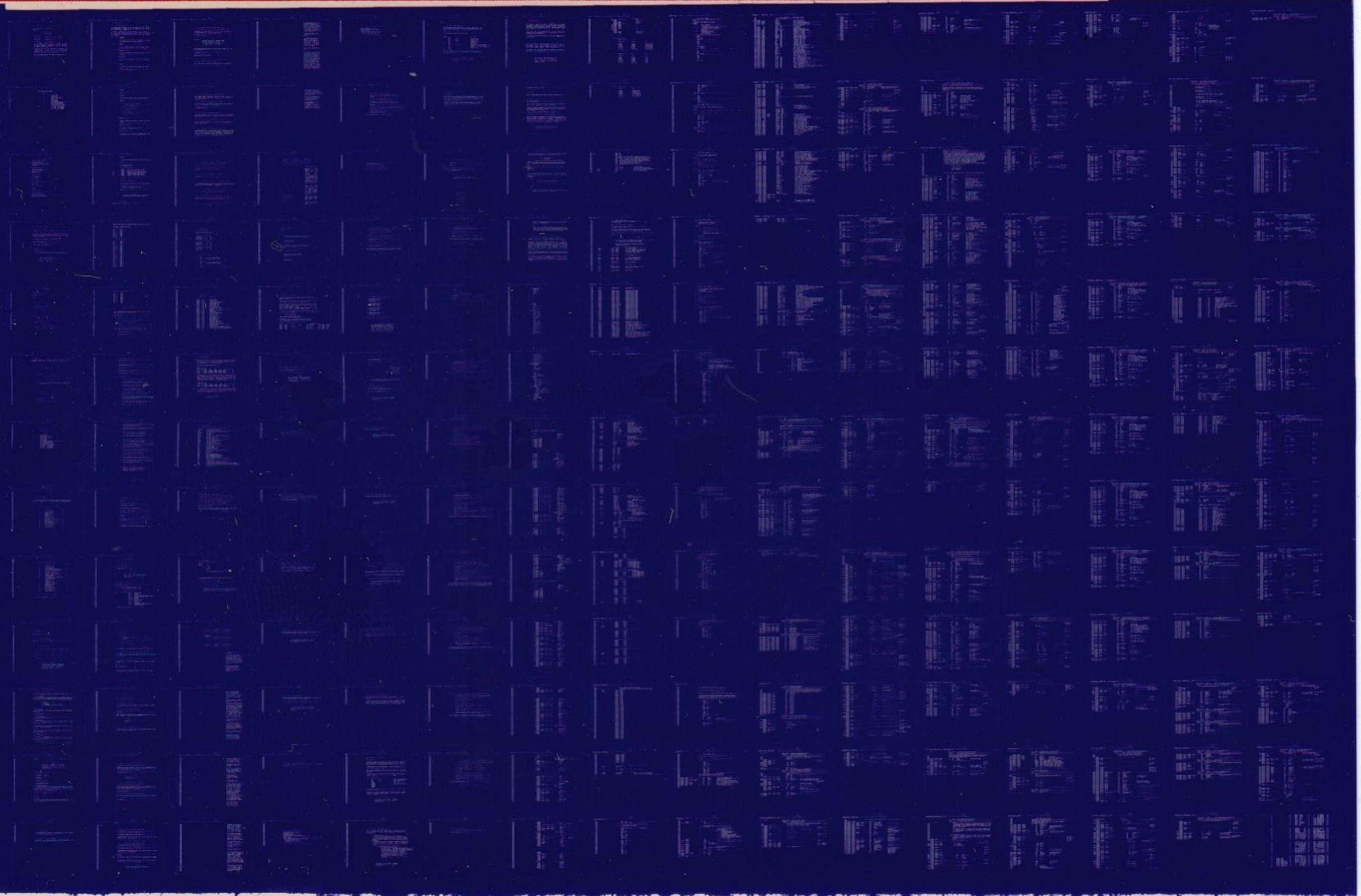


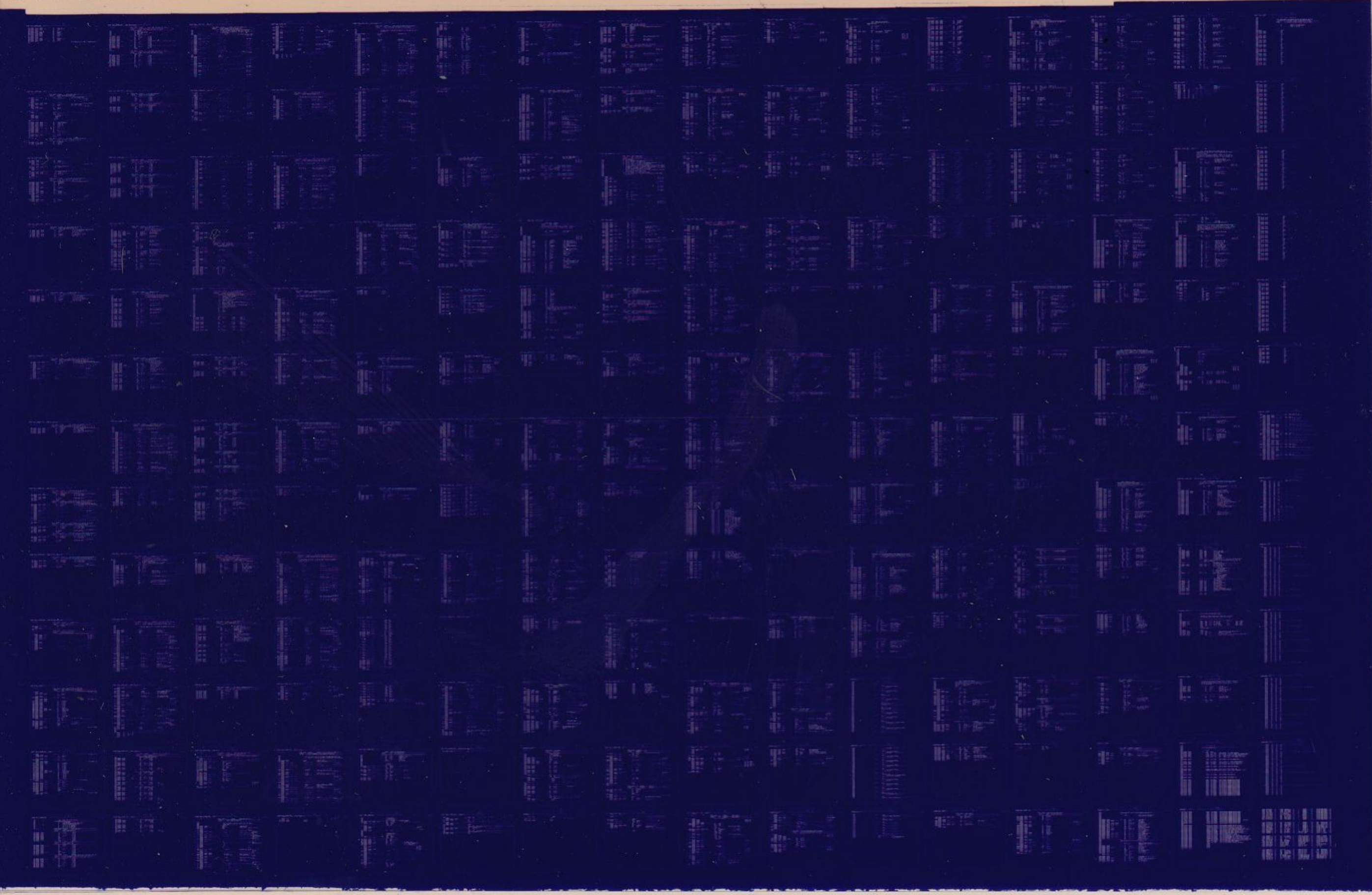
MSV11-J

MSV11-J MEMORY DIAG
CVMJAA0

AH-U136A-MC
1 OF 3 JUL 1985
COPYRIGHT© 1985

digital
MADE IN USA



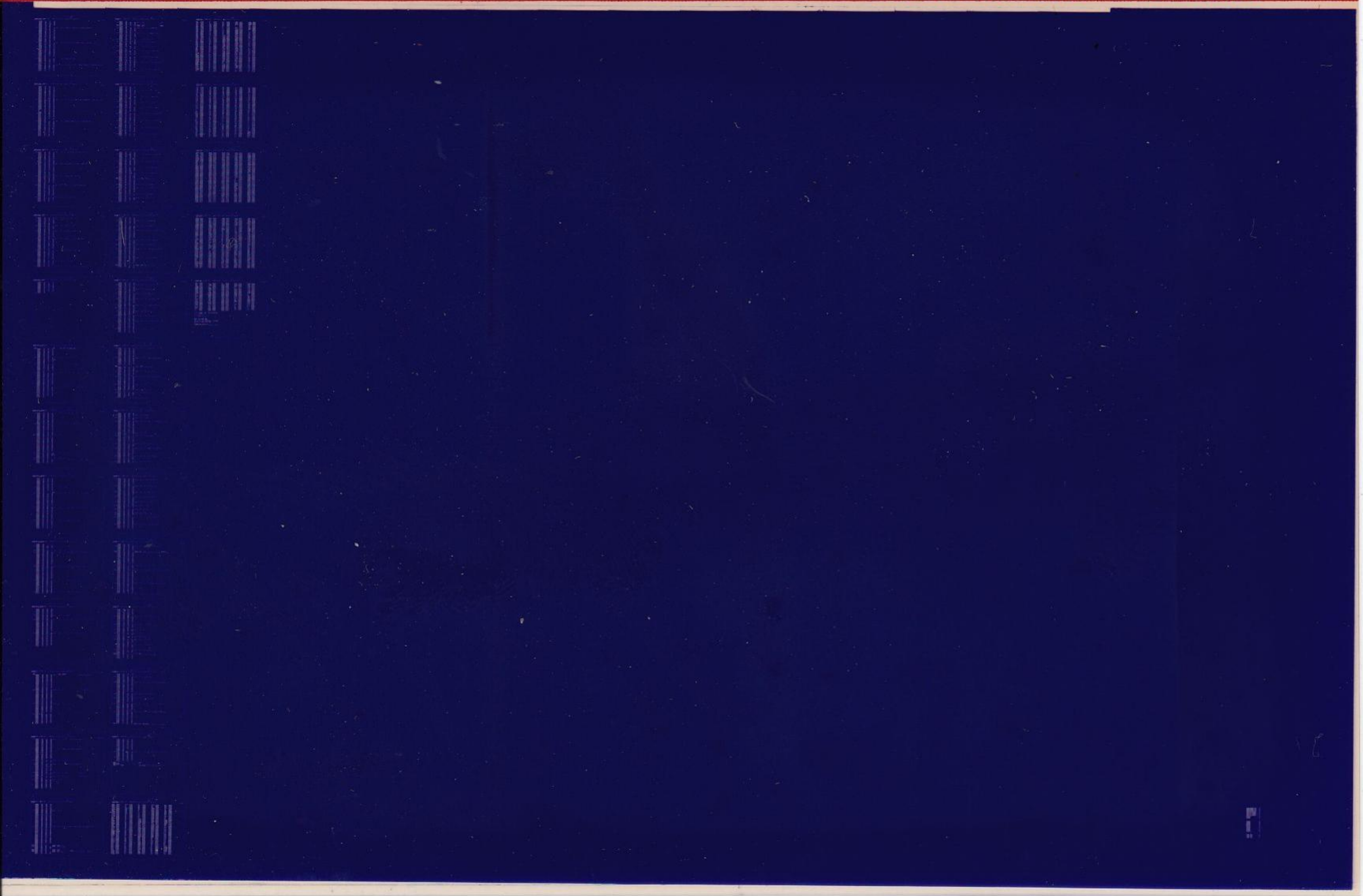


MSV11-J

MSV11-J MEMORY DIAG
CVMJAA0

AH-U136A-MC
3 OF 3 JUL 1985
COPYRIGHT© 1985

digital
MADE IN USA



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

.TITLE CVMJAAO MSV11-J MEMORY DIAG.
.NLIST TOC
REM

IDENTIFICATION

PRODUCT CODE: AC-U135A-MC
PRODUCT NAME: CVMJAA MSV11 J MEMORY DIAGNOSTIC
PRODUCT DATE: MARCH 1985
MAINTAINER: E.S.D. METHODS

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.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of Digital's copyright notice) only for use in such system, except as may otherwise be provided in writing by Digital.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

COPYRIGHT (C) 1985 Digital Equipment Corporation

42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63

OPERATIONAL SWITCH SETTINGS
SWITCH REGISTER DEFINITIONS

*	SWITCH	USE
*	15	HALT ON ERROR
*	14	LOOP ON TEST
*	13	INHIBIT ERROR TYPEOUTS
*	12	INHIBIT RELOCATION
*	11	QUICK VERIFY
*	10	BELL ON ERROR
*	9	LOOP ON ERROR
*	8	HALT PROGRAM (UNRELOCATED RESTORE LOADERS)
*	7	DETAILED ERROR REPORTS
*	6	INHIBIT CONFIGURATION MAP
*	5	LIMIT MAX ERRORS PER BANK
*	4	FAT TERMINAL (132 COLUMNS OR BETTER)
*	3	TEST MODE - SEE DOCUMENT
*	2	TEST MODE - SEE DOCUMENT
*	1	TEST MODE - SEE DOCUMENT
*	0	DETECT SINGLE BIT ERRORS

TABLE OF CONTENTS

65	
66	
67	
68	1.0 GENERAL PROGRAM INFORMATION
69	
70	1.1 Program Purpose (Abstract)
71	1.2 System Requirements
72	1.3 Related Documents And Standards
73	1.4 Diagnostic Hierarchy Prerequisites
74	1.5 Assumptions
75	
76	2.0 OPERATING INSTRUCTIONS
77	
78	2.1 Loading and Starting Procedures
79	2.2 Default Test Sequence
80	2.3 Special Environments
81	2.4 Program Options
82	2.5 Execution Times
83	
84	3.0 ERROR INFORMATION
85	
86	3.1 Error Reporting
87	3.2 Error Abbreviations
88	3.3 Error Halts
89	
90	4.0 PROGRESS REPORTS
91	
92	5.0 CSR INFORMATION TABLES
93	
94	5.1 MSV11-J CSR
95	5.2 MSV11-L/P CSR
96	
97	6.0 SUB-TEST SUMMARIES
98	
99	6.1 Tests
100	6.2 Patterns
101	
102	7.0 PROGRAM FEATURES
103	
104	7.1 Fast Data Access Rates
105	7.2 Bank Zero Testing
106	7.3 Memory Configuration Map
107	7.4 Everything You've Always Wanted To Know About SUPERMAC ...
108	7.5 Memory Management Mapping

110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151

1.0 GENERAL PROGRAM INFORMATION

1.1 Program Purpose (Abstract)

- a. Intended for use on all PDP-11/23/23B/73's which meet the conditions in 1.2.1.
- b. This program will be used by system managers and operators to determine the correct operation of main memory and also it will be primarily used by field service and manufacturing to isolate failures to the memory and to isolate failures within the memory to the correct card.
- c. The object of this software is to functionally test and verify all main memory functions as fast as possible.
- d. There is the capability of testing mixed configurations (MSV11-L, and MSV11-P) on the system.
- e. It has a special maintenance mode (Field Service Mode) to provide specific functional capabilities.

