR SCOPING	
6849	052276	010210			MOV	R2,(R0)	:LOAD DATA PATTERN INTO TEST LOCATION	
6850	052300	011103			MOV	(R1),R3	:READ TEST LOCATION VIA DIFFERENT V. A.	
6851	052302	013710	001172		MOV	\$TMP0,(R0)	:RESTORE ORIGINAL DATA USING PAGE 4	
6852	052306	020203			CMP	R2,R3	:SEE IF DATA MATCHES	
6853	052310	001401			BEQ	6\$:BRANCH IF DATA IS GOOD	
6854	052312	104047			ERROR	47	:BAD RELOCATION, UNIBUS ADDRESS	
6855								
6856					.SBTTL	TEST 'SAPN NOT CACHE ADRS H' 22-BIT MAPPING		
6857					:*			
6858					:*	ADDRESS 000000 IS GENERATED BY CARRY PROPAGATION, THIS WILL		
6859					:*	CAUSE '22BIT WRAPAROUND' TO BE ASSERTED, KNOCKING DOWN		
6860					:*	'NOT CACHE ADRS'. THEN THE SIZE REGISTER IS USED AS A PAR AND A		
6861					:*	CARRY IS PROPAGATED TO CAUSE 'ADRS OVERFLOW' TO BE ASSERTED		
6862					:*	WHICH SHOULD GENERATE 'NOT CACHE ADRS'.		
6863					:*			
6864					:*			
6865	052314	012737	052340	001112	6\$:	MOV	#16\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 16\$
6866	052322	005037	001224		CLR	CPUEXP	:NO TRAPS THRU ERRVEC EXPECTED HERE	
6867	052326	012737	177777	172350	MOV	#177777,KIPAR4	:LOAD PAR 4 WITH HIGHEST VALUE POSSIBLE	
6868	052334	005037	000000		CLR	@#000000	:CLEAR ADDRESS ZERO	
6869	052340	000240			16\$:	NOP	:THIS IS A SYNC POINT FOR SCOPING	
6870	052342	013701	100100		MOV	@#100100,R1	:THIS SHOULD READ ADDRESS ZERO INTO R1	
6871	052346	005701			TST	R1	:SEE IF YOU REALLY READ ADDRESS ZERO	
6872	052350	001401			BEQ	7\$:BRANCH IF YOU READ ADDRESS ZERO	
6873	052352	104044			ERROR	44	:DIDN'T READ ADDRESS ZERO	
6874	052354	012737	052412	001112	7\$:	MOV	#17\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 17\$
6875	052362	022737	167777	177760	CMP	#167777, \$SIZELO	:IS SIZE REGISTER L.E. TO 167777	
6876	052370	101421			BLOS	8\$:BRANCH IF MAXIMUM MEMORY IS ON SYSTEM	

```

6877 052372 012737 000040 001224      MOV      #40,CPUEXP      ;EXPECTING CACHE NON-EXISTANT MEMORY
6878 052400 005037 001260              CLR      PCPUER         ;CLEAR TRAP THRU ERRVEC FLAG
6879 052404 013737 177760 172350      MOV      SIZELO,KIPAR4  ;GET READY TO GENERATE NON-EXIST ADDR.
6880 052412 000240              NOP                     ;THIS IS A SYNC POINT FOR SCOPING
6881 052414 013701 100100              MOV      @#100100,R1    ;READ FROM NON-EXISTANT ADDRESS
6882 052420 005737 001260              TST      PCPUER         ;SEE IF TRAP THRU ERRVEC OCCURRED
6883 052424 001003              BNE      8$             ;BRANCH TO EXIT IF TRAP HAPPENED
6884 052426 012700 100100              MOV      #100100,R0    ;SAVE VIRTUAL ADDRESS FOR ERROR TYPE OUT
6885 052432 104045              ERROR    45            ;NO TRAP THRU ERRVEC
6886 052434 005037 001224 001112 8$:      CLR      CPUEXP         ;NO CPU TRAPS EXPECTED
6887 052440 012737 052006              MOV      #20$, $LPERR  ;SET LOOP POINTER TO START OF TEST
6888 052446 000472              BR       TST51         ;BRANCH TO NEXT TEST
6889
6890
6891      ;:***** TRAP TO HERE THRU ERRVEC *****
6892
6893 052450 012637 001306 10$:      MOV      (KSP)+,OLDPC   ;SAVE RETURN ADDRESS
6894 052454 012637 001300              MOV      (KSP)+,OLDPS  ;SAVE OLD PROCESSOR STATUS
6895 052460 013737 177766 001260      MOV      CPUERR,PCPUER ;SAVE CPU ERROR REGISTER FOR TYPING
6896 052466 022737 000040 001260      CMP      #40,PCPUER    ;WAS TRAP NON-EXISTANT MEMORY
6897 052474 001012              BNE      12$           ;BRANCH IF MEMORY EXISTS
6898 052476 023737 172350 177760      CMP      KIPAR4,SIZELO ;SEE IF PAR 4 MATCHES SIZE REGISTER
6899 052504 001004              BNE      11$           ;BRANCH IF NO MATCH, POSSIBLE ERROR
6900
6901 052506 012737 052212 001306      MOV      #4$,OLDPC     ;CHANGE RETURN ADDRESS IF AT TOP OF MEM
6902 052514 000430              BR       14$           ;BRANCH TO EXIT
6903 052516 104050 11$:      ERROR    50            ;COMPARE CIRCUIT FOR SIZE JUMPERS BAD
6904 052520 000426              BR       14$           ;BRANCH TO EXIT & CONTINUE
6905
6906      ;:***** COME HERE IF YOU DON'T GET CACHE NON-EXISTANT MEMORY ERROR
6907      ;:***** THERE MIGHT BE A HOLE IN MEMORY.
6908
6909 052522 022737 052006 001260 12$:      CMP      #20$,PCPUER   ;SEE IF ADDRESS TIMED OUT
6910 052530 001011              BNE      13$           ;BRANCH IF NO TIME OUT, UNEXPECTED ERROR
6911 052532 005737 001304              TST      HOLFLG        ;HAS THIS HAPPENED BEFORE?
6912 052536 001003              BNE      15$           ;BRANCH IF NOT FIRST TIMEOUT
6913 052540 013737 172352 001174      MOV      KIPAR5,$TMP1  ;SAVE PAR WHEN HOLE FIRST DISCOVERED
6914 052546 005237 001304 15$:      INC      HOLFLG        ;KEEP COUNT OF CONSECUTIVE TIMEOUTS
6915 052552 000411              BR       14$           ;BRANCH TO EXIT AND CONTINUE TEST
6916 052554 013737 001306 001262 13$:      MOV      OLDPC,BADPC   ;MOVE PC OF UNEXPECTED ERROR FOR TYPE OUT
6917 052562 104002              ERROR    2             ;UNEXPECTED TRAP THRU ERRVEC
6918 052564 013737 001110 001306      MOV      $LPADR,OLDPC  ;RETURN TO START OF TEST
6919 052572 005037 001304              CLR      HOLFLG        ;CLEAR HOLE FLAG IN CASE IT WAS SET
6920 052576 005037 177766 14$:      CLR      CPUERR        ;CLEAR THE CPU ERROR REGISTER
6921 052602 013746 001310              MOV      OLDPS,-(KSP)  ;PUSH OLD PROCESSOR STATUS ON STACK
6922 052606 013746 001306              MOV      OLDPC,-(KSP)  ;PUSH RETURN ADDRESS ON STACK
6923 052612 000002              RTI                    ;RETURN TO TEST AND CONTINUE
6924
6925      ;:***** COME HERE IF TRAP TO 114 (SIZELO HIGHER THAN ACTUAL MEMORY)
6926
6927 052614 012637 001306 30$:      MOV      (KSP)+,OLDPC  ;
6928 052620 012637 001310              MOV      (KSP)+,OLDPS  ;
6929 052624 012737 052212 001306      MOV      #4$,OLDPC     ;
6930 052632 000761              BR       14$           ;BRANCH TO EXIT
6931
6932

```

```

6933
6934
6935
6936
6937
6938
6939
6940
6941
6942
6943
6944
6945
6946
6947
6948
6949
6950
6951
6952
6953
6954
6955
6956
6957 052634
6958 052634 000004
6959 052636 012737 053166 001316
6960
6961 052644 012737 000024 001206
6962 052652
6963 052652 012737 053012 001110
6964 052660 012737 053012 001112
6965 052666 012737 000051 001102
6966 052674 013737 001102 177570
6967 052702 012737 077406 172300
6968 052710 012737 077406 172302
6969 052716 012737 077406 172304
6970 052724 012737 077406 172306
6971 052732 012737 077406 172316
6972 052740 012737 000000 172340
6973 052746 012737 000200 172342
6974 052754 012737 000400 172344
6975 052762 012737 000600 172346
6976 052770 012737 177600 172356
6977 052776 012737 000001 177572
6978 053004 012737 000020 172516
6979 053012 012737 000006 172310
6980 053020 012737 001000 172350
6981 053026 012700 172311
6982 053032 012737 053064 001112
6983 053040 005037 001250
6984 053044 012702 100000
6985 053050 010203
6986 053052 072327 177772
6987
6988 053056 042703 177600

```

```

.SBTTL ***** ENTRY POINT 5 --- STARTING ADDRESS 220 *****
.SBTTL ***** MEMORY MANAGEMENT ABORTS AND TRAPS LOGIC TESTS *****

```

```

:
: THIS GROUP OF TESTS CHECKS OUT THE MEMORY MANAGEMENT ABORT
: AND TRAP LOGIC ON PAGES 'SAPL', 'SSRC', AND 'SSRD'. IT WILL
: ALSO CHECK OUT BITS <07:01> OF MMR0, AND ALL BITS OF MMR2.
: IT THEN CHECKS THAT KERNEL MODE IS ALWAYS CLOCKED DURING
: A TRAP SEQUENCE SO THAT THE VECTOR IS PICKED UP FROM KERNEL
: SPACE.
:

```

```

:*****
:TEST 51 PAGE LENGTH FAULTS - UPWARD EXPANSION
:

```

```

: THIS TEST CHECKS OUT THE PAGE LENGTH COMPARATORS ON PAGE 'SAPL'
: AND THE LOGIC THAT GENERATES 'SAPL LENGTH FAULT'. IT TRIES
: EVERY PAGE LENGTH FIELD FROM 1 BLOCK TO 200(8) BLOCKS. THE
: TEST THEN TRIES A REFERENCE AT EVERY 100 BYTES UNTIL AN ABORT
: OCCURS. IF THE ABORT HAPPENS TOO SOON OR IF NO ABORT HAPPENS AT
: THE CORRECT BLOCK NUMBER AN ERROR IS REPORTED.
:

```

TST51:

```

SCOPE
MOV #TST52,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;DO 24 ITERATIONS
ENTPT5: MOV #24,$TIMES
MOV #20$,SLPADR ;SET LOOP ADDRESS POINTER TO 20$
MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #51,$TSTNM ;LOAD TEST NUMBER INTO MEMORY
MOV $TSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST
MOV #77406,KIPDR0 ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
MOV #77406,KIPDR1 ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
MOV #77406,KIPDR2 ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
MOV #77406,KIPDR3 ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
MOV #77406,KIPDR7 ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
MOV #000,KIPAR0 ;MAP KERNEL I PAGE 0 TO 0 - 4K
MCMV #200,KIPAR1 ;MAP KERNEL I PAGE 1 TO 4K - 8K
MOV #400,KIPAR2 ;MAP KERNEL I PAGE 2 TO 8K - 12K
MOV #600,KIPAR3 ;MAP KERNEL I PAGE 3 TO 12K - 16K
MOV #177600,KIPAR7 ;MAP KERNEL I PAGE 7 TO THE I/O PAGE
MOV #BIT0,MMR0 ;ENABLE 18-BIT RELOCATION IF NOT ON
MOV #BIT4,MMR3 ;ENABLE 22-BIT RELOCATION IF NOT ON
20$: MOV #000006,$KIPDR4 ;LOAD PDR4 FOR PAGE LENGTH OF 1
MOV #1000,KIPAR4 ;MAP PAGE 4 TO 16K
MOV #KIPDR4+1,R0 ;PUT ADDRESS OF PDR 4'S UPPER BYTE IN R0
MOV #3$,SLPERR ;SET LOOP ON ERROR POINTER TO 3$
1$: CLR PMMR0 ;CLEAR LOCATION THAT HOLDS MMR0
MOV #100000,R2 ;PUT VIRTUAL ADDRESS INTO R2
2$: MOV R2,R3 ;PUT VIRTUAL ADDRESS INTO R3 TOO
ASH #-6,R3 ;RIGHT SHIFT 6 BITS SO IT WILL
;MATCH THE PLF
BIC #177600,R3 ;CLEAR BITS THAT ARE SET IN UPPER BYTE

```



```
7045 053264 000240      NOP      ;THIS IS A SYNC POINT FOR SCOPING
7046 053266 011204      MOV      (R2),R4 ;READ USING V.A. IN R2
7047 053270 005037 001226 CLR      MMEXP ;CLEAR EXPECTED ABORT CONDITION
7048 053274 005737 001250 TST      PMMRO ;SEE IF ABORT OCCURRED YET
7049 053300 001405      BEQ      4$ ;BRANCH IF NO ABORT YET, CHANGE V.A.
7050 053302 005203      INC      R3 ;MAKE R3 EQUAL TO PLF
7051 053304 020103      CMP      R1,R3 ;SEE IF PDR 4'S PLF=R3
7052 053306 001414      BEQ      6$ ;BRANCH IF ABORT HAPPENS AT RIGHT PLACE
7053 053310 104051      ERROR   51 ;ABORT WRONG PLACE
7054 053312 000412      BR       6$ ;BRANCH TO CHANGE PLF
7055 053314 020103      4$: CMP      R1,R3 ;SEE IF PAGE LENGTH FAULT SHOULD HAVE OCCURRED
7056 053316 101402      BLOS    5$ ;BRANCH IF NO PAGE LENGTH FAULT CONDITION
7057 053320 104052      ERROR   52 ;NO ABORT, IT SHOULD HAVE HAPPENED THIS TIME
7058 053322 000406      BR       6$ ;BRANCH TO CHANGE PLF
7059 053324 120327 000000      5$: CMPB   R3,#000 ;SEE IF V.A. IS 000
7060 053330 001403      BEQ      6$ ;BRANCH TO CHECK PLF
7061 053332 162702 000100      SUB     #100,R2 ;CHANGE VIRTUAL ADDRESS
7062 053336 000741      BR       2$ ;GO TRY THE NEXT VIRTUAL ADDRESS
7063 053340 122710 000000      6$: CMPB   #000,(R0) ;SEE IF PDR 4'S PLF IS 000
7064 053344 001402      BEQ      7$ ;BRANCH TO EXIT IF PLF IS 000
7065 053346 105310      DECB   (R0) ;STEP PLF OF PDR 4 UP BY 1
7066 053350 000730      BR       1$ ;BRANCH TO START WITH V.A. OF 0
7067 053352 012737 053204 001112 7$: MOV     #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
```

```
7068
7069
7070
7071 :*****
7072 :*TEST 53 ACCESS CONTROL FIELD - 0, 3, OR 7 (ABORT ALL ACCESSES)
7073 :*
7074 :* THESE A.C.F.'S ARE ALL NON-RESIDENT, ANY REFERENCE (READ
7075 :* OR WRITE) TO A NON-RESIDENT PAGE SHOULD SET BIT 15 IN MMRO.
7076 :* BITS <06:01> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
7077 :* ON THAT PAGE. IN THIS CASE THE PAGE THAT CAUSES THE ABORT
7078 :* IS KERNEL I-SPACE PAGE 5. THE EXPECTED ERROR CODE IS 100013.
7079 :*****
```

```
7080 053360      TST53:
7081 053360 000004      SCOPE
7082 053362 012737 053644 001316      MOV     #TST54,NXTTST ;SAVE STARTING ADDRESS OF NEXT
7083 : ;TEST FOR ESCAPE ON PARITY ERRORS
7084 053370 012737 077406 172310 20$: MOV     #77406,@#KIPDR4 ;LOAD ACF 6 INTO PDR 4
7085 053376 012737 077400 172312      MOV     #77400,@#KIPDR5 ;LOAD ACF 0 INTO PDR5
7086
7087 053404 012737 001000 172350      MOV     #1000,@#KIPAR4 ;LOAD 16K INTO PAR4
7088 053412 012737 001000 172352      MOV     #1000,@#KIPAR5 ;LOAD 16K INTO PAR5
7089 053420 012700 100000      MOV     #100000,R0 ;LOAD VIRTUAL ADDRESS FOR PAR4 INTO R0
7090 053424 012701 120000      MOV     #120000,R1 ;LOAD VIRTUAL ADDRESS FOR PAR5 INTO R1
7091 053430 012702 161457      MOV     #161457,R2 ;LOAD DATA PATTERN INTO R2
7092 053434 012737 053444 001112 11$: MOV     #1$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
7093 053442 005010      CLR     (R0) ;CLEAR MEMORY LOCATION 100000
7094 053444 012737 100013 001226 1$: MOV     #100013,MMEXP ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
7095 : ;KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
7096 053452 000240      NOP ;THIS IS A SYNC POINT FOR SCOPING
7097 053454 010211      8$: MOV     R2,(R1) ;WRITE TO NON-RESIDENT OR UNUSED ACF
7098 : ;SHOULD CAUSE ABORT
7099 053456 005037 001226      CLR     MMEXP ;CLEAR EXPECTED ABORT CONDITION
7100
```



```
7101 053462 005710      TST      (R0)      ;SEE IF (100000) IS STILL ZERO
7102 053464 001401      BEQ      2$        ;BRANCH IF (100000) IS ZERO
7103 053466 104053      ERROR   53        ;ABORT DID NOT HAPPEN
7104 053470 012737 053502 001112 2$:  MOV     #4$, $LPERR ;SET LOOP ON ERROR POINTER TO 4$
7105 053476 010210      MOV     R2, (R0)   ;LOAD DATA PATTERN INTO 100000
7106 053500 005004      CLR     R4        ;CLEAR REGISTER TO RECEIVE DATA
7107 053502 012737 100013 001226 4$:  MOV     #100013, MMEXP ;LOAD EXPECTED ABORT CONDITION: NON-RESIDENT,
7108                                ;KERNEL, I-SPACE, PAGE 5, FULL RELOCATION
7109 053510 000240      NOP                                ;THIS IS A SYNC POINT FOR SCOPING
7110 053512 011104      MOV     (R1), R4   ;TRY TO READ (100000) INTO R4
7111                                ;THIS SHOULD ABORT
7112 053514 005037 001226      CLR     MMEXP     ;EXPECTED ABORT CONDITION
7113
7114 053520 005704      TST     R4        ;MAKE SURE R4 IS STILL 0
7115 053522 001401      BEQ     5$        ;BRANCH IF R4 IS 0
7116 053524 104053      ERROR   53        ;ABORT DID NOT HAPPEN
7117 053526 023727 172312 077407 5$:  CMP     KIPDR5, #77407 ;SEE IF PDR 5'S ACF=7
7118 053534 001414      BEQ     12$       ;TEST OVER IF ACF=7
7119 053536 023727 172312 077403  CMP     KIPDR5, #77403 ;SEE IF PDR 5'S ACF=3
7120 053544 001004      BNE     10$       ;BRANCH TO MAKE ACF=3 IF NOT 3
7121 053546 012737 077407 172312  MOV     #77407, KIPDR5 ;MAKE ACF=7 IF ALREADY 3
7122 053554 000727      BR      11$       ;REPEAT TEST WITH ACF=7
7123 053556 012737 077403 172312 10$:  MOV     #77403, KIPDR5 ;MAKE PDR 5'S ACF=3
7124 053564 000723      BR      11$       ;REPEAT TEST WITH ACF=3
7125
7126                                ;:
7127                                ;:
7128                                ;:
7129 053566 012737 053614 001112 12$:  MOV     #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$
7130 053574 012737 177600 172352  MOV     #177600, KIPAR5 ;MAP PAGE 5 TO I/O PAGE
7131 053602 012701 132350  MOV     #132350, R1   ;LOAD VIRTUAL ADDRESS TO REFERENCE
7132                                ;KIPAR4
7133 053606 012737 100013 001226  MOV     #100013, MMEXP ;EXPECTING NON-RESIDENT ABORT
7134                                ;KERNEL I PAGE 5
7135 053614 000240      NOP                                ;THIS IS A SYNC POINT FOR SCOPING
7136 053616 005011      CLR     (R1)     ;THIS INSTRUCTION SHOULD ABORT
7137                                ;DURING THE ABORT 'SSRC INH T3' IS
7138                                ;ASSERTED TO STOP THE CLOCKING OF THE
7139                                ;FLIP/FLOPS ON SCCX
7140 053620 022737 001000 172350  CMP     #1000, KIPAR4 ;KIPAR4 SHOULD STILL BE 1000
7141 053626 001401      BEQ     14$       ;BRANCH IF KIPAR4 IS STILL 1000
7142 053630 104127      ERROR   127      ;ABORT DID NOT STOP CLRING OF KIPAR4
7143 053632 012737 053370 001112 14$:  MOV     #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
7144 053640 005037 001226      CLR     MMEXP     ;NOT EXPECTING ANY TRAPS
7145
7146
7147
7148
7149
7150
7151
7152
7153
7154
7155
7156
```

THIS SECTION OF CODE VERIFIES THAT YOU WON'T MODIFY A MEMORY MANAGEMENT REGISTER ON A M.M. ABORT REFERENCE.

```
*****
*TEST 54      ACCESS CONTROL FIELD - 2 (ABORT ON WRITE)
*
*      THIS IS A READ ONLY A.C.F., ANY WRITE ATTEMPT TO THIS PAGE
*      WILL ABORT AND SET BIT 13 OF MMRO.
*      BITS <06:01> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
*      ON THAT PAGE. IN THIS CASE THE WRITE ATTEMPT WILL BE TO
*      KERNEL I-SPACE PAGE 4. THE EXPECTED ERROR CODE IS 020011.
*****
```



```
7157 053644          TST54:
7158 053644 000004          SCOPE
7159 053646 012737 053774 001316  MOV      #TST55,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
7160                                     ;TEST FOR ESCAPE ON PARITY ERRORS
7161 053654          20$:
7162 053654 012737 001000 172352  MOV      #1000,KIPAR5  ;MAP PAGE 5 TO 16K
7163 053662 012737 001000 172350  MOV      #1000,KIPAR4  ;MAP PAGE 4 TO 16K
7164 053670 012737 077406 172312  MOV      #77406,KIPDR5 ;PAGE 5 IS 200 BLOCKS LONG,
7165                                     ;EXPANDS UPWARD, AND IS READ/WRITE
7166                                     ;WITH NO TRAPPING
7167 053676 012737 077402 172310  MOV      #77402,KIPDR4 ;LOAD ACF 2 INTO PDR 4
7168 053704 005037 001226          CLR      MMEXP          ;NOT EXPECTING ANY TRAPS OR ABORTS YET
7169 053710 012737 002222 120000  MOV      #2222,@#120000 ;LOAD DATA INTO 16K
7170 053716 013700 100000          MOV      @#100000,R0    ;READ DATA THRU PAGE 4
7171 053722 012737 053734 001112 1$:  MOV      #11$, $LPERR  ;SET LOOP ON ERROR POINTER TO 11$
7172 053730 005037 120000          CLR      @#120000      ;CLEAR TEST LOCATION THRU PAGE 5
7173 053734 012737 020011 001226 11$: MOV      #20011,MMEXP  ;EXPECTING READ ONLY FAULT, PAGE 4
7174 053742 000240          NOP                       ;THIS IS A SYNC POINT FOR SCOPING
7175 053744 012737 017777 100000  MOV      #17777,@#100000 ;TRY TO WRITE THRU PAGE 4
7176 053752 005037 001226          CLR      MMEXP          ;NO MORE TRAPS EXPECTED
7177 053756 005737 120000          TST      @#120000      ;SEE IF TEST LOCATION IS STILL ZERO
7178 053762 001401          BEQ      2$            ;BRANCH IF WORD IS STILL ZERO
7179 053764 104054          ERROR   54            ;NO ABORT ON PAGE 4
7180 053766 012737 053654 001112 2$:  MOV      #20$, $LPERR  ;SET LOOP POINTER TO START OF TEST
```

```
7181
7182
7183 *****
7184 *TEST 55      ACCESS CONTROL FIELD = 1 (ABORT ON WRITE, TRAP ON READ)
7185 *
7186 *      THIS IS ANOTHER READ ONLY A.C.F., ALL WRITES TO THIS PAGE WILL
7187 *      ABORT AND SET BIT 13 OF MMRO.
7188 *      BITS <06:01> IN MMRO WILL LOCK UP ALL AVAILABLE INFORMATION
7189 *      ON THAT PAGE.  IN THIS CASE THE PAGE THAT CAUSES THE ABORT
7190 *      IS KERNEL I-SPACE PAGE 4.  THE EXPECTED ERROR CODE IS 020011.
7191 *
7192 *      IF BIT 09 OF MMRO (ENABLE MEMORY MANAGEMENT TRAPS) IS SET
7193 *      THEN ALL READS TO THIS PAGE WILL TRAP, AFTER THE INSTRUCTION
7194 *      IS COMPLETED, SETTING BIT 12 OF MMRO.
7195 *
7196 *      AFTER THE ABORT ON WRITE IS TESTED, A READ FROM THIS PAGE
7197 *      WITH BIT 9 CLEAR WILL BE DONE TO ENSURE THAT TRAPPING DOESN'T
7198 *      TAKE PLACE WHEN NOT ENABLED.  THEN BIT 09 IS SET AND ANOTHER
7199 *      READ IS DONE (THIS TIME IT SHOULD TRAP TO PAGE 1 KERNEL MODE).
7200 *
```

```
7201 *****
7202 TST55:
7203 053774 000004          SCOPE
7204 053776 012737 054240 001316  MOV      #TST56,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
7205                                     ;TEST FOR ESCAPE ON PARITY ERRORS
7206 054004          20$:
7207 054004 012737 001000 172352  MOV      #1000,KIPAR5  ;MAP PAGE 5 TO 16K
7208 054012 012737 001000 172350  MOV      #1000,KIPAR4  ;MAP PAGE 4 TO 16K
7209 054020 012737 077406 172312  MOV      #77406,KIPDR5 ;PAGE 5 IS 200 BLOCKS LONG,
7210                                     ;EXPANDS UPWARD, AND IS READ/WRITE
7211                                     ;WITH NO TRAPPING
7212 054026 012737 077401 172310  MOV      #77401,KIPDR4 ;SET ACF = 1 FOR PAGE 4
```

7213	054034	012737	054046	001112		MOV	#10\$, \$LPERR	;SET LOOP ON ERROR POINTER TO 10\$
7214	054042	005037	001250			CLR	PMMR0	;CLEAR M.M. ABORT FLAG
7215	054046	012737	020011	001226	10\$:	MOV	#20011, MMEXP	;READ ONLY ABORT, PAGE 4
7216	054054	000240				NOP		;THIS IS A SYNC POINT FOR SCOPING
7217	054056	012737	017777	100000		MOV	#17777, @#100000	;TRY TO WRITE THRU PAGE 4
7218	054064	005037	001226			CLR	MMEXP	;NO MORE TRAPS EXPECTED
7219	054070	005737	001250			TST	PMMR0	;SEE IF M.M. ABORT HAPPENED
7220	054074	001001				BNE	1\$;BRANCH IF ABORT HAPPENED
7221	054076	104054				ERROR	54	;NO ABORT ON PAGE 4 A.C.F.=1
7222	054100	012737	054116	001112	1\$:	MOV	#11\$, \$LPERR	;SET LOOP ON ERROR POINTER TO 11\$
7223	054106	012700	012547			MOV	#12547, R0	;PUT DATA PATTERN INTO R0
7224	054112	010037	120000			MOV	R0, @#120000	;LOAD DATA PATTERN THRU PAGE 5
7225	054116	000240			11\$:	NOP		;THIS IS A SYNC POINT FOR SCOPING
7226	054120	013701	100000			MOV	@#100000, R1	;READ TEST LOCATION THRU PAGE 4
7227	054124	020100				CMP	R1, R0	;SEE IF DATA READ TOOK PLACE
7228	054126	001401				BEQ	2\$;BRANCH IF READ CORRECT
7229	054130	104055				ERROR	55	;INCORRECT READ, BIT09 (MMR0) IS CLEAR
7230	054132	013737	177572	001250	2\$:	MOV	MMR0, PMMR0	;SAVE MMR0 IN CASE OF ERROR
7231	054140	032737	010000	001250		BIT	#BIT12, PMMR0	;BIT 12 SHOULD SET EVEN IF YOU DON'T
7232								;TAKE THE TRAP DUE TO BIT09 BEING CLR
7233	054146	001001				BNE	3\$;BRANCH IF BIT12 IS SET
7234	054150	104066				ERROR	66	;BIT 12 WAS NOT SET
7235	054152	012737	054162	001112	3\$:	MOV	#12\$, \$LPERR	;SET LOOP ON ERROR POINTER TO 12\$
7236	054160	005001				CLR	R1	;CLEAR REG 1, RECEIVES DATA ON FETCH
7237	054162	005037	001250		12\$:	CLR	PMMR0	;CLEAR M.M. TRAP FLAG
7238	054166	012737	001001	177572		MOV	#1001, MMR0	;ENABLE TRAPS, FULL RELOCATION, CLR BIT 12
7239	054174	012737	011003	001226		MOV	#11003, MMEXP	;EXPECTED TRAP CONDITION
7240								;TRAP, TRAPS ENABLED, RELOCATION
7241	054202	000240				NOP		;THIS IS A SYNC POINT FOR SCOPING
7242	054204	013701	100000			MOV	@#100000, R1	;READ THRU PAGE 4
7243	054210	005037	001226			CLR	MMEXP	;NO MORE TRAPS EXPECTED
7244	054214	005737	001250			TST	PMMR0	;SEE IF TRAP OCCURED
7245	054220	001001				BNE	4\$;BRANCH IF TRAP OCCURRED
7246	054222	104056				ERROR	56	;NO TRAP
7247	054224	020001			4\$:	CMP	R0, R1	;SEE IF DATA READ WAS CORRECT
7248	054226	001401				BEQ	5\$;BRANCH IF READ CORRECT
7249	054230	104057				ERROR	57	;INCORRECT READ, BIT09 (MMR0) WAS SET
7250	054232	012737	054004	001112	5\$:	MOV	#20\$, \$LPERR	;SET LOOP POINTER TO START OF TEST

7251
7252
7253
7254
7255
7256
7257
7258
7259
7260
7261
7262
7263
7264
7265
7266
7267
7268

```
*****  
*TEST 56 ACCESS CONTROL FIELD = 4 (TRAP ON READ OR WRITE)  
*  
* THIS A.C.F. IS READ/WRITE BUT ALL REFERENCES TO THIS PAGE  
* WILL TRAP TO VECTOR 250 SETTING BIT 12 OF MMR0 IF BIT 9  
* (ENABLE MEMORY MANAGEMENT TRAPS) IS SET.  
*  
* SINCE I HAVE ALREADY TESTED THE FACT THAT BIT 9 OF MMR0 DOES  
* INDEED ENABLE M.M. TRAPS I WILL JUST SET BIT 9 AND VERIFY THAT  
* BOTH A READ AND A WRITE TO PAGE 4 A.C.F. = 4 TRAP CORRECTLY  
*****  
TST56:  
SCOPE  
MOV #TST57, NXTTST ;SAVE STARTING ADDRESS OF NEXT  
;TEST FOR ESCAPE ON PARITY ERRORS
```

7269	054250				20\$:					
7270	054250	012737	001000	172352		MOV	#1000,KIPAR5		:MAP PAGE 5 TO 16K	
7271	054256	012737	001000	172350		MOV	#1000,KIPAR4		:MAP PAGE 4 TO 16K	
7272	054264	012737	077406	172312		MOV	#77406,KIPDR5		:PAGE 5 IS 200 BLOCKS LONG,	
7273									:EXPANDS UPWARD, AND IS READ/WRITE	
7274									:WITH NO TRAPPING	
7275	054272	012737	077404	172310		MOV	#77404,KIPDR4		:SET ACF = 4 IN PDR 4	
7276	054300	012700	013451			MOV	#13451,R0		:LOAD DATA PATTERN INTO R0	
7277	054304	010037	120000			MOV	R0,@#120000		:LOAD DATA INTO TEST LOCATION	
7278	054310	005001				CLR	R1		:WHAT DO YOU THINK	
7279	054312	012737	054320	001112		MOV	#10\$,SLPERR		:SET LOOP ON ERROR POINTER TO 10\$	
7280	054320	005037	001250		10\$:	CLR	MMR0		:CLEAR FLAG LOCATION	
7281	054324	012737	011003	001226		MOV	#11003,MMEXP		:EXPECTING TRAP WITH TRAPS ENABLED	
7282									:KERNEL I PAGE 1, FULL RELOCATION	
7283	054332	012737	001001	177572		MOV	#1001,MMR0		:ENABLE M. M. TRAPS	
7284	054340	000240				NOP			:THIS IS A SYNC POINT FOR SCOPING	
7285	054342	013701	100000			MOV	@#100000,R1		:TRY TO READ THRU PAGE 4	
7286	054346	005037	001226			CLR	MMEXP		:NO MORE TRAPS EXPECTED	
7287	054352	005737	001250			TST	MMR0		:SEE IF TRAP OCCURRED	
7288	054356	001001				BNE	1\$:BRANCH IF TRAP	
7289	054360	104056				ERROR	56		:NO TRAP	
7290	054362	020001			1\$:	CMP	R0,R1		:SEE IF READ WAS CORRECT	
7291	054364	001401				BEQ	2\$:BRANCH IF CORRECT	
7292	054366	104057				ERROR	57		:INCORRECT READ, BIT09 (MMR0) WAS SET	
7293	054370	012737	054376	001112	2\$:	MOV	#12\$,SLPERR		:SET LOOP ON ERROR POINTER TO 12\$	
7294	054376	005037	001250		12\$:	CLR	MMR0		:CLEAR FLAG LOCATION	
7295	054402	012737	001001	177572		MOV	#1001,MMR0		:ENABLE TRAPPING	
7296	054410	012737	011003	001226		MOV	#11003,MMEXP		:EXPECTING TRAP WITH TRAPS ENABLED	
7297									:KERNEL I PAGE 1, FULL RELOCATON	
7298	054416	000240				NOP			:THIS IS A SYNC POINT FOR SCOPING	
7299	054420	012737	000000	100000		MOV	#0,@#100000		:TRY TO WRITE INTO PAGE 4	
7300	054426	005037	001226			CLR	MMEXP		:NOT EXPECTING ANY TRAPS	
7301	054432	005737	001250			TST	MMR0		:SEE IF TRAP OCCURRED	
7302	054436	001001				BNE	3\$:BRANCH IF TRAP	
7303	054440	104056				ERROR	56		:NO TRAP	
7304	054442	005737	120000		3\$:	TST	@#120000		:SEE IF WRITE OCCURED	
7305	054446	001401				BEQ	4\$:BRANCH IF WRITE HAPPENED	
7306	054450	104060				ERROR	60		:NO WRITE, BIT09 (MMR0) WAS SET	
7307	054452	012737	054250	001112	4\$:	MOV	#20\$,SLPERR		:SET LOOP POINTER TO START OF TEST	
7308										
7309										

```
*****  
: *TEST 57 ACCESS CONTROL FIELD = 5 (TRAP ON WRITE)  
: *  
: * THIS TEST IS RUN WITH THE ENABLE M.M. TRAPS BIT (BIT09) SET.  
: * THE A.C.F. FOR PAGE 4 IS SET TO 5 (TRAP ON WRITE).  
: * A READ FROM PAGE 4 IS TRIED EXPECTING NO TRAP AND THEN A WRITE  
: * TO PAGE 4 IS TRIED EXPECTING A M.M. TRAP. THE TRAP IS VERIFIED  
: * BY THE USE OF A TRAP FLAG WHICH IS SET IN THE TRAP ROUTINE.  
: *  
*****
```

7310						TST57:				
7311										
7312										
7313										
7314										
7315										
7316										
7317										
7318										
7319										
7320	054460					SCOPE				
7321	054460	000004				MOV	#TST60,NXTTST		:SAVE STARTING ADDRESS OF NEXT	
7322	054462	012737	054656	001316					:TEST FOR ESCAPE ON PARITY ERRORS	
7323										
7324	054470				20\$:					

7325	054470	012737	001000	172352	MOV	#1000,KIPAR5	:MAP PAGE 5 TO 16K
7326	054476	012737	001000	172350	MOV	#1000,KIPAR4	:MAP PAGE 4 TO 16K
7327	054504	012737	077406	172312	MOV	#77406,KIPDR5	:PAGE 5 IS 200 BLOCKS LONG,
7328							:EXPANDS UPWARD, AND IS READ/WRITE
7329							:WITH NO TRAPPING
7330	054512	012737	077405	172310	MOV	#77405,KIPDR4	:SET ACF = 5 IN PDR 4
7331	054520	012700	012345		MOV	#12345,R0	:SET DATA PATTERN INTO R0
7332	054524	010037	120000		MOV	R0,@#120000	:LOAD TEST LOCATION WITH DATA
7333	054530	005037	001226		CLR	MMEXP	:NO TRAP EXPECTED
7334	054534	005001			CLR	R1	:CLEAR REGISTER 1
7335	054536	012737	054544	001112	MOV	#10\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 10\$
7336	054544	012737	001001	177572	MOV	#1001,MMRO	:ENABLE M M TRAPS
7337	054552	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
7338	054554	013701	100000		MOV	@#100000,R1	:READ TEST LOCATION THRU PAGE 4
7339	054560	020001			CMP	R0,R1	:SEE IF READ WAS CORRECT
7340	054562	001401			BEQ	1\$:BRANCH IF READ CORRECT
7341	054564	104057			ERROR	57	:INCORRECT READ, BIT09 (MMRO) WAS SET
7342	054566	012737	054574	001112	MOV	#11\$, \$LPERR	:SET LOOP ON ERROR POINTER TO 11\$
7343	054574	005037	001250		CLR	PMMRO	:CLEAR FLAG LOCATION
7344	054600	012737	001001	177572	MOV	#1001,MMRO	:ENABLE TRAPS
7345	054606	012737	011003	001226	MOV	#11003,MMEXP	:EXPECTING M.M. TRAP
7346							
7347	054614	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
7348	054616	012737	000000	100000	MOV	#0,@#100000	:TRY TO WRITE INTO PAGE 4
7349	054624	005037	001226		CLR	MMEXP	:NO MORE TRAPS EXPECTED
7350	054630	005737	001250		TST	PMMRO	:SEE IF TRAP OCCURRED
7351	054634	001001			BNE	2\$:BRANCH IF TRAP
7352	054636	104056			ERROR	56	:NO TRAP
7353	054640	005737	120000		TST	@#120000	:SEE IF WRITE OCCURRED
7354	054644	001401			BEQ	3\$:BRANCH IF WRITE HAPPENED
7355	054646	104060			ERROR	60	:NO WRITE, BIT09 (MMRO) WAS SET
7356	054650	012737	054470	001112	MOV	#20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST
7357							
7358							
7359							
7360							
7361							
7362							
7363							
7364							
7365							
7366							
7367							
7368							
7369	054656				TST60:		
7370	054656	000004			SCOPE		
7371	054660	012737	055000	001316	MOV	#TST61,NXTTST	:SAVE STARTING ADDRESS OF NEXT
7372							:TEST FOR ESCAPE ON PARITY ERRORS
7373	054666						
7374	054666	012737	001000	172352	MOV	#1000,KIPAR5	:MAP PAGE 5 TO 16K
7375	054674	012737	001000	172350	MOV	#1000,KIPAR4	:MAP PAGE 4 TO 16K
7376	054702	012737	077406	172312	MOV	#77406,KIPDR5	:PAGE 5 IS 200 BLOCKS LONG,
7377							:EXPANDS UPWARD, AND IS READ/WRITE
7378							:WITH NO TRAPPING
7379	054710	012737	077404	172310	MOV	#77404,KIPDR4	:SET ACF = 4 IN PDR 4
7380	054716	005037	001226		CLR	MMEXP	:NO TRAPS EXPECTED

7381 054722 012737 017777 120000
7382 054730 012737 054736 001112
7383 054736 012737 011001 177572 10\$:
7384 054744 000240
7385 054746 012737 000000 100000
7386 054754 012737 000001 177572
7387 054762 005737 120000
7388 054766 001401
7389 054770 104060
7390 054772 012737 054666 001112 1\$:

MOV #17777, @#120000 ;LOAD DATA INTO TEST LOCATION
MOV #10\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 10\$
MOV #11001, MMRO ;ENABLE TRAPS AND SET TRAP BIT (12)
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV #0, @#100000 ;TRY TO WRITE THRU PAGE 4
MOV #1, MMRO ;CLEAR BITS 9 & 12 OF MMRO
TST @#120000 ;SEE IF WORD WAS CHANGED
BEQ 1\$;BRANCH IF LOCATION IS CLEAR
ERROR 60 ;NO WRITE, BIT09 (MMRO) WAS SET
MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST

7391
7392
7393
7394
7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405

```

:*****
:*TEST 61          NO TRAPPING WHEN REFERENCING A MEMORY MANAGEMENT REG
:*
:*      THIS VERIFIES THE LOGIC THAT PREVENTS A M.M. TRAP WHEN
:*      REFERENCING A M.M. REGISTER.  PAGE 7 IS MAPPED TO THE I/O
:*      PAGE AND ITS A.C.F.= 4 (TRAP ON ALL REFERENCES).  EACH STATUS
:*      REGISTER AND EACH PAR7 AND PDR7 IS READ THRU PAGE 7.  IF ANY
:*      TRAPS OCCUR AN UNEXPECTED M.M. TRAP IS REPORTED.  THEN A MAP
:*      REGISTER (170200) IS REFERENCED AND THE CORRECT TRAP IS VERIFIED
:*      TO INSURE THAT A TRAP CAN OCCUR ON PAGE 7.
:*      THE SIGNAL UNDER TEST IS 'SCCC INT REG B L'.
:*****

```

7406 055000
7407 055000 000004
7408 055002 012737 055230 001316
7409
7410 055010 20\$:
7411 055010 012700 012754
7412 055014 010037 170200
7413 055020 005037 001226
7414 055024 012737 077404 172316
7415 055032 012737 001001 177572
7416 055040 013701 172356
7417 055044 013701 172256
7418 055050 013701 177656
7419 055054 013701 172316
7420 055060 013701 172216
7421 055064 013701 177616
7422 055070 013701 177572
7423 055074 013701 177574
7424 055100 013701 177576
7425 055104 013701 172516
7426 055110 012737 001001 177572
7427 055116 005037 001250
7428 055122 012737 011003 001226
7429 055130 013701 170200
7430 055134 005037 001226
7431 055140 005737 001250
7432 055144 001001
7433 055146 104061
7434 055150 020001 1\$:
7435 055152 001401
7436 055154 104062

```

TST61:
SCOPE
MOV #TST62, NXXTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$:
MOV #12754, R0 ;LOAD DATA PATTERN INTO R0
MOV R0, MAPLO ;LOAD MAP REGISTER 0
CLR MMEXP ;NOT EXPECTING ANY TRAPS
MOV #77404, KIPDR7 ;SET ACF = 4 IN PAGE 7
MOV #1001, MMRO ;ENABLE MEMORY MANAGEMENT TRAPS
MOV KIPAR7, R1 ;READ KERNEL PAR 7
MOV SIPAR7, R1 ;READ SUPERVISOR PAR 7
MOV UIPAR7, R1 ;READ USER PAR 7
MOV KIPDR7, R1 ;READ KERNEL PDR 7
MOV SIPDR7, R1 ;READ SUPERVISOR PDR 7
MOV UIPDR7, R1 ;READ USER PDR 7
MOV MMRO, R1 ;READ MMRO
MOV MMR1, R1 ;READ MMR1
MOV MMR2, R1 ;READ MMR2
MOV MMR3, R1 ;READ MMR3
MOV #1001, MMRO ;MAKE SURE TRAPS ARE ENABLED
CLR PMMRO ;CLEAR M.M. TRAP FLAG
MOV #11003, MMEXP ;EXPECTING M.M. TRAP ON NEXT REFERENCE
MOV MAPLO, R1 ;TRY TO READ MAP REGISTER 0
CLR MMEXP ;NOT EXPECTING ANY M.M. TRAPS
TST PMMRO ;SEE IF TRAP OCCURRED
BNE 1$ ;BRANCH IF TRAP OCCURED
ERROR 61 ;NO TRAP
1$:
CMP R0, R1 ;SEE IF DATA WAS READ CORRECTLY
BEQ 2$ ;BRANCH IF DATA IS RIGHT
ERROR 62 ;INCORRECT READ ON I/O PAGE

```

7437	055156	012737	001000	177572	2\$:	MOV	#BIT9,MMRO	:ENABLE TRAPS, NO RELOCATION
7438	055164	005037	001250			CLR	PMMRO	:CLEAR M.M. TRAPS FLAG
7439	055170	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
7440	055172	013701	170200			MOV	MAPLO,R1	:READ MAP REGISTER 0
7441								:THIS READ SHOULD NOT TRAP SINCE
7442								:THERE IS NO RELOCATION ENABLED.
7443	055176	005737	001250			TST	PMMRO	:SEE IF TRAP OCCURRED
7444	055202	001401				BEQ	3\$:BRANCH IF NO TRAP
7445	055204	104063				ERROR	63	:TRAPPED WHEN NO RELOCATION ENABLED
7446	055206	012737	000001	177572	3\$:	MOV	#BIT0,MMRO	:ENABLE FULL RELOCATION
7447	055214	012737	077406	172316		MOV	#77406,KIPDR7	:SET PDR 7 TO NO TRAPPING ALF
7448	055222	012737	055010	001112		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST

*TEST 62 ONLY ONE VECTOR TAKEN IF TRAP AND ABORT

IF THERE IS A M.M. TRAP CONDITION AND A M.M. ABORT ON THE SAME INSTRUCTION ONLY ONE VECTOR TO 000250 SHOULD BE TAKEN. 'SSRC KT ABORT FLG L' AND 'SSRC ABT FLG (0) H' WILL KNOCK DOWN 'SSRD MEM MGMT TRAP L' SO THAT ONLY THE ABORT VECTOR WILL BE TAKEN.
THIS TEST SETS THE VECTOR TO THE CODE AT 10\$ AND, IF TWO VECTORS ARE TAKEN ERROR 64 IS CALLED. MMRO SHOULD REPORT BOTH THE TRAP AND THE ABORT CONDITIONS WHICH ARE: PAGE LENGTH, KERNEL I-SPACE PAGE 5, AND BIT12 OF MMRO. (051013)

TST62:

7465	055230					SCOPE		
7466	055230	000004				MOV	#TST63,NXTTST	:SAVE STARTING ADDRESS OF NEXT
7467	055232	012737	055426	001316				:TEST FOR ESCAPE ON PARITY ERRORS
7468								:SET LOOP ON ERROR POINTER TO 1\$
7469	055240	012737	055304	001112	20\$:	MOV	#1\$,SLPERR	:SET M.M. VECTOR TO 10\$
7470	055246	012737	055326	000250		MOV	#10\$,MMVEC	:MAP PAGE 4 TO 16K - 20K
7471	055254	012737	001000	172350		MOV	#1000,KIPAR4	:MAP PAGE 5 TO 16K- 20K
7472	055262	012737	001000	172352		MOV	#1000,KIPARS	:SET PAGE LENGTH TO ONE FOR PAGE 5
7473	055270	012737	000006	172312		MOV	#000006,KIPDR5	:SET TRAP ON READ OR WRITE FOP PAGE 4
7474	055276	012737	077404	172310		MOV	#77404,KIPDR4	:MAKE SURE KERN STK PTR IS SETUP IN
7475	055304	012706	001100		1\$:	MOV	#KERSTK,KSP	:CASE YOU LOOP ON ERROR
7476								:ENABLE M.M. TRAPS
7477	055310	012737	001001	177572		MOV	#1001,MMRO	:THIS IS A SYNC POINT FOR SCOPING
7478	055316	000240				NOP		:TRY TO READ THRU PAGE 4 AND
7479	055320	013737	100000	120100		MOV	@#100000,@#120100	:WRITE THRU PAGE 5 (PAGE 4 TRAP &
7480								:PAGE 5 ABORT PAGE LENGTH)
7481								:HAS THE KERNEL STACK ONLY BEEN
7482	055326	020627	001074		10\$:	CMP	KSP,#1074	:PUSHED ONCE BY THE PREVIOUS INSTRUCTION
7483								:BRANCH IF IT HAS BEEN PUSHED
7484	055332	001404				BEQ	12\$:ONLY ONE TIME
7485								:SAVE THE KERNEL STACK POINTER FOR TYPE OUT
7486	055334	010637	001172			MOV	KSP,\$TMP0	:TWO PUSHES WHEN ONLY ONE SHOULD HAPPEN
7487	055340	104064				ERROR	64	:BRANCH TO EXIT TEST
7488	055342	000413				BR	15\$:SAVE MMRO FOR CHECK
7489	055344	013737	177572	001250	12\$:	MOV	MMRO,PMMRO	:MMRO SHOULD HAVE PAGE LENGTH,
7490	055352	012737	051013	001226		MOV	#51013,MMEXP	:TRAP, ENABLE TRAP, PAGE 5, RELOCATING
7491								:SEE IF ABORT CONDITION IS CORRECT
7492	055360	023737	001250	001226		CMP	PMMRO,MMEXP	

```

7493 055366 001401          BEQ      15$          ;BRANCH TO EXIT IF CORRECT
7494 055370 104065          ERROR    65          ;INCORRECT ABORT CONDITION
7495 055372 012716 055400    15$:    MOV      #16$, (KSP) ;CHANGE RETURN ADDRESS TO 16$
7496 055376 000006          RTT           ;RETURN TO 16$ AND CONTINUE PROGRAM
7497 055400 012737 032226 000250 16$:    MOV      #MMTRAP,MMVEC ;RESTORE TRAP HANDLER
7498 055406 012737 000001 177572    MOV      #BIT0,MMRO   ;CLEAR OUT MMRO, BUT LEAVE RELOC ON
7499 055414 005037 001226          CLR      MMEXP       ;NOT EXPECTING ANY M.M. TRAPS
7500 055420 012737 055240 001112    MOV      #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512
7513
7514 055426
7515 055426 000004
7516 055430 012737 055600 001316    SCOPE
7517          MOV      #TST64,NXTTST ;SAVE STARTING ADDRESS OF NEXT
7518 055436 012737 000000 172350 20$:    MOV      #000,KIPAR4 ;TEST FOR ESCAPE ON PARITY ERRORS
7519 055444 012737 077404 172310    MOV      #77404,KIPDR4 ;MAP PAGE 4 TO 0 - 4K
7520 055452 012737 055474 001112    MOV      #1$, $LPERR  ;TRAP ALL REFERENCES
7521 055460 012700 001356          MOV      #ENMMTR,R0  ;SET LOOP ON ERROR POINTER TO 1$
7522          ;PUT ADDRESS OF ENABLE M.M. TRAPS
7523 055464 052700 100000          BIS      #BIT15,R0   ;WORD INTO R0
7524 055470 005037 001226          CLR      MMEXP       ;MAKE ADDRESS IN R0 USE PAGE 4
7525 055474 000240          1$:    NOP           ;NOT EXPECTING ANY TRAPS ON THIS REF.
7526 055476 051037 177572    11$:   BIS      (R0),MMRO  ;THIS IS A SYNC POINT FOR SCOPING
7527          ;SET ENABLE TRAPS BIT IN MMRO USING
7528          ;PAGE THAT COULD CAUSE TRAP IF BIT09
7529 055502 013737 177572 001250    MOV      MMRO,PMRRO  ;WERE ON DURING SOURCE MODE
7530 055510 032737 010000 001250    BIT      #BIT12,PMRRO ;READ MMRO FOR CHECK ON BIT 12
7531 055516 001001          BNE      2$          ;SEE IF BIT12 IS SET BY INST. AT 11$
7532 055520 104066          ERROR    66          ;BRANCH IF IT IS SET
7533 055522 012737 055536 001112 2$:    MOV      #3$, $LPERR ;BIT 12 NOT SET IN MMRO
7534 055530 012737 010003 001226    MOV      #10003,MMEXP ;SET LOOP ON ERROR POINTER TO 3$
7535          ;EXPECTING TRAP, BUT ENABLE TRAPS BIT
7536          ;SHOULD BE CLEAR BEFORE END OF INST.
7537 055536 012737 001001 177572 3$:    MOV      #1001,MMRO  ;THAT CAUSES THE TRAP
7538 055544 005037 001250          CLR      PMRRO      ;ENABLE M.M. TRAPS, FULL RELOCATION
7539 055550 000240          NOP           ;CLEAR M.M. TRAP FLAG
7540 055552 041037 177572          BIC      (R0),MMRO  ;THIS IS A SYNC POINT FOR SCOPING
7541          ;CLEAR ENABLE M.M. TRAP BIT AT END
7542          ;OF INSTRUCTION, TRAP FLIP/FLOP SHOULD
7543 055556 005737 001250          TST      PMRRO      ;BE SET DURING SOURCE MODE FETCH
7544 055562 001001          BNE      4$          ;SEE IF TRAP REALLY OCCURRED ON LAST INS
7545 055564 104067          ERROR    67          ;BRANCH IF TRAP OCCURRED
7546 055566 005037 001226 4$:    CLR      MMEXP       ;NO TRAP, WHEN CLEARING BIT09 (MMRO)
7547 055572 012737 055436 001112    MOV      #20$, $LPERR ;NO M.M. TRAPS EXPECTED
7548          ;SET LOOP POINTER TO START OF TEST

```

```

:*****
:*TEST 63          PROPER TIMING OF MEMORY MANAGEMENT TRAPS
:*
:* IF THE INSTRUCTION SETTING BIT09 OF MMRO SATISFIES A M.M. TRAP
:* CONDITION, NO TRAP SHOULD OCCUR SINCE BIT09 WILL NOT BE SET
:* WHEN THE TRAP CONDITION IS SATISFIED.
:* THE SECOND HALF OF THIS TEST VERIFIES THAT IF THE M.M. TRAP
:* CONDITION IS MET DURING THE INSTRUCTION WHICH CLEARS BIT 09
:* OF MMRO THE TRAP WILL OCCUR ANYWAY.
:*
:*****

```

TST63:

7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603
7604

*TEST 64 ABORT ON ILLEGAL MODE

* IF THE MODE SET IN BITS <15:14> OF THE PROCESSOR STATUS IS
* <10> THE MODE IS ILLEGAL AND THE NEXT INSTRUCTION FETCH WILL
* SELECT NO PAR/PDR PAIR TO CONTROL THE REFERENCE. THE PDR
* LINES WILL ALL BE READ AS ONES. THE A.C.F. = 7 (NON-RESIDENT),
* THE EXPANSION DIRECTION = DOWN, THE P.L.F. = 177 OR 1 BLOCK.
* THE M.M. ABORT WILL BE NON-RESIDENT, PAGE LENGTH (IF THE
* VIRTUAL ADDRESS HAS A BLOCK NUMBER OF 176 OR LESS), MODE <10>,
* PAGE 2 (SINCE THE CODE IS ON PAGE 2).

TST64:

```
SCOPE
MOV #TST65,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$: MOV #10$,MMVEC ;SET M.M. VECTOR TO 10$
MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
1$: NOP ;THIS IS A SYNC POINT FOR SCOPING
BIS #BIT15,PSW ;SET ILLEGAL MODE IN PROCESSOR STATUS
ERROR 71 ;INSTRUCTION FETCH DIDN'T ABORT
;THIS INSTRUCTION FETCH SHOULD ABORT,
;NON-RESIDENT & PAGE LENGTH FAULT,
;ILLEGAL MODE, PAGE 1.

10$: MOV MMRO,PMRO ;READ MMRO FOR COMPARE
MOV #16$, (KSP) ;CHANGE RETURN ADDRESS TO 16$
BIC #BIT15,2(KSP) ;CLEAR ILLEGAL MODE BIT IN PSW ON STACK
RTT ;RETURN TO 16$ AND CONTINUE PROGRAM
16$: MOV #140105,R1 ;LOAD EXPECTED ABORT CONDITION IN R1:
;NON-RESIDENT, PAGE FAULT, MODE=<10>,
;PAGE 2.
CMP R1,PMRO ;DID YOU GET THE EXPECTED CONDITION
BEQ 11$ ;BRANCH IF CONDITION IS CORRECT
ERROR 72 ;WRONG ERROR CONDITION
11$: SPL 7 ;MAKE THE PRIORITY LEVEL 7
BIC #177776,MMRO ;CLEAR ALL ERROR CONDITIONS
MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
```

*TEST 65 MEMORY MANAGEMENT REGISTERS ONLY CLOCKED ONCE IF MMRO NOT CLEARED

* AS LONG AS 'SSRC NO ERROR (1) H' IS NOT ASSERTED (THAT IS AFTER
* AN ABORT AND UNTIL MMRO BITS <15:13> ARE CLEARED) MMRO, MMR1,
* AND MMR2 SHOULD NOT BE CLOCKED. THIS TEST CAUSES A NON-RESIDENT
* ABORT, SAVES THE STATUS REGISTERS, CHANGES THE VECTOR TO 10\$,
* AND THEN CAUSES A PAGE LENGTH ABORT. AT THE SECOND ABORT THE
* STATUS REGISTERS ARE COMPARED WITH THEIR FIRST CONDITIONS. IF ANY
* OF THEM CHANGE, THE OLD AND THE NEW CONDITIONS WILL BE REPORTED.

7605	055712				TST65:				
7606	055712	000004				SCOPE			
7607	055714	012737	056154	001316		MOV	#TST66,NXTTST		;SAVE STARTING ADDRESS OF NEXT
7608									;TEST FOR ESCAPE ON PARITY ERRORS
7609	055722	012737	000000	172310	20\$:	MOV	#000000,KIPDR4		;MAP PAGE 4 NON-RESIDENT, AND PAGE
7610									;LENGTH OF 1 BLOCK
7611	055730	012737	055772	000250		MOV	#5\$,MMVEC		;SET M.M. TRAP VECTOR TO 5\$
7612	055736	012737	000340	000252		MOV	#340,MMVEC+2		;SET PRIORITY TO 7 GOTO KERNEL MODE
7613	055744	012737	055752	001112		MOV	#1\$, \$LPERR		;SET LOOP ON ERROR POINTER TO 1\$
7614	055752	013700	100000		1\$:	MOV	@#100000,R0		;TRY TO READ THRU PAGE 4
7615									;THIS PAGE NON-RESIDENT SHOULD CAUSE

F 13

7616
7617 055756 012737 056022 000250 2\$:
7618 055764 000240
7619 055766 013700 100100
7620
7621
7622
7623

MOV #10\$,MMVEC
NOP
MOV @#100100,RC

;ABORT AND TRAP TO 5\$.
;SET M.M. TRAP VECTOR TO 10\$
;THIS IS A SYNC POINT FOR SCOPING
;TRY TO READ FROM BLOCK 2 OF PAGE 4
;THIS SHOULD ABORT AGAIN BUT NONE
;OF THE MEMORY MANAGEMENT STATUS
;REGISTERS SHOULD BE CLOCKED THIS TIME.

PDP-11/70-74MP MEMORY MANAGEMENT DIAGNOSTIC
CEKBED.P1 15-AUG-79 10:06 T65

G 13
MACY11 30A(1052) 05-SEP-79 15:36 PAGE 148
MEMORY MANAGEMENT REGISTERS ONLY CLOCKED ONCE IF MMRO NOT CLEARED

SEQ 0162

7624

P
C

7625	055772	013737	177572	001250	5\$:	MOV	MMR0,PMMR0	:READ MEMORY MANAGEMENT REGISTER 0
7626	056000	013737	177574	001252		MOV	MMR1,PMMR1	:READ MEMORY MANAGEMENT REGISTER 1
7627	056006	013737	177576	001254		MOV	MMR2,PMMR2	:READ MEMORY MANAGEMENT REGISTER 2
7628	056014	012716	055756			MOV	#2\$,(KSP)	:CHANGE RETURN ADDRESS TO 2\$
7629	056020	000006				RTT		:GO BACK TO 2\$ AND CAUSE PAGE FAULT
7630								
7631								
7632	056022	012716	056030		10\$:	MOV	#16\$,(KSP)	:CHANGE RETURN ADDRESS TO 16\$
7633	056026	000006				RTT		:RETURN TO 16\$ AND CONTINUE PROGRAM
7634	056030	005037	001200		16\$:	CLR	\$TMP3	:ERROR COUNTER, 3 POSSIBLE CMP FAILURES

7635	056034	013737	177572	001172		MOV	MMR0,\$TMP0	:READ MEMORY MANAGEMENT REGISTER 0
7636	056042	013737	177574	001174		MOV	MMR1,\$TMP1	:READ MEMORY MANAGEMENT REGISTER 1
7637	056050	013737	177576	001176		MOV	MMR2,\$TMP2	:READ MEMORY MANAGEMENT REGISTER 2
7638	056056	023737	001176	001254		CMP	\$TMP2,MMR2	:SEE IF MMR2 CHANGED
7639	056064	001402				BEQ	11\$:BRANCH IF MMR2 DIDN'T CHANGE
7640	056066	005237	001200			INC	\$TMP3	:ONE COMPARE FAILURE
7641	056072	023737	001174	001252	11\$:	CMP	\$TMP1,MMR1	:SEE IF MMR1 CHANGED
7642	056100	001402				BEQ	12\$:BRANCH IF MMR1 DIDN'T CHANGE
7643	056102	005237	001200			INC	\$TMP3	:ANOTHER COMPARE FAILURE
7644	056106	023737	001172	001250	12\$:	CMP	\$TMP0,MMR0	:SEE IF MMRO CHANGED
7645	056114	001402				BEQ	13\$:BRANCH IF MMRO DIDN'T CHANGE
7646	056116	005237	001200			INC	\$TMP3	:ANOTHER COMPARE FAILURE
7647	056122	005737	001200		13\$:	TST	\$TMP3	:WERE THERE ANY ERRORS ON THIS TEST
7648	056126	001401				BEQ	19\$:BRANCH IF NO ERRORS
7649	056130	104073				ERROR	73	:AT LEAST ONE M.M. REG CHANGED
7650	056132	042737	177776	177572	19\$:	BIC	#177776,MMR0	:CLEAR ALL ERROR BITS IN MMRO
7651	056140	012737	032226	000250		MOV	#MMTRAP,MMVEC	:PUT BACK REGULAR M.M. TRAP ROUTINE
7652	056146	012737	055722	001112		MOV	#20\$,SLPERK	:SET LOOP POINTER TO START OF TEST

7653
7654
7655
7656
7657
7658
7659
7660
7661
7662
7663
7664
7665
7666

```
*****  
:TEST 66 SUPERVISOR MODE, ABORT VECTOR FROM KERNEL SPACE  
:  
: THIS TEST DOES AN ABORT FROM SUPERVISOR MODE. THE VECTOR  
: SHOULD BE PICKED UP FROM KERNEL I-SPACE DUE TO 'ROM OUT06'  
: FORCING KERNEL MODE ON 'SSRB' DURING THE ABORT SEQUENCE.  
:  
: THE 'HALTS' IN THIS TEST ARE SPACE FILLERS AND SHOULD NEVER BE  
: REACHED, IF SUPERVISOR MODE IS ENABLED PROPERLY ON 'SSRB',  
: 'SAPE', AND 'SAPB'.  
:*****
```

7667
7668
7669
7670
7671
7672
7673
7674
7675
7676
7677
7678
7679
7680
7681
7682
7683
7684
7685
7686
7687
7688
7689
7690

```
TST66:  
SCOPE  
MOV #TST67,NXTTST :SAVE STARTING ADDRESS OF NEXT  
:TEST FOR ESCAPE ON PARITY ERRORS  
TBITO :MAKE SURE T-BIT IS OFF FOR THIS TEST  
:AND THE NEXT THREE (3) TESTS.  
20$: CLR MMEXP :NOT EXPECTING ANY M.M. TRAPS YET  
MOV #77400,UIPDR1 :LOAD USER PAGE 1 NON-RESIDENT  
MOV #77400,UIPDR2 :LOAD USER PAGE 2 NON-RESIDENT  
MOV #77400,UIPDR3 :LOAD USER PAGE 3 NON-RESIDENT  
MOV #77400,UIPDR7 :LOAD USER PAGE 7 NON-RESIDENT  
MOV #77406,R0 :PAGE LENGTH=200 BLOCKS EXPAND UP  
:RESIDENT READ/WRITE  
MOV R0,UIPDR0 :LOAD USER PAGE 0  
MOV R0,SIPDR0 :LOAD SUPERVISOR PAGE 0  
MOV R0,SIPDR1 :LOAD SUPERVISOR PAGE 1  
MOV R0,SIPDR2 :LOAD SUPERVISOR PAGE 2  
MOV R0,SIPDR3 :LOAD SUPERVISOR PAGE 3  
MOV R0,SIPDR7 :LOAD SUPERVISOR PAGE 7  
MOV #2,UIPAR0 :MAP USER PAGE 0 TO 00200  
MOV #177600,UIPAR7 :MAP USER PAGE 7 TO I/O PAGE  
MOV #1,SIPAR0 :MAP SUPERVISOR PAGE 0 TO 000100  
MOV #201,SIPAR1 :MAP SUPERVISOR PAGE 1 TO 020100  
MOV #401,SIPAR2 :MAP SUPERVISOR PAGE 2 TO 040100
```

```

7691 056314 012737 000601 172246 MOV #601,SIPAR3 ;MAP SUPERVISOR PAGE 3 TO 060100
7692 056322 012737 177600 172256 MOV #177600,SIPAR7 ;MAP SUPERVISOR PAGE 7 TO I/O PAGE
7693 056330 012737 077403 172210 MOV #77403,SIPDR4 ;MAKE SUPERVISOR PAGE 4 NON-RESIDENT
7694 056336 012737 001000 172250 MOV #1000,SIPAR4 ;MAP SUPERVISOR PAGE 4 TO 16K
7695 056344 012737 032506 000350 MOV #SUPVEC,350 ;SUPERVISOR SPACE VECTOR
7696 056352 012737 000140 000352 MOV #140,352 ;SUPERVISOR SPACE PSW = 140
7697 056360 012737 032530 000450 MOV #USEVEC,450 ;USER SPACE VECTOR
7698 056366 012737 000000 000452 MOV #000,452 ;USER SPACE PSW = 000
7699 056374 012737 056402 001112 MOV #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
7700 056402 012737 040000 177776 5$: MOV #40000,PSW ;GO TO SUPERVISOR MODE.
7701 ;THE NEXT INSTRUCTION EXECUTED IS AT
7702 ;3$. THE ADDRESS IS 100 OCTAL BYTES
7703 ;GREATER THAN THE ADDRESS AT 1$.
7704 056410 104074 1$: ERROR 74 ;DIDN'T GO TO SUPERVISOR MODE
7705 056412 005037 177776 CLR PSW ;GO BACK INTO KERNEL MODE
7706 056416 000137 056460 JMP 2$ ;GO TO EXIT OF TEST
7707 056422 000000 HALT ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
7708 056424 000000 HALT ;EVER BE REACHED
7709 056426 000000 HALT ;EVER BE REACHED
7710 056430 000000 HALT ;EVER BE REACHED
7711 056432 000000 HALT ;EVER BE REACHED
7712 056434 000000 HALT ;EVER BE REACHED
7713 056436 000000 HALT ;EVER BE REACHED
7714 056440 000000 HALT ;EVER BE REACHED
7715 056442 000000 HALT ;EVER BE REACHED
7716 056444 000000 HALT ;EVER BE REACHED
7717 056446 000000 HALT ;EVER BE REACHED
7718 056450 000000 HALT ;EVER BE REACHED
7719 056452 000000 HALT ;EVER BE REACHED
7720 056454 000000 HALT ;EVER BE REACHED
7721 056456 000000 HALT ;EVER BE REACHED
7722 056460 012737 032226 000250 2$: MOV #MMTRAP,MMVEC ;RESTORE NORMAL M.M. TRAP ROUTINE
7723 056466 012737 056166 001112 MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
7724 056474 000431 BR TST67 ;BRANCH TO NEXT TEST
7725 056476 000000 HALT ;THE NEXT 'SEVERAL' HALTS SHOULDN'T
7726 056500 000000 HALT ;EVER BE REACHED
7727 056502 000000 HALT ;EVER BE REACHED
7728 056504 000000 HALT ;EVER BE REACHED
7729 056506 000000 HALT ;EVER BE REACHED
7730 056510 012737 032456 000150 3$: MOV #KERVEC,<MMVEC-100> ;SET UP KERNEL SPACE VECTOR
7731 056516 012737 000340 000152 MOV #340,<MMVEC+2-100> ;KERNEL SPACE PSW = 340
7732 056524 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7733 056526 013700 100000 MOV @#100000,R0 ;READ FROM PAGE 4, SUPERVISOR
7734 056532 022737 100051 001150 CMP #100051,<MMR0-100> ;EXPECTING NON-RESIDENT PAGE 4
7735 ;ABORT IN SUPERVISOR MODE
7736 056540 001401 BEQ 10$ ;BRANCH IF CORRECT COND
7737 056542 104075 ERROR 75 ;ABORT CONDITION INCORRECT
7738 056544 042737 177776 177572 10$: BIC #177776,MMR0 ;CLEAR MMR0 FOR NEXT ABORT
7739 056552 012737 000340 177776 MOV #340,PSW ;GO BACK INTO KERNEL MODE NOW
7740 ;THE NEXT INSTRUCTION TO BE EXECUTED
7741 ;IS AT LABEL 2$
7742
7743
7744
7745
7746

```

: *TEST 67 M.M. ABORT DURING AN ODD ADDRESS ABORT SEQUENCE

```

7747 : *
7748 : * THIS TEST VERIFIES BIT07 OF MMRO (INSTRUCTION COMPLETE) AND
7749 : * 'SSRA PS RESTORE (1) H'.
7750 : * THE TEST CAUSES AN 'ODD ADDRESS' ABORT THRU 'LRRVEC' WHICH
7751 : * PUTS THE PROCESSOR INTO SUPERVISOR MODE. THE SUPERVISOR
7752 : * STACK POINTER IS AT 1104 AND ITS PAGE 0 IS 11 BLOCKS LONG,
7753 : * SO WHEN THE OLD PS AND PC ARE PUSHED ON THE STACK A M.M.
7754 : * PAGE LENGTH ABORT OCCURS. THE OLD PS SHOULD BE RESTORED SO
7755 : * THAT THE PROPER RECOVERY CAN BE MADE, AND THE M.M. ABORT
7756 : * HAPPENS.
7757 : * MMRO HAS PAGE LENGTH FAULT, SUPERVISOR I-SPACE, PAGE 0
7758 : * MMR1 HAS R6 DECREMENTED BY 2 TWICE
7759 : * MMR2 HAS 000004 (ADDRESS OF 'ERRVEC' WHERE M.M. ABORT OCCURRED)
7760 : *

```

```

7761 : *****
7762 056560 TST67:
7763 056560 000004 SCOPE
7764 056562 012737 057144 001316 MOV #TST70,NXTTST ;SAVE STARTING ADDRESS OF NEXT
7765 : TEST FOR ESCAPE ON PARITY ERRORS
7766 .EQUIV BIT4,TBIT ;BIT 4 OF P.S. IS T-BIT TRAPPING BIT
7767 056570 012737 040340 000006 20$: MOV #40340,ERRVEC+2 ;SET PRIORITY OF ERRVEC TO 7
7768 : AND MAKE NEW MODE SUPERVISOR
7769 056576 012737 000000 172240 MOV #000,SIPAR0 ;MAP SUPERVISOR PAGE 0 TO PHYS. 0
7770 056604 012737 000200 172242 MOV #200,SIPAR1 ;MAP SUPERVISOR PAGE 1 TO 4K -8K
7771 056612 012737 000400 172244 MOV #400,SIPAR2 ;MAP SUPERVISOR PAGE 2 TO 8K - 12K
7772 056620 012737 000600 172246 MOV #600,SIPAR3 ;MAP SUPERVISOR PAGE 3 TO 12K - 16K
7773 056626 012737 177600 172256 MOV #177600,SIPAR7 ;MAP SUPERVISOR PAGE 7 TO I/O PAGE
7774 056634 012737 077406 172202 MOV #77406,SIPDR1 ;MAKE SUPER PAGE 1 200 BLCKS, R/W
7775 056642 012737 077406 172204 MOV #77406,SIPDR2 ;MAKE SUPER PAGE 2 200 BLCKS, R/W
7776 056650 012737 077406 172206 MOV #77406,SIPDR3 ;MAKE SUPER PAGE 3 200 BLCKS, R/W
7777 056656 012737 077406 172216 MOV #77406,SIPDR7 ;MAKE SUPER PAGE 7 200 BLOCKS, R/W
7778 056664 012737 056706 001112 MOV #1$, $LPERR ;SET LOOP ON ERROR POINTER TO 1$
7779 056672 012737 056740 000250 MOV #10$,MMVEC ;SET M.M. VECTOR TO 10$
7780 056700 012737 004006 172200 MOV #04006,SIPDR0 ;SUPERVISOR PAGE 0 EXPANDS UPWARD
7781 : AND IS 11 BLOCKS LONG.
7782 : ANY ADDRESS ABOVE 1076 WILL CAUSE A
7783 : PAGE LENGTH FAULT. THIS MEANS THAT
7784 : WHEN THE ODD ADDRESS TRAP TRIES TO
7785 : PUSH THE OLD PS ON THE SUPERVISOR STACK
7786 : IT WILL GET A MEMORY MANAGEMENT ABORT.
7787 056706 052737 040000 177776 1$: BIS #BIT14,PSW ;GO TO SUPERVISOR MODE TO SET STK PTR.
7788 056714 012706 001104 MOV #1104,SSP ;SET THE STACK POINTER OUT OF LIMITS
7789 056720 042737 040000 177776 BIC #BIT14,PSW ;GO BACK TO KERNEL MODE AND SET UP
7790 : FOR ODD ADDRESS INSTRUCTION
7791 056726 000230 SPL 0 ;SET PRIORITY LEVEL TO 0
7792 056730 000277 SCC ;SET ALL CONDITION CODES
7793 056732 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
7794 056734 013701 000001 MOV @#000001,R1 ;TRY TO READ ADDRESS 1 INTO R1
7795 : THIS WILL ABORT TO 4, AND THE PS
7796 : AT ADDRESS 6 WILL FORCE SUPERVISOR.
7797 : THEN WHILE PUSHING THE PS ON THE STACK
7798 : YOU SHOULD GET A MEMORY MANAGEMENT
7799 : ABORT AND GO TO 10$. THE PS SHOULD
7800 : HAVE BEEN RESTORED AND WILL NOW BE ON
7801 : THE KERNEL STACK.
7802

```

7803	056740	016601	000002		10\$:	MOV	2(KSP),R1	:COPY PS ON STACK INTO R1
7804	056744	012716	056752			MOV	#16\$, (KSP)	:SET RETURN PC TO 16\$
7805	056750	000006				RTT		:RETURN TO 16\$ WITH T-BIT INTACT
7806	056752	042701	000020		16\$:	BIC	#TBIT,R1	:CLEAR T BIT IN R1 IF ON THIS PASS
7807	056756	005037	001200			CLR	\$TMP3	:THIS IS THE ERROR COUNTER FLAG
7808	056762	122701	000017			CMPB	#17,R1	:THE PROCESSOR STATUS THAT SHOULD
7809								:BE ON THE STACK IS THE ONE WITH ALL
7810								:OF THE CONDITION CODES SET.
7811	056766	001402				BEQ	11\$:BRANCH IF PS IS CORRECT
7812	056770	005237	001200			INC	\$TMP3	:WRONG PS IS ON STACK
7813	056774	013737	177572	001250	11\$:	MOV	MMR0,PMR0	:SAVE MMR0 FOR COMPARE
7814	057002	013737	177574	001252		MOV	MMR1,PMR1	:SAVE MMR1, IT SHOULD BE 173366
7815	057010	013737	177576	001254		MOV	MMR2,PMR2	:SAVE MMR2, IT SHOULD EQUAL #10
7816	057016	022737	040241	001250		CMP	#040241,PMR0	:SHOULD HAVE PAGE LENGTH FAULT,
7817								:COMPLETED, PAGE 0, SUPERVISOR I-SPACE.
7818	057024	001402				BEQ	12\$:BRANCH IF CORRECT CONDITION
7819	057026	005237	001200			INC	\$TMP3	:WRONG ABORT CONDITION
7820	057032	022737	173366	001252	12\$:	CMP	#173366,PMR1	:R6 DECREMENTED TWICE, DURING ABORTED
7821								:PUSHES TO SUPERVISOR STACK
7822	057040	001402				BEQ	13\$:BRANCH IF MMR1 IS CORRECT
7823	057042	005237	001200			INC	\$TMP3	:MMR1 IS NOT CORRECT
7824	057046	022737	000004	001254	13\$:	CMP	#4,PMR2	:SEE IF MMR2 EQUALS ADDR.000004
7825	057054	001402				BEQ	14\$:BRANCH IF ADDRESS IS CORRECT
7826	057056	005237	001200			INC	\$TMP3	:MMR2 HAS THE WRONG ADDRESS
7827	057062	005737	001200		14\$:	TST	\$TMP3	:DID ANY OF THE COMPARES FAIL?
7828	057066	001401				BEQ	15\$:BRANCH IF ALL COMPARES SUCCEEDED
7829	057070	104070				ERROR	70	:AT LEAST ONE COMPARE FAILED
7830	057072	000237			15\$:	SPL	7	:NORMAL PRIORITY IS 7
7831	057074	052737	040000	177776		BIS	#BIT14,PSW	:GO TO SUPERVISOR MODE TO RESET PTR
7832	057102	012706	000700			MOV	#700,SSP	:RESTORE SUPER STK PTR TO 700
7833	057106	042737	040000	177776		BIC	#BIT14,PSW	:RETURN TO KERNEL MODE
7834	057114	012737	000340	000006		MOV	#340,ERRVEC+2	:RESTORE CORRECT PS TO ERROR VECTOR
7835	057122	012737	056570	001112		MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
7836	057130	042737	177776	177572		BIC	#177776,MMR0	:CLEAR ERROR CONDITION IN MMR0
7837	057136	012737	032226	000250		MOV	#MMTRAP,MMVEC	:RESTORE NORMAL M.M. VECTOR

```

7838
7839
7840 :*****
7841 :*TEST 70      USER MODE, ABORT VECTOR FROM KERNEL SPACE
7842 :*
7843 :*      THIS TEST DOES AN ABORT FROM USER MODE.  THE VECTOR
7844 :*      SHOULD BE PICKED UP FROM KERNEL I-SPACE DUE TO 'ROM OUT06'
7845 :*      FORCING KERNEL MODE ON 'SSRB' DURING THE ABORT SEQUENCE.
7846 :*
7847 :*      THE 'HALTS' IN THIS TEST ARE SPACE FILLERS AND SHOULD NEVER BE
7848 :*      REACHED, IF USER MODE IS ENABLED PROPERLY ON 'SSRB', 'SAPE',
7849 :*      'SAPB', 'SAPC', AND 'SAPF'.
7850 :*
7851 :*      IT SHOULD BE NOTED THAT IF THIS TEST CODE IS EXECUTED IN SINGLE
7852 :*      INSTRUCTION, THE ABORT WILL PUSH THE PS & PC ONTO THE SUPERVISOR
7853 :*      STACK INSTEAD OF THE KERNEL STACK.  THIS IS DUE TO A FLAW IN
7854 :*      THE CPU ROM AND IS A CARRY OVER FROM THE PDP-11/45.  IF YOU NEED
7855 :*      TO SINGLE INSTRUCTION THIS TEST, SINGLE BUS CYCLE THRU THE ABORT
7856 :*      SEQUENCE FOR PROPER OPERATION OF THE STACKS.
7857 :*
7858 :*****

```



```
7859 057144          TST70:
7860 057144 000004          SCOPE
7861 057146 012737 057546 001316  MOV      #TST71,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
7862                                     ;TEST FOR ESCAPE ON PARITY ERRORS
7863 057154 005037 001226 20$: CLR      MMEXP      ;NOT EXPECTING ANY M.M. TRAPS YET
7864 057160 012737 077400 172202  MOV      #77400,SIPDR1 ;LOAD SUPERVISOR PAGE 1 NON-RESIDENT
7865 057166 012737 077400 172204  MOV      #77400,SIPDR2 ;LOAD SUPERVISOR PAGE 2 NON-RESIDENT
7866 057174 012737 077400 172206  MOV      #77400,SIPDR3 ;LOAD SUPERVISOR PAGE 3 NON-RESIDENT
7867 057202 012737 077400 172216  MOV      #77400,SIPDR7 ;LOAD SUPERVISOR PAGE 7 NON-RESIDENT
7868 057210 012700 077406  MOV      #77406,R0     ;PAGE LENGTH-200 BLOCKS EXPAND UP
7869                                     ;RESIDENT READ/WRITE
7870 057214 010037 172200  MOV      R0,SIPDR0    ;LOAD SUPERVISOR PAGE 0
7871 057220 010037 177600  MOV      R0,UIPDR0   ;LOAD USER PAGE 0
7872 057224 010037 177602  MOV      R0,UIPDR1   ;LOAD USER PAGE 1
7873 057230 010037 177604  MOV      R0,UIPDR2   ;LOAD USER PAGE 2
7874 057234 010037 177606  MOV      R0,UIPDR3   ;LOAD USER PAGE 3
7875 057240 010037 177616  MOV      R0,UIPDR7   ;LOAD USER PAGE 7
7876 057244 012737 000002 172240  MOV      #2,SIPAR0    ;MAP SUPERVISOR PAGE 0 TO 00200
7877 057252 012737 177600 172256  MOV      #177600,SIPAR7 ;MAP SUPERVISOR PAGE 7 TO I/O PAGE
7878 057260 012737 000001 177640  MOV      #1,UIPAR0   ;MAP USER PAGE 0 TO 00100
7879 057266 012737 000201 177642  MOV      #201,UIPAR1 ;MAP USER PAGE 1 TO 20100
7880 057274 012737 000401 177644  MOV      #401,UIPAR2 ;MAP USER PAGE 2 TO 40100
7881 057302 012737 000601 177646  MOV      #601,UIPAR3 ;MAP USER PAGE 3 TO 60100
7882 057310 012737 177600 177656  MOV      #177600,UIPAR7 ;MAP USER PAGE 7 TO I/O PAGE
7883 057316 012737 077400 177612  MOV      #77400,UIPDR5 ;MAKE USER PAGE 5 NON-RESIDENT
7884 057324 012737 001000 177652  MOV      #1000,UIPAR5 ;MAP USER PAGE 5 TO 16K
7885 057332 012737 032506 000450  MOV      #SUPVEC,450  ;SUPERVISOR SPACE VECTOR
7886 057340 012737 000140 000452  MOV      #140,452    ;SUPERVISOR SPACE PSW = 140
7887 057346 012737 032530 000350  MOV      #USEVEC,350 ;USER SPACE VECTOR
7888 057354 012737 000000 000352  MOV      #000,352    ;USER SPACE PSW = 000
7889 057362 012737 057370 001112  MOV      #5$, $LPERR ;SET LOOP ON ERROR POINTER TO 5$
7890 057370 012737 140000 177776 5$: MOV      #140000,PSW ;GO TO USER MODE.
7891                                     ;THE NEXT INSTRUCTION EXECUTED IS AT
7892                                     ;3$. THE ADDRESS IS 100 OCTAL BYTES
7893                                     ;GREATER THAN THE ADDRESS AT 1$.
7894 057376 104076          1$: ERROR    76
7895 057400 005037 177776  CLR      PSW
7896 057404 000137 057446  JMP      2$
7897 057410 000000          HALT
7898 057412 000000          HALT
7899 057414 000000          HALT
7900 057416 000000          HALT
7901 057420 000000          HALT
7902 057422 000000          HALT
7903 057424 000000          HALT
7904 057426 000000          HALT
7905 057430 000000          HALT
7906 057432 000000          HALT
7907 057434 000000          HALT
7908 057436 000000          HALT
7909 057440 000000          HALT
7910 057442 000000          HALT
7911 057444 000000          HALT
7912 057446 012737 032226 000250 2$: MOV      #MMTRAP,MMVEC ;RESTORE NORMAL M.M. TRAP ROUTINE
7913 057454 012737 057154 001112  MOV      #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
7914 057462 000431          BR       TST71     ;BRANCH TO NEXT TEST
```