1.2 System Requirements

1.2.1 Hardware Requirements -

PDP-11/23/23B/44/83/84 CPU with 18/22 bit addressing and at least 64K (16 Bit Words) of Memory and Memory Management.

***** NOTE *****

1. Like memory types must be on 16K word boundaries starting at physical address 0.
2. REFERENCE KTJ11 document for proper configuration of unibus memory for 11/84

1

1.2.2 Software Requirements

This program is designed to run stand alone or under any of the following monitors:

XXDP+
ACT
APT

1.3 Related Documents And Standards

1. Microcomputers and Memories (EB-20912-20)
2. PDP-11/23 User's Guide
3. MSV11-J Users guide
4. MSV11-L Users Guide
5. MSV11-P Technical Manual

1.4 Diagnostic Hierarchy Prerequisites

If the program in any way misbehaves, then:

1. Try it again with Cache off (reference Section 2.4.3.1)
2. Inhibit relocation (reference section 2.4.1)
3. Try CPU Diagnostics
4. Try Memory Management Diagnostics
5. Try Cache Diagnostics (where applicable)
6. Try QBUS Map Diagnostics (where applicable)

1.5 Assumptions

This program assumes the correct operation of the CPU, Memory Management, Cache, and the QBUS Map. This program occupies (initially) Bank 0 (0 16K). The XXDP+ loaders are in bank 1.

2.0 OPERATING INSTRUCTIONS

2.1 Loading Starting Procedures

2.1.1 Quick Starting -

1. Load address 200
2. Set switch register for options (normally 0)
3. Start

NOTE

BE SURE that the peripheral page jumper (where applicable) is in place; failure to do so sends the diagnostic to Never-Never Land.

2.1.2 Stopping -

1. Set SW8, and/or
2. Type control "C" (Reference section 2.4.4.1).

2.1.3 Restarting (Preserve Configuration Table) -

1. Load address 202
2. Set switch register for options (Normally 0)
3. Start

251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277

2.1.4 Switch Register Options -

SWITCH	USE
-----	-----
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT RELOCATION
11	QUICK VERIFY
10	BELL ON ERROR
9	LOOP ON ERROR
8	HALT PROGRAM (UNRELOCATE RESTORE LOADERS)
7	DETAILED ERROR REPORTS
6	INHIBIT CONFIGURATION MAP
5	LIMIT MAX ERRORS PER BANK
4	FAT TERMINAL (132 COLUMNS OR BETTER)
3	TEST MODE - SEE DOCUMENT
2	TEST MODE - SEE DOCUMENT
1	TEST MODE - SEE DOCUMENT
0	DETECT SINGLE BIT ERRORS

2.2 Default Test Sequence

The following two lists give the test protocol for parity and ECC Memory. Tests marked with a "*" are not normally run except under ACT or APT, or through a Field Service Command (Reference Section 2.4.4.8).

2.2.1 Test Protocol For MSV11-L/P Parity Memory -

Test	Test Name	Time (sec/16K)
34	Soft Error Test	<1
6	Initial Data Test	<1
17	Holding 1's and 0's Test	<1
7	Address Bit Test	<1
1	Address Test	<1
2	Complement Address Test	<1
3	3 XOR 9 Test	1
4	Rotating 0's Test	1
5	Rotating 1's Test	1
21	Marching 1's and 0's Test	1
35	Worst Case Noise Parity Test	n/a
* 22	Refresh Test	10
* 23	Shifting Diagonal Test	10
26	Random Data Test	<1
* 24	Fast Galloping Pattern Test	20
* 31	Sob-a-long Test	3
* 32	Write Recovery Test	<1
* 33	Branch Gobble Test	35
34	Soft Error Test	<1

2.2.2 Test Protocol for MSV11-J ECC Memory

Pattern	Pattern Name	Time (sec/16K)
5	Rotating 1's Test	1
34	Soft Error Test	<1
6	Initial Data Test	<1
44	Shifting check bits thru the CSR	5
14	Basic Double Bit Error test	<1
45	Syndromes in CSR on DBE test	<1
36	Correction code test	1
20	Syndromes in CSR on SBE test	1
37	Check ECC Disable Test	<1
41	Address to CSR on DBE test	1
42	Extended address to CSR test	<1
43	Byte write test	<1
46	Check SBE with ECC Disable test	<1
47	No CSR update on SBE with DBE test	<1
10	Byte Address Test	<1
17	Holding 1's and 0's Test	<1
7	Address Bit Test	<1
1	Address Test	<1
2	Complement Address Test	<1
4	Rotating 0's Test	1
5	Rotating 1's Test	1
21	Marching 0's and 1's Test	1
* 22	Refresh Test	10
26	Random Data Test	<1
* 24	Fast Galloping Pattern Test	20
* 31	Sob-a-long Test	3
* 32	Write Recovery Test	<1
* 33	Branch Gobble Test	35
34	Soft Error Test	<1

@ - Run only on the first Pass when under ACT or APT

At the end of each Pass the program will run cleanup Patterns #30, and #27 for all banks.

369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420

2.3 Special Environments

2.3.1 XXDP+

The first pass will be a quick verify pass if and only if it is in chain mode.

2.3.2 ACT APT Automatic Mode -

The program will not create double bit errors (DBE's) after the 1st pass.

2.3.2.1 APT Execution Times -

Here are some measured execution times for an 11/23B under APT

	1st QV Pass	2nd Pass onward
1024K MSV11-J	30 min	30 min
256K MSV11-L	20 min	20 min
128K MSV11-P	20 min	20 min

The first pass will be a quick verify pass

NOTE

Even though the first pass is a QV pass it takes longer than the subsequent non-QV passes due to the fact that it is running more patterns, some of which (patterns #24 and #33 for example) can be extremely time consuming.

L1

2.3.2.2 APT Environment Table

The following table gives some of the standard settings for the APT E-Table. They may be modified as noted as the user sees fit.

FIRST PASS RUN TIME:

This parameter should be set according to the amount and type of memory to be tested. The above table (APT Execution Times) gives some measured times. For any patterns deleted (through use of the Device Descriptor Words) reference section 2.2 for individual pattern times.

NOTE

The times given in section 2.2 are for 16K chunks of memory, not 128K boards!

LONGEST TEST TIME:

This parameter should be set to the execution time of the longest pattern being run. for the default case this is 35 seconds for Pattern #33.

ADDITIONAL RUN TIME:

Not Used By Program.

SOFTWARE ENVIRONMENT:

For APT auto mode this parameter should be set to a "1". For dump mode set this to a "0".

ENVIRONMENT MODE:

When this parameter is set to a "0" the program does it's own sizing. If the users sets bit #7 however, he must specify the types and amounts of memory to be tested.

SWITCH 1:

The default setting of this switch is "101". APT uses this as the switch register for the program. Reference section 2.4.1 for more information on switch settings.

SWITCH 2:

This switch, if set to any non-zero number, is used to limit the amount of passes APT will make. The program will hang after this count has been reached.

CPU OPTIONS:

Not Used By Program.

MEMORY TYPE n (n=1 to 4)

If bit #7 of ENVIRONMENT MODE is set these four words are used to log the different types of memory to be tested. If bit #7 is not set these location are not used.

MAXIMUM ADDRESS n (n=1 to 4)

These four words are used in conjunction with the corresponding

MEMORY TYPE words to indicate the highest address that memory type occupies.

NOTE

The above two parameters do not actually have to represent an accurate configuration of memory. All the program looks for is an accurate tally of memory amount!

INTERRUPT VECTOR n (n=1 to 2)
Not Used By Program.

BUS PRIORITY n (n=1 to 2)
Not Used By Program.

BASE ADDRESS:
Not Used By Program.

DEVICE MAP:
Not Used By Program.

CONTROLLER DESCRIPTOR CODE n (n=1 to 2)
Not Used By Program.

DEVICE DESCRIPTOR CODES:

The Device Descriptor codes are used by the program to determine which patterns it will run. The default values of these words are all "1"'s, indicating that all of the patterns shown in section 2.2 are executed (save for exceptions as noted there). Each set of words controls a table in the program as follows:

DD WORDS	PROGRAM TABLE (Symbolic location)
Words 0-1	MKCSRT
Words 2-3	MKPAT
Words 4-5	MJPAT

Bit #0 set in the first word indicates that the first pattern in the table will be executed, bit #1 the second, bit #2 the third,... bit #0 of the second word indicates that the 17th entry in the table will be executed, and so on.

529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548

2.3.3 No SBE Free Banks -

If the program cannot find any SBE (Single Bit Error) free locations (in non-protected ECC memory) it will print out an error message and continue testing by-passing the ECC logic tests.

2.3.4 Mixed Parity ECC Configurations -

The program will function normally in mixed environments. The sequence of testing may seem strange due to the recursive test mode algorithm (reference sections 2.4.1.1, 2.4.1.2, 2.4.1.3).

2.4 Program Options

2.4.1 Switch Register Details -

If a hardware switch register is not available then the software switch register is in location 176. IF under APT if BIT7 is set in the E-TABLE symbolic location "\$ENVN" the APT software switch register will be used (location \$SWREG).

To change the software switch register contents: Type "control G". This will cause display the current value of the SWR and prompt for the octal input of the new SWR value from the terminal. This routine will ignore you (not respond to control "G") if you have a hardware switch register.

SW15 = HALT ON ERROR
(100000)

Continuing from this halt will first check for a change in the software switch register ("Control G" in the TTY input buffer) then it will continue testing.

SW14 = LOOP ON TEST
(40000)

This will cause looping on the present test or pattern (back to last scope trap). If in a pattern then the looping will be for an entire bank of 16K addresses.

SW13 = INHIBIT ERROR TYPEOUTS
(20000)

This will cause returns from the error routine without the typed messages. Other on error functions are not affected.

SW12 = INHIBIT RELOCATION
(10000)

This prevents the program from moving and consequently prevents the program from testing at least 32K of memory.

SW11 = QUICK VERIFY
(4000)

If this switch is selected approximately one 64th of the possible combinations of SBE's DBE's are tested.

Each pass complete typeout will indicate this mode by preceding the pass number with "QV".

604 SW10 = BELL ON ERROR
605 (2000)
606
607 This causes a bell (or beep or click) on each error
608 trap
609
610 SW9 = LOOP ON ERROR
611 (1000)
612
613 This will cause looping from failure point back to the
614 last correctly initialized area of the current test.
615
616 SW8 = HALT PROGRAM
617 (400)
618
619 This initiates the following sequence:
620
621 1. If program is relocated it moves back to bank zero.
622
623 2. Flush out all possible DBE's.
624
625 3. Turns off Memory Management.
626
627 4. Restore loaders.
628
629 5. Unmap the Unibus Map (if there is one).
630
631 6. Halt if under APT or ACT branch sel.
632
633
634 SW7 = DETAILED ERROR REPORTS
635 (200)
636
637 After any normal error report is typed this option
638 causes the contents of the following registers to be
639 typed:
640 R0, R1, R2, R3, R4, R5, SP, "CONTROL", "CPUERR
641
642 SW6 = INHIBIT CONFIGURATION MAP
643 (100)
644
645 This inhibits the printing of a map showing the memory
646 configuration - reference section 7.3
647
648 SW5 = LIMIT MAX ERRORS PER BANK
649 (40)
650
651 This will limit the number of error typeouts per bank.
652 The default is 10. DECIMAL, however this can be
653 changed by changing location "ERRMAX" manually.
654
655

SW4 = FAT TERMINAL
(20)

This informs the program that the console terminal has
a width of at least 132 columns (LA36 with wide paper).

SW3-1 = TEST MODE

Test modes determine the recursion algorithm to be used
during pattern tests.

MODE NAME DESCRIPTION

(0)	0	BAFPAF	Banks forward, patterns forward
(2)	1	BAFPAR	Banks forward, patterns reverse
(4)	2	BAWPAF	Banks worst first, patterns forward.
(6)	3	BAWPAR	Banks worst first, patterns reverse.
(10)	4	PAFBAF	Patterns forward, banks forward
(12)	5	PAFBAW	Patterns forward, banks worst first
(14)	6	PARBAF	Patterns reverse, banks forward
(16)	7	PARBAW	Patterns reverse, banks worst first.

For more details reference section 2.4.1.1, 2.4.1.2 and
2.4.1.3.

SW0 = DETECT SINGLE BIT ERRORS (SBE's)
(1)

For manufacturing purposes this switch should always be
on. For field service purposes this switch should
always be off.

This switch will allow all ECC Single Bit errors to be
reported by disabling error correction.