7915	057464	000000				HALT		;THE NEXT 'SEVERAL' HALTS SHOULDN'T
7916	057466	000000				HALT		;EVER BE REACHED
7917	057470	000000				HALT		;EVER BE REACHED
7918	057472	000000				HALT		;EVER BE REACHED
7919	057474	000000				HALT		;EVER BE REACHED
7920	057476	012737	032456	000150	3\$:	MOV	#KERVEC,<MMVEC-100>	;SET UP KERNEL SPACE VECTOR
7921	057504	012737	000340	000152		MOV	#340,<MMVEC+2-100>	;KERNEL SPACE PSW = 340
7922	057512	000240				NOP		;THIS IS A SYNC POINT FOR SCOPING
7923	057514	013700	120000			MOV	@#120000,R0	;READ FROM PAGE 5, USER SPACE
7924	057520	022737	100153	001150		CMP	#100153,<PMMR0-100>	;EXPECTING NON-RESIDENT PAGE 5
7925								;ABORT IN USER MODE I-SPACE
7926	057526	001401				BEQ	10\$;BRANCH IF CORRECT COND
7927	057530	104077				ERROR	77	;ABORT CONDITION INCORRECT
7928	057532	042737	177776	177572	10\$:	BIC	#177776,MMR0	;CLEAR MMR0 FOR NEXT TEST
7929	057540	012737	000340	177776		MOV	#340,PSW	;GO BACK INTO KERNEL MODE NOW
7930								;THE NEXT INSTRUCTION TO BE EXECUTED
7931								;IS AT LABEL 2\$
7932								
7933								
7934								

*TEST 71 COUNT PATTERN THRU MMR2, TO TEST ALL BITS

*
* THIS TEST SETS UP ALL USER I-SPACE PAGES TO BE NON-RESIDENT
* AND THEN TRIES ALL POSSIBLE VIRTUAL ADDRESSES AS A PROCESSOR
* COUNTER IN USER MODE. EVERY INSTRUCTION FETCH WILL ABORT,
* NON RESIDENT AND THE CONTENTS OF MMR2 IS TESTED ALONG
* WITH BITS <06:01> OF MMR0.
*

TST71:

7944	057546					SCOPE		
7945	057546	000004				MOV	#TST72,NXTTST	;SAVE STARTING ADDRESS OF NEXT
7946	057550	012737	060012	001316				;TEST FOR ESCAPE ON PARITY ERRORS
7947								;DO 2 ITERATIONS
7948	057556	012737	000002	001206	20\$:	MOV	#2,\$TIMES	
7949	057564	012737	077400	177600		MOV	#77400,UIPDR0	;MAP USER PAGE 0 NON-RESIDENT
7950	057572	012737	077400	177602		MOV	#77400,UIPDR1	;MAP USER PAGE 1 NON-RESIDENT
7951	057600	012737	077400	177604		MOV	#77400,UIPDR2	;MAP USER PAGE 2 NON-RESIDENT
7952	057606	012737	077400	177606		MOV	#77400,UIPDR3	;MAP USER PAGE 3 NON-RESIDENT
7953	057614	012737	077400	177610		MOV	#77400,UIPDR4	;MAP USER PAGE 4 NON-RESIDENT
7954	057622	012737	077400	177612		MOV	#77400,UIPDR5	;MAP USER PAGE 5 NON-RESIDENT
7955	057630	012737	077400	177614		MOV	#77400,UIPDR6	;MAP USER PAGE 6 NON-RESIDENT
7956	057636	012737	077400	177616		MOV	#77400,UIPDR7	;MAP USER PAGE 7 NON-RESIDENT
7957	057644	012737	057674	001112		MOV	#2\$,SLPERR	;SET LOOP ON ERROR POINTER TO 2\$
7958	057652	012700	140000			MOV	#140000,R0	;THIS WILL FORCE USER MODE WHEN USED
7959								;AS A PROCESSOR STATUS
7960	057656	005001				CLR	R1	;R1 HOLDS USER VIRTUAL PC
7961	057660	012737	057704	000250		MOV	#10\$,MMVEC	;SET M.M. TRAP VECTOR TO 10\$
7962	057666	012737	000340	000252		MOV	#340,MMVEC+2	;MAKE SURE TRAP TAKES YOU TO KERNEL
7963	057674	010046			2\$:	MOV	R0,-(KSP)	;PUSH USER PS ON STACK
7964	057676	010146				MOV	R1,-(KSP)	;PUSH USER'S VIRTUAL PC ON STACK
7965	057700	000240				NOP		;THIS IS A SYNC POINT FOR SCOPING
7966	057702	000002				RTI		;RETURN TO USER MODE
7967								;THE FIRST INSTRUCTION FETCH WILL
7968								;CAUSE A NON-RESIDENT ABORT AND LOCK
7969								;MMR2 SO THAT ALL BITS CAN BE CHECKED.
7970								

```

7971
7972
7973 057704 062706 000004 10$: ADD #4,KSP ;CLEAN UP STACK FOR NEXT TIME
7974 057710 013737 177576 001254 MOV MMR2,PMR2 ;READ MMR2 TO TEMP LOCATION
7975 057716 013737 177572 001250 MOV MMR0,PMR0 ;READ MMR0 TO TEMP LOCATION
7976 057724 020137 001254 CMP R1,PMR2 ;SEE IF MMR2 LOCKED CORRECT V. A.
7977 057730 001401 BEQ 11$ ;BRANCH IF MMR2 HAS RIGHT V.A.
7978 057732 104100 ERROR 100 ;WRONG V.A.
7979 057734 005002 11$: CLR R2 ;R2 WILL GET THE VIRTUAL PAGE NO.
7980 057736 010103 MOV R1,R3 ;COPY VIRTUAL ADDRESS INTO R3
7981 057740 073227 000003 ASHC #3,R2 ;COMBINED LEFT SHIFT <R2,R3> 3 BITS
7982 057744 006102 ROL R2 ;ADJUST PAGE NUMBER
7983 057746 052702 100141 BIS #100141,R2 ;SET OTHER EXPECTED BITS IN MMR0
7984 057752 020237 001250 CMP R2,PMR0 ;SEE IF MMR0 RECORDED COR CT PAGE NO.
7985 057756 001401 BEQ 12$ ;BRANCH IF PAGE NUMBER WAS CORRECT
7986 057760 104101 ERROR 101 ;WRONG PAGE NO. IN MMR0
7987 057762 042737 177776 177572 12$: BIC #177776,MMR0 ;CLEAR ALL ERROR BITS IN MMR0
7988 057770 062701 000002 ADD #2,R1 ;TRY NEXT VIRTUAL ADDRESS
7989 057774 001337 BNE 2$ ;BRANCH IF NOT ALL DONE
7990 057776 012737 032226 000250 MOV #MMTRAP,MMVEC ;PUT BACK REGULAR M.M. TRAP ROUTINE
7991 060004 012737 057564 001112 MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST

```

```

7992
7993
7994 .SBTTL ***** ENTRY POINT 6 --- STARTING ADDRESS 224 *****
7995 .SBTTL ***** D-SPACE TESTS, CORRECT TIMING OF I & D SPACE *****

```

```

7996 :
7997 :
7998 : THIS GROUP OF TESTS CHECKS THE PROPER ENABLING OF D-SPACE.
7999 : IT TESTS THAT I-SPACE IS FORCED DURING INSTRUCTION FETCHES AND
8000 : ADDRESS, INDEX, OR OPERAND FETCHES IF THE REGISTER FIELD IS 7.
8001 : IT ALSO CHECKS THAT TRAPS PICK UP THE VECTOR FROM D-SPACE, IF
8002 : IT IS ENABLED.
8003 :
8004 : THROUGHOUT THIS AREA OF TESTING D-SPACE PAGES 2 & 3
8005 : ARE MAPPED NON-RESIDENT, AND I-SPACE PAGE 4 IS ALSO MAPPED
8006 : NON-RESIDENT. ALL OTHER PAGES IN BOTH I & D SPACE ARE
8007 : MAPPED RESIDENT, 4K, READ/WRITE. IF ANY ABORTS SHOULD OCCUR
8008 : DURING THESE TESTS THEY WILL VECTOR TO 'NODSPAC'. IF THE
8009 : OFFENDING PAGE IS 2 OR 3 THEN THE FAULT IS THAT I-SPACE WASN'T
8010 : FORCED WHEN IT SHOULD HAVE BEEN, BUT IF THE PAGE IS 4 THEN
8011 : THE FAULT IS THAT I-SPACE WAS FORCED WHEN IT SHOULD NOT HAVE BEEN.
8012 :
8013 :

```

```

8014 :*****
8015 :TEST 72 ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS FORCED
8016 :
8017 : THIS TEST SHOWS THAT I-SPACE IS FORCED BY EITHER 'SSRB I
8018 : SPACEA L' OR SSRB I SPACEB L' DURING THE PROPER TIMES
8019 :
8020 : ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
8021 : ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC'. THIS SUBROUTINE
8022 : WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
8023 :

```

```

8024 :*****
8025 ST72:
8026 SCOPE

```

8027	060014	012737	060606	001316	MOV	#TST73,NXTTST	:SAVE STARTING ADDRESS OF NEXT
8028							:TEST FOR ESCAPE ON PARITY ERRORS
8029	060022	104420			TBITR		:RESTORE THE T-BIT TO ITS CONDITION
8030							:BEFORE THE LAST FOUR TESTS.
8031	060024				ENTPT6:		
8032	060024	012737	060164	001110	MOV	#20\$,SLPADR	:SET LOOP ADDRESS POINTER TO 20\$
8033	060032	012737	060164	001112	MOV	#20\$,SLPERR	:SET LOOP ON ERROR POINTER TO 20\$
8034	060040	012737	000072	001102	MOV	#72,\$TSTNM	:LOAD TEST NUMBER INTO MEMORY
8035	060046	013737	001102	177570	MOV	\$TSTNM,DISPLAY	:DISPLAY TEST NUMBER FOR THIS TEST
8036	060054	012737	077406	172300	MOV	#77406,KIPDR0	:MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
8037	060062	012737	077406	172302	MOV	#77406,KIPDR1	:MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
8038	060070	012737	077406	172304	MOV	#77406,KIPDR2	:MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
8039	060076	012737	077406	172306	MOV	#77406,KIPDR3	:MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
8040	060104	012737	077406	172316	MOV	#77406,KIPDR7	:MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
8041	060112	012737	000000	172340	MOV	#000,KIPAR0	:MAP KERNEL I PAGE 0 TO 0 - 4K
8042	060120	012737	000200	172342	MOV	#200,KIPAR1	:MAP KERNEL I PAGE 1 TO 4K - 8K
8043	060126	012737	000400	172344	MOV	#400,KIPAR2	:MAP KERNEL I PAGE 2 TO 8K - 12K
8044	060134	012737	000600	172346	MOV	#600,KIPAR3	:MAP KERNEL I PAGE 3 TO 12K - 16K
8045	060142	012737	177600	172356	MOV	#177600,KIPAR7	:MAP KERNEL I PAGE 7 TO THE I/O PAGE
8046	060150	012737	000001	177572	MOV	#BIT0,MMR0	:ENABLE 18-BIT RELOCATION IF NOT ON
8047	060156	012737	000020	172516	MOV	#BIT4,MMR3	:ENABLE 22-BIT RELOCATION IF NOT ON
8048	060164	012737	077400	172310	20\$: MOV	#77400,KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
8049	060172	012737	077406	172320	MOV	#77406,KDPDR0	:MAKE KERNEL D PAGE 0 200 BLOCKS R/W
8050	060200	012737	077406	172322	MOV	#77406,KDPDR1	:MAKE KERNEL D PAGE 1 200 BLOCKS R/W
8051	060206	012737	077406	172330	MOV	#77406,KDPDR4	:MAKE KERNEL D PAGE 4 200 BLOCKS R/W
8052	060214	012737	077400	172324	MOV	#77400,KDPDR2	:MAKE KERNEL D PAGE 2 NON-RESIDENT
8053	060222	012737	077400	172326	MOV	#77400,KDPDR3	:MAKE KERNEL D PAGE 3 NON-RESIDENT
8054	060230	012737	000000	172360	MOV	#000,KDPAR0	:MAP KERNEL D PAGE 0 TO PHYSICAL 0
8055	060236	012737	000200	172362	MOV	#200,KDPAR1	:MAP KERNEL D PAGE 1 TO 4K - 8K
8056	060244	012737	001000	172370	MOV	#1000,KDPAR4	:MAP KERNEL D PAGE 4 TO 16K
8057	060252	012737	077406	172336	MOV	#77406,KDPDR7	:MAP KERNEL D PAGE 7 TO 200 BLOCKS R/W
8058	060260	012737	177600	172376	MOV	#177600,KDPAR7	:MAP KERNEL D PAGE 7 TO I/O PAGE
8059	060266	012704	172516		MOV	#MMR3,R4	:PUT ADDR OF MMR3 IN R4
8060	060272	012705	000004		MOV	#BIT2,R5	:PUT KERNEL D-SPACE ENABLE BIT INTO R5
8061	060276	012700	000002		MOV	#2,R0	:LOAD A TWO INTO R0
8062	060302	012737	032346	000250	MOV	#NODSPAC,MMVEC	:SET M.M. VECTOR TO D-SPACE SERVICE ROUTINE
8063	060310	012737	060320	001112	MOV	#10\$,SLPERR	:SET LOOP ON ERROR POINTER TO 10\$
8064	060316	050514			BIS	R5,(R4)	:ENABLE D-SPACE MAPPING IN KERNEL MODE
8065					::		
8066					::*		
8067					::*		
8068					::		
8069	060320	000400			10\$: BR	1\$:BRANCH, USE FET.00
8070	060322	000244			1\$: CLZ		:CLEAR ZERO BIT IN PROCESSOR STATUS
8071	060324	001776				1\$:NO BRANCH, USE FET.13
8072	060326	000264			2\$: SEZ		:SET ZERO BIT IN PROC. STATUS
8073	060330	001376				2\$:NO BRANCH, USE FET.12
8074	060332	000270			3\$: SEN		:SET NEGATIVE BIT IN PROC. STATUS
8075	060334	100376				3\$:NO BRANCH, USE FET.11
8076	060336	000237			4\$: SPL	7	:SET PRIOR TO 7: USE SPL.10, GOTO FET.10
8077	060340	005700				R0	:USE TST.10, GOTO FET.10
8078	060342	077003				R0,4\$:BRANCH UNTIL R0 IS ZERO:
8079							:USE SOB.20, GOTO FET.10
8080	060344	073002			ASHC	R2,R0	:COMBINED SHIFT, SHIFT COUNT ZERO
8081							:USE ASC.80, GOTO FET.10
8082	060346	005200			INC	R0	:MAKE R0 POSITIVE ONE


```
8139      : *      'TEST THAT 'ROM OUT09'' FORCES I-SPACE DURING INDEX WORD
8140      : *      FETCHES.
8141      : *
8142 060504 012737 060514 001112      MOV      #14$, $LPERR      ;SET LOOP ON ERROR POINTER TO 14$
8143 060512 050514                   BIS      R5,(R4)          ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8144 060514 012700 000000      14$:  MOV      #0,R0           ;BIN * DMO, USE D00.90, GOTO FET.10
8145 060520 010001                   MOV      R0,R1           ;BIN * DMO * SMO, USE EXC.80, GOTO FET.10
8146 060522 012767 000001 017250      MOV      #1,<77772-.>(PC) ;LOAD 1 INTO ADR 100000
8147      : *      ;DSTM = 6, DSTF = 7
8148      : *      ;USE D67.80, AND D67.00
8149 060530 112777 000000 017242      MOVB     #0,@100000      ;LOAD ZERO INTO ADR 000001
8150      : *      ;DSTM = 7, DSTF = 7
8151      : *      ;USE D67.90 AND D67.01
8152 060536 016700 017236      MOV      <77774-.>(PC),R0 ;READ FROM ADRS 100000
8153      : *      ;SRCM = 6, SRCF = 7
8154      : *      ;USE S67.00
8155 060542 040514                   BIC      R5,(R4)          ;DISABLE KERNEL D-SPACE MAPPING
8156      : *
8157      : *      TEST 'ROM OUT11'' ANDED WITH (PREV=I)
8158      : *
8159 060544 012737 060554 001112      MOV      #15$, $LPERR    ;SET LOOP ON ERROR POINTER TO 15$
8160 060552 050514                   BIS      R5,(R4)          ;ENABLE D-SPACE MAPPING IN KERNEL MODE
8161 060554 012737 000000 100000      15$:  MOV      #0,@#100000  ;CLEAR LOCATION 100000
8162 060562 005437 100000      NEG      @#100000        ;NEGATE ZERO, USE NEG.20, GOTO EXC.10
8163 060566 105437 100001      NEGB     @#100001        ;NEGATE UPPER BYTE OF 100000
8164      : *      ;USE SHR.10, GOTO EXC.10
8165 060572 005537 100000      ADC      @#100000        ;ADD CARRY BIT TO ADRS 100000
8166      : *      ;USE EXC.00, GOTO EXC.10
8167 060576 040514      19$:  BIC      R5,(R4)          ;DISABLE KERNEL D-SPACE MAPPING
8168 060600 012737 060164 001112      MOV      #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST
```

```
8169
8170
8171      : * *****
8172      : * TEST 73      ENABLE KERNEL D-SPACE AND SEE THAT I-SPACE IS NOT FORCED
8173      : *
8174      : *      THIS TEST SHOWS THAT I-SPACE IS NOT FORCED IF THE REGISTER
8175      : *      FIELD IS NOT 7 BUT THE OTHER CONDITIONS ARE STILL MET.
8176      : *
8177      : *      ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.F.U.
8178      : *      ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC'. THIS SUBROUTINE
8179      : *      WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
8180      : * *****
```

```
8181      : * *****
8182      : * TST73:
8183 060606 000004      SCOPE
8184 060610 012737 061110 001316      MOV      #TST74,NXTTST  ;SAVE STARTING ADDRESS OF NEXT
8185      : *      ;TEST FOR ESCAPE ON PARITY ERRORS
8186 060616 012737 077406 172310      20$:  MOV      #77406,KIPDR4  ;MAKE KERNEL I PAGE 4 200 BLOCKS, R/W
8187 060624 012737 001000 172350      MOV      #1000,KIPAR4   ;MAP KERNEL I PAGE 4 TO 16K
8188 060632 012737 077406 172320      MOV      #77406,KDPDR0  ;MAKE KERNEL D PAGE 0 200 BLOCKS R/W
8189 060640 012737 077406 172322      MOV      #77406,KDPDR1  ;MAKE KERNEL D PAGE 1 200 BLOCKS R/W
8190 060646 012737 077406 172330      MOV      #77406,KDPDR4  ;MAKE KERNEL D PAGE 4 200 BLOCKS R/W
8191 060654 012737 077400 172324      MOV      #77400,KDPDR2  ;MAKE KERNEL D PAGE 2 NON-RESIDENT
8192 060662 012737 077400 172326      MOV      #77400,KDPDR3  ;MAKE KERNEL D PAGE 3 NON-RESIDENT
8193 060670 012737 000000 172360      MOV      #000,KDPAR0    ;MAP KERNEL D PAGE 0 TO PHYSICAL 0
8194 060676 012737 001000 172370      MOV      #1000,KDPAR4   ;MAP KERNEL D PAGE 4 TO 16K
```

```

8195 060704 012737 077406 172336
8196 060712 012737 177600 172376
8197 060720 012704 172516
8198 060724 012705 000004
8199 060730 012700 100000
8200 060734 012737 100000 100000
8201 060742 012737 100000 100002
8202 060750 105037 172310
8203 060754 012737 060762 001112
8204
8205
8206
8207
8208 060762 012700 100000 11$:
8209 060766 050514
8210 060770 000240
8211 060772 011001
8212 060774 012001
8213 060776 013001
8214 061000 040514
8215
8216
6 7
8218
8219 061002 012737 061010 001112
8220 061010 012700 100000 12$:
8221 061014 050514
8222 061016 000240
8223 061020 005710
8224
8225 061022 005720
8226
8227 061024 021010
8228
8229 061026 122020
8230
8231 061030 040514
8232
8233
8234
8235
8236 061032 012737 061040 001112
8237 061040 112737 000006 172310 13$:
8238 061046 012700 100000
8239 061052 105037 172310
8240 061056 050514
8241 061060 000240
8242 061062 005030
8243 061064 012730 100000
8244 061070 012710 100001
8245 061074 112730 000200
8246 061100 040514 19$:
8247 061102 012737 060616 001112
8248
6249
8250

```

```

MOV #77406,KDPDR7 ;MAP KERNEL D PAGE 7 TO 200 BLOCKS R/W
MOV #177600,KDPAR7 ;MAP KERNEL D PAGE 7 TO I/O PAGE
MOV #MMR3,R4 ;PUT ADDRS OF MMR3 IN R4
MOV #BIT2,R5 ;PUT KERNEL D-SPACE ENABLE BIT INTO R5
MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0
MOV #100000,@#100000 ;LOAD ADDRESS 100000 WITH 100000
MOV #100000,@#100002 ;LOAD ADDRESS 100002 WITH 100000
CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
MOV #11$,$LPERR ;SET LOOP ON ERROR POINTER TO 11$

TEST THAT 'ROM OUT13' COMES UP WHEN SRCM = 1, 2, OR 3.
IN THIS CASE SRCF = -7.

MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0
BIS R5,(R4) ;ENABLE D-SPACE MAPPING IN KERNEL MODE
NOP ;THIS IS A SYNC POINT FOR SCOPING
MOV (R0),R1 ;USE S13.00, SOURCE MODE IS 1
MOV (R0)+,R1 ;USE S13.01, SOURCE MODE IS 2
MOV @(R0)+,R1 ;USE S13.01, SOURCE MODE IS 3
BIC R5,(R4) ;DISABLE KERNEL D-SPACE MAPPING

TEST THAT 'ROM OUT14' COMES UP WHEN DSTM = 1, OR 2.
THIS IS ANDED WITH DSTF = -7 AND NOT(MTP + MFP).

MOV #12$,$LPERR ;SET LOOP ON ERROR POINTER TO 12$
MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0
BIS R5,(R4) ;ENABLE D-SPACE MAPPING IN KERNEL MODE
NOP ;THIS IS A SYNC POINT FOR SCOPING
TST (R0) ;USE D12.00, GOTO D12.10
; (INST. IS DAC * DM1)
TST (R0)+ ;USE D12.01, GOTO D12.10
; (INST IS DAC * DM2)
CMP (R0),(R0) ;USE D12.80, GOTO D12.60
; (INST IS BIN * DM1)
CMPB (R0)+,(R0)+ ;USE D12.90, GOTO D12.60
; (INST IS BIN * DM2)
BIC R5,(R4) ;DISABLE KERNEL D-SPACE MAPPING

TEST THAT 'ROM OUT15' COMES UP WHEN DSTM = 3.
THIS IS ANDED WITH DSTF = -7.

MOV #13$,$LPERR ;SET LOOP ON ERROR POINTER TO 13$
MOVB #006,KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 RESIDENT
MOV #100000,R0 ;LOAD ADDRESS 100000 INTO R0
CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
BIS R5,(R4) ;ENABLE D-SPACE MAPPING IN KERNEL MODE
NOP ;THIS IS A SYNC POINT FOR SCOPING
CLR @(R0)+ ;USE D30.00, GOTO D30.10
MOV #100000,@(R0)+ ;USE D30.80, GOTO D30.10
MOV #100001,(R0) ;LOAD ODD ADDRESS INTO LOCATION OF R0
MOVB #200,@(R0)+ ;USE D30.90, GOTO D30.10
BIC R5,(R4) ;DISABLE KERNEL D-SPACE MAPPING
MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST

```


8251
8252
8253
8254
8255
8256
8257
8258
8259
8260
8261
8262
8263 061110
8264 061110 000004
8265 061112 012737 061562 001316
8266
8267 061120 012700 077406
8268 061124 010037 172200
8269 061130 010037 172202
8270 061134 010037 172204
8271 061140 010037 172206
8272 061144 010037 172210
8273 061150 010037 172216
8274 061154 010037 172220
8275 061160 010037 172222
8276 061164 010037 172230
8277 061170 010037 172236
8278 061174 105000
8279 061176 010037 172224
8280 061202 010037 172226
8281 061206 012737 000000 172240
8282 061214 012737 000200 172242
8283 061222 012737 000400 172244
8284 061230 012737 000600 172246
8285 061236 012737 001000 172250
8286 061244 012737 177600 172256
8287 061252 012737 000000 172260
8288 061260 012737 000200 172262
8289 061266 012737 177600 172276
8290 061274 012704 172516
8291 061300 012705 000002
8292 061304 052737 040000 177776
8293 061312 105037 172210
8294 061316 012737 061324 001112
8295
8296
8297
8298 061324 050514
8299 061326 000240
8300 061330 000400
8301 061332 040514
8302 061334 012737 061342 001112
8303 061342 050514
8304 061344 000240
8305 061346 017777 016426 016424
8306 061354 040514

```
*****
*TEST 74      PROPER ENABLING OF SUPERVISOR D-SPACE
*
*   THIS TEST SHOWS THE PROPER FUNCTIONING OF SUPERVISOR D-SPACE.
*   BOTH 'SSRB I SPACEA L' AND 'SSRB I SPACEB L' ARE ASSERTED
*   DURING THIS TEST FORCING I-SPACE.
*
*   ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.F.U.
*   ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC'. THIS SUBROUTINE
*   WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
*****
TST74:
SCOPE
MOV      #TST75,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
                          ;TEST FOR ESCAPE ON PARITY ERRORS
20$:    MOV      #17406,R0   ;MAKE THE NEXT FEW PAGES 4K, R/W, UP
        MOV      R0,SIPDR0  ;SUPERVISOR I-SPACE PAGE 0
        MOV      R0,SIPDR1  ;SUPERVISOR I-SPACE PAGE 1
        MOV      R0,SIPDR2  ;SUPERVISOR I-SPACE PAGE 2
        MOV      R0,SIPDR3  ;SUPERVISOR I-SPACE PAGE 3
        MOV      R0,SIPDR4  ;SUPERVISOR I-SPACE PAGE 4
        MOV      R0,SIPDR7  ;SUPERVISOR I-SPACE PAGE 7
        MOV      R0,SDPDR0  ;SUPERVISOR D-SPACE PAGE 0
        MOV      R0,SDPDR1  ;SUPERVISOR D-SPACE PAGE 1
        MOV      R0,SDPDR4  ;SUPERVISOR D-SPACE PAGE 4
        MOV      R0,SDPDR7  ;SUPERVISOR D-SPACE PAGE 7
        CLRB     R0         ;MAKE THE NEXT TWO PAGES NON-RESIDENT
        MOV      R0,SDPDR2  ;SUPERVISOR D-SPACE PAGE 2
        MOV      R0,SDPDR3  ;SUPERVISOR D-SPACE PAGE 3
        MOV      #000,SIPAR0 ;MAP SUPER I-SPACE PAGE 0 TO PHY 0
        MOV      #200,SIPAR1 ;MAP SUPER I-SPACE PAGE 1 TO 4K - 8K
        MOV      #400,SIPAR2 ;MAP SUPER I-SPACE PAGE 2 TO 8K - 12K
        MOV      #600,SIPAR3 ;MAP SUPER I-SPACE PAGE 3 TO 12K - 16K
        MOV      #1000,SIPAR4 ;MAP SUPER I-SPACE PAGE 4 TO 16K - 20K
        MOV      #177600,SIPAR7 ;MAP SUPER I-SPACE PAGE 7 TO I/O PAGE
        MOV      #000,SDPAR0 ;MAP SUPER D-SPACE PAGE 0 TO PHY 0
        MOV      #200,SDPAR1 ;MAP SUPER D-SPACE PAGE 1 TO 4K - 8K
        MOV      #177600,SDPAR7 ;MAP SUPER D-SPACE PAGE 7 TO I/O PAGE
        MOV      #MMR3,R4   ;PUT ADDRESS OF MMR3 IN R4
        MOV      #BIT1,R5   ;PUT ENABLE SUPERVISOR D-SPACE BIT IN R5
        BIS      #40000,PSW  ;GO INTO SUPERVISOR MODE HERE
        CLRB     SIPDR4     ;MAKE I-SPACE PAGE 4 NON-RESIDENT
        MOV      #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 10$
:
:
:   THIS SECTION CHECKS SIGNAL SPACEA FOR DISABLING D-SPACE
:
10$:    BIS      R5,(R4)     ;ENABLE SUPER D-SPACE
        NOP
        BR      1$         ;THIS IS A SYNC POINT FOR SCOPING
:
1$:     BIC      R5,(R4)     ;DISABLE SUPER D-SPACE
        MOV      #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
:
11$:    BIS      R5,(R4)     ;ENABLE SUPER D-SPACE
        NOP
        MOV      @100000,@100000 ;THIS INST USES ROM OUT09
        BIC      R5,(R4)     ;DISABLE SUPER D-SPACE
```



```
8307 061356 012737 061364 001112      MOV    #12$, $LPERR    ;SET LOOP ON ERROR POINTER TO 12$
8308 061364 050514                12$:  BIS    R5, (R4)        ;ENABLE SUPER D-SPACE
8309 061366 000240                NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8310 061370 005037 100000          CLR    @#100000        ;THIS INST USES ROM OUT15 & DSTF7
8311 061374 040514                BIC    R5, (R4)        ;DISABLE SUPER D-SPACE
8312                                     ::
8313                                     :::
8314                                     :::
8315 061376 012737 061404 001112      MOV    #13$, $LPERR    ;SET LOOP ON ERROR POINTER TO 13$
8316 061404 050514                13$:  BIS    R5, (R4)        ;ENABLE SUPER D-SPACE
8317 061406 000240                NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8318 061410 013700 100000          MOV    @#100000, R0    ;THIS INST USES ROM OUT13 & SRCF7
8319 061414 040514                BIC    R5, (R4)        ;DISABLE SUPER D-SPACE
8320 061416 012737 061424 001112      MOV    #14$, $LPERR    ;SET LOOP ON ERROR POINTER TO 14$
8321 061424 050514                14$:  BIS    R5, (R4)        ;ENABLE SUPER D-SPACE
8322 061426 000240                NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8323 061430 005727 000000          TST    #0              ;THIS INST USES ROM OUT14 & DSTF7
8324 061434 040514                BIC    R5, (R4)        ;DISABLE SUPER D-SPACE
8325                                     ::
8326                                     :::
8327                                     :::
8328                                     :::
8329 061436 012737 061444 001112      MOV    #16$, $LPERR    ;SET LOOP ON ERROR POINTER TO 16$
8330 061444 112737 000006 172210      MOVB   #006, $IPDR4    ;MAKE I-SPACE PAGE 4 RESIDENT
8331 061452 012700 100000          MOV    #100000, R0    ;LOAD ADDRESS 100000 INTO R0
8332 061456 012737 100000 100000      MOV    #100000, @#100000 ;LOAD NO. 100000 INTO ADDRESS 100000
8333 061464 105037 172210          CLRB   $IPDR4         ;MAKE I-SPACE PAGE 4 NON-RESIDENT
8334 061470 050514                BIS    R5, (R4)        ;ENABLE SUPER D-SPACE
8335 061472 000240                NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8336 061474 012730 100000          MOV    #100000, @ (R0)+ ;THIS INST USES ROM OUT15 & -DSTF7
8337 061500 040514                BIC    R5, (R4)        ;DISABLE SUPER D-SPACE
8338 061502 012737 061510 001112      MOV    #17$, $LPERR    ;SET LOOP ON ERROR POINTER TO 17$
8339 061510 012700 100000          17$:  MOV    #100000, R0    ;LOAD ADDRESS 100000 INTO R0
8340 061514 050514                BIS    R5, (R4)        ;ENABLE SUPER D-SPACE
8341 061516 000240                NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8342 061520 011001                MOV    (R0), R1       ;THIS INST USES ROM OUT13 & -SRCF7
8343 061522 040514                BIC    R5, (R4)        ;DISABLE SUPER D-SPACE
8344 061524 012737 061532 001112      MOV    #18$, $LPERR    ;SET LOOP ON ERROR POINTER TO 18$
8345 061532 012700 100000          18$:  MOV    #100000, R0    ;LOAD ADDRESS 100000 INTO R0
8346 061536 050514                BIS    R5, (R4)        ;ENABLE SUPER D-SPACE
8347 061540 000240                NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8348 061542 005720                TST    (R0)+          ;THIS INST USES ROM OUT14 & -DSTF7
8349 061544 040514                BIC    R5, (R4)        ;DISABLE SUPER D-SPACE
8350 061546 042737 040000 177776      BIC    #40000, $PSW    ;GO BACK TO KERNEL MODE
8351 061554 012737 061120 001112      MOV    #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST
```

```
8352
8353
8354 *****
8355 *TEST 75      PROPER ENABLING OF USER D-SPACE
8356 *
8357 *          THIS TEST SHOWS THE PROPER FUNCTIONING OF USER D-SPACE.
8358 *          BOTH 'SSRB I SPACEA L' AND 'SSRB I SPACEB L' ARE ASSERTED
8359 *          DURING THIS TEST FORCING I-SPACE.
8360 *
8361 *          ALL ERRORS FOUND IN THIS TEST ARE REPORTED WHEN THE C.P.U.
8362 *          ABORTS THRU 'MMVEC' TO SUBROUTINE 'NODSPAC'. THIS SUBROUTINE
```

```

8363          : *      WILL REPORT THAT D-SPACE WAS NOT ENABLED PROPERLY.
8364          : *
8365          : * *****
8366 061562    TST75:
8367 061562 000004          SCOPE
8368 061564 012737 062234 001316  MOV      #TST76,NXTTST      ;SAVE STARTING ADDRESS OF NEXT
8369          :TEST FOR ESCAPE ON PARITY ERRORS
8370 061572 012700 077406 20$:  MOV      #77406,R0          ;MAKE THE NEXT FEW PAGES 4K, R/W, UP
8371 061576 010037 177600          MOV      R0,UIPDR0          ;USER I-SPACE PAGE 0
8372 061602 010037 177602          MOV      R0,UIPDR1          ;USER I-SPACE PAGE 1
8373 061606 010037 177604          MOV      R0,UIPDR2          ;USER I-SPACE PAGE 2
8374 061612 010037 177606          MOV      R0,UIPDR3          ;USER I-SPACE PAGE 3
8375 061616 010037 177610          MOV      R0,UIPDR4          ;USER I-SPACE PAGE 4
8376 061622 010037 177616          MOV      R0,UIPDR7          ;USER I-SPACE PAGE 7
8377 061626 010037 177620          MOV      R0,UDPDR0          ;USER D-SPACE PAGE 0
8378 061632 010037 177622          MOV      R0,UDPDR1          ;USER D-SPACE PAGE 1
8379 061636 010037 177630          MOV      R0,UDPDR4          ;USER D-SPACE PAGE 4
8380 061642 010037 177636          MOV      R0,UDPDR7          ;USER D-SPACE PAGE 7
8381 061646 105000          CLR      R0                ;MAKE THE NEXT TWO PAGES NON-RESIDENT
8382 061650 010037 177624          MOV      R0,UDPDR2          ;USER D-SPACE PAGE 2
8383 061654 010037 177626          MOV      R0,UDPDR3          ;USER D-SPACE PAGE 3
8384 061660 012737 000000 177640  MOV      #000,UIPAR0        ;MAP USER I-SPACE PAGE 0 TO PHY 0
8385 061666 012737 000200 177642  MOV      #200,UIPAR1        ;MAP USER I-SPACE PAGE 1 TO 4K - 8K
8386 061674 012737 000400 177644  MOV      #400,UIPAR2        ;MAP USER I-SPACE PAGE 2 TO 8K - 12K
8387 061702 012737 000600 177646  MOV      #600,UIPAR3        ;MAP USER I-SPACE PAGE 3 TO 12K - 16K
8388 061710 012737 001000 177650  MOV      #1000,UIPAR4       ;MAP USER I-SPACE PAGE 4 TO 16K - 20K
8389 061716 012737 177600 177656  MOV      #177600,UIPAR7     ;MAP USER I-SPACE PAGE 7 TO I/O PAGE
8390 061724 012737 000000 177660  MOV      #000,JDPAR0        ;MAP USER D-SPACE PAGE 0 TO PHY 0
8391 061732 012737 000200 177662  MOV      #200,UDPDR1        ;MAP USER D-SPACE PAGE 1 TO 4K - 8K
8392 061740 012737 177600 177676  MOV      #177600,UDPDR7     ;MAP USER D-SPACE PAGE 7 TO I/O PAGE
8393 061746 012704 172516          MOV      #MMR3,R4          ;PUT ADDRESS OF MMR3 IN R4
8394 061752 012705 000001          MOV      #BIT0,R5          ;PUT ENABLE USER D-SPACE BIT IN R5
8395 061756 052737 140000 177776  BIS      #140000,PSW        ;GO INTO USER MODE HERE
8396 061764 105037 177610          CLR      UIPDR4           ;MAKE I-SPACE PAGE 4 NON-RESIDENT
8397 061770 012737 061776 001112  MOV      #10$, $LPERR       ;SET LOOP ON ERROR POINTER TO 10$
8398 061776 050514          10$:  BIS      R5,(R4)           ;ENABLE USER D-SPACE
8399 062000 000240          NOP                               ;THIS IS A SYNC POINT FOR SCOPING
8400 062002 000400          BR      1$                  ;THIS INST FETCH SHOULD USE ROM OUT09
8401 062004 040514          1$:  BIC      R5,(R4)           ;DISABLE USER D-SPACE
8402 062006 012737 062014 001112  MOV      #11$, $LPERR       ;SET LOOP ON ERROR POINTER TO 11$
8403 062014 050514          11$:  BIS      R5,(R4)           ;ENABLE USER D-SPACE
8404 062016 000240          NOP                               ;THIS IS A SYNC POINT FOR SCOPING
8405 062020 017777 015754 015752  MOV      @100000,@100000    ;THIS INST USES ROM OUT09
8406 062026 040514          BIC      R5,(R4)           ;DISABLE USER D-SPACE
8407 062030 012737 062036 001112  MOV      #12$, $LPERR       ;SET LOOP ON ERROR POINTER TO 12$
8408 062036 050514          12$:  BIS      R5,(R4)           ;ENABLE USER D-SPACE
8409 062040 000240          NOP                               ;THIS IS A SYNC POINT FOR SCOPING
8410 062042 005037 100000          CLR      @#100000          ;THIS INST USES ROM OUT15 & DSTF7
8411 062046 040514          BIC      R5,(R4)           ;DISABLE USER D-SPACE
8412          : :
8413          : : THIS SECTION CHECKS SIGNAL SPACEA FOR DISABLING D-SPACE
8414          : :
8415 062050 012737 062056 001112  MOV      #13$, $LPERR       ;SET LOOP ON ERROR POINTER TO 13$
8416 062056 050514          13$:  BIS      R5,(R4)           ;ENABLE USER D-SPACE
8417 062060 000240          NOP                               ;THIS IS A SYNC POINT FOR SCOPING
8418 062062 013700 100000          MOV      @#100000,R0       ;THIS INST USES ROM OUT13 & SRCF7

```

```
8419 062066 040514          BIC    R5,(R4)          ;DISABLE USER D-SPACE
8420 062070 012737 062076 001112  MOV    #14$,$LPFR      ;SET LOOP ON ERROR POINTER TO 14$
8421 062076 050514          BIS    R5,(R4)          ;ENABLE USER D-SPACE
8422 062100 000240          NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8423 062102 005727 000000  TST    #0                ;THIS INST USES ROM OUT14 & DSTF7
8424 062106 040514          BIC    R5,(R4)          ;DISABLE USER D-SPACE
8425                                     :::
8426                                     :::
8427                                     :::
8428                                     :::
8429 062110 012737 062116 001112  MOV    #16$,$LPERR      ;SET LOOP ON ERROR POINTER TO 16$
8430 062116 112737 000006 177610 16$:  MOVB   #006,UIPDR4      ;MAKE I-SPACE PAGE 4 RESIDENT
8431 062124 012700 100000          MOV    #100000,R0       ;LOAD ADDRESS 100000 INTO R0
8432 062130 012737 100000 100000  MOV    #100000,@#100000 ;LOAD NO. 100000 INTO ADDRESS 100000
8433 062136 105037 177610          CLRB  UIPDR4           ;MAKE I-SPACE PAGE 4 NON-RESIDENT
8434 062142 050514          BIS    R5,(R4)          ;ENABLE USER D-SPACE
8435 062144 000240          NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8436 062146 012730 100000          MOV    #100000,@(R0)+   ;THIS INST USES ROM OUT15 & -DSTF7
8437 062152 040514          BIC    R5,(R4)          ;DISABLE USER D-SPACE
8438 062154 012737 062162 001112  MOV    #17$,$LPERR      ;SET LOOP ON ERROR POINTER TO 17$
8439 062162 012700 100000 17$:  MOV    #100000,R0       ;LOAD ADDRESS 100000 INTO R0
8440 062166 050514          BIS    R5,(R4)          ;ENABLE USER D-SPACE
8441 062170 000240          NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8442 062172 011001          MOV    (R0),R1         ;THIS INST USES ROM OUT13 & -SRCF7
8443 062174 040514          BIC    R5,(R4)          ;DISABLE USER D-SPACE
8444 062176 012737 062204 001112  MOV    #18$,$LPERR      ;SET LOOP ON ERROR POINTER TO 18$
8445 062204 012700 100000 18$:  MOV    #100000,R0       ;LOAD ADDRESS 100000 INTO R0
8446 062210 050514          BIS    R5,(R4)          ;ENABLE USER D-SPACE
8447 062212 000240          NOP                                ;THIS IS A SYNC POINT FOR SCOPING
8448 062214 005720          TST    (R0)+           ;THIS INST USES ROM OUT14 & -DSTF7
8449 062216 040514          BIC    R5,(R4)          ;DISABLE USER D-SPACE
8450 062220 042737 140000 177776  BIC    #140000,PSW      ;GO BACK TO KERNEL MODE
8451 062226 012737 061572 001112  MOV    #20$,$LPERR      ;SET LOOP POINTER TO START OF TEST
```

```
8452
8453
8454 :*****
8455 :*TEST 76          TRAPPING IN D-SPACE KERNEL MODE
8456 :*
8457 :*      THIS TEST VERIFIES THAT THE ABORT VECTOR IS TAKEN FROM D-SPACE
8458 :*      AND NOT I-SPACE.  THE I-SPACE VECTOR POINTS TO 10$, AND
8459 :*      THE D-SPACE VECTOR POINTS TO 15$.  EACH PSW IN VIRTUAL 252 IS
8460 :*      DIFFERENT SO THE PROGRAM CAN TELL WHICH AREA IT IS PICKED
8461 :*      UP FROM.
8462 :*****
```

```
8463 062234          TS^76:
8464 062234 000004          SCOPE
8465 062236 012737 062502 001316  MOV    #TS^77,NXTTST   ;SAVE STARTING ADDRESS OF NEXT
8466                                     ;TEST FOR ESCAPE ON PARITY ERRORS
8467 062244 104416          TBITO          ;MAKE SURE T-BIT IS OFF FOR THIS TEST
8468 062246 012737 062304 001112 20$:  MOV    #1$,$LPERR      ;SET LOOP ON ERROR POINTER TO 1$
8469 062254 012737 062356 000250  MOV    #10$,@#250      ;SET M.M.VEC TO HOLD BAD VECTOR
8470 062262 012737 000000 000252  MOV    #000,@#252      ;PROC. STAT. IN 252 HAS PRIORITY OF ZERO
8471 062270 012737 062364 000350  MOV    #15$,@#350      ;D-SPACE M.M.VECTOR IS AT 350
8472 062276 012737 000340 000352  MOV    #340,@#352      ;PRIORITY FOR D-SPACE VECTOR IS 7
8473 062304 012706 001000 1$:    MOV    #1000,KSP        ;SET UP KERNEL VECTOR
8474 062310 042737 177776 177572  BIC    #177776,MMRO    ;CLEAR ALL ERROR BITS IN MMRO
```