Error printouts of SBE's are not distinguishable from
DBE's.

NOTE

If Double Bit Errors are found in the memory,
this switch should be set to make sure that new
data can be written to the DBE locations.

2.4.1.1 Test Mode Example -

Example analysis of mode 5 "PAFBAW". Assume Banks 0 1 are MSV11-J
and Banks 2,3,4, 5 are MSV11-L.

711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762

Assume also that Bank 3 is known bad by the program via the sizing routine or previous runs The testing sequence would be as follows:

;TEST MSV11-J MEMORY TYPES FIRST
;TEST KNOWN BAD MEMORY (BANK 3)

TEST 17,	BANK 3
TEST 7,	BANK 3
TEST 1,	BANK 3
TEST 2,	BANK 3
TEST 4,	BANK 3
TEST 5,	BANK 3
TEST 21,	BANK 3
TEST 20,	BANK 3
TEST 22,	BANK 3
TEST 26,	BANK 3

;TEST PRESUMED GOOD MEMORY (BANKS 2,4,5)

TEST 17,	BANK 2
TEST 7,	BANK 2
TEST 1,	BANK 2
TEST 2,	BANK 2
TEST 4,	BANK 2
TEST 5,	BANK 2
TEST 21,	BANK 2
TEST 20,	BANK 2
TEST 22,	BANK 2
TEST 26,	BANK 2
TEST 17,	BANK 4
TEST 7,	BANK 4
TEST 1,	BANK 4
TEST 2,	BANK 4
TEST 4,	BANK 4
TEST 5,	BANK 4
TEST 21,	BANK 4
TEST 20,	BANK 4
TEST 22,	BANK 4
TEST 26,	BANK 4
TEST 17,	BANK 5
TEST 7,	BANK 5
TEST 1,	BANK 5
TEST 2,	BANK 5
TEST 4,	BANK 5
TEST 5,	BANK 5
TEST 21,	BANK 5
TEST 20,	BANK 5
TEST 22,	BANK 5
TEST 26,	BANK 5

764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815

;RELOCATE TEST PROGRAM SPACE (BANK 0 & 1)

TEST 1, BANK 0
TEST 2, BANK 0
TEST 3, BANK 0
TEST 4, BANK 0
TEST 5, BANK 0
TEST 26, BANK 0
TEST 1, BANK 1
TEST 2, BANK 1
TEST 3, BANK 1
TEST 4, BANK 1
TEST 5, BANK 1
TEST 26, BANK 1

NOTE

This is an example not an actual sequence.

The test sequence was forward (the simple patterns first, complex tests last) sequence of patterns (MSV11-L = 17, 7, 1, 2, 4, 5, 21, 20, 22, 26)(MSV11-J = 1, 2, 3, 4, 5, 26).

If the bank selection is forward the banks will be tested in the following order:

1. ECC banks that are not protected or program space (from 0 to 167).
2. Parity banks that are not program space (from 0 to 167).
3. The program now relocates tests:
4. ECC banks that were protected or program space (from 0 to 167).
5. Parity banks that were program space (from 0 to 167).

If bank selection is worst first the configuration table will be consulted and banks will be tested in the following order.

1. ECC banks that are known bad and are not protected or program space (from 0 to 167).
2. Parity banks that are known bad and are not program space (from 0 to 167).
3. ECC banks that are presumed good and are not protected or

817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870

program space (from 0 to 167).

4. Parity banks that are presumed good and are not program space (from 0 to 167).
5. The program now relocates tests:
6. ECC banks that are known bad and were protected or program space (from 0 to 167).
7. Parity banks that are known bad and were program space (from 0 to 167).
8. ECC banks that are presumed good and were protected or program space (from 0 to 167).
9. Parity banks that are presumed good and were program space (from 0 to 167).

2.4.1.2 Test Mode Details -

MODE 0 = "BAFP AF" banks forward, patterns forward

This is the default and simplest mode.

This mode tests each bank completely from 0 to 167 except those requiring relocation*.

While testing each bank the patterns are run with the simple ones first building to the more complex.

MODE 1 = "BAFP AR" = banks forward, patterns reverse

This mode tests each bank completely from 0 to 167 except those requiring relocation*.

While testing each bank the patterns are run with the most complex ones first, working to the simple ones.

MODE 2 = "BAWP AF" = Banks worst first, patterns forward

This mode first tests each known bad bank completely from 0 to 167 except those requiring relocation*, then presumed good banks are tested from 0 to 167 except those requiring relocation*.

While testing each bank the patterns are run with the simple ones first, building to the more complex.

MODE 3 = "BAWP AR" = Banks worst first, patterns reverse

This mode first tests each known bad bank completely

872 from 0 to 167 except those requiring relocation*, then
873 presumed good banks are tested from 0 to 167 except
874 those requiring relocation*.
875
876 While testing each bank the patterns are run with the
877 most complex ones first, working to the simple ones.
878
879 MODE 4 = "PAFBAF" = Patterns forward, banks forward
880
881 This mode tests each pattern completely with the simple
882 ones first, building to the more complex.
883
884 While testing each pattern the banks are run from 0 to
885 167 except those requiring relocation*.
886
887
888 MODE 5 = "PAFBAW" = Patterns forward, banks worst first
889
890 This mode tests each pattern completely with the simple
891 ones first, building to the more complex.
892
893 While testing each pattern first each known bad bank
894 from 0 to 167 except those requiring relocation* is
895 run, then presumed good banks are run from 0 to 167
896 except those requiring relocation*.
897
898 MODE 6 = "PARBAF" = Patterns Reverse, Banks Forward
899
900 This mode tests each pattern completely with the most
901 complex ones first, working to the simple ones.
902
903 While testing each pattern the banks are run from 0 to
904 167 except those requiring relocation*.
905
906 MODE 7 = "PARBAW" = Patterns Reverse, Banks Worst First
907
908 This mode tests each pattern completely with the most
909 complex ones first, working to the simple ones.
910
911 While testing each pattern first each known bad bank
912 from 0 to 167 except those that require relocation* is
913 run, then presumed good banks are run from 0 to 167
914 except those requiring relocation*.
915
916
917
918
919
920
921
922
923
924

NOTE

* Relocation is required to test the bank(s) in program space and also to test any ECC banks protected by diagnostic checkmode with the inhibit mode pointer off (zero)!

926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966

2.4.1.3 Test Mode Applications -

1. To verify correct operation of the memory system use Mode 0 "BAFPAF".

Advantages: Easy to understand.

Disadvantages: In case of a failing Bank, it may take a long time to find the failure.

2. To get detailed error information on known bad Banks (found by sizing routine) use Mode 2 "BAWPAF".

Advantages: Seeks Bad Banks. Easy to understand.

Disadvantages: Failures other than zeros ones may take a long time to find.

3. To get good error info on any memory problem fast use Mode 4 "PAFBAPF".

Advantages: Covers all banks fast. Easy to understand.

Disadvantages: Failures from only complex patterns may take a long time to find.

4. To find any problem fast use Mode 7 "PARBAW".

Advantages: Covers all Banks fast.

Disadvantages: Difficult to understand failures reported are not necessarily the most basic failure modes.

2.4.2 Software Switch register

A software switch register exists in location 174.

2.4.3 Special Memory Locations -

2.4.3.1 CACHE Constant -

The CACHE constant is located at symbolic location "CACHK" and is used to enable CACHE.

NOTE

Bit 0 in the CACHE constant has no effect since it is unconditionally set by the program whenever it tries to enable CACHE.

2.4.3.2 Configuration Table

The configuration table is located at symbolic location "CONFIG" and has the following format:

CONFIG: First 16K Configuration words (2 each)
2nd 16K Configuration words (2 each)
200th 16K configuration words (2 each)

Configuration Words:

LOW:	BIT 0	ERRORS PRESENT
	BIT 1	MEMORY EXISTS
	BIT 2-4	RESERVED
	BIT 5	SKIP ECC LOGIC TESTS FLAG (1=SKIP)
	BIT 6	PROTECTED REGION OF AN ECC MEMORY
	BIT 7	PROTECTED (PROGRAM SPACE)
	BIT 8-11	CSR CODE
	BIT 12-15	RESERVED
MED:	BIT 0-7	NUMBER OF ERRORS
	BIT 8-10	MEMORY TYPE
	BIT 11	CSR TESTED OK
	BIT 12	RESERVED
	BIT 13	"BACKGROUND PATTERN VALID" FLAG
	BIT 14	BANK SELECTED FOR TEST BY FIELD SERVICE MODE
	BIT 15	LOADERS HOME BANK

This table is used as the source for the configuration Map (reference. section 7.3).

1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078

2.4.4 Terminal Commands

2.4.4.1 Control "C"

This command will:

1. If Switch 8 (Halt Program) in the switch register is set halt the program.
2. If Switch 8 is not set, unrellocate if program was relocated.
3. Flush out any DBE's.
4. Turn off Memory Management.
5. HALT

This command will only be recognized at the completion of the current test or pattern, or at the end of a line of an error message.

2.4.4.2 Control "K" (Kill error printout and skip pattern)

This command will allow you to stop an error printout and skip to the next pattern. This is handy, for example, when you have a whole bank full of errors, have gotten enough information, and wish to skip to the next pattern.

2.4.4.3 Control "T" (Tell me what's happening)

This command will print out the information encoded in the display register. This is mainly intended for CPU's without a hardware display register.

Example:

BANK = 17 TEST = 46
RELOCATED BANK= 0 PAT= 26

By use of Field Service Command 17 "Trace" can be set so that it will automatically type out the bank and pattern numbers as each pattern is run. (Reference section 2.4.4.8.18).

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124

2.4.4.4 Control "S" (Stop)

This command will stop typeout (soon) and will wait for a Control "Q".

2.4.4.5 Control "Q" (Continue)

This command will continue typing that has been stopped by Control "S". If there has been no Control "S" typed then this command is ignored.

2.4.4.6 Control "F" (Field Service mode)

This command will cause you to enter a mode which looks for sub commands.

When the program is looking for a sub command any number that is not a legal command will cause a mini help message to be typed. Therefore when in doubt type 99 (CR) and you will get help.

NOTE

Typing just carriage return is a default command 0.

2.4.4.7.1 Field Service Command 0 (Exit)

This command will exit Field Services Mode and return to whatever task it was in prior to typing control "F". Note typing just carriage return is a default Command 0.

1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175

2.4.4.7.2 Field Service Command 1 (Read CSR)

This command will typeout the contents of the CSR.

If there is more than one CSR on the CPU (or if the program has not determined the CSR status yet), it will Ask you "WHICH CSR(0-F)" to which you must respond with an Hexidecimal number from 0 to F. Note typing just carriage return is a default 0.

If the CSR you select causes a trap to 4 the program will type "THIS CSR DOES NOT EXIST".

NOTE

CSR references are done in accordance with section 5.0.

2.4.4.7.3 Field Service Command 2 (Load CSR)

This command will enable you to load the CSR.

If there is more than one CSR on the CPU (or if the program has not yet determined the CSR status yet) it will ask you "WHICH CSR(0-F)" to which you must respond with an Hexidecimal number from 0 to F. Note typing just carriage return is a default 0.

If the CSR you select causes a trap to 4 the program will type "THIS CSR DOES NOT EXIST".

The CSR will be read and displayed as in command 1.

The program will then ask you for the "CSR?" to which you must respond with an Octal number. Note typing just carriage return is a default 0.

The program will then load the CSR and Read it again displaying its new contents.

1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227

2.4.4.7 4 Field Service Command 3 (Examine Memory)

This command will allow you to examine any physical address and does the necessary memory management mapping for you.

The program will ask you for the "PHYSICAL ADDRESS (0-17757776)" to which you must respond with an Octal number.

If the address access causes a trap to 4 the program will type "TIMEOUT TRAP". If the address access causes a trap to 114 the program will type "PARITY ABORT".

The contents of your physical address will be typed.

2.4.4.7 5 Field Service Command 4 (Modify Memory)

This command allows you to modify any physical address and does the necessary memory management mapping for you.

The program will ask you for the "PHYSICAL ADDRESS (0-17757776)" to which you must respond with an Octal number.

If the address access causes a trap to 4 the program will type "TIMEOUT TRAP". If the address access causes a trap to 114 the program will type "PARITY ABORT".

The program will type "OLD DATA WAS" and the contents of your physical address.

The program will then type "INPUT NEW DATA" to which you must respond with an Octal number. Note typing just carriage return is a default 0.

The program will attempt to write this new data into your physical address after which it will read it again and type "DATA IS NOW" and the new contents of your physical address.

NOTE

If you can't change the data, that would indicate that you have a Double Bit Error in that double word pair.

1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278

2.4.4.7.6 Field Service Command 5 (Select Bank Test)

This command allows you to run any bank with any pattern forever.

The program will ask you "BANK(0-177)" to which you must respond with an Octal number. If the bank is not accessible. The program will type "BANK NOT ACCESSIBLE" and ask question over.

The program will then ask "TEST (0-47)" to which you must respond with an Octal number.

NOTE

Any pattern can be run including those that are not part of the APT E-TABLE defaults (reference section 6.2.1). If you select Pattern 0, the program will ask "TEST 0 DATA IS?" to which you must respond with an Octal number.

If the Bank you selected requires relocation the program will type "BANK REQUIRES RELOCATION" and exit this command. Note normally this is true for Bank 0.

The program will then arm the console keyboard for interrupts and type "TO ESCAPE TYPE ANY KEY!".

The test pattern will be entered and run until a console key is depressed to escape this loop.

2.4.4.7.7 Field Service Command 6 (Type Configuration Map)

This command types the configuration map.

This is useful after a long run (overnight) to see all the banks that are marked as bad. (Especially if your console is a video terminal).

For a detailed explanation of the map reference section 7.3.

1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332

2.4.4.7.8 Field Service Command 7 (SOB-A-LONG TEST)

This command allows execution of the SOB-A-LONG Test on all non-protected Banks reference Section 6.2.2.26. Operation is identical to command 5 except that no Pattern or Bank is entered and each pass causes a Bell.

2.4.4.7.9 Field Service Command 8 (Error Summary)

This command types out the number of passes and the total number of errors. If there were any errors it will type out the Banks and the number of errors per bank up to 255 DECIMAL.

This becomes useful after long runs (all night) on systems with a video console terminal.

2.4.4.7.10 Field Service Command 9 (Refresh TEST)

This command allows execution of the Refresh Test on all non-protected Banks reference Section 6.2.2.19. Operation is identical to command 5 except that no Pattern or Bank is entered and each pass causes a Bell.

2.4.4.7.11 Field Service Command 10 (Set Fill Count)

This command allows setting of the terminal fill count (necessary for LA30's, ASR33's, and VT05's). It is normally set to zero for LA36's, VT52's, VT100's, etc.

2.4.4.7.12 Field Service Command 11 (Enter Kamikaze Mode)

This command allows you to run patterns that are normally not executed unless under APT or ACT. They are usually very time consuming and can result in failures that are fatal to the program. In effect you are trying to find a hardware failure regardless of the consequences. Note that most crashes do not wipe out the display information which is telling you what the program was doing just prior to failure. There are two ways to die here Impatience and Crashes.

1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389

2.4.4.7.13 Field Service Command 12 (Exit Kamikaze Mode)

Return to the default mode of testing (undo Command 12).

2.4.4.7.14 Field Service Command 13 (Turn Cache Off)

This changes the Cache constant to bypass cache (reference section 2.4.3.1).

2.4.4.8.15 Field Service Command 14 (Turn Cache On)

This changes the Cache constant to use cache (reference section 2.4.3.1).

2.4.4.7.16 Field Service Command 15 (Test Only Selected Banks)

This command allows you to center the test effort on only those banks that you are troubleshooting. You may also test banks that require relocation and were inaccessible via command 5.

2.4.4.7.17 Field Service Command 16 (Resume Testing All Banks)

Return to the default mode of testing (undo Command 15).

2.4.4.7.18 Field Service Command 17 (Resume Testing All Banks)

Enable "Trace". After exiting field service mode, the program will type out the bank and pattern numbers as each pattern is run.

2.4.4.7.19 Field Service Command 18 (Resume Testing All Banks)

Disable "Trace". (undo Command 17).

1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439

2.5 Execution Times

2.5.1 Typical (System) -

Execution time depends on many variables; however here are some measured times on an 11/23B:

K words of MSV11-J Memory		
Normal Pass	Min	Sec
Quick Verify	Min	Sec
Kamikaze Mode	Min	Sec
Kamikaze QV	Min	Sec

K words of MSV11-L Memory		
Normal Pass	Min	Sec
Quick Verify	Min	Sec
Kamikaze Mode	Min	Sec
Kamikaze QV	Min	Sec

K words of MSV11-P Memory		
Normal Pass	Min	Sec
Quick Verify	Min	Sec
Kamikaze Mode	Min	Sec
Kamikaze QV	Min	Sec

2.5.2 Calculations (System)

Normal Pass		
Add	Sec per 16K BANK of MSV11-P	
Add	Sec per 16K BANK of MSV11-L	
Add	Sec per 64K BANK of MSV11-J	

Quick Verify Pass		
Add	Sec per 16K BANK of MSV11 P	
Add	Sec per 16K BANK of MSV11-L	
Add	Sec per 64K BANK of MSV11-J	

Kamikaze Mode
Add 10 min. per 128K words for approximate pass times.

1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485

2.5.3 Typical (Tests)

Test Time	Description
-----	-----
MT0000 ; <1 SEC	DATA PATTERN TEST
MT0001 ; <1 SEC	ADDRESS TEST
MT0002 ; <1 SEC	COMPLEMENT ADDRESS TEST
MT0003 ; 1 SEC	3 XOR 9 WORST CASE NOISE TEST
MT0004 ; 1 SEC	ROTATING ZEROS TEST
MT0005 ; 1 SEC	ROTATING ONES TEST
MT0006 ; <1 SEC	INITIAL DATA TEST
MT0007 ; <1 SEC	ADDRESS BIT TEST
MT0010 ; <1 SEC	BYTE ADDRESSING TEST
MT0014 ; 1 SEC	BASIC DOUBLE BIT ERROR TEST
MT0017 ; <1 SEC	HOLDING 1'S 0'S TEST
MT0020 ; 1 SEC	SYNDROMES TO CSR ON SINGLE BIT ERROR TEST
MT0021 ; 1 SEC	MARCHING 0'S 1'S TEST
MT0022 ; 10 SEC	REFRESH TEST
MT0023 ; 10 SEC	SHIFTING DIAGONAL TEST
MT0024 ; 20 SEC	FAST GALLOPING PATTERN TEST
MT0026 ; <1 SEC	RANDOM DATA TEST
MT0027 ; 1 SEC	UNIQUE BANK TEST
MT0030 ; 1 SEC	FLUSH OUT DBE'S TEST
MT0031 ; 3 SEC	SOB-A-LONG TEST
MT0032 ; <1 SEC	WRITE RECOVERY TEST
MT0033 ; 35 SEC	BRANCH GOBBLE TEST
MT0034 ; <1 SEC	SOFT ERROR TEST
MT0035 ; <1 SEC	WORST CASE PARITY TEST
MT0036 ; 1 SEC	CORRECTION CODE TEST
MT0037 ; <1 SEC	CHECK ECC DISABLE TEST
MT0041 ; 1 SEC	ADDRESS TO CSR ON DBC TEST
MT0042 ; <1 SEC	EXTENDED ADDRESS TO CSR ON ERROR TEST
MT0043 ; <1 SEC	WRITE BYTE TEST
MT0044 ; 5 SEC	SHIFTING CHECKBITS THROUGH CSR TEST
MT0045 ; <1 SEC	SYNDROME BITS TO THE CSR ON A DBE TEST
MT0046 ; 1 SEC	CHECK SINGLE BIT ERRORS WITH ECC DISABLED TEST
MT0047 ; <1 SEC	NO CSR UPDATE WITH EXISTING DBE TEST

3.0 ERROR INFORMATION

3.1 Error Reporting

Most errors are reported using the EMT trap and handler provided by SYSMAC.SML. Most errors will be of the "MEMORY DATA ERROR" type which will be described here. MEMORY DATA ERRORS will also cause the bank to be marked as Bad in the configuration table.

Other errors are best explained by referencing the specific typeout and if necessary the program listing.

Example 1:

MEMORY DATA ERROR

PC	BANK	VADD	PADD	GOOD	BAD	XOR	CSR	MTYP	PAT
022132	37	060006	03700006	000000	000100	000100	0	E	06
022132	37	060006	03700006	000000	000100	000100	0	E	06
022132	37	060006	03700006	000000	000100	000100	0	E	06
022132	37	060006	03700006	000000	000100	000100	0	E	06

While testing Bank 37 at virtual address 60006 (virtual addresses are always between 60000 and 157776 for mapping purposes), physical address 3700006 (that's Bank 37 physical 6 within the Bank) with Pattern 6 (Initial Data Test), the good data expected was 0 but the data actually read (BAD) was 100, the exclusive OR at Good Bad yields 100 which indicates only failing bit(s) (Bit 6). It is an MSV11-J (ECC) Memory. The CSR is located at 172000.

Example 2:

MEMORY DATA ERROR

PC	BANK	VADD	PADD	GOOD	BAD	XOR	CSR	MTYP	PAT
022132	35	060000	03500000	000000	000001	000001	0	E	06
022132	35	060002	03500002	000000	000100	000100	0	E	06
022132	35	060006	03500006	000000	000100	000100	0	E	06

While testing Bank 35, virtual address 60000, physical address 3500000 with Pattern 6 (Initial Data Test), the good data expected was 0 but the data actually read (BAD) was 1, the exclusive OR at Good Bad yields 1 which indicates only failing bit(s) (Bit 0). It is an MSV11-J (ECC) Memory and the CSR is located at 172000.

NOTE

Subsequent errors of the same test do not type a new heading.

3.2 Error Abbreviations

The following is a list of all abbreviations used in error reports.

# OF ERRORS	Number of Errors that were detected.
1ST ADD	First Address that failed.
ARRAY	The array number that was locked up in the MS11-M CSR.
APT#	The # of CPU's APT expects on the system.
APTCORE	APT Core size.
APTMOS	APT MOS size.
BAD	Bad data.
BAD-WD1	Bad Word #1 of a double word data value.
BAD-WD2	Bad Word #2 of a double word data value.
BAD-CHK	Bad Check Code Bits.
BANK	The Bank number. Banks are 16K words long.
BD-CC	Bad Check Code Bits.
CHKBITS	The 7 bit value of the Check Code Bits.
CONTRL	The CACHE Control register.
CPUERR	CPU Error register.
CSR	Control and Status Register.
CSRNO	CSR NUMBER (0-F Hexidecimal).
DATARG	The CACHE Data Register.
DBE	Double Bit Error (uncorrectable error).
DEV ADD	Device Address.
ECC	Error Correctable Code.
GD-CC	Good Check Code Bits.
GD-CHK	Good Check Code Bits.
GD-WD1	Good Word #1 of a double word data value.
GD WD2	Good Word #2 of a double word data value.
GOOD	Good data.
LSIZE	MSV11-J Size.
MEMERR	Memory Error register.
MMR0	Memory Management Register #0.
MMR1	Memory Management Register #1.
MMR2	Memory Management Register #2.
MMR3	Memory Management Register #3.
MTYP	Memory Type (MSV11-J,MSV11-L, or MSV11-P).
PADD	Physical Address (asserted by the program after mapping).
PAT	Pattern number.
PC	Program Counter at the time the error occurred.
SBE	Single Bit Error (correctable error).
VADD	Virtual Address (asserted by the program before mapping).
WROTE1	The data that was written into the 1st half of a double word.
WROTE2	The data that was written into the 2nd half of a double word.
XOR	Exclusive OR of the good and bad data. Shows the bad bits.
AUT	Address under test

1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607

3.3 Error Halts

There are several Halts in the program.

All unused trap vectors contain a trap catcher (.WORD .+2,HALT).

An undefined TRAP instruction halts at symbolic location "\$HALT2".

The APT down load sequence will halt at symbolic location "APTHLT".

Halt on Error option (SW15 Set) at symbolic location "\$HALT".

Halt program (SW8 Set) at symbolic location "\$EXHALT".

Power Fail will normally halt at the end of the shut down sequence (symbolic location "\$DOWN").

Power Fail has a fatal Halt at symbolic location "\$ILLUP" which can be caused by power up occurring before power down sequence completed or by power down before a power up sequence is completed.

4.0 PROGRESS REPORTS

Pass complete typeouts as follows:

END PASS	#	0
END PASS	#	1
END PASS	#QV	2

NOTE

Pass 2 was flagged as a Quick Verify
Pass. (Because of a change in SW5)

To obtain progress reports while executing, typing a Control "T" will
print out the information encoded in the display register.

Example:

BANK= 2 TEST= 34

Reference Section 2.4.4.7.18 for more information on Tracing.

5.0 CSR INFORMATION TABLES

The following is a picture view of the current control status
registers which can be tested by this program. It shows bit
assignments and definitions to provide a handy reference, and
shows the similarities and differences between each one:

NOTE

All unused bits in each CSR are equal to zero.

1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660

5.1 MSV11-J CSR

(I)	-----															
	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
	!DE!	!EA!	!					ADDRESS			!SE!	!IP!	!DC!	!EC!	!EE!	
(II)	-----															
	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
	!DE!	!EA!	!SI!	!				CHECK BITS			!SE!	!IP!	!DC!	!EC!	!EE!	
(III)	-----															
	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I	I
	!DE!	!EA!	!SI!	!				SYNDROMES			!SE!	!IP!	!DC!	!EC!	!EE!	

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00																

Bit assignments are defined
as follows:

BIT15 UNCORRECTABLE ERROR
On a read to memory (ECC Disable
Bit = 0), this bit is set if
a double error occurs. The
error address is stored in the
CSR. Setting this bit also
turns on a red LED at the rear of
the card for a visual indication.
This bit is also set in ECC
Disable Mode if a SERR or DERR
occurs.

BIT14 EUB ERROR ADDRESS
With Bit 14 = 1, a read to the CSR
will fetch address A21
through A18. When Bit 14 = 1,
diagnostic data may not be
loaded into the syndrome
register.

1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715

1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770

BIT13 SET INHIBIT MODE
When this bit is set to
a "1", it enables the inhibit
mode pointer to inhibit
either the first or second
16K from ever going into the
Diag Check or ECC Disable
Mode.

BITS05-10 CHECK BIT STORAGE (CSR II)
Check Bit Storage (Diag CK
Bit 2 = 1)
When in the diagnostic check
mode these bits are used to store
the check bits to be written
into memory or the check bits
read from memory. If a double
error or single error occurs
when in the diagnostic check
mode and ECC disable Bit 1 = 0,
then the check bits are stored
in the CSR together with
the double or single error
bit. These bits are writeable
in diagnostic mode. A "1"
is stored in Bit 11 if CSR
02, CSR 13, and CSR 14 are
set to indicate that the
memory under test is a MSV11-J.

BITS05 11 QBUS ADDRESS STORAGE (CSR I)
(Diag Ck Bits 2 - 0, ECC
Disable Bit 1 = 0)

If a double or single error
occurs on a Read cycle, then
address Bits A11 through
A17 are stored in these bits
These bits are read only on
the condition that SERR (CSR 4)
or DERR (CSR 15) is set but
CSR 14 is not set.

EQB Address Storage (Diag Ck
Bit 2 = 0), ECC Disable Bit
1 = 0 or 1).