8531	062512	104420			TBITR		:RESTORE THE T-BIT TO ITS CONDITION :BEFORE THE LAST TEST.
8532							
8533	062514				ENTPT7:		
8534	062514	012737	062654	001110	MOV	#20\$,SLPADR	:SET LOOP ADDRESS POINTER TO 20\$
8535	062522	012737	062654	001112	MOV	#20\$,SLPERR	:SET LOOP ON ERROR POINTER TO 20\$
8536	062530	012737	000077	001102	MOV	#77,\$TSTNM	:LOAD TEST NUMBER INTO MEMORY
8537	062536	013737	001102	177570	MOV	\$TSTNM,DISPLAY	:DISPLAY TEST NUMBER FOR THIS TEST
8538	062544	012737	077406	172300	MOV	#77406,KIPDR0	:MAKE KERNEL I PAGE 0 200 BLOCKS, R/W
8539	062552	012737	077406	172302	MOV	#77406,KIPDR1	:MAKE KERNEL I PAGE 1 200 BLOCKS, R/W
8540	062560	012737	077406	172304	MOV	#77406,KIPDR2	:MAKE KERNEL I PAGE 2 200 BLOCKS, R/W
8541	062566	012737	077406	172306	MOV	#77406,KIPDR3	:MAKE KERNEL I PAGE 3 200 BLOCKS, R/W
8542	062574	012737	077406	172316	MOV	#77406,KIPDR7	:MAKE KERNEL I PAGE 7 200 BLOCKS, R/W
8543	062602	012737	000000	172340	MOV	#000,KIPAR0	:MAP KERNEL I PAGE 0 TO 0 - 4K
8544	062610	012737	000200	172342	MOV	#200,KIPAR1	:MAP KERNEL I PAGE 1 TO 4K - 8K
8545	062616	012737	000400	172344	MOV	#400,KIPAR2	:MAP KERNEL I PAGE 2 TO 8K - 12K
8546	062624	012737	000600	172346	MOV	#600,KIPAR3	:MAP KERNEL I PAGE 3 TO 12K - 16K
8547	062632	012737	177600	172356	MOV	#177600,KIPAR7	:MAP KERNEL I PAGE 7 TO THE I/O PAGE
8548	062640	012737	000001	177572	MOV	#BITC,MMR0	:ENABLE 18-BIT RELOCATION IF NOT ON
8549	062646	012737	000020	172516	MOV	#BIT4,MMR3	:ENABLE 22-BIT RELOCATION IF NOT ON
8550					.EQUIV	BIT6,WBIT	:LABEL 'WBIT' IS EQUIVALENT TO BIT 6
8551					.EQUIV	BIT7,ABIT	:LABEL 'ABIT' IS EQUIVALENT TO BIT 7
8552	062654	012737	000006	172310	20\$: MOV	#00006,KIPDR4	:PAGE 4 HAS A PAGE LENGTH OF 1
8553	062662	012737	001000	172350	MOV	#1000,KIPAR4	:MAP PAGE 4 TO 16K
8554	062670	012737	063304	000250	MOV	#10\$,MMVEC	:SET M.M. TRAP VECTOR TO 10\$
8555	062676	012737	062704	001112	MOV	#11\$,SLPERR	:SET LOOP ON ERROR POINTER TO 11\$
8556	062704	000240			11\$: NOP		:THIS IS A SYNC POINT FOR SCOPING
8557	062706	012737	012345	100000	1\$: MOV	#12345,@#100000	:WRITE INTO PAGE 4
8558	062714	013737	172310	001172	MOV	KIPDR4,\$TMP0	:READ PDR 4 TO CHECK W-BIT
8559	062722	032737	000100	001172	BIT	#WBIT,\$TMP0	:SEE IF W-BIT IS SET
8560	062730	001001			BNE	2\$:BRANCH IF W-BIT IS SET
8561	062732	104102			ERROR	102	:W-BIT NOT SET
8562	062734	012737	062742	001112	2\$: MOV	#12\$,SLPERR	:SET LOOP ON ERROR POINTER TO 12\$
8563	062742	000240			12\$: NOP		:THIS IS A SYNC POINT FOR SCOPING
8564	062744	012737	012345	100100	MOV	#12345,@#100100	:WRITE INTO PAGE 4, BLOCK 2
8565	062752	013737	172310	001172	MOV	KIPDR4,\$TMP0	:READ PDR 4 TO SEE IF W-BIT REMAINED SET
8566	062760	032737	000100	001172	BIT	#WBIT,\$TMP0	:SEE IF W-BIT REMAINED SET
8567	062766	001001			BNE	3\$:BRANCH IF W-BIT IS SET
8568	062770	104103			ERROR	103	:W-BIT DID NOT REMAIN SET
8569	062772	012737	063000	001112	3\$: MOV	#13\$,SLPERR	:SET LOOP ON ERROR POINTER TO 13\$
8570	063000	000240			13\$: NOP		:THIS IS A SYNC POINT FOR SCOPING
8571	063002	112737	000004	172310	MOVB	#004,KIPDR4	:CHANGE PAGE 4'S ACCESS CONTROL FIELD
8572	063010	013737	172310	001172	MOV	KIPDR4,\$TMP0	:READ PDR 4 TO CHECK W-BIT CLEARING
8573	063016	022737	000004	001172	CMP	#00004,\$TMP0	:SEE IF W-BIT GOT CLEARED ON PDR WRITE
8574	063024	001401			BEQ	4\$:BRANCH IF W-BIT GOT CLEARED
8575	063026	104104			ERROR	104	:W-BIT DIDN'T GET CLEARED ON PDR WRITE
8576	063030	012737	063036	001112	4\$: MOV	#14\$,SLPERR	:SET LOOP ON ERROR POINTER TO 14\$
8577	063036	052737	001000	177572	14\$: BIS	#BIT9,MMR0	:ENABLE M.M. TRAPS
8578	063044	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
8579	063046	013700	100000		MOV	@#100000,R0	:READ FROM PAGE 4, BLOCK 1
8580	063052	013737	172310	001172	MOV	KIPDR4,\$TMP0	:READ PDR 4 TO CHECK A-BIT
8581	063060	032737	000200	001172	BIT	#ABIT,\$TMP0	:SEE IF A-BIT WAS SET DURING READ
8582	063066	001001			BNE	5\$:BRANCH IF A-BIT WAS SET
8583	063070	104105			ERROR	105	:A-BIT DIDN'T GET SET
8584	063072	012737	063100	001112	5\$: MOV	#15\$,SLPERR	:SET LOOP ON ERROR POINTER TO 15\$
8585	063100	052737	001000	177572	15\$: BIS	#BIT9,MMR0	:ENABLE M.M. TRAPS
8586	063106	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING

8587	063110	013700	100100			MOV	#100100,R0	;READ FROM PAGE 4, BLOCK 2
8588	063114	013737	172310	001172		MOV	KIPDR4,\$TMP0	;READ PDR 4 TO SEE IF A-BIT REMAINED SET
8589	063122	032737	000200	001172		BIT	#ABIT,\$TMP0	;SEE IF A-BIT IS STILL SET
8590	063130	001001				BNE	6\$;BRANCH IF A-BIT IS STILL SET
8591	063132	104106				ERROR	106	;A-BIT DIDN'T REMAIN SET
8592	063134	012737	063142	001112	6\$:	MOV	#16\$,\$LPERR	;SET LOOP ON ERROR POINTER TO 16\$
8593	063142	000240			16\$:	NOP		;THIS IS A SYNC POINT FOR SCOPING
8594	063144	012737	001000	172350		MOV	#1000,KIPAR4	;WRITE PAR 4 TO SEE IF A-BIT CLEARS
8595	063152	013737	172310	001172		MOV	KIPDR4,\$TMP0	;READ PDR 4 TO CHECK A-BIT
8596	063160	022737	000004	001172		CMP	#00004,\$TMP0	;SEE IF A-BIT IS CLEAR
8597	063166	001401				BEQ	7\$;BRANCH IF A-BIT IS CLEAR
8598	063170	104107				ERROR	107	;A-BIT DIDN'T CLEAR ON PAR WRITE
8599	063172	012737	063200	001112	7\$:	MOV	#17\$,\$LPERR	;SET LOOP ON ERROR POINTER TO 17\$
8600	063200	000240			17\$:	NOP		;THIS IS A SYNC POINT FOR SCOPING
8601	063202	012737	123456	170200		MOV	#123456,MAPLO	;LOAD A MAP REGISTER TO CHECK 'INT REG'
8602	063210	013737	172316	001172		MOV	KIPDR7,\$TMP0	;READ PDR 7 TO CHECK W-BIT
8603	063216	032737	000100	001172		BIT	#WBIT,\$TMP0	;SEE IF W-BIT SET ON MAP REGISTER WRITE
8604	063224	001001				BNE	8\$;BRANCH IF W-BIT WAS SET
8605	063226	104110				ERROR	110	;W-BIT DID NOT GET SET
8606	063230	012737	063236	001112	8\$:	MOV	#18\$,\$LPERR	;SET LOOP ON ERROR POINTER TO 18\$
8607	063236	000240			18\$:	NOP		;THIS IS A SYNC POINT FOR SCOPING
8608	063240	012737	077406	172310		MOV	#77406,KIPDR4	;LOAD AN INTERNAL REGISTER, SHOULDN'T
8609								;AFFECT PAGE 7'S W-BIT
8610	063246	013737	172316	001172		MOV	KIPDR7,\$TMP0	;READ PDR 7 TO CHECK W-BIT AGAIN
8611	063254	032737	000100	001172		BIT	#WBIT,\$TMP0	;SEE IF W-BIT IS STILL SET
8612	063262	001001				BNE	9\$;BRANCH IF W-BIT IS STILL SET
8613	063264	104111				ERROR	111	;W-BIT NOT SET AFTER WRITING PDR 4
8614	063266	012737	062654	001112	9\$:	MOV	#20\$,\$LPERR	;SET LOOP POINTER TO START OF TEST
8615	063274	012737	032226	000250		MOV	#MMTRAP,MMVEC	;RESTORE NORMAL M.M. TRAP ROUTINE
8616	063302	000404				BR	TST100	;BRANCH TO NEXT TEST
8617								
8618	063304	042737	177776	177572	10\$:	BIC	#177776,MMR0	;CLEAR MMR0 FOR NEXT READ OR WRITE
8619	063312	000002				RTI		;RETURN TO TEST AND CHECK A OR W BIT

8620
8621
8622
8623
8624
8625
8626
8627
8628
8629
8630
8631
8632
8633
8634
8635
8636
8637
8638
8639
8640
8641
8642

*TEST 100 DUAL MAPPING KERNEL MODE I-SPACE

THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405 (4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO 010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN THE KERNEL I-SPACE P.D.R. UNDER TEST. NOW ALL OF THE P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.

8643
8644
8645
8646
8647 063314
8648 063314 000004
8649 063316 012737 064540 001316
8650
8651
8652
8653
8654
8655 063324 012737 000000 172340
8656 063332 012737 000200 172342
8657 063340 012737 000400 172344
8658 063346 012737 000600 172346
8659 063354 012737 001000 172350
8660 063362 012737 001000 172352
8661 063370 012737 001000 172354
8662 063376 012737 177600 172356
8663 063404 012737 000000 172360
8664 063412 012737 000200 172362
8665 063420 012737 000400 172364
8666 063426 012737 000600 172366
8667 063434 012737 001000 172370
8668 063442 012737 001000 172372
8669 063450 012737 001000 172374
8670 063456 012737 177600 172376
8671 063464 012737 000000 172240
8672 063472 012737 000200 172242
8673 063500 012737 000400 172244
8674 063506 012737 000600 172246
8675 063514 012737 001000 172250
8676 063522 012737 001000 172252
8677 063530 012737 001000 172254
8678 063536 012737 177600 172256
8679 063544 012737 000000 172260
8680 063552 012737 000200 172262
8681 063560 012737 000400 172264
8682 063566 012737 000600 172266
8683 063574 012737 001000 172270
8684 063602 012737 001000 172272
8685 063610 012737 001000 172274
8686 063616 012737 177600 172276
8687 063624 012737 000000 177640
8688 063632 012737 000200 177642
8689 063640 012737 000400 177644
8690 063646 012737 000600 177646
8691 063654 012737 001000 177650
8692 063662 012737 001000 177652
8693 063670 012737 001000 177654
8694 063676 012737 177600 177656
8695 063704 012737 000000 177660
8696 063712 012737 000200 177662
8697 063720 012737 000400 177664
8698 063726 012737 000600 177666

.* WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
.* ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
.*

TST100:

SCOPE
MOV #TST101,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
THE FOLLOWING CODE IS USED TO INITIALIZE THE PAR'S FOR
THE NEXT TESTS, SO THAT THE CODE WILL RUN WITH MEMORY
MANAGEMENT FULLY ENABLED.

MOV #0,KIPAR0
MOV #200,KIPAR1
MOV #400,KIPAR2
MOV #600,KIPAR3
MOV #1000,KIPAR4
MOV #1000,KIPAR5
MOV #1000,KIPAR6
MOV #177600,KIPAR7
MOV #0,KDPAR0
MOV #200,KDPAR1
MOV #400,KDPAR2
MOV #600,KDPAR3
MOV #1000,KDPAR4
MOV #1000,KDPAR5
MOV #1000,KDPAR6
MOV #177600,KDPAR7
MOV #0,SIPAR0
MOV #200,SIPAR1
MOV #400,SIPAR2
MOV #600,SIPAR3
MOV #1000,SIPAR4
MOV #1000,SIPAR5
MOV #1000,SIPAR6
MOV #177600,SIPAR7
MOV #0,SDPAR0
MOV #200,SDPAR1
MOV #400,SDPAR2
MOV #600,SDPAR3
MOV #1000,SDPAR4
MOV #1000,SDPAR5
MOV #1000,SDPAR6
MOV #177600,SDPAR7
MOV #0,UIPAR0
MOV #200,UIPAR1
MOV #400,UIPAR2
MOV #600,UIPAR3
MOV #1000,UIPAR4
MOV #1000,UIPAR5
MOV #1000,UIPAR6
MOV #177600,UIPAR7
MOV #0,UDPAR0
MOV #200,UDPAR1
MOV #400,UDPAR2
MOV #600,UDPAR3

8699	053734	012737	001000	177670		MOV	#1000,UDPAR4	:	
8700	063742	012737	001000	177672		MOV	#1000,UDPAR5	:	
8701	063750	012737	001000	177674		MOV	#1000,UDPAR6	:	
8702	063756	012737	177600	177676		MOV	#177600,UDPAR7	:	
8703	063764	012737	064002	001110		MOV	#20\$, \$LPADR	:	SET LOOP ON TEST POINTER TO 20\$
8704	063772	012737	064002	001112		MOV	#20\$, \$LPERR	:	SET LOOP ON ERROR POINTER TO 20\$
8705	064000	104416				TBITO		:	MAKE SURE THAT T-BIT IS OFF FOR
8706								:	THE SIX DUAL MAPPING TESTS
8707	064002	012737	064152	001112	20\$:	MOV	#21\$, \$LPERR	:	SET LOOP ON ERROR POINTER TO 21\$
8708	064010	012737	000000	177776		MOV	#00000,PSW	:	GO TO KERNEL MODE
8709	064016	012703	172300			MOV	#KIPDR0,R3	:	LOAD FIRST ADDRESS OF KERNEL PDR'S
8710								:	THAT WILL BE TESTED IN THIS TEST
8711	064022	012704	000010			MOV	#10,R4	:	TEST THE NEXT EIGHT PDR'S
8712	064026	012705	010200			MOV	#10200,R5	:	LOAD STARTING VIRTUAL ADDRESS INTO R5
8713	064032	012700	077405		19\$:	MOV	#77405,R0	:	ALL PAGES WILL BE TRAP ON WRITE
8714	064036	005037	001172			CLR	\$TMP0	:	CLEAR CORRECT PDR SET INDICATOR
8715	064042	012702	000010			MOV	#10,R2	:	SET COUNT TO LOAD 8 ADDRESSES
8716	064046	012701	177620			MOV	#UDPDR0,R1	:	PUT ADDRESS OF FIRST PDR IN R1
8717	064052	010021			1\$:	MOV	R0,(R1)+	:	LOAD R0 INTO PDR ADDRESSED BY R1
8718	064054	077202				SOB	R2,1\$:	BRANCH BACK TO 1\$ IF R2 IS NOT ZERO
8719	064056	012702	000010			MOV	#10,R2	:	SET COUNT TO LOAD 8 ADDRESSES
8720	064062	012701	177600			MOV	#UIPDR0,R1	:	PUT ADDRESS OF FIRST PDR IN R1
8721	064066	010021			2\$:	MOV	R0,(R1)+	:	LOAD R0 INTO PDR ADDRESSED BY R1
8722	064070	077202				SOB	R2,2\$:	BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
8723	064072	012702	000010			MOV	#10,R2	:	SET COUNT TO LOAD 8 ADDRESSES
8724	064076	012701	172220			MOV	#SDPDR0,R1	:	PUT ADDRESS OF FIRST PDR IN R1
8725	064102	010021			3\$:	MOV	R0,(R1)+	:	LOAD R0 INTO PDR ADDRESSED BY R1
8726	064104	077202				SOB	R2,3\$:	BRANCH BACK TO 3\$ IF R2 IS NOT ZERO
8727	064106	012702	000010			MOV	#10,R2	:	SET COUNT TO LOAD 8 ADDRESSES
8728	064112	012701	172200			MOV	#SIPDR0,R1	:	PUT ADDRESS OF FIRST PDR IN R1
8729	064116	010021			4\$:	MOV	R0,(R1)+	:	LOAD R0 INTO PDR ADDRESSED BY R1
8730	064120	077202				SOB	R2,4\$:	BRANCH BACK TO 4\$ IF R2 IS NOT ZERO
8731	064122	012702	000010			MOV	#10,R2	:	SET COUNT TO LOAD 8 ADDRESSES
8732	064126	012701	172320			MOV	#KDPDR0,R1	:	PUT ADDRESS OF FIRST PDR IN R1
8733	064132	010021			5\$:	MOV	R0,(R1)+	:	LOAD R0 INTO PDR ADDRESSED BY R1
8734	064134	077202				SOB	R2,5\$:	BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
8735	064136	012702	000010			MOV	#10,R2	:	SET COUNT TO LOAD 8 ADDRESSES
8736	064142	012701	172300			MOV	#KIPDR0,R1	:	PUT ADDRESS OF FIRST PDR IN R1
8737	064146	010021			6\$:	MOV	R0,(R1)+	:	LOAD R0 INTO PDR ADDRESSED BY R1
8738	064150	077202				SOB	R2,6\$:	BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
8739	064152	012700	077405		21\$:	MOV	#77405,R0	:	MUST RE-INIT THESE PDRS IF ERROR
8740	064156	010037	172300			MOV	R0,KIPDR0	:	LOAD KERNEL PDR0
8741	064162	010037	172302			MOV	R0,KIPDR1	:	LOAD KERNEL PDR1
8742	064166	010037	172316			MOV	R0,KIPDR7	:	LOAD KERNEL PDR7
8743	064172	010037	172300			MOV	R0,KIPDR0	:	LOAD PDR0 OF PRESENT SPACE
8744	064176	010037	172316			MOV	R0,KIPDR7	:	LOAD PDR7 OF PRESENT SPACE
8745	064202	000240				NOP		:	THIS IS A SYNC POINT FOR SCOPING
8746	064204	011515				MOV	(R5),(R5)	:	WRITE INTO PAGE UNDER TEST
8747	064206	012702	000020			MOV	#20,R2	:	SET COUNTER TO READ NEXT 20 REGISTERS
8748	064212	012701	172300			MOV	#KIPDR0,R1	:	LOAD ADDRESS OF BEGINNING PDR
8749	064216	011100			7\$:	MOV	(R1),R0	:	READ PDR INTO R0
8750	064220	022700	077705			CMP	#77705,R0	:	SEE IF THIS WAS THE PDR WITH A&W BITS ON
8751	064224	001023				BNE	9\$:	BRANCH IF THIS IS NOT THE ONE
8752	064226	020103				CMP	R1,R3	:	SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8753	064230	001417				BEQ	8\$:	BRANCH IF ADDRESS IS CORRECT
8754	064232	104112				ERROR	112	:	A & W BITS GOT SET IN WRONG PDR

8755	064234	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8756	064240	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8757	064244	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8758	064250	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8759	064254	010037	172300		MOV	R0,KIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8760	064260	010037	172316		MOV	R0,KIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
8761	064264	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8762							:WERE TESTING PAGE SEVEN
8763	064266	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
8764	064270	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8765	064274	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8766	064300	077232			SQB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
8767	064302	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8768	064306	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8769	064312	011100		10\$:	MOV	(R1),R0	:READ PDR INTO R0
8770	064314	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
8771	064320	001023			BNE	12\$:BRANCH IF THIS IS NOT THE ONE
8772	064322	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8773	064324	001417			BEQ	11\$:BRANCH IF ADDRESS IS CORRECT
8774	064326	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
8775	064330	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8776	064334	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8777	064340	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8778	064344	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8779	064350	010037	172300		MOV	R0,KIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8780	064354	010037	172316		MOV	R0,KIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
8781	064360	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8782							:WERE TESTING PAGE SEVEN
8783	064362	000402			BR	12\$:GO UPDATE R1 FOR NEXT READ
8784	064364	005237	001172	11\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8785	064370	062701	000002	12\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8786	064374	077232			SQB	R2,10\$:BRANCH TO 10\$ IF ALL PDR'S NOT READ
8787	064376	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8788	064402	012701	177600		MOV	#UIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8789	064406	011100		13\$:	MOV	(R1),R0	:READ PDR INTO R0
8790	064410	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
8791	064414	001023			BNE	15\$:BRANCH IF THIS IS NOT THE ONE
8792	064416	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8793	064420	001417			BEQ	14\$:BRANCH IF ADDRESS IS CORRECT
8794	064422	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
8795	064424	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8796	064430	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8797	064434	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8798	064440	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8799	064444	010037	172300		MOV	R0,KIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8800	064450	010037	172316		MOV	R0,KIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
8801	064454	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8802							:WERE TESTING PAGE SEVEN
8803	064456	000402			BR	15\$:GO UPDATE R1 FOR NEXT READ
8804	064460	005237	001172	14\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8805	064464	062701	000002	15\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8806	064470	077232			SQB	R2,13\$:BRANCH TO 13\$ IF ALL PDR'S NOT READ
8807	064472	005737	001172		TST	\$TMP0	:SEE IF THERE WAS A CORRECT PDR
8808	064476	001002			BNE	16\$:BRANCH IF THERE WAS
8809	064500	011300			MOV	(R3),R0	:SAVE CONTENTS OF PDR UNDER TEST
8810	064502	104113			ERROR	113	:NO PDR ADDRESSES MATCHED

```
8811 064504 062703 000002 16$: ADD #2,R3 ;POINT TO NEXT PDR UNDER TEST
8812 064510 062705 020000 ADD #20000,R5 ;CHANGE PAGE NUMBER IN VIRT. ADDR.
8813 064514 005304 DEC R4 ;DECREMENT COUNTER
8814 064516 001402 BEQ 17$ ;BRANCH IF COUNTER IS ZERO
8815 064520 000137 064032 JMP 19$ ;JUMP TO LOAD PDR'S AGAIN
8816 064524 17$:
8817 064524 012737 000340 177776 MOV #340,PSW ;RETURN TO KERNEL MODE, PRIORITY 7
8818 064532 012737 064002 001112 MOV #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
8819
8820
```

```
8821 *****
8822 *TEST 101 DUAL MAPPING SUPERVISOR MODE I-SPACE
8823
8824 THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
8825 (4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
8826 010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
8827 SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
8828 SINCE BIT09 OF MMRO IS CLEAR) AND SET BOTH THE A & W BITS IN
8829 THE SUPERVISOR I-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
8830 P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
8831 ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
8832 WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
8833 ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
8834
8835 *****
```

```
8836 064540 TST101:
8837 064540 000004 SCOPE
8838 064542 012737 065306 001316 MOV #TST102,NXTTST ;SAVE STARTING ADDRESS OF NEXT
8839 ;TEST FOR ESCAPE ON PARITY ERRORS
8840 064550 012737 064720 001112 20$: MOV #21$,$LPERR ;SET LOOP ON ERROR POINTER TO 21$
8841 064556 012737 040000 177776 MOV #40000,PSW ;GO TO SUPERVISOR MODE
8842 064564 012703 172200 MOV #SIPDR0,R3 ;LOAD FIRST ADDRESS OF SUPERVISOR PDR'S
8843 ;THAT WILL BE TESTED IN THIS TEST
8844 064570 012704 000010 MOV #10,R4 ;TEST THE NEXT EIGHT PDR'S
8845 064574 012705 010200 MOV #10200,R5 ;LOAD STARTING VIRTUAL ADDRESS INTO R5
8846 064600 012700 077405 19$: MOV #77405,R0 ;ALL PAGES WILL BE TRAP ON WRITE
8847 064604 005037 001172 CLR $TMP0 ;CLEAR CORRECT PDR SET INDICATOR
8848 064610 012702 000010 MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
8849 064614 012701 172320 MOV #KDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
8850 064620 010021 1$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
8851 064622 077202 SOB R2,1$ ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
8852 064624 012702 000010 MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
8853 064630 012701 172300 MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
8854 064634 010021 2$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
8855 064636 077202 SOB R2,2$ ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
8856 064640 012702 000010 MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
8857 064644 012701 177620 MOV #UDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
8858 064650 010021 3$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
8859 064652 077202 SOB R2,3$ ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
8860 064654 012702 000010 MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
8861 064660 012701 177600 MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
8862 064664 010021 4$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
8863 064666 077202 SOB R2,4$ ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
8864 064670 012702 000010 MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
8865 064674 012701 172220 MOV #SDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
8866 064700 010021 5$: MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
```

8867	064702	077202		SOB	R2,5\$:BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
8868	064704	012702	000010	MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8869	064710	012701	172200	MOV	#SIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8870	064714	010021		6\$: MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8871	064716	077202		SOB	R2,6\$:BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
8872	064720	012700	077405	21\$: MOV	#77405,R0	:MUST RE-INIT THESE PDRS IF ERROR
8873	064724	010037	172300	MOV	R0,KIPDR0	:LOAD KERNEL PDR0
8874	064730	010037	172302	MOV	R0,KIPDR1	:LOAD KERNEL PDR1
8875	064734	010037	172316	MOV	R0,KIPDR7	:LOAD KERNEL PDR7
8876	064740	010037	172200	MOV	R0,SIPDR0	:LOAD PDR0 OF PRESENT SPACE
8877	064744	010037	172216	MOV	R0,SIPDR7	:LOAD PDR7 OF PRESENT SPACE
8878	064750	000240		NOP		:THIS IS A SYNC POINT FOR SCOPING
8879	064752	011515		MOV	(R5),(R5)	:WRITE INTO PAGE UNDER TEST
8880	064754	012702	000020	MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8881	064760	012701	172300	MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8882	064764	011100		7\$: MOV	(R1),R0	:READ PDR INTO R0
8883	064766	022700	077705	CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
8884	064772	001023		BNE	9\$:BRANCH IF THIS IS NOT THE ONE
8885	064774	020103		CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8886	064776	001417		BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
8887	065000	104112		ERROR	112	:A & W BITS GOT SET IN WRONG PDR
8888	065002	012700	077405	MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8889	065006	010037	172300	MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8890	065012	010037	172302	MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8891	065016	010037	172316	MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8892	065022	010037	172200	MOV	R0,SIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8893	065026	010037	172216	MOV	R0,SIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
8894	065032	011515		MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8895						:WERE TESTING PAGE SEVEN
8896	065034	000402		BR	9\$:GO UPDATE R1 FOR NEXT READ
8897	065036	005237	001172	8\$: INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8898	065042	062701	000002	9\$: ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8899	065046	077232		SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
8900	065050	012702	000020	MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8901	065054	012701	172200	MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8902	065060	011100		10\$: MOV	(R1),R0	:READ PDR INTO R0
8903	065062	022700	077705	CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
8904	065066	001023		BNE	12\$:BRANCH IF THIS IS NOT THE ONE
8905	065070	020103		CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8906	065072	001417		BEQ	11\$:BRANCH IF ADDRESS IS CORRECT
8907	065074	104112		ERROR	112	:A & W BITS GOT SET IN WRONG PDR
8908	065076	012700	077405	MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
8909	065102	010037	172300	MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
8910	065106	010037	172302	MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
8911	065112	010037	172316	MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
8912	065116	010037	172200	MOV	R0,SIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
8913	065122	010037	172216	MOV	R0,SIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
8914	065126	011515		MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
8915						:WERE TESTING PAGE SEVEN
8916	065130	000402		BR	12\$:GO UPDATE R1 FOR NEXT READ
8917	065132	005237	001172	11\$: INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
8918	065136	062701	000002	12\$: ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
8919	065142	077232		SOB	R2,10\$:BRANCH TO 10\$ IF ALL PDR'S NOT READ
8920	065144	012702	000020	MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
8921	065150	012701	177600	MOV	#UIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
8922	065154	011100		13\$: MOV	(R1),R0	:READ PDR INTO R0

8923	065156	022700	077705			CMP	#77705,R0	;SEE IF THIS WAS THE PDR WITH A&W BITS ON
8924	065162	001023				BNE	15\$;BRANCH IF THIS IS NOT THE ONE
8925	065164	020103				CMP	R1,R3	;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
8926	065166	001417				BEQ	14\$;BRANCH IF ADDRESS IS CORRECT
8927	065170	104112				ERROR	112	;A & W BITS GOT SET IN WRONG PDR
8928	065172	012700	077405			MOV	#77405,R0	;RE-SET PAGES MODIFIED BY ERROR
8929	065176	010037	172300			MOV	R0,KIPDR0	;RELOAD KERNEL PDR0
8930	065202	010037	172302			MOV	R0,KIPDR1	;RELOAD KERNEL PDR1
8931	065206	010037	172316			MOV	R0,KIPDR7	;RELOAD KERNEL PDR7
8932	065212	010037	172200			MOV	R0,SIPDR0	;RELOAD PAGE 0 OF PRESENT SPACE
8933	065216	010037	172216			MOV	R0,SIPDR7	;RE-LOAD I/O PAGE PDR IF ERROR
8934	065222	011515				MOV	(R5),(R5)	;TRY WRITE AGAIN, IN CASE YOU
8935								;WERE TESTING PAGE SEVEN
8936	065224	000402				BR	15\$;GO UPDATE R1 FOR NEXT READ
8937	065226	005237	001172	14\$:		INC	\$TMP0	;SET FLAG SINCE ADDRESSES MATCHED
8938	065232	062701	000002	15\$:		ADD	#2,R1	;POINT TO NEXT PDR TO BE READ
8939	065236	077232				SOB	R2,13\$;BRANCH TO 13\$ IF ALL PDR'S NOT READ
8940	065240	005737	001172			TST	\$TMP0	;SEE IF THERE WAS A CORRECT PDR
8941	065244	001002				BNE	16\$;BRANCH IF THERE WAS
8942	065246	011300				MOV	(R3),R0	;SAVE CONTENTS OF PDR UNDER TEST
8943	065250	104113				ERROR	113	;NO PDR ADDRESSES MATCHED
8944	065252	062703	000002	16\$:		ADD	#2,R3	;POINT TO NEXT PDR UNDER TEST
8945	065256	062705	020000			ADD	#20000,R5	;CHANGE PAGE NUMBER IN VIRT. ADDR.
8946	065262	005304				DEC	R4	;DECREMENT COUNTER
8947	065264	001402				BEQ	17\$;BRANCH IF COUNTER IS ZERO
8948	065266	000137	064600			JMP	19\$;JUMP TO LOAD PDR'S AGAIN
8949	065272			17\$:				
8950	065272	012737	000340	177776		MOV	#340,PSW	;RETURN TO KERNEL MODE, PRIORITY 7
8951	065300	012737	064550	001112		MOV	#20\$,\$LPERR	;SET LOOP POINTER TO START OF TEST

```

8952
8953
8954
8955
8956
8957
8958
8959
8960
8961
8962
8963
8964
8965
8966
8967
8968
8969
8970
8971
8972
8973
8974
8975
8976
8977
8978

```

*TEST 102 DUAL MAPPING USER MODE I-SPACE

THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
(4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
SINCE BIT09 OF MMRO IS CLEAR) AND SET BOTH THE A & W BITS IN
THE USER I-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.

```

TST102:
8969 065306
8970 065306 000004
8971 065310 012737 066054 001316
8972
8973 065316 012737 065466 001112 20$:
8974 065324 012737 140000 177776
8975 065332 012703 177600
8976
8977 065336 012704 000010
8978 065342 012705 010200

```

SCOPE
MOV #TST103,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
MOV #21\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 21\$
MOV #140000,PSW ;GO TO USER MODE
MOV #UIPDR0,R3 ;LOAD FIRST ADDRESS OF USER PDR'S
;THAT WILL BE TESTED IN THIS TEST
MOV #10,R4 ;TEST THE NEXT EIGHT PDR'S
MOV #10200,R5 ;LOAD STARTING VIRTUAL ADDRESS INTO R5

8979	065346	012700	077405	19\$:	MOV	#77405,R0	:ALL PAGES WILL BE TRAP ON WRITE
8980	065352	005037	001172		CLR	\$TMP0	:CLEAR CORRECT PDR SET INDICATOR
8981	065356	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8982	065362	012701	172220		MOV	#SDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8983	065366	010021		1\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8984	065370	077202			SOB	R2,1\$:BRANCH BACK TO 1\$ IF R2 IS NOT ZERO
8985	065372	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8986	065376	012701	172200		MOV	#SIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8987	065402	010021		2\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8988	065404	077202			SOB	R2,2\$:BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
8989	065406	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8990	065412	012701	172320		MOV	#KDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8991	065416	010021		3\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8992	065420	077202			SOB	R2,3\$:BRANCH BACK TO 3\$ IF R2 IS NOT ZERO
8993	065422	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8994	065426	012701	172300		MOV	#KIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8995	065432	010021		4\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
8996	065434	077202			SOB	R2,4\$:BRANCH BACK TO 4\$ IF R2 IS NOT ZERO
8997	065436	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
8998	065442	012701	177620		MOV	#UDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
8999	065446	010021		5\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9000	065450	077202			SOB	R2,5\$:BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
9001	065452	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9002	065456	012701	177600		MOV	#UIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9003	065462	010021		6\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9004	065464	077202			SOB	R2,6\$:BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
9005	065466	012700	077405	21\$:	MOV	#77405,R0	:MUST RE-INIT THESE PDRS IF ERROR
9006	065472	010037	172300		MOV	R0,KIPDR0	:LOAD KERNEL PDR0
9007	065476	010037	172302		MOV	R0,KIPDR1	:LOAD KERNEL PDR1
9008	065502	010037	172316		MOV	R0,KIPDR7	:LOAD KERNEL PDR7
9009	065506	010037	177600		MOV	R0,UIPDR0	:LOAD PDR0 OF PRESENT SPACE
9010	065512	010037	177616		MOV	R0,UIPDR7	:LOAD PDR7 OF PRESENT SPACE
9011	065516	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
9012	065520	011515			MOV	(R5),(R5)	:WRITE INTO PAGE UNDER TEST
9013	065522	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9014	065526	012701	172300		MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9015	065532	011100		7\$:	MOV	(R1),R0	:READ PDR INTO R0
9016	065534	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9017	065540	001023			BNE	9\$:BRANCH IF THIS IS NOT THE ONE
9018	065542	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9019	065544	001417			BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
9020	065546	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9021	065550	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9022	065554	010037	172300		MOV	R0,KIPDR0	:RELOAD KERNEL PDR0
9023	065560	010037	172302		MOV	R0,KIPDR1	:RELOAD KERNEL PDR1
9024	065564	010037	172316		MOV	R0,KIPDR7	:RELOAD KERNEL PDR7
9025	065570	010037	177600		MOV	R0,UIPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9026	065574	010037	177616		MOV	R0,UIPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9027	065600	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9028							:WERE TESTING PAGE SEVEN
9029	065602	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
9030	065604	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9031	065610	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9032	065614	077232			SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
9033	065616	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9034	065622	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR

```
9035 065626 011100          10$: MOV      (R1),R0      ;READ PDR INTO R0
9036 065630 022700 077705  CMP      #77705,R0    ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
9037 065634 001023          BNE      12$          ;BRANCH IF THIS IS NOT THE ONE
9038 065636 020103          CMP      R1,R3        ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9039 065640 001417          BEQ      11$          ;BRANCH IF ADDRESS IS CORRECT
9040 065642 104112          ERROR    112         ;A & W BITS GOT SET IN WRONG PDR
9041 065644 012700 077405  MOV      #77405,R0    ;RE-SET PAGES MODIFIED BY ERROR
9042 065650 010037 172300  MOV      R0,KIPDR0    ;RELOAD KERNEL PDR0
9043 065654 010037 172302  MOV      R0,KIPDR1    ;RELOAD KERNEL PDR1
9044 065660 010037 172316  MOV      R0,KIPDR7    ;RELOAD KERNEL PDR7
9045 065664 010037 177600  MOV      R0,UIPDR0    ;RELOAD PAGE 0 OF PRESENT SPACE
9046 065670 010037 177616  MOV      R0,UIPDR7    ;RE-LOAD I/O PAGE PDR IF ERROR
9047 065674 011515          MOV      (R5),(R5)    ;TRY WRITE AGAIN, IN CASE YOU
9048                                ;WERE TESTING PAGE SEVEN
9049 065676 000402          BR       12$          ;GO UPDATE R1 FOR NEXT READ
9050 065700 005237 001172  11$: INC      $TMP0     ;SET FLAG SINCE ADDRESSES MATCHED
9051 065704 062701 000002  12$: ADD      #2,R1     ;POINT TO NEXT PDR TO BE READ
9052 065710 077232          SOB      R2,#0$       ;BRANCH TO 10$ IF ALL PDR'S NOT READ
9053 065712 012702 000020  MUV      #20,R2       ;SET COUNTER TO READ NEXT 20 REGISTERS
9054 065716 012701 177600  MOV      #UIPDR0,R1   ;LOAD ADDRESS OF BEGINNING PDR
9055 065722 011100          13$: MOV      (R1),R0    ;READ PDR INTO R0
9056 065724 022700 077705  CMP      #77705,R0    ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
9057 065730 001023          BNE      15$          ;BRANCH IF THIS IS NOT THE ONE
9058 065732 020103          CMP      R1,R3        ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9059 065734 001417          BEQ      14$          ;BRANCH IF ADDRESS IS CORRECT
9060 065736 104112          ERROR    112         ;A & W BITS GOT SET IN WRONG PDR
9061 065740 012700 077405  MOV      #77405,R0    ;RE-SET PAGES MODIFIED BY ERROR
9062 065744 010037 172300  MOV      R0,KIPDR0    ;RELOAD KERNEL PDR0
9063 065750 010037 172302  MOV      R0,KIPDR1    ;RELOAD KERNEL PDR1
9064 065754 010037 172316  MOV      R0,KIPDR7    ;RELOAD KERNEL PDR7
9065 065760 010037 177600  MOV      R0,UIPDR0    ;RELOAD PAGE 0 OF PRESENT SPACE
9066 065764 010037 177616  MOV      R0,UIPDR7    ;RE-LOAD I/O PAGE PDR IF ERROR
9067 065770 011515          MOV      (R5),(R5)    ;TRY WRITE AGAIN, IN CASE YOU
9068                                ;WERE TESTING PAGE SEVEN
9069 065772 000402          BR       15$          ;GO UPDATE R1 FOR NEXT READ
9070 065774 005237 001172  14$: INC      $TMP0     ;SET FLAG SINCE ADDRESSES MATCHED
9071 066000 062701 000002  15$: ADD      #2,R1     ;POINT TO NEXT PDR TO BE READ
9072 066004 077232          SOB      R2,#13$      ;BRANCH TO 13$ IF ALL PDR'S NOT READ
9073 066006 005737 001172  TST      $TMP0        ;SEE IF THERE WAS A CORRECT PDR
9074 066012 001002          BNE      16$          ;BRANCH IF THERE WAS
9075 066014 011300          MOV      (R3),R0      ;SAVE CONTENTS OF PDR UNDER TEST
9076 066016 104113          ERROR    113         ;NO PDR ADDRESSES MATCHED
9077 066020 062703 000002  16$: ADD      #2,R3     ;POINT TO NEXT PDR UNDER TEST
9078 066024 062705 020000  ADD      #20000,R5    ;CHANGE PAGE NUMBER IN VIRT. ADDR.
9079 066030 005304          DEC      R4           ;DECREMENT COUNTER
9080 066032 001402          BEQ      17$          ;BRANCH IF COUNTER IS ZERO
9081 066034 000137 065346  JMP      19$          ;JUMP TO LOAD PDR'S AGAIN
9082 066040          17$:
9083 066040 012737 000340 177776 MOV      #340,PSW     ;RETURN TO KERNEL MODE, PRIORITY 7
9084 066046 012737 065316 001112 MOV      #20$,$LPERR  ;SET LOOP POINTER TO START OF TEST
9085
9086
9087
9088
9089
9090
```

```
::*****
:*TEST 103      DUAL MAPPING KERNEL MODE D-SPACE
::
::
::      THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
```



```

9091 :: (4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
9092 :: 010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
9093 :: SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
9094 :: SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
9095 :: THE KERNEL D-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
9096 :: P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
9097 :: ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
9098 :: WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
9099 :: ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
9100 ::
9101 ::