If a double or single error
occurs on a Read cycle,
address Bits A17 through
A11 are stored in CSR Bits
11 through 5 and address
Bits A21 through A18 are

1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825

stored in a backup register.
The EQB Error Address
Retrieval Bit (CSR 14) is
used to obtain the total
error address as follows:

With CSR Bit 14 = 0 a read to
the CSR will obtain A17
through A11 from CSR Bits 11
through 5.

CSR Bit 14 can then be set
to a "1" and a read to the
CSR will then read A21
through A18 from CSR Bits 8
through 5 and 0's from CSR
Bits 11 through 9.

Address Bits A21 through
A11 are obtained
to locate the double
error to a 1K segment of
memory.

The EQB address A21
through A18 is read only
whenever CSR 14 = 1.

BIT05-10 SYNDROME STORAGE (CSR III)
If a double or single error
occurs on a read or write
byte cycle, and if CSR bit
2 is set to a "0" syndrome
Bits X, 0, 1, 2, 4 and 8
and stored in CSR bits 5
through 10. To read the
syndrome bits from CSR, Bit
you must read the error
address, then set 2 of
the CSR must be set to
a "1" (diagnostic mode) and
the CSR read again. This operation
will allow syndrome bits
for a single or double
failure to be read instead
of the address bits normally
read when CSR 02 is set to "0".

BIT04 SINGLE ERROR
If on a read to memory a
SBE occurs, the error
address A21-A11 and
the error syndromes will

1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879

be logged in CSR Bits 5 11 unless the uncorrectable error CSR 15 is set. The error address will be logged unconditionally in the ECC Disable Mode. This bit is not set if Inhibit Mode (Bit 13 = 1) is set and Diagnostic Mode (Bit 02 = 1) is set.

BIT03 INHIBIT MODE POINTER
This bit works in conjunction with the Set Inhibit Mode (Bit 13). When Bit 13 is set to a 1, a 16K portion of memory is inhibited from operating in the ECC Disable Mode or Diagnostic Check Mode.

The Inhibit Mode Pointer indicates which 16K is being inhibited, i.e., Bit 3 = 0 the first 16K of memory is inhibited, Bit 3 = 1, the second 16K of memory is inhibited.

With Bit 13 set to a 0, Bit 3 becomes inoperative.

Bit03, in conjunction with Bit 13, therefore allows a 16K chunk of memory to always have ECC coverage. The systems diagnostic can therefore reside in this protected portion of memory and can disable ECC and/or run the Diagnostic Check Mode in the rest of memory without itself becoming vulnerable to single errors. This bit is a Read/Write bit reset by power up and BUS INIT.

BIT02 Diagnostic Check Mode
This mode allows a means of forcing a single or double error in a desired location. It also provides a means of examining the check bits and the syndrome in a given location.

1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933

The check bits desired for a given data pattern are written into Bits 5 through 11 of the CSR. A word or write byte memory will write the check bits from the CSR to the MOS array (CSR 2 = 1) instead of the check bits generated on the data to be written. Single errors on the read portion of the DATOB cycle are corrected.

A read to the memory will read the check bits stored in memory and clock them into the CSR.

If a double error or single error occurs the DERR or SERR bit in the CSR is set and the error syndrome bits read from ECC are stored in CSR Bits 10-5 as well as the address bits. In Diagnostic Check Mode the error syndrome bits will be read when CSR Bits 10-5 are read.

This bit is a Read/Write bit and is reset on power up and BUS INIT

BIT01 DISABLE CORRECTION MODE
If this bit is set, no single errors will be corrected. A single error will set CSR 4 and CSR 15 or a double error will set CSR 15 and assert BUS PBL if CSR 00 is asserted. The 1K block of address where the error occurs will also be stored in the CSR. The priority of a SERR and DERR will be the same, i.e., the last error information will always be stored unless a DERR precedes a SERR. If a double error occurs during a write byte cycle, the write portion of the cycle will not be aborted. The check bits written will

1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958

have been generated on the data written. This means that if a single or double error existed in the location accessed, it would be cleared (unless the errors were hard).

This bit is a diagnostic aid to allow writing and reading data from memory without interference from the error correction logic.

BIT00 UNCORRECTABLE ERROR
INDICATION ENABLE
If a double error occurs with ECC enabled or a single error or double error with ECC disabled, on a Read cycle to the memory and this bit is set, then BUS PBL will be asserted.

1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014

5.2 MSV11-L/P CSR

```

-----
I I I I I I I I I I I I I I I I I I I I I I
!PE!EU! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! ! !
I I I I I I I I I I I I I I I I I I I I I I
-----
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

```

Bit assignments are defined as follows:

BIT15 PARITY ERROR

BIT14 EQB ERROR
RETRIEVAL If the
memory is on an
Extended QBUS, when
BIT14 is zero, the low
order failing
addresses are
available (Bits
11-17); when BIT14 is
one, the high order
failing addresses are
available (Bits 18 21
of address). If the
memory is on a QBUS,
a jumper disables this
bit so that it is read
only, and equal to ze-
ro.

BITS 11-5 ERROR
ADDRESS With BIT14
set, they contain the
high order parity er-
ror address (Bits
21-18 of address);
with BIT14 cleared,
they contain the low
order parity error ad-
dress (Bits 17-11 of
address).

BIT02 WRITE WRONG
PARITY Normal parity
(odd) when clear;
other parity (even)
when set.

BIT00 ACTION ENABLE No
action when clear;
trap to vector 114
when set.

2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061

6.0 SUB-TEST SUMMARIES

6.1 Tests

TEST 1

BIT TEST OF ALL CSR'S/MATCH ALL CSR'S WITH MEMORY
(CSR Access may cause wrong Type of Traps)

TEST 2

TEST BANK 0 ACCESSES
Failures are fatal.

TEST 3

TEST BANKS 1-177 (OCTAL) FOR ZEROS AND ONES
Errors are not typed here - only logged in
the configuration table

TEST 4

ECC INHIBIT MODE POINTER TEST

TEST 5

DIAGNOSTIC MODE DISPATCH ROUTINE
This test runs all the patterns in the
mode selected.

TEST 6

UNIQUE BANK TEST
Pattern 27 is run

2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108

6.2 Tests

6.2.1 General Test Information

Actual Tests are identified by symbolic locations "MTPXYY" where X may be any sub program indicator (A,B,C,etc) or 0 and YY will be the number of the test.

Setup procedures for each test are identified by symbolic locations "MT00YY" where YY will be the number of the test.

Tests reside in 4 scripts that are scanned for execution. Symbolic location "MKCSRT" is a table of tests that can run once for each ECC bank. Symbolic location "MKPAT" is a table of tests that can run on each Bank of ECC memory. Symbolic location "MJPAT" is a table of tests that can run on each Bank of Parity memory. Symbolic location "FSPAT" is a table of tests that can be run in Field Service Mode (command 5).

The 1st 3 scripts are completely controlled by the APT E-table (even if not running under APT). Modifications to this table can be made (1) with APT, or (2) manually.

Example E-table Segment:

```
;THE FOLLOWING LOCATIONS SPECIFY WHICH TESTS
;ARE TO BE RUN FOR PARTICULAR MEMORIES
;
;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO TESTS.
;BIT0 SET WILL RUN THE FIRST ENTRY IN THE TABLE, BIT0 SET
;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE...
;
;NOTE**NULL TESTS DO NOT TAKE ANY TIME
```

			RECOMMENDED VALUE	
\$DDW0:	.WORD	177777	;ECC CSR TESTS	177777 TABLE = MKCSRT:
\$DDW1:	.WORD	177777	;ECC CSR TESTS	177777 TABLE = MKCSRT:
\$DDW2:	.WORD	177777	;ECC TESTS	103777 TABLE = MKPAT:
\$DDW3:	.WORD	177777	;ECC TESTS	177777 TABLE = MKPAT:
\$DDW4:	.WORD	177777	;PARITY TESTS	003777 TABLE = MJPAT:
\$DDW5:	.WORD	177777	;PARITY TESTS	177774 TABLE = MJPAT:

2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139

6.2.2 Specific Tests

6.2.2.1 Test 0 Basic Data Test

Writes Reads R2 into a 16K Bank.

This is used for Zeros and Ones testing and in Field Service Mode for any console selected Test.

It can execute out of the USER Instruction PAR's.

NOTE

It is frequently modified dynamically such that (1) it returns after writing only (the 1st NOP is replaced with a RETURN) or (2) it only counts Errors (the code PERRO2 and NOP are replaced with INC @#PATERR).

2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158

6.2.2.2 Test 1 Address Test

Writes Reads an incrementing pattern equivalent to physical
addressed into a 16K Bank.

It can execute out of the USER Instruction PAR's.

2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173

6.2.2.3 Test 2 Complement Address Test

Writes the complement of the physical address from high addresses to low (write down) and reads from low addresses to high (read up).

This provides the complement of the coverage of Test 1 in both data pattern and addressing sequence.

It can execute out of the USER Instruction PAR's.

2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192

6.2.2.4 Test 3 3 XOR 9

Writes Reads a Test that complements as address bits 3 and 9 change.

This test is run 4 times (1) with Zeros Ones, (2) with Ones Zeros, (3) with 401 Ones, and (4) with Ones 401. The test of the 401 is to force a the parity bits to become involved.

It can execute out of the USER Data PDR's, the User Instruction PAR's, the Kernel Data PAR's and the Supervisor Data PAR's.

2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217

6.2.2.5 Test 4 Rotating Zeros Test

Writes a background pattern of Ones. Rotates a Zero Carry Bit left thru each par of bytes (18 times) and then checks that the carry is Zero and the word (2 bytes) is still all Ones.

It can execute out of the User Data PAR's and the Kernel Data PAR's.

NOTE

It is not uncommon to observe the good data equal to the bad data. This indicates that the carry was not clear after 18 ROLB's.

2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243

6.2.2.6 Test 5 Rotating Ones Test

Writes a background pattern of Zeros. Rotates a One carry bit left thru each pair of bytes (18 times) and then checks that the Carry is a One and the Word (2 Bytes) is still all Zeros.

This provides the complement of the coverage of Test 4 in data.

It can execute out of the User Data PAR's and the Kernel Data PAR's.

NOTE

It is not uncommon to observe the good data equal the bad data. This indicates that the Carry was not set after 18 ROLB's.

2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258

6.2.2.7 Test 6 Initial Data Test

Writes Reads a double word first with all bits 0 except 1 (for every bit position), Second with all bits 1 except 1 (for every bit position)

This is a very quick check of the data paths.

2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284

6.2.2.8 Test 7 Address Bit Test

Writes a background of all Zeros.

Read Address 1 for a 0 Byte.

Complement Address 1.

Read Address 1 for a non 0 Byte.

For each Address Bit position from Bit 1:

Virtual (2, 4, 10, 20, 40, 100, 200, 400, 1000, 2000, 4000, 10000,
60000, 20000)

Physical (60002, 60004, 60010, 60020, 60040, 60100, 60200, 60400,
61000, 62000, 64000, 70000, 140000, 100000)

Read Address for a 0 word.

Complement Address contents.

Read Address for a non-zero word.

This is a very quick check of the address bit uniqueness.

2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309

6.2.2.9 Test 10 Byte Addressing Test

With ECC Disabled.

Writes all ones to a double word.

For each of the 4 Bytes in the Double Word.

Clears one byte.

Reads all 4 bytes from double word.

Checks for only proper byte clear.

All other bytes set to all ones.

This is only done on one double word address.

NOTE

This is run for ECC memory only

2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349

6.2.2.10

Test 14 Basic Double Bit Error Test

1. Write the CSR to enable diag mode with a double bit error check bits of 110011 and Uncorrectable Error Indication enabled.
2. Write first AUT in a 16k bank with data of all zero's. This will write the check bits in (1)
3. Read address, this should cause a double bit error. BUS PBL is asserted and we check for a parity trap to occur.
4. Read the CSR for check bits in (1) and Uncorrectable Error Indicator.
5. Write ones to the high byte of the address under test. Since a DBE exists at this address the write should be aborted.
6. Read address and check for a Parity trap to occur as a result of (5)
7. Repeat 5 and 6 for data of ones in the low byte and check for write abort and parity trap.

This test checks to see if a double bit error will be aborted and a byte write of a double bit error will be aborted.

NOTE

This test is only run for the MSV11-J

2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367

6.2.2.11 Test 17 Holding 1's 0's Test

1. Write a 16K Bank with alternating Bytes of Zeros Ones
writing a Byte at a time.
2. Read each Word for correct Test.
3. Do (1-2) again for a complement Test.

This checks the memory for the capability of holding 0's 1's.

2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406

6.2.2.12 Test 20 Syndrome Bits to the CSR on a SBE Test

1. Write CSR with check bits to correct bit 0 of the first AUT 16k bank from a 0 to a 1 with diag mode.
2. Write AUT with data of 0's creating a SBE.
3. Clear CSR.
4. Read the AUT to clock the address and syndromes into the CSR.
5. Read the CSR for the SBE indicator, bit 4.
6. Write the CSR to diag mode to clock the syndrome bits into CSR bits 5-11.
7. Read the CSR for the proper syndrome bits.
8. Repeat 1-7 for all 16 data bits.
9. Repeat 1-8 for data of ones so that a correction will occur from a 1 to a zero.

This test checks to see that the EDC chip can detect Single Bit errors for all 16 data bits by checking for CSR bit#4 and that the proper syndrome bits are placed in the CSR.

NOTE

This test is only run for the MSV11-J

2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457

6.2.2.13 Test 21 Marching 0's 1's Test

1. Write a Background of alternating Bytes of Zeros Ones
2. For the 16K Bank addressing Down
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word
3. For the 16K Bank addressing Up
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word
4. For the 16K Bank addressing Up
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word
5. For the 16K Bank addressing Down
 - (a) Read check a word
 - (b) Byte Swap a word
 - (c) Read check a word

This checks the integrity of the 32 Bit Double Words.

It can execute out of the User Data PAR's.

NOTE

It is not uncommon to see a misleading error typeout because the second test in each case is based upon a byteswap of the first test which may or may not have failed. If the error report indicates errors in pairs with the bad bit in the second report being the same bit position relative to a byte then you should ignore the second error report.

6.2.2.14 Test 22 Refresh Test

1. Write a diagonal Test of ones on every KDIAG(TH) stripe write zeros elsewhere.

This Test is on addresses not bit positions.

Example:

Address

LSB's

MSB's

!	0	0	0	1	0	0	0	1
!	0	0	1	0	0	0	1	0
!	0	1	0	0	0	1	0	0
!	1	0	0	0	1	0	0	0
!	0	0	0	1	0	0	0	1
!	0	0	1	0	0	0	1	0
!	0	1	0	0	0	1	0	0
!	1	0	0	0	1	0	0	0

NOTE

Example uses KDIAG of value 4 more typical is a value of 8. Consult the symbolic definition of "KDIAG" in the program listing to be sure.

2. Disturb each row for > 3.2ms
3. Read check diagonal pattern
4. Do (1-3) KDIAG times moving the placement of the diagonal stripe to cover all address positions.
5. Do (1-4) for a complement pattern (zeros in a background of ones)

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511

2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533

6.2.2.15 Test 23 Shifting Diagonal Test

Similar in overall operation to test 22 except it does not delay for refresh and disturb rows.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554

6.2.2 16 Test 24 Fast Galloping Pattern Test

This does a classical galloping pattern except that addressing is incremented by 400 Octal (every 64th double word)

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587

6.2.2.17 Test 26 Random Data Test

Write Random Data in a 16K Bank while incrementing the Addresses.

Read check Random Data.

This routine regenerates the same random numbers by using the same

seed as the write sequence. After the read check the seed is updated so that the next use of this pattern will not invoke the same sequence of random numbers.

If you wish to change the random sequence so that it is different than any other run in the same configuration then there are 2 ways of doing so.

1. Modify symbolic locations "SEEDHI" and "SEEDLO" to any number you like.
2. Enter Field Service Mode and execute this Test (command 5) on some (any good) bank for a short time (30 sec or so).

This can execute out of the User Data PAR's, the Kernel Data PAR's, and the Supervisor Data PAR's.

2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605

6.2.2.18 Test 27 Unique Bank Test

This Test uses Test 0 to write read the Bank number in each bank.

It does not test Banks that require relocation to test.

It does not run as part of any script but rather is always run after normal pattern tests are complete.

2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623

6.2.2.19 Test 30 Flush Out DBE's Test

This Reads each location then moves the old value back in. This is done with ECC Disabled and therefore corrects any DBE's or SBE's (if possible).

It does not run as part of any script but rather is always run just prior to the End of Pass Code, as part of a Control "C" (Boot) command, as part of End of Pass shutdown for ACT or XXDP Chain Mode, as part of hanging sequence after an error if under ACT or APT, and as part of a shutdown sequence directed by Switch 8 (Halt Program).

6.2.2.20 Test 31 SOB-A-LONG Test

Rationalization

In order to concentrate the memory cycles of a test into a particular address, we must cut the overhead cycles to a minimum. Frequently, the instruction itself may provide adequate data or set up a background in which any complemented bit may find it hard to survive.

The SOB instruction is the only PDP-11 instruction that is (1) a single operand, (2) can be repeatedly executed at the same PC and, (3) can escape this repetitious loop.

Hence, it can be possible to SOB a MOS cell to death (or at least brain wash him), and to SOB a core into over-heating (or at least warm discomfort).

The SOB Routine will be loaded and called with R0 set equal to the SOB constant "SOBK", R1 set equal to the complement of a "SOB R0,.." Instruction "100776".

Simplified SOB Example:

1\$:	SOB	R0,1\$;SOB till R0 underflows
	MOV	R1,1\$;Write complement of SOB
	CMP	R1,1\$;Read check not SOB
	BEQ	2\$;Skip if OK
	SOBFAIL		;Trap report error
2\$:	SOBMOV1		;Code to get self moved
	SOBMOV2		;Forward 1 word and run again
	SOBMOV3		
	SOBMOV4		
	SOBMOV...		

The value of the SOB constant can be found at symbolic location "SOBK" (typical 25 decimal).

This test is not in the normal script of execution but may be added via the APT E-TABLE, reference symbolic locations "MKPAT", "MJPAT", "\$DDW2-5". Field Service Mode command 8 is the normal method of running this pattern.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728

6.2.2.21 Test 32 Write Recovery Test

This test causes a WRITE, READ, WRITE, READ, ... to occur in memory and if the 1st, 3rd, 5th, ... READ is bad the program may bomb or if the 2nd, 4th, 6th, ... READ is bad the program will gracefully type out the error.

Write Recovery Test

This test differs from other tests in that it consists of a small test program actually running in the bank under test.

The program is self modifying and may be difficult to debug. To aid in the debug, remember that the bank and margin are being displayed. This will allow the user to at least see which memory bank failed.

The test consists of 1/2 of the bank stored with "MOV R2, -(PC)" and the other 1/2 containing "177667". "177667" is the complement of "JMP (R0)" instruction. R2 contains "COM -(R1)" instruction on entry to the bank and R1 contains the highest test address in that bank.

If you understand this so far the rest is easy.

The test execution is as follows:

1. The "MOV R2, -(PC)" instruction executes storing the contents of R2 in the address it vacated (due to -(PC)).
2. Since R2 contains a "COM -(R1)" instruction it complements the highest address under test. this address contained "177667" so after the COM -(R1) it equals 110 cleverly this is the "JMP (R0)" instruction.
3. This sequence continues until the "MOV R2, -(PC)" instructions reach the middle of the test bank. then the "JMP (R0)" instruction is met and executed. R0 contained the return address back to test 13.
4. These steps are repeated for each bank under test.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

6.2.2.22 Test 33 Branch Gobble Test

This test loads a small routine into the memory under test. The routine moves itself along in memory one word after each pass so that when it reaches the end every instruction has executed from every location with the exception of the beginning and end of each test area.

The Branch Gobble's general format after you eliminate setup code and code to move the program along is as follows.

BGTEST:	0		;TEST WORD
BRGOBB:	SEC		
	ADCB	BGTEST	;INC LOW BYTE
	BMI	1\$;END LOOP AFTER 128 TIMES
	INCB	BGTEST+1	;INC HIGH BYTE
	BR	BRGOBB	;LOOP 128 TIMES
1\$:	BVS	2\$;BRANCH IF V-BIT SET (SHOULD BE)
	ERROR		;ERROR TRAP
2\$:	CLV		;CLEAR V-BIT
	INCB	BGTEST	;INC HIGH BYTE ONE LAST TIME
	BCS	3\$;BRANCH IF C-BIT SET (SHOULD NOT BE)
	BVC	3\$;BRANCH IF V-BIT CLEAR (SHOULD NOT BE)
	BMI	4\$;BRANCH IF N-BIT SET (SHOULD BE)
3\$:	ERROR		;ERROR TRAP
4\$:	RETURN		

This code originally came from the PDP-11 Family Instruction Exerciser DZQKA-A. The first MOS memorys fell succceptable to this section of that diagnostic and it has been an important memory exerciser ever since.

NOTE

This test is not normally executed except under APT or ACT. It may be invoked VIA Field Service Command 13 (Kamikaze Mode).

2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778

2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804

6.2.2.23 Test 34 Soft Error Test

Rationalization

MOS chips have a failure mode in which they can randomly pick or drop bits. This is caused by Alpha particles bombarding the cell. If the cell is very small (and they are) then the electrons displaced by the Alpha particle are sufficient to cause the cell to change from a one to a zero or from a zero to a one.

This test is controlled by the main program so that it is used to create a Test of 125252 and 52525 on alternate passes of the program. The configuration table is used to flag banks that have the Test invalidated because another Test was written over this background.

This Test is nothing more than a clever use of Test 0.

2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834

6.2.2.24 Test 35 Worst Case Parity Test

1. Force Write Wrong Parity in each 1K word block of the Memory Under Test.
2. Read with Parity Trapping enabled, making sure that a trap occurs.
3. Make sure error address bits are set correctly.
4. Write good parity without trapping, and make sure no trap occurs when read.

NOTE

This test is run for parity memory which is not controlled by the same CSR as the program.

2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862

6.2.2.25 Test 36 Correction Code Test

1. Write CSR with check bits to correct bit 0 of the first address in a 16k bank from a 0 to a 1 with diag mode.
2. Write AUT with data of 0's.
3. Read AUT for correction of bit 0 from a 0 to a 1.
4. Repeat 1 3 for all 16 data bits.
5. Repeat 1-4 for data of ones so that a correction will occur from a 1 to a zero.

This test checks to see that the EDC chip can correct Single Bit Errors for all 16k data bits from a 1 to a 0 and visa versa.

NOTE

This test is only run for the MSV11-J

2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888

6.2.2.26 Test 37 Check ECC Disable Test

1. Write CSR with ECC disable, Diag mode, and SBE check bits of 000010.
2. Write AUT with data of zero's. This should write check bits to memory.
3. Read AUT for data of zeros insuring no correction was made.

NOTE

This test is run on the MSV11-J only.

2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908
2909
2910
2911
2912
2913
2914
2915
2916
2917

6.2.2.27 Test 41 Address to CSR on DBE Test

1. Write CSR with ECC disable, Diag mode, and Double Bit error check bits of 010011
2. Write AUT with data of zeros creating a DBE.
3. Read AUT to detect DBE and to clock address into CSR
4. Read CSR for correct address in bits 5-11.
5. Increment address by 1k and repeat 1-4 until 16k is done.

This test insures that the correct address appears in CSR bits 5-11 on a DBE

NOTE

This test is run on a MSV11-J only.

2919
2920
2921
2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940
2941
2942
2943
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953

6.2.2.28 Test 42 Extended Address to CSR on Error Test

1. Write CSR with SBE check bits of 000010 with Diagnostic Mode.
2. Write low address in a 16k bank with data of zeros creating a SBE.
3. Clear the CSR.
4. Read address to detect SBE.
5. Read CSR for correct address and the SBE indicator bit #4.
6. Enable CSR bit 14 to check the Extended address bits.
7. Read CSR for correct address bits
8. Repeat 1-7 with a test address that is the highest in a 16k bank.

This test checks to see that the correct address bits appear in the CSR.
This is also repeated for the Extended Address function in the CSR.

NOTE

This test is only run for the MSV11 J

2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984
2985
2986
2987
2988
2989
2990
2991
2992
2993
2994

6.2.2.29 Test 43 Write Byte Test

1. Write CSR to Diag mode with check bits of 001100. These correspond to data of zeros.
2. Write first AUT with data of one in bit zero. The write effectively creates a SBE in byte 0.
3. Clear the CSR
4. Write byte 1 of the AUT with data of all ones.
5. Read CSR to check for SBE indication.
6. Write the CSR to Diag mode.
7. Read the AUT to check for the correct data -- all ones in high byte and all zeros in low byte.
8. Read the CSR to check for correct check bits corresponding to the data read in (7). These check bits are 000110.
9. Repeat (1)-(8) this time creating an error in byte 1 (2) and writing byte 0 in (4).

This test checks to see that a SBE will be corrected during the read portion of the byte write and that correct checkbits will be generated on the write.

NOTE

This test is only run for the MSV11-J

J6

2996
2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039

6.2.2.30 Test 44 Shifting Checkbits through the CSR Test

1. Write CSR to Diag Mode to Enable Checkbit register.
2. Write CSR with check bits of 000001, ECC disable and Diag mode
3. Write memory with data of zeros. This should write the check bits into memory.
4. Complement check bits pattern and write CSR as in (2).
5. Read CSR for complement check bit pattern.
6. Read memory to read check bits written in (2) into CSR.
7. Read CSR for corret check bits written in (2).
8. Shift check bit pattern and repeat (1-7) till CSR bits 5-10 are done.
9. Complement check bit pattern in (2) and repeat (1 8) shifting a zero through a field of ones.
10. Repeat 1-9 for every 100 octal locations in 16k

This test checks the ability to read check bits from the CSR to memory and back. The test is done twice. Once shifting a field of a one through a field of zeros and a zero through a field of ones. This tests the Checkbit/Syndrome bit register and Check bit RAM's. This test is done for every 100 octal locations in 16K.

NOTE

This test is only run for the MSV11-J

3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056
3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076

6.2.2.31 Test 45 Syndrome Bits to the CSR on a DBE Test

1. Write CSR with Diag mode to enable Check/Syndrome bit Register.
2. Write CSR with DBE check bits of 110011 with Diag mode.
3. Write memory with data of zeros creating a DBE.
4. Clear CSR.
5. Read memory to detect DBE.
6. Read CSR for Uncorrectable error indicator.
7. Write CSR to Diag mode to read Syndrome bits into CSR.
8. Read CSR for correct Syndrome bits of 111111.
9. Repeat (1-8) with Multiple Bit Error check bits of 111100 and corresponding Syndrome Bits of 110000.

This test checks the ability of the CSR to detect a DBE and read for the proper Syndrome bits generated by the EDC chip. This test is then repeated with check bits corresponding to a Multiple Bit error.

NOTE

This test is only run for the MSV11-J

3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107
3108
3109
3110
3111
3112
3113
3114

6.2.2.32 Test 46 Check Single Bit Errors with ECC Disabled Test

1. Write CSR with check bits to correct bit 0 of the first address in a 16k bank from a 0 to a 1 with diag mode and ECC disabled.
2. Write AUT with data of 0's thus creating a SBE.
3. Write the CSR to ECC disable.
4. Read AUT to detect SBE.
5. Check to see that no trap occurred.
6. Read CSR to see that uncorrectable error (CSR15) is set.
7. Repeat 1-6 for all 16 data bits.
8. Repeat 1-7 for data of ones so that a correction will occur from a 1 to a zero.
9. Repeat 1-8 except in steps (3) the CSR is written to ECC Disable and BUS PBL enable and (5) we check for traps.