```

TST103:

9102	066054					SCOPE		
9103	066054	000004				MOV	#TST104,NXTTST	;SAVE STARTING ADDRESS OF NEXT
9104	066056	012737	066636	001316				;TEST FOR ESCAPE ON PARITY ERRORS
9105								;SET LOOP ON ERROR POINTER TO 21\$
9106	066064	012737	066242	001112	20\$:	MOV	#21\$, \$LPERR	;GO TO KERNEL MODE
9107	066072	012737	000000	177776		MOV	#00000,PSW	;ENABLE ALL D-SPACE MAPPING
9108	066100	052737	000007	172516		BIS	#7,MMR3	;LOAD FIRST ADDRESS OF KERNEL PDR'S
9109	066106	012703	172320			MOV	#KDPDR0,R3	;THAT WILL BE TESTED IN THIS TEST
9110								;TEST THE NEXT EIGHT PDR'S
9111	066112	012704	000010			MOV	#10,R4	;LOAD STARTING VIRTUAL ADDRESS INTO R5
9112	066116	012705	010200			MOV	#10200,R5	;ALL PAGES WILL BE TRAP ON WRITE
9113	066122	012700	077405		19\$:	MOV	#77405,R0	;CLEAR CORRECT PDR SET INDICATOR
9114	066126	005037	001172			CLR	\$TMP0	;SET COUNT TO LOAD 8 ADDRESSES
9115	066132	012702	000010			MOV	#10,R2	;PUT ADDRESS OF FIRST PDR IN R1
9116	066136	012701	177600			MOV	#UIPDR0,R1	;LOAD R0 INTO PDR ADDRESSED BY R1
9117	066142	010021			1\$:	MOV	R0,(R1)+	;BRANCH BACK TO 1\$ IF R2 IS NOT ZERO
9118	066144	077202				SOB	R2,1\$;SET COUNT TO LOAD 8 ADDRESSES
9119	066146	012702	000010			MOV	#10,R2	;PUT ADDRESS OF FIRST PDR IN R1
9120	066152	012701	177620			MOV	#UDPDR0,R1	;LOAD R0 INTO PDR ADDRESSED BY R1
9121	066156	010021			2\$:	MOV	R0,(R1)+	;BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
9122	066160	077202				SOB	R2,2\$;SET COUNT TO LOAD 8 ADDRESSES
9123	066162	012702	000010			MOV	#10,R2	;PUT ADDRESS OF FIRST PDR IN R1
9124	066166	012701	172200			MOV	#SIPDR0,R1	;LOAD R0 INTO PDR ADDRESSED BY R1
9125	066172	010021			3\$:	MOV	R0,(R1)+	;BRANCH BACK TO 3\$ IF R2 IS NOT ZERO
9126	066174	077202				SOB	R2,3\$;SET COUNT TO LOAD 8 ADDRESSES
9127	066176	012702	000010			MOV	#10,R2	;PUT ADDRESS OF FIRST PDR IN R1
9128	066202	012701	172220			MOV	#SDPDR0,R1	;LOAD R0 INTO PDR ADDRESSED BY R1
9129	066206	010021			4\$:	MOV	R0,(R1)+	;BRANCH BACK TO 4\$ IF R2 IS NOT ZERO
9130	066210	077202				SOB	R2,4\$;SET COUNT TO LOAD 8 ADDRESSES
9131	066212	012702	000010			MOV	#10,R2	;PUT ADDRESS OF FIRST PDR IN R1
9132	066216	012701	172300			MOV	#KIPDR0,R1	;LOAD R0 INTO PDR ADDRESSED BY R1
9133	066222	010021			5\$:	MOV	R0,(R1)+	;BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
9134	066224	077202				SOB	R2,5\$;SET COUNT TO LOAD 8 ADDRESSES
9135	066226	012702	000010			MOV	#10,R2	;PUT ADDRESS OF FIRST PDR IN R1
9136	066232	012701	172320			MOV	#KDPDR0,R1	;LOAD R0 INTO PDR ADDRESSED BY R1
9137	066236	010021			6\$:	MOV	R0,(R1)+	;BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
9138	066240	077202				SOB	R2,6\$;MUST RE-INIT THESE PDRS IF ERROR
9139	066242	012700	077405		21\$:	MOV	#77405,R0	;LOAD KERNEL PDR0
9140	066246	010037	172320			MOV	R0,KDPDR0	;LOAD KERNEL PDR1
9141	066252	010037	172322			MOV	R0,KDPDR1	;LOAD KERNEL PDR7
9142	066256	010037	172336			MOV	R0,KDPDR7	;LOAD PDR0 OF PRESENT SPACE
9143	066262	010037	172320			MOV	R0,KDPDR0	;LOAD PDR7 OF PRESENT SPACE
9144	066266	010037	172336			MOV	R0,KDPDR7	;THIS IS A SYNC POINT FOR SCOPING
9145	066272	000240				NOP		;WRITE INTO PAGE UNDER TEST
9146	066274	011515				MOV	(R5),(R5)	

9147	066276	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9148	066302	012701	172300		MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9149	066306	011100		7\$:	MOV	(R1),R0	:READ PDR INTO R0
9150	066310	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9151	066314	001023			BNE	9\$:BRANCH IF THIS IS NOT THE ONE
9152	066316	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9153	066320	001417			BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
9154	066322	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9155	066324	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9156	066330	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9157	066334	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9158	066340	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9159	066344	010037	172320		MOV	R0,KDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9160	066350	010037	172336		MOV	R0,KDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9161	066354	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9162							:WERE TESTING PAGE SEVEN
9163	066356	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
9164	066360	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9165	066364	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9166	066370	077232			SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
9167	066372	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9168	066376	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9169	066402	011100		10\$:	MOV	(R1),R0	:READ PDR INTO R0
9170	066404	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9171	066410	001023			BNE	12\$:BRANCH IF THIS IS NOT THE ONE
9172	066412	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9173	066414	001417			BEQ	11\$:BRANCH IF ADDRESS IS CORRECT
9174	066416	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9175	066420	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9176	066424	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9177	066430	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9178	066434	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9179	066440	010037	172320		MOV	R0,KDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9180	066444	010037	172336		MOV	R0,KDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9181	066450	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9182							:WERE TESTING PAGE SEVEN
9183	066452	000402			BR	12\$:GO UPDATE R1 FOR NEXT READ
9184	066454	005237	001172	11\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9185	066460	062701	000002	12\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9186	066464	077232			SOB	R2,10\$:BRANCH TO 10\$ IF ALL PDR'S NOT READ
9187	066466	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9188	066472	012701	177600		MOV	#UIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9189	066476	011100		13\$:	MOV	(R1),R0	:READ PDR INTO R0
9190	066500	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9191	066504	001023			BNE	15\$:BRANCH IF THIS IS NOT THE ONE
9192	066506	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9193	066510	001417			BEQ	14\$:BRANCH IF ADDRESS IS CORRECT
9194	066512	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9195	066514	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9196	066520	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9197	066524	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9198	066530	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9199	066534	010037	172320		MOV	R0,KDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9200	066540	010037	172336		MOV	R0,KDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9201	066544	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9202							:WERE TESTING PAGE SEVEN

K 15

9203	066546	000402			BR	15\$:GO UPDATE R1 FOR NEXT READ
9204	066550	005237	001172	14\$:	INC	\$TMP0		:SET FLAG SINCE ADDRESSES MATCHED
9205	066554	062701	000002	15\$:	ADD	#2,R1		:POINT TO NEXT PDR TO BE READ
9206	066560	077232			SOB	R2,13\$:BRANCH TO 13\$ IF ALL PDR'S NOT READ
9207	066562	005737	001172		TST	\$TMP0		:SEE IF THERE WAS A CORRECT PDR
9208	066566	001002			BNE	16\$:BRANCH IF THERE WAS
9209	066570	011300			MOV	(R3),R0		:SAVE CONTENTS OF PDR UNDER TEST
9210	066572	104113			ERROR	113		:NO PDR ADDRESSES MATCHED
9211	066574	062703	000002	16\$:	ADD	#2,R3		:POINT TO NEXT PDR UNDER TEST
9212	066600	062705	020000		ADD	#20000,R5		:CHANGE PAGE NUMBER IN VIRT. ADDR.
9213	066604	005304			DEC	R4		:DECREMENT COUNTER
9214	066606	001402			BEQ	17\$:BRANCH IF COUNTER IS ZERO
9215	066610	000137	066122	17\$:	JMP	19\$:JUMP TO LOAD PDR'S AGAIN
9216	066614							
9217	066614	042737	000007	172516	BIC	#7,MMR3		:DISABLE ALL D-SPACE MAPPING
9218	066622	012737	000340	177776	MOV	#340,PSW		:RETURN TO KERNEL MODE, PRIORITY 7
9219	066630	012737	066064	001112	MOV	#20\$,SLPERR		:SET LOOP POINTER TO START OF TEST
9220								

```

9221
9222
9223
9224
9225
9226
9227
9228
9229
9230
9231
9232
9233
9234
9235
9236
9237 066636
9238 066636 000004
9239 066640 012737 067420 001316
9240
9241 066646 012737 067024 001112 20$:
9242 066654 012737 040000 177776
9243 066662 052737 000007 172516
9244 066670 012703 172220
9245
9246 066674 012704 000010
9247 066700 012705 010200
9248 066704 012700 077405 19$:
9249 066710 005037 001172
9250 066714 012702 000010
9251 066720 012701 172300
9252 066724 010021 1$:
9253 066726 077202
9254 066730 012702 000010
9255 066734 012701 172320
9256 066740 010021 2$:
9257 066742 077202
9258 066744 012702 000010
9259 066750 012701 177600
9260 066754 010021 3$:
9261 066756 077202
9262 066760 012702 000010
9263 066764 012701 177620
9264 066770 010021 4$:
9265 066772 077202
9266 066774 012702 000010
9267 067000 012701 172200
9268 067004 010021 5$:
9269 067006 077202
9270 067010 012702 000010
9271 067014 012701 172220
9272 067020 010021 6$:
9273 067022 077202
9274 067024 012700 077405 21$:
9275 067030 010037 172320
9276 067034 010037 172322

```

```

*****
*TEST 104      DUAL MAPPING SUPERVISOR MODE D-SPACE
*****
THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
(4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
THE SUPERVISOR D-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.
*****

```

```

TST104:
SCOPE
MOV #TST105,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
;SET LOOP ON ERROR POINTER TO 21$
MOV #21$,SLPERR
MOV #40000,PSW ;GO TO SUPERVISOR MODE
BIS #7,MMR3 ;ENABLE ALL D-SPACE MAPPING
MOV #SDPDR0,R3 ;LOAD FIRST ADDRESS OF SUPERVISOR PDR'S
;THAT WILL BE TESTED IN THIS TEST
MOV #10,R4 ;TEST THE NEXT EIGHT PDR'S
MOV #10200,R5 ;LOAD STARTING VIRTUAL ADDRESS INTO R5
MOV #77405,R0 19$: ;ALL PAGES WILL BE TRAP ON WRITE
CLR $TMP0 ;CLEAR CORRECT PDR SET INDICATOR
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #KIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
MOV R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,1$ ;BRANCH BACK TO 1$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #KDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
MOV R0,(R1)+ 2$: ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,2$ ;BRANCH BACK TO 2$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #UIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
MOV R0,(R1)+ 3$: ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,3$ ;BRANCH BACK TO 3$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #UDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
MOV R0,(R1)+ 4$: ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,4$ ;BRANCH BACK TO 4$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #SIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
MOV R0,(R1)+ 5$: ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,5$ ;BRANCH BACK TO 5$ IF R2 IS NOT ZERO
MOV #10,R2 ;SET COUNT TO LOAD 8 ADDRESSES
MOV #SDPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
MOV R0,(R1)+ 6$: ;LOAD R0 INTO PDR ADDRESSED BY R1
SOB R2,6$ ;BRANCH BACK TO 6$ IF R2 IS NOT ZERO
MOV #77405,R0 21$: ;MUST RE-INIT THESE PDRS IF ERROR
MOV R0,KDPDR0 ;LOAD KERNEL PDR0
MOV R0,KDPDR1 ;LOAD KERNEL PDR1

```

9277	067040	010037	172336		MOV	R0,KDPDR7	:LOAD KERNEL PDR7
9278	067044	010037	172220		MOV	R0,SDPDR0	:LOAD PDR0 OF PRESENT SPACE
9279	067050	010037	172236		MOV	R0,SDPDR7	:LOAD PDR7 OF PRESENT SPACE
9280	067054	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
9281	067056	011515			MOV	(R5),(R5)	:WRITE INTO PAGE UNDER TEST
9282	067060	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9283	067064	012701	172300		MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9284	067070	011100		7\$:	MOV	(R1),R0	:READ PDR INTO R0
9285	067072	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9286	067076	001023			BNE	9\$:BRANCH IF THIS IS NOT THE ONE
9287	067100	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9288	067102	001417			BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
9289	067104	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9290	067106	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9291	067112	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9292	067116	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9293	067122	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9294	067126	010037	172220		MOV	R0,SDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9295	067132	010037	172236		MOV	R0,SDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9296	067136	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9297							:WERE TESTING PAGE SEVEN
9298	067140	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
9299	067142	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9300	067146	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9301	067152	077232			SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
9302	067154	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9303	067160	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9304	067164	011100		10\$:	MOV	(R1),R0	:READ PDR INTO R0
9305	067166	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9306	067172	001023			BNE	12\$:BRANCH IF THIS IS NOT THE ONE
9307	067174	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9308	067176	001417			BEQ	11\$:BRANCH IF ADDRESS IS CORRECT
9309	067200	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9310	067202	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9311	067206	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9312	067212	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9313	067216	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9314	067222	010037	172220		MOV	R0,SDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9315	067226	010037	172236		MOV	R0,SDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9316	067232	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9317							:WERE TESTING PAGE SEVEN
9318	067234	000402			BR	12\$:GO UPDATE R1 FOR NEXT READ
9319	067236	005237	001172	11\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9320	067242	062701	000002	12\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9321	067246	077232			SOB	R2,10\$:BRANCH TO 10\$ IF ALL PDR'S NOT READ
9322	067250	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9323	067254	012701	177600		MOV	#UIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9324	067260	011100		13\$:	MOV	(R1),R0	:READ PDR INTO R0
9325	067262	022700	077705		CMP	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9326	067266	001023			BNE	15\$:BRANCH IF THIS IS NOT THE ONE
9327	067270	020103			CMP	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9328	067272	001417			BEQ	14\$:BRANCH IF ADDRESS IS CORRECT
9329	067274	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9330	067276	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9331	067302	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9332	067306	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1

9333	067312	010037	172336		MOV	R0,KDPDR7	;RELOAD KERNEL PDR7
9334	067316	010037	172220		MOV	R0,SDPDR0	;RELOAD PAGE 0 OF PRESENT SPACE
9335	067322	010037	172236		MOV	R0,SDPDR7	;RE-LOAD I/O PAGE PDR IF ERROR
9336	067326	011515			MOV	(R5),(R5)	;TRY WRITE AGAIN, IN CASE YOU
9337							;WERE TESTING PAGE SEVEN
9338	067330	000402			BR	15\$;GO UPDATE R1 FOR NEXT READ
9339	067332	005237	001172	14\$:	INC	\$TMP0	;SET FLAG SINCE ADDRESSES MATCHED
9340	067336	062701	000002	15\$:	ADD	#2,R1	;POINT TO NEXT PDR TO BE READ
9341	067342	077232			SOB	R2,13\$;BRANCH TO 13\$ IF ALL PDR'S NOT READ
9342	067344	005737	001172		TST	\$TMP0	;SEE IF THERE WAS A CORRECT PDR
9343	067350	001002			BNE	16\$;BRANCH IF THERE WAS
9344	067352	011300			MOV	(R3),R0	;SAVE CONTENTS OF PDR UNDER TEST
9345	067354	104113			ERROR	113	;NO PDR ADDRESSES MATCHED
9346	067356	062703	000002	16\$:	ADD	#2,R3	;POINT TO NEXT PDR UNDER TEST
9347	067362	062705	020000		ADD	#20000,R5	;CHANGE PAGE NUMBER IN VIRT. ADDR.
9348	067366	005304			DEC	R4	;DECREMENT COUNTER
9349	067370	001402			BEQ	17\$;BRANCH IF COUNTER IS ZERO
9350	067372	000137	066704		JMP	19\$;JUMP TO LOAD PDR'S AGAIN
9351	067376			17\$:			
9352	067376	042737	000007	172516	BIC	#7,MMR3	;DISABLE ALL D-SPACE MAPPING
9353	067404	012737	000340	177776	MOV	#340,PSW	;RETURN TO KERNEL MODE, PRIORITY 7
9354	067412	012737	066646	001112	MOV	#20\$,SLPERR	;SET LOOP POINTER TO START OF TEST

```

9355
9356
9357
9358
9359
9360
9361
9362
9363
9364
9365
9366
9367
9368
9369
9370
9371
9372
9373
9374
9375
9376
9377
9378
9379
9380
9381
9382
9383
9384
9385
9386
9387
9388

```

*TEST 105 DUAL MAPPING USER MODE D-SPACE

THIS TEST STARTS BY LOADING ALL THE P.D.R.'S WITH 77405
(4K PAGE, TRAP ON WRITE). THEN THE VIRTUAL ADDRESS IS SET TO
010200 AND THAT WORD IS WRITTEN INTO ITSELF. THIS WRITE WILL
SATISFY THE TRAP CONDITION OF THE A.C.F. (BUT IT WILL NOT TRAP
SINCE BIT09 OF MMR0 IS CLEAR) AND SET BOTH THE A & W BITS IN
THE USER D-SPACE P.D.R. UNDER TEST. NOW ALL OF THE
P.D.R.'S ARE COMPARED WITH 77705 AND ANY THAT MATCH, EXCEPT THE
ONE THAT IS UNDER TEST, ARE REPORTED AS DUAL MAPPING ERRORS.
WHEN THE I/O PAGE IS REACHED THE VIRTUAL ADDRESS GENERATES THE
ADDRESS OF 'MAPLOO' (17770200) WHICH SHOULD ALWAYS EXIST.

```

9371
9372
9373
9374
9375
9376
9377
9378
9379
9380
9381
9382
9383
9384
9385
9386
9387
9388

```

TST105:

9372	067420				SCOPE		
9373	067420	000004			MOV	#TST106,NXTTST	;SAVE STARTING ADDRESS OF NEXT
9374	067422	012737	070202	001316			;TEST FOR ESCAPE ON PARITY ERRORS
9375							;SET LOOP ON ERROR POINTER TO 21\$
9376	067430	012737	067606	001112	20\$:	MOV	#21\$,SLPERR
9377	067436	012737	140000	177776		MOV	#140000,PSW
9378	067444	052737	000007	172516		BIS	#7,MMR3
9379	067452	012703	177620			MOV	#UDPDR0,R3
9380							;ENABLE ALL D-SPACE MAPPING
9381	067456	012704	000010			MOV	#10,R4
9382	067462	012705	010200			MOV	#10200,R5
9383	067466	012700	077405		19\$:	MOV	#77405,R0
9384	067472	005037	001172			CLR	\$TMP0
9385	067476	012702	000010			MOV	#10,R2
9386	067502	012701	172200			MOV	#SIPDR0,R1
9387	067506	010021			1\$:	MOV	R0,(R1)+
9388	067510	077202				SOB	R2,1\$

;LOAD FIRST ADDRESS OF USER PDR'S
;THAT WILL BE TESTED IN THIS TEST
;TEST THE NEXT EIGHT PDR'S
;LOAD STARTING VIRTUAL ADDRESS INTO R5
;ALL PAGES WILL BE TRAP ON WRITE
;CLEAR CORRECT PDR SET INDICATOR
;SET COUNT TO LOAD 8 ADDRESSES
;PUT ADDRESS OF FIRST PDR IN R1
;LOAD R0 INTO PDR ADDRESSED BY R1
;BRANCH BACK TO 1\$ IF R2 IS NOT ZERO

9389	067512	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9390	067516	012701	172220		MOV	#SDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9391	067522	010021		2\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9392	067524	077202			SOB	R2,2\$:BRANCH BACK TO 2\$ IF R2 IS NOT ZERO
9393	067526	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9394	067532	012701	172300		MOV	#KIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9395	067536	010021		3\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9396	067540	077202			SOB	R2,3\$:BRANCH BACK TO 3\$ IF R2 IS NOT ZERO
9397	067542	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9398	067546	012701	172320		MOV	#KDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9399	067552	010021		4\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9400	067554	077202			SOB	R2,4\$:BRANCH BACK TO 4\$ IF R2 IS NOT ZERO
9401	067556	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9402	067562	012701	177600		MOV	#UIPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9403	067566	010021		5\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9404	067570	077202			SOB	R2,5\$:BRANCH BACK TO 5\$ IF R2 IS NOT ZERO
9405	067572	012702	000010		MOV	#10,R2	:SET COUNT TO LOAD 8 ADDRESSES
9406	067576	012701	177620		MOV	#UDPDR0,R1	:PUT ADDRESS OF FIRST PDR IN R1
9407	067602	010021		6\$:	MOV	R0,(R1)+	:LOAD R0 INTO PDR ADDRESSED BY R1
9408	067604	077202			SOB	R2,6\$:BRANCH BACK TO 6\$ IF R2 IS NOT ZERO
9409	067606	012700	077405	21\$:	MOV	#77405,R0	:MUST RE-INIT THESE PDRS IF ERROR
9410	067612	010037	172320		MOV	R0,KDPDR0	:LOAD KERNEL PDR0
9411	067616	010037	172322		MOV	R0,KDPDR1	:LOAD KERNEL PDR1
9412	067622	010037	172336		MOV	R0,KDPDR7	:LOAD KERNEL PDR7
9413	067626	010037	177620		MOV	R0,UDPDR0	:LOAD PDR0 OF PRESENT SPACE
9414	067632	010037	177636		MOV	R0,UDPDR7	:LOAD PDR7 OF PRESENT SPACE
9415	067636	000240			NOP		:THIS IS A SYNC POINT FOR SCOPING
9416	067640	011515			MOV	(R5),(R5)	:WRITE INTO PAGE UNDER TEST
9417	067642	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9418	067646	012701	172300		MOV	#KIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9419	067652	011100		7\$:	MOV	(R1),R0	:READ PDR INTO R0
9420	067654	022700	077705		CMF	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9421	067660	001023			BNE	9\$:BRANCH IF THIS IS NOT THE ONE
9422	067662	020103			CMF	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9423	067664	001417			BEQ	8\$:BRANCH IF ADDRESS IS CORRECT
9424	067666	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR
9425	067670	012700	077405		MOV	#77405,R0	:RE-SET PAGES MODIFIED BY ERROR
9426	067674	010037	172320		MOV	R0,KDPDR0	:RELOAD KERNEL PDR0
9427	067700	010037	172322		MOV	R0,KDPDR1	:RELOAD KERNEL PDR1
9428	067704	010037	172336		MOV	R0,KDPDR7	:RELOAD KERNEL PDR7
9429	067710	010037	177620		MOV	R0,UDPDR0	:RELOAD PAGE 0 OF PRESENT SPACE
9430	067714	010037	177636		MOV	R0,UDPDR7	:RE-LOAD I/O PAGE PDR IF ERROR
9431	067720	011515			MOV	(R5),(R5)	:TRY WRITE AGAIN, IN CASE YOU
9432							:WERE TESTING PAGE SEVEN
9433	067722	000402			BR	9\$:GO UPDATE R1 FOR NEXT READ
9434	067724	005237	001172	8\$:	INC	\$TMP0	:SET FLAG SINCE ADDRESSES MATCHED
9435	067730	062701	000002	9\$:	ADD	#2,R1	:POINT TO NEXT PDR TO BE READ
9436	067734	077232			SOB	R2,7\$:BRANCH TO 7\$ IF ALL PDR'S NOT READ
9437	067736	012702	000020		MOV	#20,R2	:SET COUNTER TO READ NEXT 20 REGISTERS
9438	067742	012701	172200		MOV	#SIPDR0,R1	:LOAD ADDRESS OF BEGINNING PDR
9439	067746	011100		10\$:	MOV	(R1),R0	:READ PDR INTO R0
9440	067750	022700	077705		CMF	#77705,R0	:SEE IF THIS WAS THE PDR WITH A&W BITS ON
9441	067754	001023			BNE	12\$:BRANCH IF THIS IS NOT THE ONE
9442	067756	020103			CMF	R1,R3	:SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9443	067760	001417			BEQ	11\$:BRANCH IF ADDRESS IS CORRECT
9444	067762	104112			ERROR	112	:A & W BITS GOT SET IN WRONG PDR

```
9445 067764 012700 077405      MOV      #77405,R0      ;RE-SET PAGES MODIFIED BY ERROR
9446 067770 010037 172320      MOV      R0,KDPDR0     ;RELOAD KERNEL PDR0
9447 067774 010037 172322      MOV      R0,KDPDR1     ;RELOAD KERNEL PDR1
9448 070000 010037 172336      MOV      R0,KDPDR7     ;RELOAD KERNEL PDR7
9449 070004 010037 177620      MOV      R0,UDPDR0     ;RELOAD PAGE 0 OF PRESENT SPACE
9450 070010 010037 177636      MOV      R0,UDPDR7     ;RE-LOAD I/O PAGE PDR IF ERROR
9451 070014 011515              MOV      (R5),(R5)     ;TRY WRITE AGAIN, IN CASE YOU
9452                          ;WERE TESTING PAGE SEVEN
9453 070016 000402              BR       12$           ;GO UPDATE R1 FOR NEXT READ
9454 070020 005237 001172      11$: INC      $TMP0     ;SET FLAG SINCE ADDRESSES MATCHED
9455 070024 062701 000002      12$: ADD      #2,R1     ;POINT TO NEXT PDR TO BE READ
9456 070030 077232              SOB      R2,10$       ;BRANCH TO 10$ IF ALL PDR'S NOT READ
9457 070032 012702 000020      MOV      #20,R2       ;SET COUNTER TO READ NEXT 20 REGISTERS
9458 070036 012701 177600      MOV      #UIPDR0,R1   ;LOAD ADDRESS OF BEGINNING PDR
9459 070042 011100      13$: MOV      (R1),R0    ;READ PDR INTO R0
9460 070044 022700 077705      CMP      #77705,R0    ;SEE IF THIS WAS THE PDR WITH A&W BITS ON
9461 070050 001023              BNE     15$           ;BRANCH IF THIS IS NOT THE ONE
9462 070052 020103              CMP      R1,R3        ;SEE IF THE ADDRESS MATCHES PDR UNDER TEST
9463 070054 001417              BEQ     14$           ;BRANCH IF ADDRESS IS CORRECT
9464 070056 104112              ERROR   112          ;A & W BITS GOT SET IN WRONG PDR
9465 070060 012700 077405      MOV      #77405,R0    ;RE-SET PAGES MODIFIED BY ERROR
9466 070064 010037 172320      MOV      R0,KDPDR0     ;RELOAD KERNEL PDR0
9467 070070 010037 172322      MOV      R0,KDPDR1     ;RELOAD KERNEL PDR1
9468 070074 010037 172336      MOV      R0,KDPDR7     ;RELOAD KERNEL PDR7
9469 070100 010037 177620      MOV      R0,UDPDR0     ;RELOAD PAGE 0 OF PRESENT SPACE
9470 070104 010037 177636      MOV      R0,UDPDR7     ;RE-LOAD I/O PAGE PDR IF ERROR
9471 070110 011515              MOV      (R5),(R5)     ;TRY WRITE AGAIN, IN CASE YOU
9472                          ;WERE TESTING PAGE SEVEN
9473 070112 000402              BR       15$           ;GO UPDATE R1 FOR NEXT READ
9474 070114 005237 001172      14$: INC      $TMP0     ;SET FLAG SINCE ADDRESSES MATCHED
9475 070120 062701 000002      15$: ADD      #2,R1     ;POINT TO NEXT PDR TO BE READ
9476 070124 077232              SOB      R2,13$       ;BRANCH TO 13$ IF ALL PDR'S NOT READ
9477 070126 005737 001172      TST      $TMP0        ;SEE IF THERE WAS A CORRECT PDR
9478 070132 001002              BNE     16$           ;BRANCH IF THERE WAS
9479 070134 011300      MOV      (R3),R0      ;SAVE CONTENTS OF PDR UNDER TEST
9480 070136 104113              ERROR   113          ;NO PDR ADDRESSES MATCHED
9481 070140 062703 000002      16$: ADD      #2,R3     ;POINT TO NEXT PDR UNDER TEST
9482 070144 062705 020000      ADD      #20000,R5    ;CHANGE PAGE NUMBER IN VIRT. ADDR.
9483 070150 005304              DEC      R4            ;DECREMENT COUNTER
9484 070152 001402              BEQ     17$           ;BRANCH IF COUNTER IS ZERO
9485 070154 000137 067466      JMP      19$           ;JUMP TO LOAD PDR'S AGAIN
9486 070160      17$:
9487 070160 042737 000007 172516      BIC      #7,MMR3      ;DISABLE ALL D-SPACE MAPPING
9488 070166 012737 000340 177776      MOV      #340,PSU    ;RETURN TO KERNEL MODE, PRIORITY 7
9489 070174 012737 067430 001112      MOV      #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
9490
9491
9492      .SBTTL ***** ENTRY POINT 8 --- STARTING ADDRESS 234 *****
9493      .SBTTL ***** MOVE FROM AND MOVE TO PREVIOUS MODE INSTRUCTION TEST *****
9494      *
9495      ;* THIS GROUP OF TESTS WILL TEST ALL THE LOGIC ASSOCIATED WITH
9496      ;* THE 'MOVE FROM PREVIOUS' AND MOVE TO PREVIOUS' INSTRUCTIONS.
9497      ;* THE LOGIC IS PRIMARILY ON 'SSRB', THE 'ROM OUTXX' SIGNALS ARE
9498      ;* GENERATED BY THE ROMS ON 'SSRA'.
9499      ;*
9500
```

9501
9502
9503
9504
9505
9506
9507
9508
9509
9510
9511
9512
9513
9514 070202
9515 070202 000004
9516 070204 012737 071004 001316
9517
9518 070212 104420
9519
9520 070214
9521 070214 012737 070416 001110
9522 070222 012737 070416 001112
9523 070230 012737 000106 001102
9524 070236 013737 001102 177570
9525 070244 012737 077406 172300
9526 070252 012737 077406 172302
9527 070260 012737 077406 172304
9528 070266 012737 077406 172306
9529 070274 012737 077406 172316
9530 070302 012737 000000 172340
9531 070310 012737 000200 172342
9532 070316 012737 000400 172344
9533 070324 012737 000600 172346
9534 070332 012737 177600 172356
9535 070340 012737 000001 177572
9536 070346 012737 000020 172516
9537 070354 042737 000007 172516
9538 070362 012700 077406
9539
9540 070366 012702 000000
9541 070372 012701 172300
9542 070376 010021
9543 070400 077202
9544 070402 012737 070416 001110
9545 070410 012737 070416 001112
9546 070416 012700 077400
9547
9548 070422 010037 172330
9549 070426 010037 172230
9550 070432 010037 177630
9551 070436 010037 177610
9552 070442 012737 077406 172310
9553 070450 012737 077406 172210
9554 070456 012737 001000 172350
9555 070464 012737 001000 172250
9556 070472 012700 036514

```
*****  
:TEST 106      MOVE FROM PREVIOUS (SUPERVISOR) I-SPACE  
:*****  
:      THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE  
:      PREVIOUS MODE IS CLOCKED CORRECTLY BY 'ROM OUT05'.  
:      THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,  
:      WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH  
:      THEIR ADDRESSES).  
:      IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT  
:      WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.  
:*****  
TST106:  
SCOPE  
MOV      #TST107,NXTTST ;SAVE STARTING ADDRESS OF NEXT  
;TEST FOR ESCAPE ON PARITY ERRORS  
TBITR    ;RESTORE T-BIT TO ITS STATUS BEFORE  
;THE SIX DUAL MAPPING TESTS  
ENTPT8:  
MOV      #20$,SLPADR    ;SET LOOP ADDRESS POINTER TO 20$  
MOV      #20$,SLPERR    ;SET LOOP ON ERROR POINTER TO 20$  
MOV      #106,$TSTNM    ;LOAD TEST NUMBER INTO MEMORY  
MOV      $TSTNM,DISPLAY ;DISPLAY TEST NUMBER FOR THIS TEST  
MOV      #77406,KIPDR0  ;MAKE KERNEL I PAGE 0 200 BLOCKS, R/W  
MOV      #77406,KIPDR1  ;MAKE KERNEL I PAGE 1 200 BLOCKS, R/W  
MOV      #77406,KIPDR2  ;MAKE KERNEL I PAGE 2 200 BLOCKS, R/W  
MOV      #77406,KIPDR3  ;MAKE KERNEL I PAGE 3 200 BLOCKS, R/W  
MOV      #77406,KIPDR7  ;MAKE KERNEL I PAGE 7 200 BLOCKS, R/W  
MOV      #000,KIPAR0    ;MAP KERNEL I PAGE 0 TO 0 - 4K  
MOV      #200,KIPAR1    ;MAP KERNEL I PAGE 1 TO 4K - 8K  
MOV      #400,KIPAR2    ;MAP KERNEL I PAGE 2 TO 8K - 12K  
MOV      #600,KIPAR3    ;MAP KERNEL I PAGE 3 TO 12K - 16K  
MOV      #177600,KIPAR7 ;MAP KERNEL I PAGE 7 TO THE I/O PAGE  
MOV      #BIT0,MMR0     ;ENABLE 18-BIT RELOCATION IF NOT ON  
MOV      #BIT4,MMR3     ;ENABLE 22-BIT RELOCATION IF NOT ON  
BIC      #7,MMR3        ;MAKE SURE THAT ALL D-SPACE IS DISABLED  
MOV      #77406,R0      ;MAKE ALL KERNEL I-SPACE PAGES RESIDENT  
;READ/WRITE, LENGTH 200 BLOCKS  
MOV      #10,R2         ;SET COUNT TO LOAD 8 ADDRESSES  
MOV      #KIPDR0,R1    ;PUT ADDRESS OF FIRST PDR IN R1  
19$:    MOV      R0,(R1)+ ;LOAD R0 INTO PDR ADDRESSED BY R1  
;BRANCH BACK TO 19$ IF R2 IS NOT ZERO  
SOB     R2,19$  
MOV      #20$,SLPADR    ;SET LOOP POINTER TO 20$  
MOV      #20$,SLPERR    ;SET LOOP ON ERROR TO 20$  
20$:    MOV      #77400,R0 ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I  
;AND KERNEL I NON-RESIDENT  
MOV      R0,KDPDR4     ;KERNEL D-SPACE PAGE 4  
MOV      R0,SDPDR4     ;SUPERVISOR D-SPACE PAGE 4  
MOV      R0,UDPDR4     ;USER D-SPACE PAGE 4  
MOV      R0,UIPDR4     ;USER I-SPACE PAGE 4  
MOV      #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE  
MOV      #77406,SIPDR4 ;SUPER I-SPACE PAGE 4 READ/WRITE  
MOV      #1000,KIPAR4   ;MAP KERNEL I PAGE 4 TO 16K  
MOV      #1000,SIPAR4   ;MAP SUPER I PAGE 4 TO 16K  
MOV      #36514,R0     ;LOAD DATA PATTERN INTO R0
```



```

9557 070476 010037 100000      MOV      R0,#100000      ;LOAD DATA PATTERN INTO PHY 100000
9558 070502 012737 070756 000250  MOV      #10$,MMVEC      ;SET M.M. VECTOR TO 10$
9559 070510 105037 172310      CLR      KIPDR4          ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
9560 070514 012737 070522 001112  MOV      #11$,SLPERR      ;SET LOOP ON ERROR POINTER TO 11$
9561 070522 012737 010340 177776 11$:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9562 070530 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9563 070532 006506      *$:  MFPI      SSP          ;PUT SUPERVISOR STACK POINTER ON KERNEL
9564                                ;STACK
9565 070534 022706 001100      CMP      #KERSTK,KSP      ;WAS SOMETHING PUSHED ON STACK AT 1$
9566 070540 001407      BEQ      3$              ;BRANCH IF NOTHING WAS PUSHED
9567 070542 012601      MOV      (KSP)+,R1        ;POP KERNEL STACK INTO R1
9568 070544 012702 000700      MOV      #SUPSTK,R2       ;EXPECTING TO GET 700 AS SSP
9569 070550 020201      CMP      R2,R1           ;DID YOU GET THE RIGHT POINTER?
9570 070552 001403      BEQ      2$              ;BRANCH IF YOU DID
9571 070554 104114      ERROR   114             ;WRONG THING WAS PUSHED ON STACK
9572 070556 000401      BR      2$              ;BRANCH TO NEXT TRY
9573 070560 104115      3$:  ERROR   115             ;NOTHING PUSHED ON STACK
9574 070562      2$:  ;THE FOLLOWING WILL TEST DSTM=1 MFPI.  BELOW ARE THE
9575                                ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9576                                ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9577                                ;INTO THE F/F'S ON SSRB.
9578                                :  * D12.00 (001)
9579                                :  * D12.10 (175)
9580                                :  * MFP.00 (066)
9581                                :  MFP.10 (250)
9582                                :  SVC.80 (222)
9583                                :  SVC.90 (300)
9584                                :  FET.00 (217)
9585 070562 012737 070570 001112  MOV      #12$,SLPERR      ;SET LOOP ON ERROR POINTER TO 12$
9586 070570 012737 010340 177776 12$:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9587 070576 012702 100000      MOV      #100000,R2       ;LOAD VIRTUAL ADDRESS INTO R2
9588 070602 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9589 070604 006512      MFPI      (R2)          ;READ FROM PHYSICAL 100000
9590 070606 012601      MOV      (KSP)+,R1        ;POP KERNEL STACK INTO R1
9591 070610 020001      CMP      R0,R1           ;WAS DATA FETCHED SAME AS STORED
9592 070612 001401      BEQ      4$              ;BRANCH IF CORRECT DATA WAS FETCHED
9593 070614 104116      ERROR   116             ;WRONG DATA WAS FETCHED
9594 070616      4$:  ;THE FOLLOWING WILL TEST DSTM=2 MFPI.  BELOW ARE THE
9595                                ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9596                                ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9597                                ;INTO THE F/F'S ON SSRB.
9598                                :  * D12.01 (002)
9599                                :  * D12.10 (175)
9600                                :  * MFP.00 (066)
9601                                :  MFP.10 (250)
9602                                :  SVC.80 (222)
9603                                :  SVC.90 (300)
9604                                :  FET.00 (217)
9605 070616 012737 070624 001112  MOV      #14$,SLPERR      ;SET LOOP ON ERROR POINTER TO 14$
9606 070624 012737 010340 177776 14$:  MOV      #010340,PSW      ;MAKE PREVIOUS MODE SUPERVISOR
9607 070632 012702 100000      MOV      #100000,R2       ;LOAD VIRTUAL ADDRESS INTO R2
9608 070636 000240      NOP                      ;THIS IS A SYNC POINT FOR SCOPING
9609 070640 006522      MFPI      (R2)+         ;READ FROM PHYSICAL 100000
9610 070642 012601      MOV      (KSP)+,R1        ;POP KERNEL STACK INTO R1
9611 070644 020001      CMP      R0,R1           ;WAS DATA FETCHED SAME AS STORED
9612 070646 001401      BEQ      5$              ;BRANCH IF CORRECT DATA WAS FEICED