This test checks to see that SBE are treated as uncorrectable errors with ECC Disable. The test is repeated 2 times, once with traps disabled and again with it enabled. This is done for all 16 possible SBE conditions.

NOTE

This test is only run for the MSV11-J

3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149

6.2.2.33 Test 47 NO CSR Update On SBE with exsisting DBE test

1. Write the CSR to Diag mode to enable Checkbit/Syndrome bit Register.
2. Write the CSR with DBE check bits of 110011 and Diag mode.
3. Write memory with data of zeros creating a DBE.
4. Write CSR with SBE check bits of 000010 and Diag mode.
5. Write memory 4k above address in (3) creating a SBE.
6. Clear CSR.
7. Read memory with address in (3) to detect DBE.
8. Read CSR for correct address and Uncorrectable error indicator
9. Read memory with address in (5) to detect SBE.
10. Read CSR for SBE indicator and no change in DBE status in CSR in (8)

This test checks to see that no update will occur in the CSR with a SBE in memory when a DBE already exists.

NOTE

This test is only run for the MSV11-J

3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163

6.2.2.34 Test 999 Null Test

This is an instant return added to preserve the software structure.

This test replaces any real tests when the APT E-Table does not specify a test to be run.

7.0 PROGRAM FEATURES

7.1 Fast Data Access Rates

One of the main areas of concern in testing memory in systems environments is speed. One of the prime reasons that system programs like RSTS, IAS and MUMPS can crash due to memory failures not detectable by memory diagnostics (0-124K, 0-2 MEG, etc.) is because of multiple NPR devices contending for the bus. After some delay a NPR device becomes bus master and does several memory transfers at memory data rates.

On the other hand most diagnostics when writing reading and/or checking patterns spend most of their time fetching instructions and operands out of their program space and proportionally little time accessing the memory under test.

This diagnostic's error detecting abilities have been optimized around the primary design criteria of speed. To this end the following steps have been taken.

7.1.1 Fast City

Utilization of Memory Management Registers as Non Memory Bus, Non QBUS, Bipolar Memory. Since User Mode is only used for relocation and Data Space is never used, then subroutines can be executed from the UIPAR's, UDPAR's, KDPAR's, SDPAR's and with some Bit Pattern restrictions the UIPDR's, UDPDR's, KDPDR's, and SDPDR's.

The program runs in Kernel mode and Patterns are executed in Supervisor mode for mapping purposes. All core patterns and some MOS Patterns are subroutines that are moved to this Bipolar region referred to in the program as Fast City.

NOTE

18-Bit PDP-11's cannot execute from the PAR's because their PAR's are only 12 bits wide; they also have no Supervisor Mode. Therefore, all patterns are executed in memory, using User Mode (reference Section 7.5).

7.1.2 SOB's

Utilization of the full PDP-11 Instruction Set to speed pattern algorithms (principally the SOB).

7.1.3 CACHE

CACHE is used between pattern tests to decrease program pass times. CACHE can be defeated by the operator (reference section 2.4.3.1).

7.2 Bank Zero Testing

Bank Zero has been traditionally neglected by memory diagnostics for the following reason.

The vector space exists there and ALL traps must not access test pattern data. If the area is tested the diagnostic must not use any traps, and it is against the rules for power to fail.

Systems with Memory Management can overcome this because all traps are to Kernel Virtual space even if the power should fail (caution must be observed because power up goes to physical address 24 (because the Memory Management Unit comes up off)).

However, Catch 22 is that the diagnostic is not APT compatible in this mode because APT Accesses Physical Memory Locations.

The PDP-11/83 can overcome this because the QBUS Map can fool APT.

Because of the previous arguments this program does not relocate in the true sense of the word (i.e. no position independent code was written (at least not on purpose)), but rather this program moves and remaps (hereafter referred to as relocates). This enables the complete testing of Bank Zero or any other program space or privileged space exactly as all other banks are tested. (The conditional test to see if a bank is protected is complemented when relocated).

NOTE

The program will relocate only in the first pass under APT; after this, the program will remain fixed in Banks 0 and 1.

7.3 Memory Configuration Map

Example:

16K BANKS

[illegible]

Displayed are Banks 0-77 Octal (1 meg words). If the Fat Terminal Switch was set (reference section 2.4.1) then all Banks (0-177) would be shown. If this was an 18-Bit PDP-11 (eg - 11/23), only Banks 0-7 would be printed.

The fields:

ERRORS:

The sizing routine could not write zeros and ones in Banks 10 11, hence they are marked as bad with X's.

ERRORS:

MEMTYPE:

Banks 0-7 are Memory Type E (MSV11-J), and Banks 10-37 are Memory Type P (MSV11-L/P) and Banks 40-77 are memory type E(MSV11-J). Banks 100-167 do not exist.

CSR:

3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348
3349
3350
3351
3352
3353
3354
3355
3356
3357
3358
3359
3360
3361
3362
3363
3364
3365
3366

Banks 0-7 are assigned to CSR 172100, 10-17 to CSR 172102, and 20-37 to interleaved CSR's 172104 and 172106 and banks 40-77 are assigned CSR 17210.

PROTECT:

Banks 0 and 1 are protected because they are program space. Bank 0 and 1 can also be protected because they are in the bottom 16K of an MSV11-L/P CSR. The protection is hierarchical and program space overshadows MSV11-L/P protection. Banks 0 and 1 will not be tested until the program relocates.

7.4 Everything You've Always Wanted To Know About SUPERMAC ...

SUPER-MAC is a set of structured programming macros that allows programs to be written in a high level, easily understood language.

As a general rule, most SUPER-MAC statements can be single-line statements or multiple-line (nested) block statements. A single-line statement must be completed on one source line; no continuation lines are allowed. Single-line statements should be as short and simple as possible. Comments may also be included on a source line. All the general rules, conditions, etc., that govern MACRO 11 also govern SUPER-MAC. Spacing on a source line is very important. The elements should be separated by a comma or a space. Tabs should never be used for spacing. For example: The expression $A+B$ is interpreted different than $A + B$.

All the conditional statements can be written as multiple-line nested blocks. Each level of nesting within a block must be terminated with an associated END statement. Each level of nesting should be indented two spaces.

User written macros or assembly language instructions may be included in a program if desired. As a debugging aid, if the symbol LST\$\$ is defined, it will cause generated code and labels to be listed. All programs must begin with the macro call SMACIT. This call initializes SUPER-MAC. All legal PDP 11 source and destination operands are legal in SUPER-MAC.

```

3368
3369       7.4.1 Sample Source File -
3370       .ENABL ABS
3371       .ENABL AMA
3372       .MCALL .SUPER
3373       .SUPER
3374       ;LST$$=0
3375       BIT5=40
3376       A:      0
3377       B:      0
3378       C:      0
3379       D:      0
3380       E:      0
3381       F:      0
3382       G:      0
3383       H:      0
3384       I:      0
3385       J:      0
3386       .PAGE
3387       ;LET EXAMPLES
3388       LET RO := A
3389       LET B := C + D
3390       LET E := F + 1
3391       LET G := H + 2
3392       LET J := J + 01
3393       LET A :B= B
3394       ;IF EXAMPLES
3395       IF A IS TRUE
3396       MOV 23,D
3397       END ;OF IF A
3398       IF B IS FALSE
3399       MOV 34,E
3400       END ;OF IF B
3401       IF A EQ B THEN LET C := D
3402       IF A LT B
3403       MOV C,D
3404       ELSE
3405       MOV E,D
3406       END ;OF IF A
3407       IF A EQ B AND C NE D
3408       MOV F,G
3409       END ;OF IF A
3410       IF A EQ B OR C NE D
3411       MOV F,G
3412       END ;OF IF A
3413       IFB A EQ B AND C EQ 1
3414       MOV H,J
3415       ELSE
3416       MOV E,J
3417       END ;OF IFB A
3418       IFB A EQ B ANDB C EQ 1
3419       MOV H,J
3420       ELSE
3421       MOV E,J

```

```

3423      END ;OF IFB A
3424      IF RESULT IS EQ
3425      MOV   A,B
3426      END ;OF IF RESULT
3427      IF BITS SET.IN A
3428      MOV   B,C
3429      END ;OF IF BITS
3430      IF BITS OFF.IN A
3431      MOV   C,D
3432      END ;OF IF BITS
3433      ;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
3434      ;ON.ERROR EXAMPLES
3435      ON.ERROR
3436      MOV   A,B
3437      ELSE
3438      MOV   C,B
3439      END ;OF ON.ERROR
3440      ON.NOERROR
3441
3442
3443      MOV   C,B
3444      ELSE
3445      MOV   A,B
3446      END ;OF ON.NOERROR
3447      ON.ERROR THEN LET A :B= B
3448      ;FOR EXAMPLES
3449      FOR I := -5 TO 23
3450      INC   A
3451      END ;OF FOR I
3452      FOR RO := 0 TO 140 BY 4
3453      DEC   A(RO)
3454      END ;OF FOR RO
3455      FOR I := 133 DOWNT0 3 BY 2
3456      ADD   A,B
3457      END ;OF FOR I
3458      ;BEGIN EXAMPLES
3459      BEGIN ALPHA
3460      FOR RO := 0 TO 167
3461      MOVB   A(RO),B
3462      IF B LT 0 THEN LEAVE ALPHA
3463      END ;OF FOR RO
3464      FOR RO := 400 TO 567
3465      IF B GE 0 THEN LEAVE ALPHA
3466      END ;OF FOR RO
3467      END ALPHA
3468      ;$RETURN EXAMPLES
3469      $RETURN
3470      $RETURN ERROR
3471      $RETURN NOERROR
3472      ;CASE EXAMPLES
3473      MOV   A,RO
3474      CASE RO
3475      A

```

3477
3478
3479
3480
3481
3482
3483
3484
3485
3486
3487
3488
3489
3490
3491
3492
3493
3494
3495
3496
3497
3498
3499
3500
3501
3502
3503
3504
3505
3506
3507
3508
3509
3510
3511
3512
3513
3514
3515
3516
3517
3518
3519
3520
3521
3522
3523
3524
3525
3526
3527
3528
3529

B
C
D
E
F
END ;OF CASE R0
.END

7.4.2 Sample Listing File (with No Expanded Macros) - -
.MAIN. MACRO M1111 01 APR-79 16:41 PAGE 2

1	000000			.ENABL ABS
2				.ENABL AMA
3				.MCALL .SUPER
4	000000			.SUPER
5				;LST\$\$=0
6		000040		BIT5=40
7	000000	000000	A:	0
8	000002	000000	B:	0
9	000004	000000	C:	0
10	000006	000000	D:	0
11	000010	000000	E:	0
12	000012	000000	F:	0
13	000014	000000	G:	0
14	000016	000000	H:	0
15	000020	000000	I:	0
16	000022	000000	J:	0

.MAIN. MACRO M1111 01-APR-79 16:41 PAGE 3

18					;LET EXAMPLES
19	000024				LET RO := A
20	000030				LET B := C + D
21	000044				LET E := F + 1
22	000056				LET G := H + 2
23	000072				LET J := J + 01
24	000100				LET A :B= B
25					;IF EXAMPLES
26	000106				IF A IS TRUE
27	000114	012737	000023	000006	MOV 23,D
28	000122				END ;OF IF A
29	000122				IF B IS FALSE
30	000130	012737	000034	000010	MOV 34,E
31	000136				END ;OF IF B
32	000136				IF A EQ B THEN LET C := D
33	000154				IF A LT B
34	000164	013737	000004	000006	MOV C,D
35	000172				ELSE

3531	36	000174	013737	000010	000006
3532	37	000202			
3533	38	000202			
3534	39	000222	013737	000012	000014
3535	40	000230			
3536	41	000230			
3537	42	000250	013737	000012	000014
3538	43	000256			
3539	44	000256			
3540	45	000276	013737	000016	000022
3541	46	000304			
3542	47	000306	013737	000010	000022
3543	48	000314			
3544	49	000314			
3545	50	000334	013737	000016	000022
3546	51	000342			
3547	52	000344	013737	000010	000022
3548	53	000352			
3549	54	000352			
3550	55	000354	013737	000000	000002
3551	56	000362			
3552	57	000362			
3553	58	000372	013737	000002	000004
3554	59	000400			
3555	60	000400			
3556	61	000410	013737	000004	000006
3557	62	000416			
3558	63				
3559	64				
3560	65	000416			
3561	66	000420	013737	000000	000002
3562	67	000426			
3563	68	000430	013737	000004	000002
3564	69	000436			
3565	70	000436			
3566	71	000440	013737	000004	000002
3567	72	000446			
3568	73	000450	013737	000000	000002
3569	74	000456			

```

MOV E,D
END ;OF IF A
IF A EQ B AND C NE D
MOV F,G
END ;OF IF A
IF A EQ B OR C NE D
MOV F,G
END ;OF IF A
IFB A EQ B AND C EQ 1
MOV H,J
ELSE
MOV E,J
END ;OF IFB A
IFB A EQ B ANDB C EQ 1
MOV H,J
ELSE
MOV E,J
END ;OF IFB A
IF RESULT IS EQ
MOV A,B
END ;OF IF RESULT
IF BITS SET.IN A
MOV B,C
END ;OF IF BITS
IF BITS OFF.IN A
MOV C,D
END ;OF IF BITS
;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
;ON.ERROR EXAMPLES
ON.ERROR
MOV A,B
ELSE
MOV C,B
END ;OF ON.ERROR
ON.NOERROR
MOV C,B
ELSE
MOV A,B
END ;OF ON.NOERROR

```

.MAIN. MACRO M1111 01-APR-79 16:41 PAGE 3-1

3571					
3572					
3573					
3574					
3575	75	000456			
3576	76				
3577	77	000466			
3578	78	000474	005237	000000	
3579	79	000500			
3580	80	000514			
3581	81	000516	005360	000000	
3582	82	000522			
3583	83	000534			

```

ON.ERROR THEN LET A :B= B
;FOR EXAMPLES
FOR I := -5 TO 23
INC A
END ;OF FOR I
FOR RO := 0 TO 140 BY 4
DEC A(RO)
END ;OF FOR RO
FOR I := 133 DOWNT0 3 BY 2