```


9613	070650	104116				ERROR 116 ;WRONG DATA WAS FETCHED
9614	070652				5\$:	;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
9615						;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9616						;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9617						;INTO THE F/F'S ON SSRB.
9618						: D30.00 (003)
9619						: D30.10 (221)
9620						: D10.20 (233)
9621						: * D10.50 (311)
9622						: * D10.60 (177)
9623						: * MFP.00 (066)
9624						: MFP.10 (250)
9625						: SVC.80 (222)
9626						: SVC.90 (300)
9627						: FET.00 (217)
9628	070652	012737	070660	001112		MOV #15\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 15\$
9629	070660	012737	010340	177776	15\$:	MOV #010340, PSW ;MAKE PREVIOUS MODE SUPERVISOR
9630	070666	000240				NOP ;THIS IS A SYNC POINT FOR SCOPING
9631	070670	006537	100000			MFPI @#100000 ;READ FROM PHYSICAL 100000
9632	070674	012601				MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
9633	070676	020001				CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
9634	070700	001401				BEQ 6\$;BRANCH IF CORRECT DATA WAS FETCHED
9635	070702	104116				ERROR 116 ;WRONG DATA WAS FETCHED
9636	070704				6\$:	;THE FOLLOWING WILL TEST DSTM=4 MFPI. BELOW ARE THE
9637						;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9638						;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9639						;INTO THE F/F'S ON SSRB.
9640						: D45.00 (004)
9641						: * D10.30 (122)
9642						: * D10.60 (177)
9643						: * MFP.00 (066)
9644						: MFP.10 (250)
9645						: SVC.80 (222)
9646						: SVC.90 (300)
9647						: FET.00 (217)
9648	070704	012737	070712	001112		MOV #16\$, \$LPERR ;SET LOOP ON ERROR POINTER TO 16\$
9649	070712	012737	010340	177776	16\$:	MOV #010340, PSW ;MAKE PREVIOUS MODE SUPERVISOR
9650	070720	012702	100002			MOV #100002, R2 ;LOAD VIRTUAL ADDRESS INTO R2
9651	070724	000240				NOP ;THIS IS A SYNC POINT FOR SCOPING
9652	070726	006542				MFPI -(R2) ;READ FROM PHYSICAL 100000
9653	070730	012601				MOV (KSP)+, R1 ;POP KERNEL STACK INTO R1
9654	070732	020001				CMP R0, R1 ;WAS DATA FETCHED SAME AS STORED
9655	070734	001401				BEQ 7\$;BRANCH IF CORRECT DATA WAS FETCHED
9656	070736	104116				ERROR 116 ;WRONG DATA WAS FETCHED
9657	070740	012737	032226	000250	7\$:	MOV #MMTRAP, MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
9658	070746	012737	070416	001112		MOV #20\$, \$LPERR ;SET LOOP POINTER TO START OF TEST
9659	070754	000413				BR TST107 ;BRANCH TO NEXT TEST
9660						
9661						
9662	070756	013737	177572	001250	10\$:	MOV MMR0, PMMR0 ;SAVE MMR0 FOR ERROR TYPEOUT
9663	070764	013737	177574	001252		MOV MMR1, PMMR1 ;SAVE MMR1 FOR ERROR TYPEOUT
9664	070772	013737	177576	001254		MOV MMR2, PMMR2 ;SAVE MMR2 FOR ERROR TYPEOUT
9665	071000	104117				ERRUR 117 ;TRIED TO READ NON-RESIDENT PAGE
9666	071002	000002				RTI
9667						
9668						

```

9659
9670
9671
9672
9673
9674
9675
9676
9677
9678
9679
9680
9681 071004
9682 071004 000004
9683 071006 012737 071464 001316
9684
9685 071014 012700 077400
9686
9687 071020 010037 172330
9688 071024 010037 172230
9689 071030 010037 177630
9690 071034 010037 177610
9691 071040 012737 077406 172310
9692 071046 012737 077406 172210
9693 071054 012737 001000 172350
9694 071062 012737 001000 172250
9695 071070 012737 071436 000250
9696 071076 012737 010340 177776
9697 071104 012746 007777
9698 071110 006606
9699 071112 006506
9700 071114 012601
9701 071116 022701 007777
9702 071122 001401
9703 071124 104120
9704 071126 012737 010340 177776
9705 071134 012746 000700
9706 071140 006606
9707 071142
9708
9709
9710
9711
9712
9713
9714
9715 071142 012737 071160 001112
9716 071150 012702 100000
9717 071154 012700 125252
9718 071160 010046
9719 071162 105037 172310
9720 071166 000240
9721 071170 006612
9722 071172 112737 000006 172310
9723 071200 011201
9724 071202 020001

```

```

*****
:TEST 107 MOVE TO PREVIOUS (SUPERVISOR) I-SPACE
:
: THIS TEST USES THE 'MTPJ' INSTRUCTION TO ENSURE THAT THE
: PREVIOUS MODE IS CLOCKED CORRECTLY BY 'ROM OUT05'.
: THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
: WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
: THEIR ADDRESSES).
: IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
: WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
*****
TST107:
SCOPE
MOV #TST110,NXTTST ;SAVE STARTING ADDRESS OF NEXT
;TEST FOR ESCAPE ON PARITY ERRORS
20$: MOV #77400,R0 ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
;AND KERNEL I NON-RESIDENT
;KERNEL D-SPACE PAGE 4
MOV R0,KDPDR4 ;SUPERVISOR D-SPACE PAGE 4
MOV R0,SDPDR4 ;USER D-SPACE PAGE 4
MOV R0,UDPDR4 ;USER I-SPACE PAGE 4
MOV R0,UIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
MOV #77406,KIPDR4 ;SUPER I-SPACE PAGE 4 READ/WRITE
MOV #77406,SIPDR4 ;MAP KERNEL I PAGE 4 TO 16K
MOV #1000,KIPAR4 ;MAP SUPER I PAGE 4 TO 16K
MOV #1000,SIPAR4 ;SET M.M. VECTOR TO 10$
1$: MOV #10$,MMVEC ;MAKE PREVIOUS MODE SUPERVISOR
MOV #010340,PSW ;PUSH DATA ON KERNEL STACK
MOV #7777,-(KSP) ;LOAD SUPERVISOR STACK POINTER
MTPJ SSP ;READ SUPERVISOR STACK POINTER
MFPJ SSP ;POP KERNEL STACK INTO R1
MOV (KSP)+,R1 ;WAS SUPERVISOR STACK POINTER CHANGED
CMP #7777,R1 ;BRANCH IF IT WAS
BEQ 2$ ;SUPER STACK POINTER NOT CHANGED
ERROR 120 ;MAKE PREVIOUS MODE SUPERVISOR
2$: MOV #010340,PSW ;GET READY TO RESTORE SUPERVISOR S. POINT
MOV #SUPSTK,-(KSP) ;RESTORE SUPERVISOR STACK POINTER
MTPJ SSP
3$: ;THIS WILL TEST DSTM = 1 MTPJ. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
;INTO THE FLIP/FLOPS ON SSRB.
; * D12.80 (111)
; * D12.60 (155)
; D12.20 (312)
; FET.10 (260)
MOV #13$,SLPERR ;SET LOOP ON ERROR POINTER TO 13$
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
MOV #125252,R0 ;LOAD TEST DATA INTO R0
13$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
CLR B KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
NOP ;THIS IS A SYNC POINT FOR SCOPING
MTPJ (R2) ;LOAD TEST DATA INTO PHYSICAL 100000
MOV B #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
MOV (R2),R1 ;READ FROM ADDRESS 100000
CMP R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE

```

```

9725 071204 001401
9726 071206 104121
9727 0712'0
9728
9729
9730
9731
9732
9733
9734
9735
9736
9737 071210 012737 071230 001112
9738 071216 012737 010340 177776
9739 071224 012700 052525
9740 071230 010046
9741 071232 105037 172310
9742 071236 000240
9743 071240 006637 100000
9744 071244 112737 000006 172310
9745 071252 013701 100000
9746 071256 020001
9747 071260 001401
9748 071262 104121
9749 071264
9750
9751
9752
9753
9754
9755
9756
9757 071264 012737 071304 001112
9758 071272 012737 010340 177776
9759 071300 012700 125252
9760 071304 010046
9761 071306 012702 100002
9762 071312 105037 172310
9763 071316 000240
9764 071320 006642
9765 071322 112737 000006 172310
9766 071330 013701 100000
9767 071334 020001
9768 071336 001401
9769 071340 104121
9770 071342
9771
9772
9773
9774
9775
9776
9777
9778
9779
9780 071342 012737 071364 001112
    
```

```

BEQ 4$ ;BRANCH IF STORE WAS CORRECT
ERROR 121 ;INCORRECT STORE
;THIS WILL TEST DSTM = 3 MTPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
;INTO THE FLIP/FLOPS ON SSRB.
: D30.80 (113)
: D30.10 (221)
: D10.20 (233)
: * D10.50 (311)
: * D10.40 (157)
: FET.01 (331)
MOV #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$
MOV #010340, PSW ;MAKE PREVIOUS MODE SUPERVISOR
MOV #52525, R0 ;LOAD TEST DATA INTO R0
MOV R0, -(KSP) ;PUSH TEST DATA ON KERNEL STACK
CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
NOP ;THIS IS A SYNC POINT FOR SCOPING
MTPI @#100000 ;LOAD TEST DATA INTO PHYSICAL 100000
MOVB #006, KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
MOV @#100000, R1 ;READ FROM ADDRESS 100000
CMP R0, R1 ;SEE IF DATA WAS STORED CORRECTLY
BEQ 5$ ;BRANCH IF STORE WAS CORRECT
ERROR 121 ;INCORRECT STORE
;THIS WILL TEST DSTM = 4 MTPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
;INTO THE FLIP/FLOPS ON SSRB.
: D45.80 (115)
: * D40.20 (121)
: * D10.40 (157)
: FET.01 (331)
MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
MOV #010340, PSW ;MAKE PREVIOUS MODE SUPERVISOR
MOV #125252, R0 ;LOAD TEST DATA INTO R0
MOV R0, -(KSP) ;PUSH TEST DATA ON KERNEL STACK
MOV #100002, R2 ;LOAD VIRTUAL ADDRESS INTO R2
CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
NOP ;THIS IS A SYNC POINT FOR SCOPING
MTPI -(R2) ;LOAD TEST DATA INTO PHYSICAL 100000
MOVB #006, KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
MOV @#100000, R1 ;READ FROM ADDRESS 100000
CMP R0, R1 ;SEE IF DATA WAS STORED CORRECTLY
BEQ 6$ ;BRANCH IF STORE WAS CORRECT
ERROR 121 ;INCORRECT STORE
;THIS WILL TEST DSTM = 6 MTPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
;INTO THE FLIP/FLOPS ON SSRB.
: D67.80 (117)
: D67.00 (006)
: D67.10 (251)
: * D10.30 (122)
: * D10.40 (157)
: FET.01 (331)
MOV #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
    
```

9781	071350	012737	010340	177776		MOV	#010340,PSW	:MAKE PREVIOUS MODE SUPERVISOR
9782	071356	012700	052525			MOV	#52525,R0	:LOAD TEST DATA INTO R0
9783	071362	005002				CLR	R2	:MAKE REGISTER 2 ZERO
9784	071364	010046			16\$:	MOV	R0,-(KSP)	:PUSH TEST DATA ON KERNEL STACK
9785	071366	105037	172310			CLRB	KIPDR4	:MAKE KERNEL I PAGE 4 NON-RESIDENT
9786	071372	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
9787	071374	006662	100000			MTPJ	100000(R2)	:LOAD TEST DATA INTO PHYSICAL 100000
9788	071400	112737	000006	172310		MOVB	#006,KIPDR4	:MAKE KERNEL PAGE 4 RESIDENT
9789	071406	013701	100000			MOV	#100000,R1	:READ FROM ADDRESS 100000
9790	071412	020001				CMP	R0,R1	:SEE IF DATA WAS STORED CORRECTLY
9791	071414	001401				BEQ	7\$:BRANCH IF STORE WAS CORRECT
9792	071416	104121				ERROR	121	:INCORRECT STORE
9793	071420	012737	071014	001112	7\$:	MOV	#20\$,SLPERR	:SET LOOP POINTER TO START OF TEST
9794	071426	012737	032226	000250		MOV	#MMTRAP,MMVEC	:RESTORE M.M. VECTOR TO NORMAL ROUTINE
9795	071434	000413				BR	TST110	:BRANCH TO NEXT TEST
9796								
9797								
9798	071436	013737	177572	001250	10\$:	MOV	MMR0,PMR0	:SAVE MMR0 FOR ERROR TYPEOUT
9799	071444	013737	177574	001252		MOV	MMR1,PMR1	:SAVE MMR1 FOR ERROR TYPEOUT
9800	071452	013737	177576	001254		MOV	MMR2,PMR2	:SAVE MMR2 FOR ERROR TYPEOUT
9801	071460	104117				ERROR	117	:TRIED TO LOAD A N.R. PAGE 4
9802	071462	000002				RT!		:RETURN TO TEST

9803
9804
9805
9806
9807
9808
9809
9810
9811
9812
9813
9814
9815
9816
9817
9818

```
*****  
*TEST 110 MFPI (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED  
*  
* THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE  
* PREVIOUS MODE IS CLOKED CORRECTLY BY 'ROM OUT05', AND THAT  
* D-SPACE IS NOT ENABLED. THIS IS DONE BY 'ROM OUT08 H' AND  
* 'SSRB IR15 L' NOT ASSERTED GENERATING 'SSRB I SPACEB L'.  
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,  
* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH  
* THEIR ADDRESSES).  
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT  
* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.  
*****
```

9819	071464					TST110:	SCOPE	
9820	071464	000004					MOV	#TST111,NXTTST
9821	071466	012737	072064	001316				:SAVE STARTING ADDRESS OF NEXT
9822								:TEST FOR ESCAPE ON PARITY ERRORS
9823	071474	012700	077406				MOV	#77406,R0
9824								:MAKE ALL KERNEL I-SPACE PAGES RESIDENT
9825	071500	012702	000010				MOV	#10,R2
9826	071504	012701	172300				MOV	#KIPDR0,R1
9827	071510	010021			19\$:		MOV	R0,(R1)+
9828	071512	077202					SOB	R2,19\$
9829	071514	012737	071530	001110			MOV	#20\$,SLPADR
9830	071522	012737	071530	001112			MOV	#20\$,SLPERR
9831	071530	012700	077400		20\$:		MOV	#77400,R0
9832								:MAKE PAGE 4 IN ALL BUT SUPERVISOR I
9833	071534	010037	172330					:AND KERNEL I NON-RESIDENT
9834	071540	010037	172230				MOV	R0,KDPDR4
9835	071544	010037	177630				MOV	R0,SDPDR4
9836	071550	010037	177610				MOV	R0,UDPDR4
							MOV	R0,UIPDR4

```
*****  
*TEST 110 MFPI (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED  
*  
* THIS TEST USES THE 'MFPI' INSTRUCTION TO ENSURE THAT THE  
* PREVIOUS MODE IS CLOKED CORRECTLY BY 'ROM OUT05', AND THAT  
* D-SPACE IS NOT ENABLED. THIS IS DONE BY 'ROM OUT08 H' AND  
* 'SSRB IR15 L' NOT ASSERTED GENERATING 'SSRB I SPACEB L'.  
* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,  
* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH  
* THEIR ADDRESSES).  
* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT  
* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.  
*****
```

9837 071554 012737 077406 172310
9838 071562 012737 077406 172210
9839 071570 012737 001000 172350
9840 071576 012737 001000 172250
9841 071604 012700 036514
9842 071610 010037 100000
9843 071614 012737 072036 000250
9844 071622 052737 000002 172516
9845 071630 105037 172310
9846
9847
9848
9849
9850
9851
9852
9853
9854
9855
9856
9857 071634 012737 071642 001112
9858 071642 012737 010340 177776 12\$:
9859 071650 012702 100000
9860 071654 000240
9861 071656 006512
9862 071660 012601
9863 071662 020001
9864 071664 001401
9865 071666 104116
9866 071670 4\$:
9867
9868
9869
9870
9871
9872
9873
9874
9875
9876
9877 071670 012737 071676 001112
9878 071676 012737 010340 177776 14\$:
9879 071704 012702 100000
9880 071710 000240
9881 071712 006522
9882 071714 012601
9883 071716 020001
9884 071720 001401
9885 071722 104116
9886 071724 5\$:
9887
9888
9889
9890
9891
9892

MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
MOV #77406,SIPDR4 ;SUPER I-SPACE PAGE 4 READ/WRITE
MOV #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
MOV #1000,SIPAR4 ;MAP SUPER I PAGE 4 TO 16K
MOV #36514,R0 ;LOAD DATA PATTERN INTO R0
MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 100000
MOV #10\$,MMVEC ;SET M.M. VECTOR TO 10\$
BIS #BIT1,MMR3 ;ENABLE SUPERVISOR D-SPACE
CLRB KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
;THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
: * D12.00 (001)
: * D12.10 (175)
: * MFP.00 (066)
: MFP.10 (250)
: SVC.80 (222)
: SVC.90 (300)
: FET.00 (217)
MOV #12\$,SLPERR ;SET LOOP ON ERROR POINTER TO 12\$
MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
NOP ;THIS IS A SYNC POINT FOR SCOPING
MFPI (R2) ;READ FROM PHYSICAL 100000
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
BEQ 4\$;BRANCH IF CORRECT DATA WAS FETCHED
ERROR 116 ;WRONG DATA WAS FETCHED
;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
: * D12.01 (002)
: * D12.10 (175)
: * MFP.00 (066)
: MFP.10 (250)
: SVC.80 (222)
: SVC.90 (300)
: FET.00 (217)
MOV #14\$,SLPERR ;SET LOOP ON ERROR POINTER TO 14\$
MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
NOP ;THIS IS A SYNC POINT FOR SCOPING
MFPI (R2)+ ;READ FROM PHYSICAL 100000
MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
BEQ 5\$;BRANCH IF CORRECT DATA WAS FETCHED
ERROR 116 ;WRONG DATA WAS FETCHED
;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
: D30.00 (003)
: D30.10 (221)
: D10.20 (233)

```
9893 : * D10.50 (311)
9894 : * D10.60 (177)
9895 : * MFP.00 (066)
9896 : MFP.10 (250)
9897 : SVC.80 (222)
9898 : SVC.90 (300)
9899 : FET.00 (217)
9900 071724 012737 071732 001112 MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
9901 071732 012737 010340 177776 15$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
9902 071740 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
9903 071742 006537 100000 MFPI @#100000 ;READ FROM PHYSICAL 100000
9904 071746 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
9905 071750 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
9906 071752 001401 BEQ 6$ ;BRANCH IF CORRECT DATA WAS FETCHED
9907 071754 104116 ERROR 116 ;WRONG DATA WAS FETCHED
9908 071756 6$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI. BELOW ARE THE
9909 ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
9910 ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
9911 ;INTO THE F/F'S ON SSRB.
9912 : D45.00 (004)
9913 : * D10.30 (122)
9914 : * D10.60 (177)
9915 : * MFP.00 (066)
9916 : MFP.10 (250)
9917 : SVC.80 (222)
9918 : SVC.90 (300)
9919 : FET.00 (217)
9920 071756 012737 071764 001112 MOV #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
9921 071764 012737 010340 177776 16$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
9922 071772 012702 100002 MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
9923 071776 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
9924 072000 006542 MFPI -(R2) ;READ FROM PHYSICAL 100000
9925 072002 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
9926 072004 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
9927 072006 001401 BEQ 7$ ;BRANCH IF CORRECT DATA WAS FETCHED
9928 072010 104116 ERROR 116 ;WRONG DATA WAS FETCHED
9929 072012 012737 032226 000250 7$: MOV #MMTRAP,MMVEC ;SET M.M.VECTOR TO NORMAL ROUTINE
9930 072020 012737 071530 001112 MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
9931 072026 042737 000002 172516 BIC #BIT1,MMR3 ;DISABLE SUPERVISOR D-SPACE
9932 072034 000413 BR TST111 ;:BRANCH TO NEXT TEST
9933
9934
9935 072036 013737 177572 001250 10$: MOV MMR0,PMR0 ;SAVE MMR0 FOR ERROR TYPEOUT
9936 072044 013737 177574 001252 MOV MMR1,PMR1 ;SAVE MMR1 FOR ERROR TYPEOUT
9937 072052 013737 177576 001254 MOV MMR2,PMR2 ;SAVE MMR2 FOR ERROR TYPEOUT
9938 072060 104117 ERROR 117 ;TRIED TO READ NON-RESIDENT PAGE
9939 072062 000002 RTI ;RETURN TO TEST
9940
9941
9942
9943
9944
9945 :*****
9946 :*TEST 111 MTP1 (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED
9947 :*
9948 :* THIS TEST USES THE 'MTP1' INSTRUCTION TO ENSURE(TIQT)IE
:PREVIOUS MODE IS CLOKED CORRECTLY BY 'ROM OUT05', AND 'HA'
```

```
9949 : * D-SPACE IS NOT ENABLED. THIS IS DONE BY 'ROM OUT08 H' AND
9950 : * 'SSRB IR15 L' NOT ASSERTED GENERATING 'SSRB I SPACEB L'.
9951 : * THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
9952 : * WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
9953 : * THEIR ADDRESSES).
9954 : * IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
9955 : * WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
9956 : *
9957 : *****
9958 072064 :TST 11:
9959 072064 000004 SCOPE
9960 072066 012737 072514 001316 MOV #TST112,NXTTST ;SAVE STARTING ADDRESS OF NEXT
9961 : ;TEST FOR ESCAPE ON PARITY ERRORS
9962 072074 012700 077400 20$: MOV #77400,R0 ;MAKE PAGE 4 IN ALL BUT SUPERVISOR I
9963 : ;AND KERNEL I NON-RESIDENT
9964 072100 010037 172330 MOV R0,KDPDR4 ;KERNEL D-SPACE PAGE 4
9965 072104 010037 172230 MOV R0,SDPDR4 ;SUPERVISOR D-SPACE PAGE 4
9966 072110 010037 177630 MOV R0,UDPDR4 ;USER D-SPACE PAGE 4
9967 072114 010037 177610 MOV R0,UIPDR4 ;USER I-SPACE PAGE 4
9968 072120 012737 077406 172310 MOV #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
9969 072126 012737 077406 172210 MOV #77406,SIPDR4 ;SUPER I-SPACE PAGE 4 READ/WRITE
9970 072134 012737 001000 172350 MOV #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
9971 072142 012737 001000 172250 MOV #1000,SIPAR4 ;MAP SUPER I PAGE 4 TO 16K
9972 072150 012737 072466 000250 MOV #10$,MMVEC ;SET M.M. VECTOR TO 10$
9973 072156 052737 000002 172516 BIS #BIT1,MMR3 ;ENABLE SUPERVISOR D-SPACE
9974 : ;THIS WILL TEST DSTM = 1 MTPI. BELOW ARE THE
9975 : ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
9976 : ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9977 : ;INTO THE FLIP/FLOPS ON SSRB.
9978 : * D12.80 (111)
9979 : * D12.60 (155)
9980 : D12.20 (312)
9981 : FET.10 (260)
9982 072164 012737 072202 001112 MOV #13$, $LPERR ;SET LOOP ON ERROR POINTER TO 13$
9983 072172 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
9984 072176 012700 125252 MOV #125252,R0 ;LOAD TEST DATA INTO R0
9985 072202 010046 13$: MOV R0,-(KSP) ;PUSH TEST DATA ON KERNEL STACK
9986 072204 105037 172310 CLRB KIPDR4 ;MAKE KERNEL I PAGE 4 NON-RESIDENT
9987 072210 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
9988 072212 006612 MTPI (R2) ;LOAD TEST DATA INTO PHYSICAL 100000
9989 072214 112737 000006 172310 MOVB #006,KIPDR4 ;MAKE KERNEL PAGE 4 RESIDENT
9990 072222 011201 MOV (R2),R1 ;READ FROM ADDRESS 100000
9991 072224 020001 CMP R0,R1 ;SEE IF DATA WAS STORED AT CORRECT PLACE
9992 072226 001401 BEQ 4$ ;BRANCH IF STORE WAS CORRECT
9993 072230 104121 ERROR 121 ;INCORRECT STORE
9994 072232 4$: ;THIS WILL TEST DSTM = 3 MTPI. BELOW ARE THE
9995 : ;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK
9996 : ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
9997 : ;INTO THE FLIP/FLOPS ON SSRB.
9998 : D30.80 (113)
9999 : D30.10 (221)
10000 : D10.20 (233)
10001 : * D10.50 (311)
10002 : * D10.40 (157)
10003 : FET.01 (331)
10004 072232 012737 072252 001112 MOV #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$
```


10005	072240	012737	010340	177776	MOV	#010340,PSW	;MAKE PREVIOUS MODE SUPERVISOR	
10006	072246	012700	052525		MOV	#52525,R0	;LOAD TEST DATA INTO R0	
10007	072252	010046			MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK	
10008	072254	105037	172310		CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT	
10009	072260	000240			NOP		;THIS IS A SYNC POINT FOR SCOPING	
10010	072262	006637	100000		MTPI	@#100000	;LOAD TEST DATA INTO PHYSICAL 100000	
10011	072266	112737	000006	172310	MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT	
10012	072274	013701	100000		MOV	@#100000,R1	;READ FROM ADDRESS 100000	
10013	072300	020001			CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY	
10014	072302	001401			BEQ	5\$;BRANCH IF STORE WAS CORRECT	
10015	072304	104121			ERROR	121	;INCORRECT STORE	
10016	072306				5\$:		;THIS WILL TEST DSTM = 4 MTPI. BELOW ARE THE	
10017							;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK	
10018							;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED	
10019							;INTO THE FLIP/FLOPS ON SSRB.	
10020						D45.80	(115)	
10021						* D40.20	(121)	
10022						* D10.40	(157)	
10023						FET.01	(331)	
10024	072306	012737	072326	001112	MOV	#15\$,SLPERR	;SET LOOP ON ERROR POINTER TO 15\$	
10025	072314	012737	010340	177776	MOV	#010340,PSW	;MAKE PREVIOUS MODE SUPERVISOR	
10026	072322	012700	125252		MOV	#125252,R0	;LOAD TEST DATA INTO R0	
10027	072326	010046			MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK	
10028	072330	012702	100002		MOV	#100002,R2	;LOAD VIRTUAL ADDRESS INTO R2	
10029	072334	105037	172310		CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT	
10030	072340	000240			NOP		;THIS IS A SYNC POINT FOR SCOPING	
10031	072342	006642			MTPI	-(R2)	;LOAD TEST DATA INTO PHYSICAL 100000	
10032	072344	112737	000006	172310	MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT	
10033	072352	013701	100000		MOV	@#100000,R1	;READ FROM ADDRESS 100000	
10034	072356	020001			CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY	
10035	072360	001401			BEQ	6\$;BRANCH IF STORE WAS CORRECT	
10036	072362	104121			ERROR	121	;INCORRECT STORE	
10037	072364				6\$:		;THIS WILL TEST DSTM = 6 MTPI. BELOW ARE THE	
10038							;ROM STATE NAMES AND ADDRESSES, FROM THE C-FORK	
10039							;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED	
10040							;INTO THE FLIP/FLOPS ON SSRB.	
10041						D67.80	(117)	
10042						D67.00	(006)	
10043						D67.10	(251)	
10044						* D10.30	(122)	
10045						* D10.40	(157)	
10046						FET.01	(331)	
10047	072364	012737	072406	001112	MOV	#16\$,SLPERR	;SET LOOP ON ERROR POINTER TO 16\$	
10048	072372	012737	010340	177776	MOV	#010340,PSW	;MAKE PREVIOUS MODE SUPERVISOR	
10049	072400	012700	052525		MOV	#52525,R0	;LOAD TEST DATA INTO R0	
10050	072404	005002			CLR	R2	;MAKE REGISTER 2 ZERO	
10051	072406	010046			MOV	R0,-(KSP)	;PUSH TEST DATA ON KERNEL STACK	
10052	072410	105037	172310		CLRB	KIPDR4	;MAKE KERNEL I PAGE 4 NON-RESIDENT	
10053	072414	000240			NOP		;THIS IS A SYNC POINT FOR SCOPING	
10054	072416	006662	100000		MTPI	100000(R2)	;LOAD TEST DATA INTO PHYSICAL 100000	
10055	072422	112737	000006	172310	MOVB	#006,KIPDR4	;MAKE KERNEL PAGE 4 RESIDENT	
10056	072430	013701	100000		MOV	@#100000,R1	;READ FROM ADDRESS 100000	
10057	072434	020001			CMP	R0,R1	;SEE IF DATA WAS STORED CORRECTLY	
10058	072436	001401			BEQ	7\$;BRANCH IF STORE WAS CORRECT	
10059	072440	104121			ERROR	121	;INCORRECT STORE	
10060	072442	012737	072074	001112	7\$:	MOV	#20\$,SLPERR	;SET LOOP POINTER TO START OF TEST

10061	072450	012737	032226	000250		MOV	#MMTRAP,MMVEC	:RESTORE M.M. VECTOR TO NORMAL ROUTINE
10062	072456	042737	000002	172516		BIC	#BIT1,MMR3	:DISABLE SUPERVISOR D-SPACE
10063	072464	000413				BR	TST112	:BRANCH TO NEXT TEST
10064								
10065								
10066	072466	013737	177572	001250	10\$:	MOV	MMR0,PMR0	:SAVE MMR0 FOR ERROR TYPEOUT
10067	072474	013737	177574	001252		MOV	MMR1,PMR1	:SAVE MMR1 FOR ERROR TYPEOUT
10068	072502	013737	177576	001254		MOV	MMR2,PMR2	:SAVE MMR2 FOR ERROR TYPEOUT
10069	072510	104117				ERROR	117	:TRIED TO LOAD A N.R. PAGE 4
10070	072512	000002				RTI		:RETURN TO TEST
10071								
10072								
10073								
10074								
10075								
10076								
10077								
10078								
10079								
10080								
10081								
10082								
10083								
10084								
10085								
10086								

```
*****
:TEST 112      MOVE FROM PREVIOUS (USER) I-SPACE
:
: THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS USER THE
: 'USER SPACE (1) L' FLIP-FLOP IS SET AND THE FETCH IS FROM
: USER MODE.
: THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
: WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
: THEIR ADDRESSES).
: IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
: WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
*****
```

10087	072514					TST112:		
10088	072514	000004				SCOPE		
10089	072516	012737	073106	001316		MOV	#TST113,NXTTST	:SAVE STARTING ADDRESS OF NEXT
10090								:TEST FOR ESCAPE ON PARITY ERRORS
10091	072524	012700	077400		20\$:	MOV	#77400,R0	:MAKE PAGE 4 NON-RESIDENT IN ALL MODES
10092								:EXCEPT KERNEL I-SPACE & USER I-SPACE
10093	072530	010037	172330			MOV	R0,KDPDR4	:KERNEL D-SPACE PAGE 4
10094	072534	010037	177630			MOV	R0,UDPDR4	:USER D-SPACE PAGE 4
10095	072540	010037	172230			MOV	R0,SDPDR4	:SUPERVISOR D-SPACE PAGE 4
10096	072544	010037	172210			MOV	R0,SIPDR4	:SUPERVISOR I-SPACE PAGE 4
10097	072550	012737	077406	172310		MOV	#77406,KIPDR4	:KERNEL I-SPACE PAGE 4 READ/WRITE
10098	072556	012737	077406	177610		MOV	#77406,UIPDR4	:USER I-SPACE PAGE 4 READ/WRITE
10099	072564	012737	001000	172350		MOV	#1000,KIPAR4	:MAP KERNEL I PAGE 4 TO 16K
10100	072572	012737	001000	177650		MOV	#1000,UIPAR4	:MAP USER I PAGE 4 TO 16K
10101	072600	012700	036514			MOV	#36514,R0	:LOAD DATA PATTERN INTO R0
10102	072604	010037	100000			MOV	R0,#100000	:LOAD DATA PATTERN INTO PHY 100000
10103	072610	012702	100000			MOV	#100000,R2	:LOAD VIRTUAL ADDRESS INTO R2
10104	072614	012737	073060	000250		MOV	#10\$,MMVEC	:SET M.M. VECTOR TO 10\$
10105	072622	105037	172310			CLRB	KIPDR4	:MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
10106	072626	012737	030340	177776		MOV	#030340,PSW	:MAKE PREVIOUS MODE USER
10107	072634	006506			1\$:	MFPI	USP	:PUT USER STACK POINTER ON KERNEL STACK
10108	072636	022706	001100			CMF	#KERSTK,KSP	:WAS SOMETHING PUSHED ON STACK AT 1\$
10109	072642	001407				BEQ	3\$:BRANCH IF NOTHING WAS PUSHED
10110	072644	012601				MOV	(KSP)+,R1	:POP KERNEL STACK INTO R1
10111	072646	012702	000600			MOV	#USESTK,R2	:EXPECTING 600 AS USP
10112	072652	020201				CMF	R2,R1	:DID YOU GET THE RIGHT POINTER?
10113	072654	001403				BEQ	2\$:BRANCH IF YOU DID
10114	072656	104114				ERROR	114	:WRONG THING WAS PUSHED ON STACK
10115	072660	000401				BR	2\$:BRANCH TO NEXT TRY
10116	072662	104115			3\$:	ERROR	115	:NOTHING WAS PUSHED ON THE STACK

```
10117 072664 2$: ;THE FOLLOWING WILL TEST DSTM=' MFPI. BELOW ARE THE
10118 ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10119 ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10120 ;INTO THE F/F'S ON SSRB.
10121 : * D12.00 (001)
10122 : * D12.10 (175)
10123 : * MFP.00 (066)
10124 : MFP.10 (250)
10125 : SVC.80 (222)
10126 : SVC.90 (300)
10127 : FET.00 (217)
10128 072664 012737 072672 001112 MOV #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
10129 072672 012737 030340 177776 12$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
10130 072700 012702 100000 MOV #100000,R2 ;PUT VIRTUAL ADDRESS IN R2
10131 072704 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
10132 072706 006512 MFPI (R2) ;READ FROM PHYSICAL 100000
10133 072710 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
10134 072712 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10135 072714 001401 BEQ 4$ ;BRANCH IF CORRECT DATA WAS FETCHED
10136 072716 1041'6 ERROR 116 ;WRONG DATA WAS FETCHED
10137 072720 4$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE
10138 ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10139 ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10140 ;INTO THE F/F'S ON SSRB.
10141 : * D12.01 (002)
10142 : * D12.10 (175)
10143 : * MFP.00 (066)
10144 : MFP.10 (250)
10145 : SVC.80 (222)
10146 : SVC.90 (300)
10147 : FET.00 (217)
10148 072720 012737 072726 001112 MOV #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$
10149 072726 012737 030340 177776 14$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
10150 072734 012702 100000 MOV #100000,R2 ;PUT VIRTUAL ADDRESS IN R2
10151 072740 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING
10152 072742 006522 MFPI (R2)+ ;READ FROM PHYSICAL 100000
10153 072744 012601 MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
10154 072746 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10155 072750 001401 BEQ 5$ ;BRANCH IF CORRECT DATA WAS FETCHED
10156 072752 104116 ERROR 116 ;WRONG DATA WAS FETCHED
10157 072754 5$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
10158 ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10159 ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10160 ;INTO THE F/F'S ON SSRB.
10161 : D30.00 (003)
10162 : D30.10 (221)
10163 : D10.20 (233)
10164 : * D10.50 (311)
10165 : * D10.60 (177)
10166 : * MFP.00 (066)
10167 : MFP.10 (250)
10168 : SVC.80 (222)
10169 : SVC.90 (300)
10170 : FET.00 (217)
10171 072754 012737 072762 001112 MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
10172 072762 012737 030340 177776 15$: MOV #030340,PSW ;MAKE PREVIOUS MODE USER
```

```
10173 072770 000240      NOP                ;THIS IS A SYNC POINT FOR SCOPING
10174 072772 006537 100000 MFPI #100000      ;READ FROM PHYSICAL 100000
10175 072776 012601      MOV (KSP)+,R1      ;POP KERNEL STACK INTO R1
10176 073000 020001      CMP R0,R1          ;WAS DATA FETCHED SAME AS STORED
10177 073002 001401      BEQ 6$             ;BRANCH IF CORRECT DATA WAS FETCHED
10178 073004 104116      ERROR 116         ;WRONG DATA WAS FETCHED
10179 073006                6$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI. BELOW ARE THE
10180                                ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10181                                ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10182                                ;INTO THE F/F'S ON SSRB.
10183                                : D45.00 (004)
10184                                : * D10.30 (122)
10185                                : * D10.60 (177)
10186                                : * MFF.00 (066)
10187                                : MFP.10 (250)
10188                                : SVC.80 (222)
10189                                : SVC.90 (300)
10190                                : FET.00 (217)
10191 073006 012737 073014 001112 MOV #16$, $LPERR    ;SET LOOP ON ERROR POINTER TO 16$
10192 073014 012737 030340 177776 16$: MOV #030340,PSW     ;MAKE PREVIOUS MODE USER
10193 073022 012702 100002      MOV #100002,R2     ;LOAD VIRTUAL ADDRESS INTO R2
10194 073026 000240      NOP                ;THIS IS A SYNC POINT FOR SCOPING
10195 073030 006542      MFPI -(R2)        ;READ FROM PHYSICAL 100000
10196 073032 012601      MOV (KSP)+,R1      ;POP KERNEL STACK INTO R1
10197 073034 020001      CMP R0,R1          ;WAS DATA FETCHED SAME AS STORED
10198 073036 001401      BEQ 7$             ;BRANCH IF CORRECT DATA WAS FETCHED
10199 073040 104116      ERROR 116         ;WRONG DATA WAS FETCHED
10200 073042 012737 032226 000250 7$: MOV #MMTRAP,MMVEC  ;SET M.M. VECTOR TO NORMAL ROUTINE
10201 073050 012737 072524 001112 MOV #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST
10202 073056 000413      BR TST113         ;BRANCH TO NEXT TEST
10203
10204
10205 073060 013737 177572 001250 10$: MOV MMR0,PMR0      ;SAVE MMR0 FOR ERROR TYPEOUT
10206 073066 013737 177574 001252 MOV MMR1,PMR1      ;SAVE MMR1 FOR ERROR TYPEOUT
10207 073074 013737 177576 001254 MOV MMR2,PMR2      ;SAVE MMR2 FOR ERROR TYPEOUT
10208 073102 104117      ERROR 117         ;TRIED TO READ NON-RESIDENT PAGE
10209 073104 000002      RTI                ;RETURN TO TEST
10210
10211 :*****
10212 :*TEST 113 MOVE FROM PREVIOUS (KERNEL) I-SPACE TO SUPERVISOR MODE
10213 :*
10214 :* THIS TEST CHECKS THAT IF THE PREVIOUS MODE IS KERNEL THE
10215 :* 'KERNEL SPACE (1) L' FLIP-FLOP IS SET AND THE FETCH IS FROM
10216 :* KERNEL MODE.
10217 :* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
10218 :* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
10219 :* THEIR ADDRESSES).
10220 :* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
10221 :* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
10222 :*
10223 :*****
10224 073106                ;TST113:
10225 073106 000004      SCOPE
10226 073110 012737 073552 001316 MOV #TST114,NXTTST ;SAVE STARTING ADDRESS OF NEXT
10227                                ;TEST FOR ESCAPE ON PARITY ERRORS
10228 073116 012700 077406      MOV #77406,R0       ;MAKE ALL SUPER I-SPACE PAGES RESIDENT
```

```

10229                                     ;READ/WRITE, LENGTH 200 BLOCKS
10230 073122 012702 000010      MOV    #10,R2      ;SET COUNT TO LOAD 8 ADDRESSES
10231 073126 012701 172200      MOV    #SIPDR0,R1 ;PUT ADDRESS OF FIRST PDR IN R1
10232 073132 010021          19$: MOV    R0,(R1)+    ;LOAD R0 INTO PDR ADDRESSED BY R1
10233 073134 077202          SOB    R2,19$     ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
10234 073136 012737 073152 001110  MOV    #20$,SLPADR ;SET LOOP POINTER TO 20$
10235 073144 012737 073152 001112  MOV    #20$,SLPERR ;SET LOOP ON ERROR TO 20$
10236 073152 012737 040340 177776 20$: MOV    #040340,PSW ;GO TO SUPERVISOR MODE FOR THIS TEST
10237 073160 012700 077400      MOV    #77400,R0  ;MAKE PAGE 4 NON-RESIDENT IN ALL MODES
10238                                     ;EXCEPT SUPER I-SPACE & KERNEL I-SPACE
10239 073164 010037 172330      MOV    R0,KDPDR4  ;KERNEL D-SPACE PAGE 4
10240 073170 010037 172230      MOV    R0,SDPDR4  ;SUPERVISOR D-SPACE PAGE 4
10241 073174 010037 177630      MOV    R0,UDPDR4  ;USER D-SPACE PAGE 4
10242 073200 010037 177610      MOV    R0,UIPDR4  ;USER I-SPACE PAGE 4
10243 073204 012737 077406 172310  MOV    #77406,KIPDR4 ;KERNEL I-SPACE PAGE 4 READ/WRITE
10244 073212 012737 077406 172210  MOV    #77406,SIPDR4 ;SUPER I-SPACE PAGE 4 READ/WRITE
10245 073220 012737 001000 172350  MOV    #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
10246 073226 012737 001000 172250  MOV    #1000,SIPAR4 ;MAP SUPER I PAGE 4 TO 16K
10247 073234 012700 036514      MOV    #36514,R0  ;LOAD DATA PATTERN INTO R0
10248 073240 010037 100000      MOV    R0,#100000 ;LOAD DATA PATTERN INTO PHY 100000
10249 073244 012702 100000      MOV    #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
10250 073250 012737 073524 000250  MOV    #10$,MMVEC  ;SET M.M. VECTOR TO 10$
10251 073256 105037 172210      CLRB   SIPDR4    ;MAKE SUPER I-SPACE PAGE 4 NON-RESIDENT
10252 073262 012737 040340 177776  MOV    #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
10253 073270 000240          NOP                    ;THIS IS A SYNC POINT FOR SCOPING
10254 073272 006506          1$: MFPI   KSP        ;PUT KERNEL STACK POINTER ON SUPER STACK
10255 073274 022706 000700      CMP    #SUPSTK,SSP ;WAS SOMETHING PUSHED ON STACK AT 1$
10256 073300 001407          BEQ    3$          ;BRANCH IF NOTHING WAS PUSHED
10257 073302 012601          MOV    (SSP)+,R1    ;POP SUPERVISOR STACK INTO R1
10258 073304 012702 001100      MOV    #KERSTK,R2  ;EXPECTING 1100 AS KSP
10259 073310 020201          CMP    R2,R1       ;DID YOU GET THE RIGHT POINTER?
10260 073312 001403          BEQ    2$          ;BRANCH IF YOU DID
10261 073314 104114          ERROR  114       ;WRONG THING WAS PUSHED ON STACK
10262 073316 000401          BR    2$          ;BRANCH TO NEXT TRY
10263 073320 104115          3$: ERROR  115       ;NOTHING PUSHED ON STACK
10264 073322          2$: ;THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
10265                                     ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10266                                     ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED
10267                                     ;INTO THE F/F'S ON SSRB.
10268                                     ; * D12.00 (001)
10269                                     ; * D12.10 (175)
10270                                     ; * MFP.00 (066)
10271                                     ; MFP.10 (250)
10272                                     ; SVC.80 (222)
10273                                     ; SVC.90 (300)
10274                                     ; FET.00 (217)
10275 073322 012737 073330 001112  MOV    #12$,SLPERR ;SET LOOP ON ERROR POINTER TO 12$
10276 073330 012737 040340 177776 12$: MOV    #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER
10277 073336 012702 100000      MOV    #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
10278 073342 000240          NOP                    ;THIS IS A SYNC POINT FOR SCOPING
10279 073344 006512          MFPI   (R2)       ;READ FROM PHYSICAL 100000
10280 073346 012601          MOV    (SSP)+,R1    ;POP SUPER STACK INTO R1
10281 073350 020001          CMP    R0,R1       ;WAS DATA FETCHED SAME AS STORED
10282 073352 001401          BEQ    4$          ;BRANCH IF CORRECT DATA WAS FETCHED
10283 073354 104116          ERROR  116       ;WRONG DATA WAS FETCHED
10284 073356          4$: ;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE

```

```
10285 :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.  
10286 :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED  
10287 :INTO THE F/F'S ON SSRB.  
10288 : * D12.01 (002)  
10289 : * D12.10 (175)  
10290 : * MFP.00 (066)  
10291 : MFP.10 (250)  
10292 : SVC.80 (222)  
10293 : SVC.90 (300)  
10294 : FET.00 (217)  
10295 073356 012737 073364 001112 MOV #14$, $LPERR ;SET LOOP ON ERROR POINTER TO 14$  
10296 073364 012737 040340 177776 14$: MOV #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER  
10297 073372 012702 100000 MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2  
10298 073376 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING  
10299 073400 006522 MFPI (R2)+ ;READ FROM PHYSICAL 100000  
10300 073402 012601 MOV (SSP)+,R1 ;POP SUPER STACK INTO R1  
10301 073404 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED  
10302 073406 001401 BEQ 5$ ;BRANCH IF CORRECT DATA WAS FETCHED  
10303 073410 104116 ERROR 116 ;WRONG DATA WAS FETCHED  
10304 073412 5$: ;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE  
10305 :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.  
10306 :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED  
10307 :INTO THE F/F'S ON SSRB.  
10308 : D30.00 (003)  
10309 : D30.10 (221)  
10310 : D10.20 (233)  
10311 : * D10.50 (311)  
10312 : * D10.60 (177)  
10313 : * MFP.00 (066)  
10314 : MFP.10 (250)  
10315 : SVC.80 (222)  
10316 : SVC.90 (300)  
10317 : FET.00 (217)  
10318 073412 012737 073420 001112 MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$  
10319 073420 012737 040340 177776 15$: MOV #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER  
10320 073426 000240 NOP ;THIS IS A SYNC POINT FOR SCOPING  
10321 073430 006537 100000 MFPI @#100000 ;READ FROM PHYSICAL 100000  
10322 073434 012601 MOV (SSP)+,R1 ;POP SUPERVISOR STACK INTO R1  
10323 073436 020001 CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED  
10324 073440 001401 BEQ 6$ ;BRANCH IF CORRECT DATA WAS FETCHED  
10325 073442 104116 ERROR 116 ;WRONG DATA WAS FETCHED  
10326 073444 6$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI. BELOW ARE THE  
10327 :ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.  
10328 :THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOCKED  
10329 :INTO THE F/F'S ON SSRB.  
10330 : D45.00 (004)  
10331 : * D10.30 (122)  
10332 : * D10.60 (177)  
10333 : * MFP.00 (066)  
10334 : MFP.10 (250)  
10335 : SVC.80 (222)  
10336 : SVC.90 (300)  
10337 : FET.00 (217)  
10338 073444 012737 073452 001112 MOV #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$  
10339 073452 012737 040340 177776 16$: MOV #040340,PSW ;MAKE PREVIOUS MODE KERNEL PRESENT SUPER  
10340 073460 012702 100002 MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
```

10341	073464	000240				NOP		:THIS IS A SYNC POINT FOR SCOPING
10342	073466	006542				MFPI	-(R2)	:READ FROM PHYSICAL 100000
10343	073470	012601				MOV	(SSP)+,R1	:POP SUPERVISOR STACK INTO R1
10344	073472	020001				CMP	R0,R1	:WAS DATA FETCHED SAME AS STORED
10345	073474	001401				BEQ	7\$:BRANCH IF CORRECT DATA WAS FETCHED
10346	073476	104116				ERROR	116	:WRONG DATA WAS FETCHED
10347	073500	012737	032226	000250	7\$:	MOV	#MMTRAP,MMVEC	:SET M.M.VECTOR TO NORMAL ROUTINE
10348	073506	012737	000340	177776		MOV	#00340,PSW	:GO BACK TO KERNEL MODE, PREVIOUS KERNEL
10349	073514	012737	073152	001112		MOV	#20\$, \$LPERR	:SET LOOP POINTER TO START OF TEST
10350	073522	000413				BR	TST114	:BRANCH TO NEXT TEST

10351								
10352								
10353	073524	013737	177572	001250	10\$:	MOV	MMR0,PMR0	:SAVE MMR0 FOR ERROR TYPEOUT
10354	073532	013737	177574	001252		MOV	MMR1,PMR1	:SAVE MMR1 FOR ERROR TYPEOUT
10355	073540	013737	177576	001254		MOV	MMR2,PMR2	:SAVE MMR2 FOR ERROR TYPEOUT
10356	073546	104117				ERROR	117	:TRIED TO READ NON-RESIDENT PAGE
10357	073550	000002				RTI		:RETURN TO TEST

10358
10359
10360
10361
10362
10363
10364
10365
10366
10367
10368
10369
10370
10371
10372
10373
10374

```
*****  
:TEST 114      MFPD (SUPERVISOR) WITH SUPERVISOR D-SPACE ENABLED  
:*
```

```
:* THIS TEST CHECKS THAT 'SSRB IR15 L' CAN INHIBIT THE ASSERTING  
:* OF 'SSRB I SPACEB L' SO THAT THE REFERENCE IS TO D-SPACE IF  
:* THE INSTRUCTION IS MFPD (OR MTPD). [THESE INSTRUCTIONS HAVE  
:* BIT 15 SET IN THE I.R.]  
:* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,  
:* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH  
:* THEIR ADDRESSES).  
:* IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT  
:* WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.  
:*
```

```
*****  
:TST114.  
*****
```

10375	073552					SCOPE		
10376	073552	000004				MOV	#TST115,NXTTST	:SAVE STARTING ADDRESS OF NEXT
10377	073554	012737	074152	001316				:TEST FOR ESCAPE ON PARITY ERRORS
10378								:MAKE ALL KERNEL I-SPACE PAGES RESIDENT
10379	073562	012700	077406			MOV	#77406,R0	:READ/WRITE, LENGTH 200 BLOCKS
10380								:SET COUNT TO LOAD 8 ADDRESSES
10381	073566	012702	000010			MOV	#10,R2	:PUT ADDRESS OF FIRST PDR IN R1
10382	073572	012701	172300			MOV	#KIPDR0,R1	:LOAD R0 INTO PDR ADDRESSED BY R1
10383	073576	010021			19\$:	MOV	R0,(R1)+	:BRANCH BACK TO 19\$ IF R2 IS NOT ZERO
10384	073600	077202				SUB	R2,19\$:SET LOOP POINTER TO 20\$
10385	073602	012737	073616	001110		MOV	#20\$, \$LPADR	:SET LOOP ON ERROR TO 20\$
10386	073610	012737	073616	001112		MOV	#20\$, \$LPERR	:MAKE PAGE 4 IN ALL BUT SUPERVISOR D
10387	073616	012700	077400		20\$:	MOV	#77400,R0	:AND KERNEL I NON-RESIDENT
10388								:KERNEL D-SPACE PAGE 4
10389	073622	010037	172330			MOV	R0,KDPDR4	:SUPERVISOR I-SPACE PAGE 4
10390	073626	010037	172210			MOV	R0,SIPDR4	:USER D-SPACE PAGE 4
10391	073632	010037	177630			MOV	R0,UDPDR4	:USER I-SPACE PAGE 4
10392	073636	010037	177610			MOV	R0,UIPDR4	:KERNEL I-SPACE PAGE 4 READ/WRITE
10393	073642	012737	077406	172310		MOV	#77406,KIPDR4	:SUPER D-SPACE PAGE 4 READ/WRITE
10394	073650	012737	077406	172230		MOV	#77406,SDPDR4	:MAP KERNEL I PAGE 4 TO 16K
10395	073656	012737	001000	172350		MOV	#1000,KIPAR4	:MAP SUPER D PAGE 4 TO 16K
10396	073664	012737	001000	172270		MOV	#1000,SDPAR4	


```
10453 : SVC.80 (222)
10454 : SVC.90 (300)
10455 : FET.00 (217)
10456 074012 012737 074020 001112 : MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
10457 074020 012737 010340 177776 15$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
10458 074026 000240 : NOP ;THIS IS A SYNC POINT FOR SCOPING
10459 074030 106537 100000 : MFPD @#100000 ;READ FROM PHYSICAL 100000
10460 074034 012601 : MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
10461 074036 020001 : CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10462 074040 001401 : BEQ 6$ ;BRANCH IF CORRECT DATA WAS FETCHED
10463 074042 104116 : ERROR 116 ;WRONG DATA WAS FETCHED
10464 074044 6$: ;THE FOLLOWING WILL TEST DSTM=4 MFPD. BELOW ARE THE
10465 : ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10466 : ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10467 : ;INTO THE F/F'S ON SSRB.
10468 : D45.00 (004)
10469 : * D10.30 (122)
10470 : * D10.60 (177)
10471 : * MFP.00 (066)
10472 : MFP.10 (250)
10473 : SVC.80 (222)
10474 : SVC.90 (300)
10475 : FET.00 (217)
10476 074044 012737 074052 001112 : MOV #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
10477 074052 012737 010340 177776 16$: MOV #010340,PSW ;MAKE PREVIOUS MODE SUPERVISOR
10478 074060 012702 100002 : MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
10479 074064 000240 : NOP ;THIS IS A SYNC POINT FOR SCOPING
10480 074066 106542 : MFPD -(R2) ;READ FROM PHYSICAL 100000
10481 074070 012601 : MOV (KSP)+,R1 ;POP KERNEL STACK INTO R1
10482 074072 020001 : CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10483 074074 001401 : BEQ 7$ ;BRANCH IF CORRECT DATA WAS FETCHED
10484 074076 104116 : ERROR 116 ;WRONG DATA WAS FETCHED
10485 074100 012737 032226 000250 7$: MOV #MMTRAP,MMVEC ;SET M.M.VECTOR TO NORMAL ROUTINE
10486 074106 012737 073616 001112 : MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
10487 074114 042737 000002 172516 : BIC #BIT1,MMR3 ;DISABLE SUPERVISOR D-SPACE
10488 074122 000413 : BR TST115 ;BRANCH TO NEXT TEST
10489
10490
10491 074124 013737 177572 001250 10$: MOV MMR0,PMR0 ;SAVE MMR0 FOR ERROR TYPEOUT
10492 074132 013737 177574 001252 : MOV MMR1,PMR1 ;SAVE MMR1 FOR ERROR TYPEOUT
10493 074140 013737 177576 001254 : MOV MMR2,PMR2 ;SAVE MMR2 FOR ERROR TYPEOUT
10494 074146 104117 : ERROR 117 ;TRIED TO READ NON-RESIDENT PAGE
10495 074150 000002 : RTI ;RETURN TO TEST
10496
10497
10498
10499
10500 :*****
10501 :*TEST 115 MFPD (USER/PREV.USER) WITH USER D-SPACE ENABLED
10502 :*
10503 :* THIS TEST CHECKS THAT, IF THE INSTRUCTION IS EITHER MFPD OR
10504 :* MTPD AND BOTH THE PRESENT AND PREVIOUS MODES ARE USER
10505 :* (PS=17XXXX), THEN 'SSRB I SPACEB L' IS NOT ASSERTED AND
10506 :* D-SPACE IS USED IF IT WAS ENABLED. [IN THIS WAY AN OPERATING
10507 :* SYSTEM CAN MAKE SOME PROPRIETARY CODE 'EXECUTE ONLY' FOR
10508 :* THE USER.]
10509 :* THERE IS A DESCRIPTION BEFORE EACH DESTINATION MODE TESTED,
10510 :* WHICH LISTS THE ROM STATES THAT SHOULD COME UP (ALONG WITH
```



```

10509          : *    THEIR ADDRESSES).
10510          : *    IF THE CORRECT MODE IS NOT ENABLED A NON-RESIDENT ABORT
10511          : *    WILL OCCUR AND TRAP TO 10$, WHERE THE ERRORS ARE REPORTED.
10512          : *
10513          : *  *****
10514  074152  000004  TST115: SCOPE
10515  074154  012737  074576  001316  MOV    #TST116,NXTTST ;SET ESCAPE POINTER TO EOP ROUTINE
10516  074162  012700  077406          MOV    #77406,R0      ;MAKE ALL USER I-SPACE PAGES RESIDENT
10517          : *                               ;READ/WRITE, LENGTH 200 BLOCKS
10518  074166  012702  000010          MOV    #10,R2        ;SET COUNT TO LOAD 8 ADDRESSES
10519  074172  012701  177600          MOV    #UIPDR0,R1    ;PUT ADDRESS OF FIRST PDR IN R1
10520  074176  010021          :19$: MOV    R0,(R1)+   ;LOAD R0 INTO PDR ADDRESSED BY R1
10521  074200  077202          SOB    R2,19$        ;BRANCH BACK TO 19$ IF R2 IS NOT ZERO
10522  074202  012737  074216  001110          MOV    #20$,$LPADR   ;SET LOOP POINTER TO 20$
10523  074210  012737  074216  001112          MOV    #20$,$LPERR   ;SET LOOP ON ERROR TO 20$
10524  074216  012737  077406  172310  :20$: MOV    #77406,KIPDR4 ;MAKE KERNEL I PAGE 4 R/W, 200 BLOCKS
10525  074224  012700  077400          MOV    #77400,R0     ;MAKE PAGE 4 IN ALL BUT USER D
10526          : *                               ;AND USER I NON-RESIDENT
10527  074230  010037  172330          MOV    R0,KDPDR4    ;KERNEL D-SPACE PAGE 4
10528  074234  010037  172210          MOV    R0,SIPDR4    ;SUPERVISOR I-SPACE PAGE 4
10529  074240  010037  172230          MOV    R0,SDPDR4    ;SUPERVISOR D-SPACE PAGE 4
10530  074244  012737  077406  177630          MOV    #77406,UDPDR4 ;USER D-SPACE PAGE 4 READ/WRITE
10531  074252  012737  001000  172350          MOV    #1000,KIPAR4 ;MAP KERNEL I PAGE 4 TO 16K
10532  074260  012737  001000  177650          MOV    #1000,UIPAR4 ;MAP USER I PAGE 4 TO 16K
10533  074266  012737  001000  177670          MOV    #1000,UDPAR4 ;MAP USER D PAGE 4 TO 16K
10534  074274  012700  036514          MOV    #36514,R0     ;LOAD DATA PATTERN INTO R0

```

P
C
A
A
B
B
B
B
B
B
B
B
B
B
B
B
B
B
B
C
C
C
C
C
C
C
C
C
C
D

10535 074300 010037 100000
10536 074304 012737 074540 000250
10537 074312 052737 000001 172516
10538 074320 105037 172310
10539 074324 105037 177610
10540 074330
10541
10542
10543
10544
10545
10546
10547
10548
10549
10550
10551 074330 012737 074336 001112
10552 074336 012737 170340 177776 12\$:
10553 074344 012702 100000
10554 074350 000240
10555 074352 006512
10556 074354 012601
10557 074356 020001
10558 074360 001401
10559 074362 104116
10560 074364
10561
10562
10563
10564
10565
10566
10567
10568
10569
10570
10571 074364 012737 074372 001112
10572 074372 012737 170340 177776 14\$:
10573 074400 012702 100000
10574 074404 000240
10575 074406 006522
10576 074410 012601
10577 074412 020001
10578 074414 001401
10579 074416 104116
10580 074420
10581
10582
10583
10584
10585
10586
10587
10588
10589
10590

MOV R0,#100000 ;LOAD DATA PATTERN INTO PHY 100000
MOV #10\$,MMVEC ;SET M.M. VECTOR TO 10\$
BIS #BIT0,MMR3 ;ENABLE USER D-SPACE
CLRB KIPDR4 ;MAKE KERNEL I-SPACE PAGE 4 NON-RESIDENT
CLRB UIPDR4 ;MAKE USER I-SPACE PAGE 4 NON-RESIDENT
2\$:
;THE FOLLOWING WILL TEST DSTM=1 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
: * D12.00 (001)
: * D12.10 (175)
: * MFP.00 (066)
: MFP.10 (250)
: SVC.80 (222)
: SVC.90 (300)
: FET.00 (217)
MOV #12\$,SLPERR ;SET LOOP ON ERROR POINTER TO 12\$
MOV #170340,PSW ;MAKE PREVIOUS MODE USER
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
NOP ;THIS IS A SYNC POINT FOR SCOPING
MFPI (R2) ;READ FROM PHYSICAL 100000
MOV (USP)+,R1 ;POP USER STACK INTO R1
CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
BEQ 4\$;BRANCH IF CORRECT DATA WAS FETCHED
ERROR 116 ;WRONG DATA WAS FETCHED
4\$:
;THE FOLLOWING WILL TEST DSTM=2 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
: * D12.01 (002)
: * D12.10 (175)
: * MFP.00 (066)
: MFP.10 (250)
: SVC.80 (222)
: SVC.90 (300)
: FET.00 (217)
MOV #14\$,SLPERR ;SET LOOP ON ERROR POINTER TO 14\$
MOV #170340,PSW ;MAKE PREVIOUS MODE USER
MOV #100000,R2 ;LOAD VIRTUAL ADDRESS INTO R2
NOP ;THIS IS A SYNC POINT FOR SCOPING
MFPI (R2)+ ;READ FROM PHYSICAL 100000
MOV (USP)+,R1 ;POP USER STACK INTO R1
CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
BEQ 5\$;BRANCH IF CORRECT DATA WAS FETCHED
ERROR 116 ;WRONG DATA WAS FETCHED
5\$:
;THE FOLLOWING WILL TEST DSTM=3 MFPI. BELOW ARE THE
;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
;INTO THE F/F'S ON SSRB.
: D30.00 (003)
: D30.10 (221)
: D10.20 (233)
: * D10.50 (311)
: * D10.60 (177)
: * MFP.00 (066)
: MFP.10 (250)

```
10591 : SVC.80 (222)
10592 : SVC.90 (300)
10593 : FET.00 (217)
10594 074420 012737 074426 001112 : MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
10595 074426 012737 170340 177776 15$: MOV #170340,PSW ;MAKE PREVIOUS MODE USER
10596 074434 000240 : NOP ;THIS IS A SYNC POINT FOR SCOPING
10597 074436 006537 100000 : MFPI @#100000 ;READ FROM PHYSICAL 100000
10598 074442 012601 : MOV (USP)+,R1 ;POP USER STACK INTO R1
10599 074444 020001 : CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10600 074446 001401 : BEQ 6$ ;BRANCH IF CORRECT DATA WAS FETCHED
10601 074450 104116 : ERROR 116 ;WRONG DATA WAS FETCHED
10602 074452 : 6$: ;THE FOLLOWING WILL TEST DSTM=4 MFPI. BELOW ARE THE
10603 : ;ROM STATE NAMES AND ADDRESSES, FROM THE A-FORK.
10604 : ;THE * INDICATES WHEN THE PREVIOUS MODE GETS CLOKED
10605 : ;INTO THE F/F'S ON SSRB.
10606 : D45.00 (004)
10607 : * D10.30 (122)
10608 : * D10.60 (177)
10609 : * MFP.00 (066)
10610 : MFP.10 (250)
10611 : SVC.80 (222)
10612 : SVC.90 (300)
10613 : FET.00 (217)
10614 074452 012737 074460 001112 : MOV #16$, $LPERR ;SET LOOP ON ERROR POINTER TO 16$
10615 074460 012737 170340 177776 16$: MOV #170340,PSW ;MAKE PREVIOUS MODE USER
10616 074466 012702 100002 : MOV #100002,R2 ;LOAD VIRTUAL ADDRESS INTO R2
10617 074472 000240 : NOP ;THIS IS A SYNC POINT FOR SCOPING
10618 074474 006542 : MFPI -(R2) ;READ FROM PHYSICAL 100000
10619 074476 012601 : MOV (USP)+,R1 ;POP USER STACK INTO R1
10620 074500 020001 : CMP R0,R1 ;WAS DATA FETCHED SAME AS STORED
10621 074502 001401 : BEQ 7$ ;BRANCH IF CORRECT DATA WAS FETCHED
10622 074504 104116 : ERROR 116 ;WRONG DATA WAS FETCHED
10623 074506 012737 032226 000250 7$: MOV #MMTRAP,MMVEC ;SET M.M. VECTOR TO NORMAL ROUTINE
10624 074514 012737 074216 001112 : MOV #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
10625 074522 042737 000001 172516 : BIC #BIT0,MMR3 ;DISABLE USER D-SPACE
10626 074530 012737 000340 177776 : MOV #340,PSW ;MAKE PRESENT MODE KERNEL
10627 074536 000413 : BR 21$ ;BRANCH TO EXIT
10628
10629
10630 074540 013737 177572 001250 10$: MOV MMR0,PMR0 ;SAVE MMR0 FOR ERROR TYPEOUT
10631 074546 013737 177574 001252 : MOV MMR1,PMR1 ;SAVE MMR1 FOR ERROR TYPEOUT
10632 074554 013737 177576 001254 : MOV MMR2,PMR2 ;SAVE MMR2 FOR ERROR TYPEOUT
10633 074562 104117 : ERROR 117 ;TRIED TO READ NON-RESIDENT PAGE
10634 074564 000002 : RTI ;RETURN TO TEST
10635
10636 074566 005037 177572 21$: CLR @MMR0 ;DISABLE K^
10637 074572 005037 172516 : CLR @MMR3
10638
10639
10640
10641 :*****
10642 :*TEST 116 CHECK DPARS READ BACK CORRECTLY
10643 :*
10644 :* DATA ERRORS MAY OCCUR WHEN A PAR,PDR, OR MEMORY MANAGEMENT
10645 :* REGISTER TO BE READ IS ADDRESSED USING MEMORY MANAGEMENT.
10646 :* DATA FROM AN INTERNAL REGISTER (SYS SIZE-LO, SYS SIZE-HI, SYS
```

10647 : * ID, OR CPU ERROR) MAY BE ENABLED ONTO THE INTERNAL DATA BUS AT
10648 : * THE SAME TIME AS DATA FROM THE ADDRESSED PAR, PDR, OR
10649 : * MEMORY MANAGEMENT REGISTER. SYMPTOM: INCORRECT DATA READ
10650 : * FROM A PAR, PDR, OR MEMORY MANAGEMENT REGISTER WHEN
10651 : * ADDRESSED USING MEMORY MANAGEMENT. CORRECTION: INSTALL
10652 : * ECO M8140-00002.
10653 : *
10654 : *

10655 074576 000004 : *****
10656 074600 013746 177776 TST116: SCOPE
10657 074604 042716 000020 MOV @#PS, -(SP)
10658 074610 012746 074616 BIC #20, (SP)
10659 074614 000002 MOV #8\$, -(SP)
10660 074616 RTI
10661 074616 012737 075354 001316 8\$: MOV #DONE, NXTTST ;POINT TO NEXT TEST
10662 074624 20\$: CLR MMR0 ;SET CONDITIONS FOR THIS TEST
10663 074624 005037 177572 CLR MMR3 ;MEM MAN OFF
10664 074630 005037 172516 CLR PS ;KERNAL MODE
10665 074634 005037 177776 MOV #1014, @#CONTRL ;DISABLE CACHE
10666 074640 012737 001014 177746 CLR KIPAR0 ;KI VIRTUAL-PHYSICAL
10667 074646 005037 172340 MOV #177406, KIPDR0
10668 074652 012737 177406 172300 MOV #200, KIPAR1
10669 074660 012737 000200 172342 MOV #177406, KIPDR1
10670 074666 012737 177406 172302 MOV #400, KIPAR2
10671 074674 012737 000400 172344 MOV #177406, KIPDR2
10672 074702 012737 177406 172304 MOV #600, KIPAR3
10673 074710 012737 000600 172346 MOV #177406, KIPDR3
10674 074716 012737 177406 172306 MOV #1000, KIPAR4
10675 074724 012737 001000 172350 MOV #177406, KIPDR4
10676 074732 012737 177406 172310 MOV #1200, KIPAR5
10677 074740 012737 001200 172352 MOV #177406, KIPDR5
10678 074746 012737 177406 172312 MOV #1400, KIPAR6
10679 074754 012737 001400 172354 MOV #177406, KIPDR6
10680 074762 012737 177406 172314 MOV #7600, KIPAR7
10681 074770 012737 007600 172356 MOV #177406, KIPDR7
10682 074776 012737 177406 172316 CLR KDPAR0 ;KD VIRTUAL=PHYSICAL
10683 075004 005037 172360 MOV #177406, KDPDR0
10684 075010 012737 177406 172320 MOV #200, KDPAR1
10685 075016 012737 000200 172362 MOV #177406, KDPDR1
10686 075024 012737 177406 172322 MOV #400, KDPAR2
10687 075032 012737 000400 172364 MOV #177406, KDPDR2
10688 075040 012737 177406 172324 MOV #600, KDPAR3
10689 075046 012737 000600 172366 MOV #177406, KDPDR3
10690 075054 012737 177406 172326 MOV #1000, KDPAR4
10691 075062 012737 001000 172370 MOV #177406, KDPDR4
10692 075070 012737 177406 172330 MOV #1200, KDPAR5
10693 075076 012737 001200 172372 MOV #177406, KDPDR5
10694 075104 012737 177406 172332 MOV #1400, KDPAR6
10695 075112 012737 001400 172374 MOV #177406, KDPDR6
10696 075120 012737 177406 172334 MOV #7600, KDPAR7
10697 075126 012737 007600 172376 MOV #177406, KDPDR7
10698 075134 012737 177406 172336 MOV #14, MMR3 ;ENABLE DPARS
10699 075142 012737 000014 172516 MOV #MAGIC, WORK ;BEGIN WITH USER DPARS
10700 075150 012737 075336 075352 1\$: CMP #WORK, WORK ;TESTED ALL THE DPARS?
10701 075156 022737 075352 075352 BLOS 7\$;BRANCH IF YES
10702 075164 101461

10703	075166	012737	000001	177572	MOV	#1,	MMR0		;TURN ON MEM MAN
10704	075174	017703	000152		MOV	@WORK,	R3		;GET ADDRESS OF FIRST DPAR
10705	075200	017700	000146		MOV	@WORK,	R0		
10706	075204	017701	000142		MOV	@WORK,	R1		
10707	075210	062701	000016		ADD	#16,	R1		;POINT TO LAST DPAR
10708	075214	062737	000002	075352	ADD	#2,	WORK		;POINT TO MAGIC NUMBER
10709	075222	017702	000124		MOV	@WORK,	R2		;GET THE MAGIC NUMBER
10710	075226	020103		2\$:	CMP	R1,	R3		;ARE ALL DPARS LOADED?
10711	075230	101402			BLOS	3\$;BRANCH IF YES
10712	075232	010223			MOV	R2,	(R3)+		;LOAD DPAR USING KDPAR7
10713	075234	000774			BR	2\$;DO DPAR0 THRU DPAR6
10714	075236	010003		3\$:	MOV	R0,	R3		;POINT BACK TO DPAR0
10715	075240	020103		4\$:	CMP	R1,	R3		;ARE ALL DPARS READ?
10716	075242	101421			BLOS	5\$;BRANCH IF YES
10717	075244	020213			CMP	R2,	(R3)		;READ THE MAGIC NUMBER BACK
10718	075246	001425			BEQ	6\$;BRANCH IF OK
10719	075250	011301			MOV	(R3),	R1		;GET THE RECEIVED DATA
10720	075252	005037	177572		CLR	MMR0			;TURN OFF MEM MAN
10721	075256	010137	001176		MOV	R1,	\$TMP2		;GET THE RECEIVED DATA
10722	075262	010337	001172		MOV	R3,	\$TMP0		;GET THE DPAR ADDRESS
10723	075266	010237	001174		MOV	R2,	\$TMP1		;GET THE MAGIC NUMBER
10724	075272	005037	172516		CLR	MMR3			;DISABLE DPAR
10725	075276	104145			ERROR	145			;DPAR DID NOT READ CORRECTLY
10726	075300	104400	014365		TYPE	,ECO			;TYPE ECO MESSAGE
10727	075304	000423			BR	DONE			;REPORT ONLY FIRST ERROR
10728	075306	005037	177572	5\$:	CLR	MMR0			;TURN OFF MEM MAN
10729	075312	062737	000002	075352	ADD	#2,	WORK		;POINT TO NEXT PAR0
10730	075320	000716			BR	1\$			
10731	075322	062703	000002	6\$:	ADD	#2,	R3		;POINT TO NEXT DPAR
10732	075326	000744			BR	4\$			
10733	075330	005037	172516	7\$:	CLR	MMR3			;DISABLE DPARS
10734	075334	000407			BR	DONE			
10735									
10736	075336	177660		MAGIC:	.WORD	UDPAR0			;DPARS AND MAGIC NUMBER TABLE
10737	075340	007601			.WORD	7601			
10738	075342	172260			.WORD	SDPAR0			
10739	075344	007655			.WORD	7655			
10740	075346	172360			.WORD	KDPAR0			
10741	075350	007654			.WORD	7654			
10742	075352	000000		WORK:	.WORD	0			;WORK LOCATION
10743									
10744	075354			DONE:					
10745				:					
10746				:					
10747				.SBTTL	MEMORY MANAGEMENT SETUP				
10748				.*					
10749				.*	THIS ROUTINE SETS UP THE KERNEL AND SUPERVISOR PAR'S AND				
10750				.*	PDR'S TO MAP VIRTUAL ADDRESSES TO THE SAME PHYSICAL ADDRESSES.				
10751									
10752	075354			MMSET:					
10753	075354	012703	172340	MOV	#KIPAR0,	R3			;GET ADDRESS OF KIPAR0
10754	075360	012704	172300		MOV	#KIPDR0,	R4		
10755	075364	005005			CLR	R5			
10756	075366	010300		4\$:	MOV	R3,R0			;GET ADDRESS OF KIPAR0 OR SIPARC
10757	075370	012720	000000		MOV	#0,(R0)+			;SETUP
10758	075374	012720	000200		MOV	#200,(R0)+			;THE

```

10759 075400 012720 000400      MOV      #400,(R0)+      ;KERNEL OR SUPERVISOR
10760 075404 012720 000600      MOV      #600,(R0)+      ;PAR'S
10761 075410 012720 001000      MOV      #1000,(R0)+     ;TO MAP
10762 075414 012720 001200      MOV      #1200,(R0)+     ;THE PROGRAM
10763 075420 012720 001400      MOV      #1400,(R0)+     ;TO
10764 075424 012710 177600      MOV      #177600,(R0)    ;ITSELF
10765 075430 010400              MOV      R4,R0           ;GET ADDRESS OF KIPDR0 OR SIF^RO
10766 075432 012701 077406      MOV      #77406,R1       ;GET PDR DATA
10767 075436 012702 000010      MOV      #10,R2          ;SETUP SOB COUNT
10768 075442 010120              SOB      R2,2$           ;INSTRUCTION TO SETUP PDR'S
10769 075444 077202              TST      R5               ;EXECUTE EIGHT TIMES
10770 075446 005705              TST      R5               ;IS SETUP COMPLETE?
10771 075450 001006              BNE      TST117          ;BRANCH IF YES
10772 075452 012703 172240      MOV      #SIPAR0,R3      ;GET ADDRESS OF SUPER PAR0
10773 075456 012704 172200      MOV      #SIPDR0,R4      ;GET ADDRESS OF SUPER PDR0
10774 075462 005205              INC      R5               ;SET PASS COUNT
10775 075464 000740              BR      4$               ;GO SETUP SUPER PAR'S AND PDR'S
    
```

```

*****
:TEST 117      MEMORY MANAGEMENT ABORT
:
:      THIS TEST SETS UP PDR6 TO CAUSE A MEMORY MANAGEMENT ABORT.
:      IF TMCC AERF(1) DOES NOT GO HIGH IT WILL LOOK LIKE THE TRAP FAILED.
:
:      IF TMCC ABORT DOES NOT GO HIGH THE TEST WILL NOT TRAP AT ALL.
:      IF TMCB SEGT DOES NOT GO LOW A TRAP TO 4 WILL OCCUR.
:
:      IF TMCE CACHE BEND DOES NOT GO HIGH A TRAP TO 350 WILL OCCUR.
*****
    
```

```

10787 075466 000004              TST117: SCOPE
10788 075470 012737 075662 001210      MOV      #TST120,$ESCAPE ;SAVE START ADDRESS OF NEXT TEST
10789 075476 012737 075662 001316      MOV      #TST120,NXTTST ;SAVE START ADDRESS OF NEXT TEST
10790 075504 012737 075620 000004      MOV      #3$,@ERRVEC     ;SETUP LOCATION 4
10791 075512 012737 000340 000006      MOV      #PR7,@ERRVEC+2 ;RESTOR ERROR VEC PSW
10792 075520 012737 075650 000250      MOV      #4$,@MMVEC      ;SETUP ABORT VECTOR
10793 075526 012737 075626 000010      MOV      #5$,@RESVEC     ;SETUP LOCATION 10
10794 075534 012737 000340 000012      MOV      #PR7,@RESVEC+2 ;RESTORE RESVEC PSW
10795 075542 012737 075634 000240      MOV      #6$,@240        ;SETUP LOCATION 240
10796 075550 012737 075642 000350      MOV      #7$,@350        ;SETUP LOCATION 350
10797 075556 012737 075572 001112      MOV      #1$, $LPERR     ;SETUP ERROR LOOP
10798 075564 012737 077401 172314      MOV      #77401,@KIPDR6 ;MAKE PAGE 6 CAUSE ABORT
10799 075572 012706 001100      1$:      MOV      #STACK,SP       ;SETUP THE SP
10800 075576 012737 000001 177572      MOV      #BIT0,@MMRO     ;TURN RELOCATION ON
10801 075604 012737 177777 140000      MOV      #-1,@140000    ;EXECUTE ABORT INSTRUCTION
10802              ;ABORT FAILED
10803 075612 005037 177572      CLR      @MMRO           ;TURN RELOCATION OFF
10804 075616 104146              ERROR 146                ;NO KT ABORT
10805              ;TRAPPED TO LOCATION 4
10806 075620 005037 177572      3$:      CLR      @MMRO           ;TURN OFF MM
10807 075624 104150              ERROR 150                ;TRAPPED TO 4
10808 075626 005037 177572      5$:      CLR      @MMRO           ;TURN OFF MM
10809 075632 104151              ERROR 151                ;TRAPPED TO 10
10810 075634 005037 177572      6$:      CLR      @MMRO           ;TURN OFF MM
10811 075640 104152              ERROR 152                ;TRAPPED TO 240
10812 075642 005037 177572      7$:      CLR      @MMRO           ;TURN RELOCATION OFF
10813 075646 104166              ERROR 166
10814 075650 005037 177572      4$:      CLR      @MMRO           ;TURN MM OFF
    
```

```

10815 075654 012737 000012 000010
10816
10817
10818
10819
10820
10821
10822
10823
10824
10825
10826
10827
10828
10829
10830
10831 075662 000004
10832 075664 012737 076022 001210
10833 075672 012737 076022 001316
10834 075700 012737 026252 000004
10835 075706 012737 075754 000250
10836 075714 012737 075730 001112
10837 075722 012737 077404 172314
10838 075730 012706 001100
10839 075734 012737 001001 177572
10840 075742 005037 140000
10841
10842 075746 005037 177572
10843 075752 104153
10844
10845 075754 012737 076016 000250
10846 075762 012737 075770 001112
10847 075770 012706 001100
10848 075774 012737 001001 177572
10849 076002 013737 140000 001172
10850
10851 076010 005037 177572
10852 076014 104155
10853
10854 076016 005037 177572
10855
10856
10857
10858
10859
10860
10861
10862
10863
10864
10865
10866 076022 000004
10867 076024 012737 076202 001316
10868 076032 005037 177766
10869 076036 022737 167777 177760
10870 076044 003456
    
```

```

MOV #12,@WRESVEC
;CONTINUE
*****
*TEST 120 MEMORY MANAGEMENT TRAP
*****
*
* THIS TEST ENSURES THAT THE MEMORY MANAGEMENT TRAP LOGIC WORKS.
* IF TMCA HONOR SEGTF DOES NOT GO LOW OR DOES NOT GET THRU
* TO TMCB BRQ TRUE THE TRAP WILL NOT OCCUR.
*
* IF TMCB SEGT DOES NOT GO LOW BEN13 WILL FAIL TO BRK.20.
* THE FLOW WILL THEN GO TO RTI.60 WHICH MEANS AN ACKNOWLEDGE
* IS NEVER GIVEN AND THE PROCESSOR WILL HANG UP.
*
* AN INSTRUCTION IS THEN EXECUTED THAT CAUSES A MEMORY MANAGEMENT
* TRAP ON THE SOURCE OPERAND BUT NOT ON THE DESTINATION.
*****
TST120: SCOPE
MOV #TST121,$ESCAPF ;SAVE START ADDRESS OF NEXT TEST
MOV #TST121,NXTTST ;SAVE START ADDRESS OF NEXT TEST
MOV #CPUSPUR,@WERRVEC ;RESTORE LOCATION 4
MOV #2$,@MMVEC ;SETUP LOCATION 250
MOV #3$, $LPERR ;SETUP ERROR LOOP
MOV #77404,@KIPDR6 ;SET ACF 4 IN PDR6
3$: MOV #STACK,SP ;INITIALIZE THE SP
MOV #1001,@MMRO ;TURN RELOCATION ON
CLR @140000 ;EXECUTE TRAP TYPE INSTRUCTION
:NO TRAP
CLR @MMRO ;TURN RELOCATION OFF
ERROR 153 ;NO KT TRAP
;TRAP OK, NOW TRY TRAP ON SOURCE NO TRAP ON DESTINATION
2$: MOV #4$,@MMVEC ;SETUP VECTOR
MOV #5$, $LPERR ;SETUP ERROR LOOP
5$: MOV #STACK,SP ;INITIALIZE THE SP
MOV #1001,@MMRO ;INITIALIZE MMRO
MOV @140000,$TMP0 ;EXECUTE TRAP ON SOURCE
:NO TRAP
CLR @MMRO ;TURN RELOCATION OFF
ERROR 155 ;NO TRAP ON SOURCE
;TRAP OK
4$: CLR @MMRO ;TURN RELOCATION OFF
;CONTINUE
*****
*TEST 121 NON EXISTANT MEMORY ABORT
*****
*
* THIS TEST ENSURES THAT A NON EXISTANT MEMORY
* REFERENCE FUNCTIONS PROPERLY.
*
* IF TMCC AERF(1) DOES NOT GO HIGH IT WILL LOOK LIKE THE ABORT FAILED.
* IF TMCC ABORT DOES NOT GO HIGH THE ABORT WILL STILL OCCUR,
* BUT THE ERROR REGISTER WILL NOT BE LOADED.
*****
TST121: SCOPE
MOV #TST122,NXTTST ;SAVE ADDRESS OF NEXT TEST
CLR @CPUERR ;CLEAR ERROR REG
CMP #167777,@$SIZELO-2 MILLION WORDS ON SYSTEM?
BLE TST122 ;BRANCH IF YES
    
```


10891 076172 005737 177766
10892 076176 001401
10893 076200 104161
10894
10895
10896
10897
10898
10899
10900
10901
10902
10903
10904
10905 076202 000004
10906 076204 012737 076524 001316
10907 076212 022737 167777 177760
10908 076220 003446
10909 076222 012737 076302 001112
10910 076230 012737 076320 000250
10911 076236 012737 076336 000004
10912 076244 005737 001360
10913 076250 001406
10914 076252 012737 076336 000250
10915 076260 012737 076330 000004
10916 076266 052737 000007 172314
10917 076274 012737 000060 172516
10918 076302 012706 001100
10919 076306 012737 000001 177572
10920 076314 005037 140100
10921
10922 076320 005037 177572
10923 076324 104162
10924 076326 000403
10925
10926 076330 005037 177572
10927 076334 104165
10928
10929 076336 005037 172516
10930 076342 052737 000007 172314
10931 076350 012737 076414 000250
10932 076356 012737 076414 000004
10933 076364 012737 076372 001112
10934 076372 012737 000001 177572
10935 076400 012706 140336
10936 076404 012737 140000 177774
10937 076412 005016
10938 076414 032737 100000 177572
10939 076422 001407
10940 076424 005037 177572
10941 076430 005037 177774
10942 076434 012706 001100
10943 076440 104163
10944
10945 076442 005037 177774
10946 076446 012737 076506 000250

TST @#CPUERR ;DID REGISTER CLEAR?
BEQ TST122 ;:BRANCH IF YES
ERROR 161 ;NEXM BIT DID NOT CLEAR

*TEST 122 KT BEND
*
* THIS TEST ENSURES THAT TMCE KT BEND GOES LOW ON AN ODD
* ADDRESS ERROR, SL RED, AND NEXM. THIS IS DONE BY EXECUTING
* AN INSTRUCTION FOR EACH OF THESE THREE CASES THAT ALSO
* CAUSES A KT ABORT. THE ABORT SHOULD NOT BE HONORED.
*
* NOTE: ON A KB11-E OR KB11-EM THE KT ABORT SHOULD BE HONORED OVER A
* NEXM TRAP. IF THIS IS A KB11-E/EM THEN THIS FEATURE IS TESTED.

TST122: SCOPE
MOV #TST123,NXTTST ;SAVE ADDRESS OF NEXT TEST
CMF #167777,@#SIZELO;2 MILLION WORDS ON SYSTEM?
BLE 3\$;BRANCH IF YES
MOV #1\$,SLPERR ;SETUP ERROR LOOP
MOV #2\$,@#MMVEC ;SETUP MEMORY VECTOR
MOV #3\$,@#ERRVEC ;SETUP LOCATION 4
TST @#KB11E ;IS THIS A KB11-E OR KB11-EM?
BEQ 20\$;BR IF NOT
MOV #3\$,@#MMVEC ;SETUP MEMORY VECTOR
MOV #21\$,ERRVEC ;SETUP LOCATION 4
20\$: BIS #7,@#KIPDR6 ;SET ACF FIELD TO 7 IN PDR6
MOV #60,@#MMR3 ;SETUP MMR3
1\$: MOV #STACK,SP ;INITIALIZE THE SP
MOV #BIT0,@#MMR0 ;TURN RELOCATION ON
CLR @#140100 ;MAKE A REFERENCE TO NEXM
;FAILURE, KT ABORT CAME IN (KB11-B/C)
2\$: CLR @#MMR0 ;TURN RELOCATION OFF
ERROR 162 ;KT ABORT OCCURRED ON NEXM
BR 3\$;SKIP KB11-E ERROR MESSAGE
;FAILURE, KT ABORT OVER-RIDDEN BY NEXM (KB11-E/EM ONLY)
21\$: CLR @#MMR0 ;TURN RELOCATION OFF
ERROR 165
;NEXM OK, TRY SL RED
3\$: CLR @#MMR3 ;GO BACK TO 18 BIT MODE
BIS #7,@#KIPDR6 ;MAKE KERNEL PAGE 6 NON RESIDENT
MOV #5\$,@#MMVEC ;SETUP MEM VECTOR
MOV #5\$,@#ERRVEC ;SETUP LOCATION 4
MOV #6\$,SLPERR ;SETUP ERROR LOOP
6\$: MOV #BIT0,@#MMR0 ;TURN RELOCATION ON
MOV #140336,KSP ;PUT SP IN RED ZONE PAGE 6
MOV #140000,@#STKLMT ;SET STACK BOUNDARY AT 24K + 400
CLR (SP) ;EXECUT INSTRUCTION TO RED ZONE NON-RESIDENT
5\$: BIT #BIT15,@#MMR0 ;DID KT ABORT FLAG COME ON?
BEQ 10\$;BRANCH IF NO
CLR @#MMR0 ;TURN RELOCATION OFF
CLR @#STKLMT ;
MOV #STACK,SP ;RESTORE THE SP
ERROR 163
;SL RED OK, TRY ODD ADDRESS
10\$: CLR @#STKLMT ;CLEAR THE SL REG
MOV #7\$,@#MMVEC ;SETUP MM VECTOR

```
10947 076454 012737 076514 000004      MOV    #8$, @ERRVEC      ;SETUP LOCATION 4
10948 076462 012737 076470 001112      MOV    #9$, $LPERR      ;SETUP ERROR LOOP
10949 076470 012737 000001 177572 9$:    MOV    #BIT0, @MMRO     ;TURN RELOCATION ON
10950 076476 012706 001100      MOV    #STACK, SP      ;INITIALIZE THE SP
10951 076502 005037 140001      CLR    @#140001        ;EXECUTE ODD ADDRESS AND KT ABORT
10952                                     ;FAILURE, KT ABORT CAME IN
10953 076506 005037 177572 7$:    CLR    @MMRO           ;TURN RELOCATION OFF
10954 076512 104164      ERROR  164             ;KT ABORT OCCURRED ON ODD ADDRESS
10955                                     ;ODD ADDRESS OK
10956 076514 005037 177572 8$:    CLR    @MMRO           ;TURN RELOCATION OFF
10957 076520 005037 177766      CLR    @#CPUERR        ;ENSURE ERROR REG CLEAR
10958                                     ;CONTINUE
10959
10960                                     ;*****
10961                                     ;*TEST 123      SL REGISTER COMPARATOR TEST 2
10962                                     ;*
10963                                     ;* THIS TEST IS THE SAME AS TEST 153 EXCEPT IT TESTS THE ADDRESSES
10964                                     ;* ON EVERY PAGE. THIS IS DONE BY MAPPING I/O PAGE ADDRESSES TO
10965                                     ;* MEMORY IN KERNEL MODE. THIS MAKES THE I/O PAGE INACCESSABLE
10966                                     ;* IN KERNEL MODE SO AN IOT INSTRUCTION IS USED TO RETURN TO
10967                                     ;* SUPERVISOR MODE WHEN THE I/O PAGE IS NEEDED.
10968                                     ;*****
10968 076524 000004      TST123: SCOPE
10969 076526 012737 077266 001316      MOV    #TST124, NXXTST ;SAVE ADDRESS OF NEXT TEST
10970 076534 005037 001176      CLR    $TMP2           ;CLEAR BUFFER OVERFLOW FLAG
10971 076540 012737 003706 001106      MOV    #^D1990, $ICNT  ;SETUP ITERATION COUNT
10972 076546 012737 076554 001110      MOV    #23$, $LPADR    ;SETUP LOOP ADDRESS
10973 076554 012737 077240 001210 23$:   MOV    #22$, $ESCAPE   ;SETUP ESCAPE
10974 076562 012737 077234 000020      MOV    #12$, @IOTVEC   ;SETUP THE IOT VECTOR
10975 076570 012737 040340 000022      MOV    #40340, @IOTVEC+2 ;SETUP THE IOT VEC PSW
10976 076576 012737 077406 172314      MOV    #77406, @KIPDR6 ;ENSURE PDR6 OK
10977 076604 012737 001400 172354      MOV    #1400, @KIPAR6  ;ENSURE PAR6 OK
10978 076612 012737 076736 000004      MOV    #1$, @ERRVEC    ;SETUP ERROR VECTOR
10979 076620 012737 040340 000006      MOV    #40340, @ERRVEC+2 ;SETUP ERRVEC PSW
10980 076626 012737 001600 172356      MOV    #1600, @KIPAR7  ;MAP KERNEL PAGE 7 TO 28K
10981 076634 052737 040000 177776      BIS    #BIT14, @PSW    ;GO TO SUPER MODE
10982 076642 012737 000001 177572      MOV    #BIT0, @MMRO    ;TURN RELOCATION ON
10983 076650 012706 001100      MOV    #STACK, SSP    ;INITIALIZE THE SSP
10984 076654 105037 177777      CLR    @#PSW+1        ;GO BACK TO KERNEL
10985 076660 012703 000400      MOV    #400, R3        ;SET SOB COUNT FOR SL REGISTER
10986 076664 012700 120000      MOV    #120000, R0     ;INITIALIZE ERROR DATA POINTER
10987 076670 005060 000002      CLR    2(R0)          ;INITIALIZE ERROR DATA BUFFER
10988 076674 012701 000340      MOV    #340, R1       ;INITIALIZE YELLOW ZONE ADDRESS
10989 076700 012702 000400 2$:    MOV    #400, R2        ;SET SOB COUNT FOR SP
10990 076704 012705 000340      MOV    #340, R5       ;INITIALIZE SECONDARY STORAGE FOR SP
10991 076710 010506 3$:    MOV    R5, KSP        ;SET THE SSP
10992 076712 011616      MOV    (KSP), (KSP)   ;EXECUTE TEST INSTRUCTION
10993                                     ;NO TRAP. DETERMINE IF THIS IS CORRECT.
10994 076714 020601      CMP    KSP, R1        ;IS ADDRESS > YELL ZONE BOUNDRY?
10995 076716 101075      BHI    5$             ;BRANCH IF YES
10996 076720 001403      BEQ    6$             ;BRANCH IF ADDRESS = YELL ZONE BOUNDRY
10997                                     ;NO TRAP ADDRESS IS LESS THAN YELLOW ZONE BOUNDRY
10998 076722 052710 000010      BIS    #BIT3, (R0)    ;SET ERROR TYPE IN DATA BUFFER
10999 076726 000443      BR    10$            ;GO RECORD DATA
11000                                     ;NO TRAP EQUALS YELLOW ZONE BOUNDRY
11001 076730 052710 000012 6$:    BIS    #12, (R0)     ;SET ERROR TYPE IN DATA BUFFER
11002 076734 000440      BR    10$            ;GO RECORD DATA
```

```

11003
11004      : GOT A TRAP. NOW DETERMINE IF IT IS CORRECT.
11005      : NOW IN SUPER MODE
11006 076736 032737 000004 177766 1$: BIT #BIT2,@#CPUERR ;WAS IT A RED ZONE?
11007 076744 001406      BEQ 8$ ;BRANCH IF NO
11008 076746 020005      CMP R1,R5 ;IS ADDRESS < YELL ZONE BOUNDRY?
11009 076750 101060      BHI 5$ ;BRANCH IF YES
11010 076752 001431      BEQ 10$ ;BRANCH IF ADDRESS - YELL ZONE BOUNDRY
11011      : RED ZONE TRAP ON LEGAL ADDRESS
11012 076754 052710 000002      BIS #BIT1,(R0) ;SET ERROR TYPE IN DATA BUFFER
11013 076760 000426      BR 10$ ;GO RECORD DATA
11014      : NOT A RED ZONE. IS IT A YELLOW ZONE?
11015 076762 032737 000010 177766 8$: BIT #BIT3,@#CPUERR ;IS THIS A YELLOW ZONE TRAP?
11016 076770 001011      BNE 31$ ;BRANCH IF YES
11017 076772 012737 177600 172356      MOV #177600,@#KIPAR7 ;REMAP KERNEL I/O PAGE
11018 077000 005037 177777      CLR @#PSW+1 ;GO BACK TO KERNEL
11019 077004 005037 177572      CLR @#MMRO ;TURN RELOCATION OFF
11020 077010 000137 026252      JMP CPUSPUR
11021 077014 020105      31$: CMP R1,R5 ;IS ADDRESS = YELL ZONE BOUNDRY?
11022 077016 001435      BEQ 5$ ;BRANCH IF YES
11023 077020 101003      BHI 9$ ;BRANCH IF ADDRESS IS < YELL ZONE BOUNDRY
11024      : YELLOW ZONE TRAP ON LEGAL ADDRESS
11025 077022 052710 000006      BIS #6,(R0) ;SET ERROR TYPE IN DATA BUFFER
11026 077026 000403      BR 10$ ;GO RECORD DATA
11027      : YELLOW ZONE TRAP ON RED ZONE ADDRESS
11028 077030 052710 000004      9$: BIS #BIT2,(R0) ;SET ERROR TYPE IN DATA BUFFER
11029 077034 000400      BR 10$ ;GO RECORD DATA
11030
11031      : RECORD ERROR DATA
11032 077036 000004      10$: IOT ;GO TO SUPER
11033 077040 032737 001000 177570      BIT #SW9,@#SWR ;IS LOOP ON ERROR ENABLED?
11034 077046 001405      BEQ 20$ ;BRANCH IF NO
11035 077050 012706 001100      MOV #STACK,SSP ;INITIALIZE THE SSP
11036 077054 105037 177777      CLRB @#PSW+1 ;GO BACK TO KERNEL
11037 077060 000713      BR 3$ ;LOOP
11038 077062 005737 001176      20$: TST $TMP2 ;HAS BUFFER OVERFLOWED?
11039 077066 001011      BNE 5$ ;BRANCH IF YES
11040 077070 005200      INC R0 ;SET POINTER TO HIGH BYTE
11041 077072 113720 177775      MOVB @#STKLM+1,(R0)+ ;SAVE ERROR STACK LIMIT
11042 077076 010520      MOV R5,(R0)+ ;SAVE ERROR SP
11043 077100 020027 157774      CMP R0,#157774 ;HAS BUFFER REACHED PAGE ??
11044 077104 001002      BNE 5$ ;BRANCH IF NO
11045 077106 005237 001176      INC $TMP2 ;SET BUFFER OVERFLOW FLAG
11046
11047      : CONTINUE TEST
11048 077112 062705 000400      5$: ADD #400,R5 ;GO TO NEXT STACK ADDRESS
11049 077116 000004      21$: IOT ;GO TO SUPERVISOR MODE
11050 077120 012706 001100      MOV #STACK,SSP ;RESET THE SSP
11051 077124 005037 177766      CLR @#CPUERR ;CLEAR ERROR REGISTER
11052 077130 105037 177777      CLRB @#PSW+1 ;GO BACK TO KERNEL
11053 077134 005302      DEC R2 ;REPLACES A
11054 077136 001264      BNE 3$ ;SOB
11055 077140 000004      IOT ;GO TO SUPERVISOR MODE
11056 077142 062701 000400      16$: ADD #400,R1 ;SET NEXT YELLOW ZONE ADDRESS
11057 077146 062737 000400 177774      ADD #400,@#STKLMT ;GO TO NEXT SL ADDRESS
11058 077154 105037 177777      CLRB @#PSW+1 ;GO BACK TO KERNEL

```

```

11059 077160 005303          DEC      R3          ;THIS REPLACES
11060 077162 001246          BNE      2$          ;A SOB
11061
11062          ;DONE WITH TEST.  WAS THERE AN ERROR?
11063 077164 000004          IOT          ;GO TO SUPERVISOR MODE
11064 077166 012737 177600 172356  MOV      #177600,@#KIPAR7 ;RESTORE KERNEL I/O PAGE
11065 077174 105037 177777          CLR      @#PSW+1      ;GO TO KERNEL MODE
11066 077200 005037 177774          CLR      @#STKLMT     ;RESET THE SL REG
11067 077204 012706 001100          MOV      #STACK,SP    ;AND SP
11068 077210 012737 026252 000004  MOV      #CPUSPUR,@#ERRVEC ;RESTORE ERRVEC
11069 077216 005737 120002          TST      @#120002     ;WAS THERE AN ERROR?
11070 077222 001406          BEQ      22$          ;BRANCH IF NO
11071 077224 010037 001156          MOV      R0,$REGO     ;SAVE ERROR DATA POINTER
11072 077230 104167          ERROR    167         ;STACK LIMIT COMPARATORS FAILED
11073 077232 000402          BR       22$
11074          ;IOT ROUTINE
11075 077234 000176 000000 12$: JMP      @($SP)      ;RETURN IN SUPER MODE
11076
11077          ;TEST FINISHED, CLEAN UP VECTORS
11078 077240 005037 177572 22$: CLR      @#MMRO          ;TURN RELOCATION OFF
11079 077244 012737 025474 000020  MOV      #$$SCOPE,@#IOTVEC ;RESTORE IOT VECTOR
11080 077252 012737 000340 000022  MOV      #PR7,@#IOTVEC+2
11081 077260 012737 000340 000006  MCV     #PR7,@#ERRVEC+2
11082          ;CONTINUE
11083          ;*****
11084          ;*TEST 124      PS RESTORE
11085          ;*
11086          ;* THIS TEST ENSURES THAT BEN6 WORKS ON A PS RESTORE.
11087          ;* THIS IS DONE BY SETTING THE STACK TO A NON RESIDENT PAGE
11088          ;* AND DOING A TRAP INSTRUCTION.  WHEN THE PROCESSOR TRYS TO
11089          ;* PUSH THE OLD PSW ON THE STACK A KT ABORT WILL OCCUR.
11090          ;* SINCE IT WAS A KERNEL R6 OPERATION THIS WILL CAUSE BEN13
11091          ;* TO GO TO STATE SER.00 WHICH WILL PUSH THE PSW AND PC INTO
11092          ;* LOCATIONS 2 AND 0, AND THEN TRAP TO LOCATION 4.
11093          ;* THE PSW IN LOCATION 2 SHOULD BE THE PSW BEFORE THE
11094          ;* TRAP INSTRUCTION AND NOT THE PSW IN THE TRAP VECTOR.
11095          ;*****
11096 077266 000004          TST124: SCOPE
11097 077270 012737 077372 001316  MOV      #CACHE,NXTTST
11098 077276 012737 077350 000004  MOV      #1$,@#ERRVEC   ;SETUP ERROR VECTOR
11099 077304 012737 000340 000036  MOV      #PR7,@#TRAPVEC+2 ;ENSURE PR7 IN LOCATION 36
11100 077312 042737 000007 172314  BIC      #7,@#KIPDR6    ;MAKE PAGE 6 NON-RESIDENT
11101 077320 012737 077326 001112  MOV      #2$,$LPERR     ;SETUP ERROR LOOP
11102 077326 005037 000002          CLR      @#2           ;ENSURE LOCATION 2 CLEAR
11103 077332 012737 000001 177572  MOV      #BIT0,@#MMRO   ;TURN RELOCATION ON
11104 077340 012706 150004          MOV      #150004,$SP    ;SETUP THE SP
11105 077344 000236          SPL      6             ;SET CPU PRIORITY AT 6
11106 077346 104400          TRAP          ;EXECUTE TRAP INSTRUCTION
11107 077350 005037 177572          CLR      @#MMRO        ;TURN RELOCATION OFF
11108 077354 012706 001100          MOV      #STACK,SP     ;RETSORE SP
11109 077360 022737 000310 000002  CMP      #310,@#2      ;DID CORRECT PSW GET STACKED?
11110 077366 001401          BEQ      CACHE        ;BRANCH IF YES
11111 077370 104170          ERROR    170         ;BEN6 FAILED ON PS RESTORE
11112 077372          CACHE:
11113
11114          ;*****
    
```


EM131	013335	1186	2155#
EM132	013416	1192	2164#
EM133	013462	1198	2170#
EM134	013530	1203	2177#
EM135	013572	1209	2183#
EM136	013632	1215	2189#
EM137	013763	1222	2204#
EM14	004503	714	1530#
EM140	014112	1228	2220#
EM141	014160	1234	2227#
EM142	014222	1240	2233#
EM143	014270	1246	2240#
EM145	014332	1258	2246#
EM146	014461	1264	2262#
EM147	014637	1270	2281#
EM15	004537	720	1535#
EM150	014671	1276	2286#
EM151	014775	1282	2298#
EM152	015044	1288	2305#
EM153	015110	1294	2311#
EM155	015217	1306	2324#
EM156	015337	1312	2339#
EM16	004613	726	1543#
EM160	015472	1324	2355#
EM161	015600	1330	2367#
EM162	015651	1336	2374#
EM163	015730	1342	2382#
EM164	016005	1348	2390#
EM165	016070	1354	2399#
EM166	016154	1360	2408#
EM167	016231	1366	2417#
EM17	004646	732	1548#
EM170	016425	1372	2440#
EM171	016521	1378	2451#
EM2	003537	650	1441#
EM20	004714	738	1555#
EM201	016546	1385	2455#
EM202	016623	1393	2463#
EM203	016714	1402	2473#
EM21	004755	744	1561#
EM22	005017	751	1567#
EM23	005061	757	1573#
EM24	005125	763	1580#
EM25	005164	769	1586#
EM26	005242	775	1594#
EM27	005310	781	1601#
EM3	003620	656	1450#
EM30	005347	787	1607#
EM31	005436	793	1617#
EM32	005477	799	1623#
EM32M	005606	1252	1635#
EM33	005711	805	1647#
EM34	005770	811	1655#
EM35	006054	817	1664#
EM36	006142	823	1673#
EM37	006211	830	1680#

PDP-11/70-74MP MEMORY MANAGEMENT DIAGNOSTIC		MACY11 30A(1052)		D 3		05-SEP-79 15:36		PAGE 223								
CEKBFDP.P11 15-AUG-79 10:06		CROSS REFERENCE TABLE -- USER SYMBOLS										SEQ 0236				
ER200	003274	1382#	3459	3462												
FCAC =	000400	425#														
FSTTST	001222	551#	4598*	4600*	4602*	4604*	4606*	4608*	4610*	4612*	4692					
FVPE =	002000	423#														
GNS =	***** J	451	3175	3182	3362	3368	3390	3396	3873	3874	3875	3876	3877	3878		
		3879	3880	3881	4556	4560	4564	4573	4581	4669						
HIADRS=	177742	155#	4314													
HITMIS=	177752	159#	440#	4318												
HOLFLG	001304	579#	6629*	6647	6652*	6665	6752	6755*	6760*	6799*	6810	6815*	6828	6911		
		6914*	6919*													
HT =	000011	43#	3664	3702												
IOTVEC=	000020	141#	4623*	4624*	10974*	10975*	11079*	11080*								
IVSS =	040000	419#														
KBTST	032602	4487#	4644													
KB11CM	001362	619#	4487*	4535*	4677	6015	6064	6112								
KB11E	001360	617#	4488*	4492*	4531	4533*	4675	4683	4842	4851	6017	6066	6114	10912		
KB11EM	001361	618#														
KDPAR0=	172360	313#	8054*	8193*	8480*	8486*	8663*	10683*	10740							
KDPAR1=	172362	314#	8055*	8664*	10685*											
KDPAR2=	172364	315#	8665*	10687*												
KDPAR3=	172366	316#	8666*	10689*												
KDPAR4=	172370	317#	8056*	8194*	8478*	8667*	10691*									
KDPAR5=	172372	318#	8668*	10693*												
KDPAR6=	172374	319#	8669*	10695*												
KDPAR7=	172376	320#	5299	5517	5532	5891	8058*	8196*	8670*	10697*						
KDPDRO	172320	291#	8049*	8188*	8732	8849	8990	9109	9136	9140*	9143*	9156*	9159*	9176*		
		9179*	9196*	9199*	9255	9275*	9291*	9311*	9331*	9398	9410*	9426*	9446*	9466*		
		10684*														
KDPDR1=	172322	292#	8050*	8189*	9141*	9157*	9177*	9197*	9276*	9292*	9312*	9332*	9411*	9427*		
		9447*	9467*	10686*												
KDPDR2=	172324	293#	8052*	8191*	10688*											
KDPDR3=	172326	294#	8053*	8192*	10690*											
KDPDR4=	172330	295#	8051*	8190*	8477*	9548*	9687*	9833*	9964*	10093*	10239*	10389*	10527*	10692*		
KDPDR5=	172332	296#	10694*													
KDPDR6=	172334	297#	10696*													
KDPDR7=	172336	298#	5395	5704	5719	6025	8057*	8195*	9142*	9144*	9158*	9160*	9178*	9180*		
		9198*	9200*	9277*	9293*	9313*	9333*	9412*	9428*	9448*	9468*	10698*				
KERSTK=	001100	30#	4617	4662	4957	4964	7475	8487	9565	10108	10258					
KERVEC	032456	4446#	7730	7920												
KIPAR0=	172340	302#	3564	3935	3969	5295	5498	5500	5514	5515	5529	5882	6158	6159*		
		6162	6167	6171	6424	6439*	6972*	8041*	8543*	8655*	9530*	10667*	10753			
KIPAR1=	172342	303#	6440*	6973*	8042*	8544*	8656*	9531*	10669*							
KIPAR2=	172344	304#	6441*	6974*	8043*	8545*	8657*	9532*	10671*							
KIPAR3=	172346	305#	6442*	6975*	8044*	8546*	8658*	9533*	10673*							
KIPAR4=	172350	306#	2986	6446*	6460*	6474*	6488*	6502*	6516*	6530*	6544*	6558*	6572*	6586*		
		6600*	6632*	6660*	6685*	6709*	6721*	6739	6802*	6824*	6843*	6867*	6879*	6898		
		6980*	7033*	7087*	7140	7163*	7208*	7271*	7326*	7375*	7471*	7518*	8187*	8553*		
		8594*	8659*	9554*	9693*	9839*	9970*	10099*	10245*	10395*	10531*	10675*				
KIPAR5=	172352	307#	6633*	6649	6661*	6663	6754	6803*	6812	6825*	6826	6844*	6913	7088*		
		7130*	7162*	7207*	7270*	7325*	7374*	7472*	8660*	10677*						
KIPAR6	172354	308#	8661*	10679*	10873*	10977*										
KIPAR7=	172356	309#	3983	6443*	6976*	7416	8045*	8547*	8662*	9534*	10681*	10980*	11017*	11064*		
KIPDRO	172300	280#	4512*	4513	4515*	5391	5685	5687	5701	5702	5716	6010	6263	6264*		
		6267	6272	6276	6422	6436	6967*	8036*	8538*	8709	8736	8740*	8743*	8748		
		8756*	8759*	8776*	8779*	8796*	8799*	8853	8873*	8881	8889*	8909*	8929*	8994		
		9006*	9014	9022*	9042*	9062*	9132	9148	9251	9283	9394	9418	9525*	9541		

P
C
A
B
B
C
C
O
D
E
D
E
E

E
L
T
E
R
N
E
T
I
V
E
S
T
M
E
N
T
S

TST111	072064	9821	9932	9958#	
TST112	072514	9960	10063	10087#	
TST113	073106	10089	10202	10224#	
TST114	073552	10226	10350	10375#	
TST115	074152	10377	10488	10514#	
TST116	074576	10515	10655#		
TST117	075466	10771	10787#		
TST12	042032	4994	5055#		
TST120	075662	10788	10789	10831#	
TST121	076022	10832	10833	10866#	
TST122	076202	10867	10870	10892	10905#
TST123	076524	10906	10968#		
TST124	077266	10969	11096#		
TST13	042464	5057	5168#		
TST14	042640	5170	5203	5216#	
TST15	043004	5218	5283#		
TST16	043140	5285	5304	5320#	
TST17	043244	5322	5336	5352#	
TST2	041012	4727	4739	4756#	
TST20	043350	5354	5368	5384#	
TST21	043454	5386	5400	5416#	
TST22	043560	5418	5432	5448#	
TST23	043664	5450	5464	5492#	
TST24	044074	5494	5536	5555#	
TST25	044304	5557	5599	5617#	
TST26	044514	5619	5661	5679#	
TST27	044724	5681	5723	5741#	
TST3	041146	4758	4798#		
TST30	045134	5743	5785	5803#	
TST31	045344	5805	5847	5874#	
TST32	045474	5876	5899	5917#	
TST33	045624	5919	5942	5960#	
TST34	045754	5962	5985	6002#	
TST35	046132	6004	6033	6051#	
TST36	046310	6053	6082	6099#	
TST37	046466	6101	6130	6152#	
TST4	041212	4800	4826#		
TST40	046604	6154	6187#		
TST41	046722	6189	6222#		
TST42	047040	6224	6257#		
TST43	047156	6259	6292#		
TST44	047274	6294	6327#		
TST45	047412	6329	6367#		
TST46	047514	6369	6408#		
TST47	051130	6410	6625#		
TST5	041334	4828	4870#		
TST50	051760	6627	6730	6786#	
TST51	052634	6788	6888	6957#	
TST52	053166	6959	7027#		
TST53	053360	7029	7080#		
TST54	053644	7082	7157#		
TST55	053774	7159	7202#		
TST56	054240	7204	7265#		
TST57	054460	7267	7320#		
TST6	041412	4872	4898#		
TST60	054656	7322	7369#		

\$OVER	025742	3239	3257	3265	3275	3283#								
\$PASS	001100	507#	3165*	3166*	3177	3217	3271	3287						
\$PWAD	030512	3917#												
\$PWADN	030370	3888#	3912	4629										
\$PWARMG	030506	3915#												
\$PWUP	030436	3897	3902#											
\$QUES	001216	548#	3352											
\$RDCHR=	***** U	3878												
\$RDDEC=	***** U	3878												
\$RDLIN=	***** U	3878												
\$RDOCT=	***** U	3878												
\$REGAD	001154	531#												
\$REGO	001156	533#	2953	2961	2963	2965	2967	2969	2971	2973	2979	2984	2991	3019
		3023	3029	3034	3036	3044	3045	3047	3306*	11071*				
\$REG1	001160	534#	2940	2942	2948	2961	2965	2967	2971	2973	2982	2988	2991	3001
		3004	3011	3017	3021	3023	3027	3034	3036	3045	3047	3307*		
\$REG2	001162	535#	2942	2944	2946	2949	2953	2961	2967	2971	2973	2979	3013	3021
		3036	3047	3308*										
\$REG3	001164	536#	2946	2979	2988	3017	3019	3036	3309*					
\$REG4	001156	537#	3036	3047	3310*									
\$REG5	001170	538#	3311*											
\$RESRE	027406	3618#	3379											
\$RTRN	025460	3193	3212#	4635	4637*									
\$SAVRE	027350	3602#	7878											
\$SAVR6	030522	3896*	3902	3903*	3904*	3921#								
\$SCOPE	025474	3235#	4623	11079										
\$SETUP=	000037	444#	3163	3329	4665									
\$SIZE	030570	3951#	4550											
\$SIZEX	031036	3987	3997#											
\$STUP =	177777	444#												
\$SVLAD	025714	3247	3278#											
\$SVPC -	000240	491#	496											
\$SWR	177400	1#	11	16	17	18	19	20	21	22	23	545	546	547
		3157	3164	3191	3204	3217	3227	3228	3229	3230	3231	3238	3250	3252
		3253	3258	3259	3260	3267	3268	3269	3280	3283	3286	3294	3295	3296
		3297	3298	3299	3315	3318	3325	3329	3335	3352	4727	4760	4802	4830
		4874	4902	4928	4956	4996	5059	5172	5220	5287	5324	5356	5388	5420
		5452	5496	5559	5621	5683	5745	5807	5878	5921	5964	6006	6055	6103
		6156	6191	6226	6261	6296	6331	6371	6412	6629	6790	6961	7031	7084
		7161	7206	7269	7324	7373	7410	7469	7518	7568	7609	7671	7766	7863
		7948	8029	8186	8267	8370	8467	8531	8651	8840	8973	9106	9241	9376
		9518	9685	9823	9962	10091	10228	10379	10515	10656	10788	10832	10867	10906
		10969	11097											
\$SWRMK -	000200	1#	23	24	3231	3232	3254	3255						
\$TBIT	025466	3208*	3217#	4638*										
\$TIMES	001206	545#	3164*	3267*	3274	3277*	3286	4632*	5878*	5921*	5964*	6006*	6055*	6103*
		6961*	7031*	7948*										
\$TKB	001142	524#												
\$TKS	001140	523#	4621											
\$TMP0	001172	539#	2995	3006	3015	3039	3042	3370*	3380	3398*	3408	5126*	5136	5140*
		5150	5520*	5527*	5583*	5590*	5645*	5652*	5707*	5714*	5769*	5776*	5831*	5838*
		6445*	6457	6459*	6471	6473*	6485	6487*	6499	6501*	6513	6515*	6527	6529*
		6541	6543*	6555	6557*	6569	6571*	6583	6585*	6597	6599*	6611	6644*	6659
		6807*	6822	6847*	6851	7486*	7635*	7644	8558*	8559	8565*	8566	8572*	8573
		8580*	8581	8588*	8589	8595*	8596	8602*	8603	8610*	8611	8714*	8764*	8784*
		8804*	8807	8847*	8897*	8917*	8937*	8940	8980*	9030*	9050*	9070*	9073	9114*

		9164*	9184*	9204*	9207	9249*	9299*	9319*	9339*	9342	9384*	9434*	9454*	9474*
\$TMP1	001174	9477	10722*	10849*	10887*									
\$TMP2	001176	540#	2975	2977	3006	3039	3042	6754*	6913*	7636*	7641	10723*	10886*	
\$TMP3	001200	541#	2975	3006	3039	6649*	6812*	7637*	7638	10721*	10970*	11038	11045*	
\$TMP4	001202	542#	7634*	7640*	7643*	7646*	7647	7807*	7812*	7819*	7823*	7826*	7827	
\$TMP5	001204	543#												
		544#	5305*	5337*	5369*	5401*	5433*	5465*	5537*	5600*	5662*	5724*	5786*	5848*
\$TN	- 000125	5900*	5943*	5986*	6034*	6083*	6131*							
		1#	11	4693	4711	4727#	4739	4747	4757	4760#	4790	4799	4802#	4813
		4827	4830#	4860	4871	4874#	4889	4899	4902#	4916	4925	4928#	4943	4953
		4956#	4965	4980	4993	4996#	5038	5056	5059#	5156	5169	5172#	5203	5206
		5217	5220#	5271	5284	5287#	5304	5308	5321	5324#	5336	5340	5353	5356#
		5368	5372	5385	5388#	5400	5404	5417	5420#	5432	5436	5449	5452#	5464
		5477	5493	5496#	5536	5540	5556	5559#	5599	5603	5618	5621#	5661	5665
		5680	5683#	5723	5727	5742	5745#	5785	5789	5804	5807#	584-	5860	5875
		5878#	5899	5903	5918	5921#	5942	5946	5961	5964#	5985	5989	6003	6006#
		6033	6038	6052	6055#	6082	6086	6100	6103#	6130	6143	6153	6156#	6178
		6188	6191#	6213	6223	6226#	6248	6258	6261#	6283	6293	6296#	6318	6328
		6331#	6354	6368	6371#	6399	6409	6412#	6417	6614	6626	6629#	6730	6775
		6787	6790#	6888	6946	6958	6961#	6962	7016	7028	7031#	7070	7081	7084#
		7147	7158	7161#	7183	7203	7206#	7253	7266	7269#	7310	7321	7324#	7359
		7370	7373#	7393	7407	7410#	7451	7466	7469#	7503	7515	7518#	7551	7565
		7568#	7593	7606	7609#	7655	7668	7671#	7724	7745	7763	7766#	7840	7860
		7863#	7914	7934	7945	7948#	8014	8026	8029#	8031	8171	8183	8186#	8251
		8264	8267#	8354	8367	8370#	8454	8464	8467#	8517	8528	8531#	8533	8616
		8632	8648	8651#	8821	8837	8840#	8954	8970	8973#	9087	9103	9106#	9222
		9238	9241#	9357	9373	9376#	9502	9515	9518#	9520	9659	9669	9682	9685#
		9795	9805	9820	9823#	9932	9944	9959	9962#	10063	10074	10088	10091#	10202
		10211	10225	10228#	10350	10361	10376	10379#	10488	10498	10515#	10641	10656#	10771
		10776	10788#	10817	10832#	10856	10867#	10892	10894	10906#	10959	10969#	11083	11097#
\$TPB	001146	526#	3691*	3702										
\$TPFLG	001153	530#	3652	3702										
\$TPS	001144	525#	3689	3702										
\$TRAP	030326	3857#	4627											
\$TRP =	000022	3864#	3874#	3875#	3876#	3877#	3878#	3879#	3880#	3881#	3882#			
\$TRPAD	030346	3861	3872#											
\$TSTNM	001102	508#	3163*	3226	3256	3278*	3283	3287	3304	3314	3352	3371	3399	4725*
		4726	4999*	5000	5290*	5291	6420*	6421	6965*	6966	8034*	8035	8536*	8537
		9523*	9524											
\$TYPBN-	***** U	3878												
\$TYPDS	030102	3793#	3877											
\$TYPE	027444	3652#	3864	3873										
\$TYPEC	027604	3671	3678	3684	3689#	3690								
\$TYPEX	027652	3695	3697	3700#										
\$TYPOC	027700	3733#	3874											
\$TYPON	027714	3732	3735#	3876										
\$TYPOS	027654	3728#	3875											
\$XTSTR	025512	3241#												
\$\$GET4=	000001	3191#	3196											
\$\$TRP =	000002	3863#	3874	3875	3876	3877	3878	3879	3880	3881	3882			
\$\$TSTN	001104	510#	3371*	3376	3399*	3404								
\$OFILL	030077	3729*	3733*	3743	3778#									
	- 101404	447#	451	453#	491	492#	494#	496#	504#	551	2926#	3176#	3217	3219#
		3286	3287	3352	3363#	3369#	3391#	3397#	3543#	3702	3847#	3899	3920	4061#
		4557#	4561#	4574#	4582#	4596#	4639	4640	8146*	8152	11120	11123#	11125#	

MSG15	5271#	5273
MSG16	5308#	5310
MSG161	10776#	10778
MSG162	10817#	10819
MSG163	10856#	10858
MSG164	10894#	10896
MSG165	10959#	10961
MSG166	11083#	11085
MSG17	5340#	5342
MSG2	4747#	4749
MSG20	5372#	5374
MSG200	10641#	10643
MSG21	5404#	5406
MSG22	5436#	5438
MSG23	5477#	5479
MSG24	5540#	5542
MSG25	5603#	5605
MSG26	5665#	5667
MSG27	5727#	5729
MSG3	4790#	4792
MSG30	5789#	5791
MSG31	5860#	5862
MSG32	5903#	5905
MSG33	5946#	5948
MSG34	5989#	5991
MSG35	6038#	6040
MSG36	6086#	6088
MSG37	6143#	6145
MSG4	4813#	4815
MSG40	6178#	6180
MSG41	6213#	6215
MSG42	6248#	6250
MSG43	6283#	6285
MSG44	6318#	6320
MSG45	6354#	6356
MSG46	6399#	6401
MSG47	6614#	6616
MSG5	4860#	4862
MSG50	6775#	6777
MSG512	6946#	6948
MSG53	7070#	7072
MSG54	7147#	7149
MSG55	7183#	7185
MSG56	7253#	7255
MSG57	7310#	7312
MSG6	4889#	4891
MSG60	7359#	7361
MSG61	7393#	7395
MSG62	7451#	7453
MSG63	7503#	7505
MSG64	7551#	7553
MSG65	7593#	7595
MSG66	7655#	7657
MSG67	7745#	7747
MSG7	4916#	4918
MSG70	7840#	7842

7018

MSG71	7934#	7936													
MSG72	8014#	8016													
MSG73	8171#	8173													
MSG74	8251#	8253													
MSG75	8354#	8356													
MSG76	8454#	8456													
MSG77	8517#	8519													
MULT	1#	415#													
NEWTST	1#	415#	4711	4747	4790	4813	4860	4889	4916	4943	4980	5038	5156	5206	5271
	5308	5340	5372	5404	5436	5477	5540	5603	5665	5727	5789	5860	5903	5946	5989
	6038	6086	6143	6178	6213	6248	6283	6318	6354	6399	6614	6775	6946	7016	7070
	7147	7183	7253	7310	7359	7393	7451	7503	7551	7593	7655	7745	7840	7934	8014
	8171	8251	8354	8454	8517	8632	8821	8954	9087	9222	9357	9502	9669	9805	9944
	10074	10211	10361	10498	10641	10776	10817	10856	10894	10959	11083				
PAGEF	1#	6979	7032												
PARMSG	5903#	5905	5948												
PDRMSG	6038#	6040	6088												
POP	1#	415#	3623	3834	3906										
PUSH	1#	415#	3603	3793	3890										
RESPMG	5308#	5310	5342	5374	5406	5438									
RESPON	1#	5292	5324	5356	5388	5420	5452								
SAVEAD	1#	10788	10832												
SAVTST	1#	3304													
SCOPE	34#	4757	4799	4827	4871	4899	4925	4953	4993	5056	5169	5217	5284	5321	5353
	5385	5417	5449	5493	5556	5618	5680	5742	5804	5875	5918	5961	6003	6052	6100
	6153	6188	6223	6258	6293	6328	6368	6409	6626	6787	6958	7028	7081	7158	7203
	7266	7321	7370	7407	7466	7515	7565	7606	7668	7763	7860	7945	8026	8183	8264
	8367	8464	8528	8648	8837	8970	9103	9238	9373	9515	9682	9820	9959	10088	10225
	10376	10514	10655	10787	10831	10866	10905	10968	11096	11115					
SETACF	1#	7161	7206	7269	7324	7373									
SETTRA	3864#	3874	3875	3876	3877	3878	3879	3880	3881						
SETUP	1#	415#													
SKIP	1#	415#	4693	4739	4965	5203	5304	5336	5368	5400	5432	5464	5536	5599	5661
	5723	5785	5847	5899	5942	5985	6033	6082	6130	6730	6888	7724	7914	8616	9659
	9795	9932	10063	10202	10350	10488	10771	10892	11110						
SLASH	1#	415#													
SPACE	415#														
SSCOPE	1#	3236													
STARS	1#	415#	443	471	497	624	3149	3220	3287	3357	3584	3630	3702	3780	3848
	3883	3928	4022	4542	4547	4711	4721	4747	4755	4790	4797	4813	4825	4860	4869
	4889	4897	4916	4923	4943	4951	4980	4991	5038	5054	5156	5167	5206	5215	5271
	5282	5308	5319	5340	5351	5372	5383	5404	5415	5436	5447	5477	5491	5540	5554
	5603	5616	5665	5678	5727	5740	5789	5802	5860	5873	5903	5916	5946	5959	5989
	6001	6038	6050	6086	6098	6143	6151	6178	6186	6213	6221	6248	6256	6283	6291
	6318	6326	6354	6366	6399	6407	6614	6624	6775	6785	6946	6956	7016	7026	7070
	7079	7147	7156	7183	7201	7253	7264	7310	7319	7359	7368	7393	7405	7451	7464
	7503	7513	7551	7563	7593	7604	7655	7666	7745	7761	7840	7858	7934	7943	8014
	8024	8171	8181	8251	8262	8354	8365	8454	8462	8517	8526	8632	8646	8821	8835
	8954	8968	9087	9101	9222	9236	9357	9371	9502	9513	9669	9680	9805	9818	9944
	9957	10074	10086	10211	10223	10361	10374	10498	10513	10641	10654	10776	10786	10817	10830
	10856	10865	10894	10904	10959	10967	11083	11095	11114						
TRACK	1#	4756	4798	4826	4870	4898	4924	4952	4992	5055	5168	5216	5283	5320	5352
	5384	5416	5448	5492	5555	5617	5679	5741	5803	5874	5917	5960	6002	6051	6099
	6152	6187	6222	6257	6292	6327	6367	6408	6625	6786	6957	7027	7080	7157	7202
	7265	7320	7369	7406	7465	7514	7564	7605	7667	7762	7859	7944	8025	8182	8263
	8366	8463	8527	8647	8836	8969	9102	9237	9372	9514	9681	9819	9958	10087	10224

.\$SIZE	1#	3928
.\$SUPR	1#	
.\$STRAP	1#	3848
.\$TYPB	1#	
.\$TYPD	1#	3780
.\$TYPE	1#	3630
.\$TYPO	1#	3702
.\$1170	1#	25

. ABS. 101404 000

ERRORS DETECTED: 0

CEKBED,CEKBED/CRF/SOL=CEKBED.SML,CEKBED.P11
RUN-TIME: 86 128 10 SECONDS
RUN-TIME RATIO: 566/225=2.5
CORE USED: 41K (81 PAGES)