```


J7

3585	84	000542	063737	000000	000002	ADD A,B
3586	85	000550				END ;OF FOR I
3587	86					;BEGIN EXAMPLES
3588	87	000566				BEGIN ALPHA
3589	88	000566				FOR RO := 0 TO 167
3590	89	000570	116037	000000	000002	MOVB A(RO),B
3591	90	000576				IF B LT 0 THEN LEAVE ALPHA
3592	91	000604				END ;OF FOR RO
3593	92	000614				FOR RO := 400 TO 567
3594	93	000620				IF B GE 0 THEN LEAVE ALPHA
3595	94	000626				END ;OF FOR RO
3596	95	000636				END ALPHA
3597	96					;\$RETURN EXAMPLES
3598	97	000636				\$RETURN
3599	98	000640				\$RETURN ERROR
3600	99	000644				\$RETURN NOERROR
3601	100					;CASE EXAMPLES
3602	101	000650	013700	000000		MOV A,RO
3603	102	000654				CASE RO
3604	103	000664	000000			A
3605	104	000666	000002			B
3606	105	000670	000004			C
3607	106	000672	000006			D
3608	107	000674	000010			E
3609	108	000676	000012			F
3610	109	000700				END ;OF CASE RO
3611	110					
3612	111		000001			.END

7.4.3 Sample Listing File (with Expanded Macros) - -
 .MAIN. MACRO M1111 01-APR-79 16:10 PAGE 2

3617	1	000000				.ENABL ABS
3618	2					.ENABL AMA
3619	3					.MCALL .SUPER
3620	4	000000				.SUPER
3621	5		000000			LST\$\$=0
3622	6		000040			BIT5=40
3623	7	000000	000000		A:	0
3624	8	000002	000000		B:	0
3625	9	000004	000000		C:	0
3626	10	000006	000000		D:	0
3627	11	000010	000000		E:	0
3628	12	000012	000000		F:	0
3629	13	000014	000000		G:	0
3630	14	000016	000000		H:	0
3631	15	000020	000000		I:	0
3632	16	000022	000000		J:	0

.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 3

3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654
3655
3656
3657
3658
3659
3660
3661
3662
3663
3664
3665
3666
3667
3668
3669
3670
3671
3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688

18
19 000024 013700 000000
20 000030 000030 013737 000004 000002
000036 063737 000006 000002
21 000044 000044 013737 000012 000010
000052 005237 000010
22 000056 000056 013737 000016 000014
000064 062737 000002 000014
23 000072 000072 062737 000001 000022
24 000100 000100 113737 000002 000000
25
26 000106 000106 005737 000000
000112 001403
27 000114 012737 000023 000006
28 000122 000122
29 000122 000122 005737 000002
000126 001003
30 000130 012737 000034 000010
31 000136 000136
32 000136 000136 023737 000000 000002
000144 001003
000146 013737 000006 000004
000154
33 000154 000154 023737 000000 000002
000162 002004
34 000164 013737 000004 000006
35 000172 000172 000403
000174
36 000174 013737 000010 000006
37 000202 000202
38 000202 000202 023737 000000 000002
000210 001007
000212 023737 000004 000006
000220 001403
39 000222 013737 000012 000014
40 000230

;LET EXAMPLES

LET R0 := A
MOV A,R0
LET B := C + D
MOV C,B
ADD D,B
LET E := F + 1
MOV F,E
INC E
LET G := H + 2
MOV H,G
ADD 2,G
LET J := J + 01
ADD 01,J
LET A := B + B
MOV B,A

;IF EXAMPLES

IF A IS TRUE
TST A
BEQ L0
MOV 23,D
END ;OF IF A

L0:

IF B IS FALSE
TST B
BNE L1
MOV 34,E
END ;OF IF B

L1:

IF A EQ B THEN LET C := D
CMP A,B
BNE L2
MOV D,C

L2:

IF A LT B
CMP A,B
BGE L3
MOV C,D
ELSE
BR L4

L3:

MOV E,D
END ;OF IF A

L4:

IF A EQ B AND C NE D
CMP A,B
BNE L5
CMP C,D
BEQ L5
MOV F,G
END ;OF IF A

L7

SEQ 0089

3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709
3710
3711
3712
3713
3714
3715
3716
3717
3718
3719
3720
3721
3722
3723
3724
3725
3726
3727
3728
3729
3730
3731
3732
3733
3734
3735
3736
3737
3738
3739
3740
3741
3742

000230
41 000230
000230 023737 000000 000002
000236 001404
000240 023737 000004 000006
000246 001403

L5:

IF A EQ B OR C NE D
CMP A,B
BEQ L6
CMP C,D
BEQ L7

.MAIN. MACRO M1111 01-APR 79 16:10 PAGE 3-1

000250
42 000250 013737 000012 000014
43 000256
000256
44 000256
000256 123737 000000 000002
000264 001010
000266 023727 000004 000001
000274 001004
45 000276 013737 000016 000022
46 000304
000304 000400
000306
47 000306 013737 000010 000022
48 000314
000314
49 000314
000314 123737 000000 000002
000322 001010
000324 123727 000004 000001
000332 001004
50 000334 013737 000016 000022
51 000342
000342 000403
000344
52 000344 013737 000010 000022
53 000352
000352
54 000352
000352 001003
55 000354 013737 000000 000002
56 000362
000362
57 000362
000362 032737 000040 000000
000370 001403
58 000372 013737 000002 000004
59 000400
000400
60 000400
000400 032737 000040 000000

L6:

MOV F,G
END ;OF IF A

L7:

IFB A EQ B AND C EQ 1
CMPB A,B
BNE L10
CMP C, 1
BNE L10

MOV H,J

ELSE

BR L11

L10:

MOV E,J
END ;OF IFB A

L11:

IFB A EQ B ANDB C EQ 1
CMPB A,B
BNE L12
CMPB C, 1
BNE L12

MOV H,J

ELSE

BR L13

L12:

MOV E,J
END ;OF IFB A

L13:

IF RESULT IS EQ
BNE L14

MOV A,B

END ;OF IF RESULT

L14:

IF BITS SET.IN A
BIT BITS,A
BEQ L15

MOV B,C

END ;OF IF BITS

L15:

IF BITS OFF.IN A

BIT BITS,A

3744	61	000410	013737	000004	000006	MOV C,D
3745						
3746	62	000416				END ;OF IF BITS
3747		000416				L16:
3748	63					;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
3749	64					;ON.ERROR EXAMPLES
3750	65	000416				ON.ERROR
3751		000416	103004			BCC L17
3752	66	000420	013737	000000	000002	MOV A,B
3753	67	000426				ELSE
3754		000426	000403			BR L20
3755		000430				L17:
3756						
3757	68	000430	013737	000004	000002	MOV C,B
3758						
3759	69	000436				END ;OF ON.ERROR
3760		000436				L20:
3761	70	000436				ON.NOERROR

.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 3-2

3762						
3763						
3764						
3765						
3766						
3767		000436	103404			BCS L21
3768	71	000440	013737	000004	000002	MOV C,B
3769	72	000446				ELSE
3770		000446	000403			BR L22
3771		000450				L21:
3772	73	000450	013737	000000	000002	MOV A,B
3773	74	000456				END ;OF ON.NOERROR
3774		000456				L22:
3775	75	000456				ON.ERROR THEN LET A :B B
3776		000456	103003			BCC L23
3777		000460	113737	000002	000000	MOVB B,A
3778		000466				L23:
3779	76					;FOR EXAMPLES
3780	77	000466				FOR I := -5 TO 23
3781		000466	012737	177773	000020	MOV -5,I
3782		000474				B0:
3783	78	000474	005237	000000		INC A
3784	79	000500				END ;OF FOR I
3785		000500	005237	000020		INC I
3786		000504	023727	000020	000023	CMP I, 23
3787		000512	003770			BLE B0
3788		000514				E0:
3789	80	000514				FOR R0 := 0 TO 140 BY 4
3790		000514	005000			CLR R0
3791		000516				B1:
3792	81	000516	005360	000000		DEC A(R0)
3793	82	000522				END ;OF FOR R0
3794		000522	062700	000000		ADD 4,R0
3795		000526	020027	000140		CMP R0, 140

3797	000532	003771							
3798	000534								
3799	83 000534					E1:	BLE B1		
3800	000534	012737	000133	000020			FOR I := 133 DOWNT0 3 BY 2		
3801	000542						MOV 133,I		
3802	84 000542	063737	000000	000002		B2:	ADD A,B		
3803	85 000550						END ;OF FOR I		
3804	000550	162737	000002	000020			SUB 2,I		
3805	000556	023727	000020	000003			CMP I, 3		
3806	000564	002366					BGE B2		
3807	000566					E2:			
3808	86						;BEGIN EXAMPLES		
3809	87 000566						BEGIN ALPHA		
3810	000566					B3:			
3811	88 000566						FOR R0 := 0 TO 167		
3812	000566	005000					CLR R0		
3813	000570					B4:			
3814	89 000570	116037	000000	000002			MOVB A(R0),B		
3815	90 000576						IF B LT 0 THEN LEAVE ALPHA		
3816	000576	005737	000002				TST B		
3817	000602	002415					BLT E3		
3818	91 000604						END ;OF FOR R0		
3819	000604	005200					INC R0		
3820	000606	020027	000167				CMP R0, 167		
3821	000612	003766					BLE B4		
3822	000614					E4:			
3823	92 000614						FOR R0 := 400 TO 567		
3824	000614	012700	000400				MOV 400,R0		
3825									
3826									
3827									
3828									
3829									
3830									
3831	000620					B5:			
3832	93 000620						IF B GE 0 THEN LEAVE ALPHA		
3833	000620	005737	000002				TST B		
3834	000624	002004					BGE E3		
3835	94 000626						END ;OF FOR R0		
3836	000626	005200					INC R0		
3837	000630	020027	000567				CMP R0, 567		
3838	000634	003771					BLE B5		
3839	000636					E5:			
3840	95 000636						END ALPHA		
3841	000636					E3:			
3842	96						;RETURN EXAMPLES		
3843	97 000636						\$RETURN		
3844	000636	000207					RTS PC		
3845	98 000640						\$RETURN ERROR		
3846	000640	000261					SEC		
3847	000642	000207					RTS PC		
3848	99 000644						\$RETURN NOERROR		
3849	000644	000241					CLC		
	000646	000207					RTS PC		

.MAIN. MACRO M1111 01-APR-79 16:10 PAGE 3-3

3851	100				;CASE EXAMPLES
3852	101	000650	013700	000000	MOV A,R0
3853	102	000654			CASE R0
3854		000654	010046		MOV R0,-(SP)
3855		000656	006316		ASL @SP
3856		000660	004737	000700	JSR PC,L24
3857	103	000664	000000		A
3858	104	000666	000002		B
3859	105	000670	000004		C
3860	106	000672	000006		D
3861	107	000674	000010		E
3862	108	000676	000012		F
3863	109	000700			END ;OF CASE R0
3864		000700			L24:
3865		000700	062616		ADD (SP)+,@SP
3866		000702	013646		MOV @SP+,-(SP)
3867		000704	004736		JSR PC,@SP+
3868	110				
3869	111		000001		.END

7.5 Memory Management Mapping

7.5.1 Memory Management Mapping For The 11/83 -

PAR	SUPERVISOR	KERNEL	USER
---	-----	-----	----
0	Program	Program	Dst Bk/Fst Mem
1	Program	Program	Src Bk/Fst Mem
2	Program	Program	Src Bk/Fst Mem
3	Test Area	Program	Src Bk/Fst Mem
4	Test Area	Program	Dst Bk/Fst Mem
5	Test Area	Program	Dst Bk/Fst Mem
6	Test Area	Map to CSR's	Dst Bk/Fst Mem
7	Perif Page	Perif Page	Dst Bk/Fst Mem

7.5.2 Memory Management Mapping For QBUS-11's With Supervisor Mode (eg 11/23B) -

PAR	SUPERVISOR	KERNEL	USER
---	-----	-----	----
0	Program	Program	Dst Bk
1	Program	Program	Src Bk
2	Program	Program	Src Bk
3	Test Area	Program	Src Bk
4	Test Area	Program	Dst Bk
5	Test Area	Program	Dst Bk
6	Test Area	Map to CSR's	Dst Bk
7	Perif Page	Perif Page	Dst Bk

3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3916

7.5.3 Memory Management Mapping For QBUS-11's W/o Supervisor Mode (eg 11/2³) -

PAR	KERNEL	USER
---	-----	-
0	Program	Program/Dst Bk
1	Program	Program/Src Bk
2	Program	Program/Src Bk
3	Program	Test Area/Src Bk
4	Program	Test Area/Dst Bk
5	Program	Test Area/Dst Bk
6	Map to CSR's	Test Area/Dst Bk
7	Perif Page	Perif Page/Dst Bk

```

3918 .LIST TOC
3919 000000 .ENABL ABS
3920 .ENABL AMA
3921 .DSABL GBL
3922 ;NOTE: CVMJAA0.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
3923 ;THIS PROGRAM. ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
3924 .MCALL SMACIT,..PUSH,..POP,..TAG,..BRAN,..EMIT,..EMITN,..EMITL,..EMITR
3925 .MCALL .IFOPR,..IS,..GENBR,..OPADD,..OPSUB,CLEAR,SET,CLEARB,SETB
3926 .MCALL RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
3927 .MCALL IF,..OR,..IFARI,..LEAVE,..GOTO,OR,AND,THEN,ELSE,WHILE,CASE
3928 .MCALL FOR,TO,DOWNT0,REPEAT,UNTIL,THRU,END,BEGIN
3929 .MCALL $$END,LEAVE,JUMPT0,GOTO,PUSH,POP,LET
3930 .MCALL .SIMPLE,..ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
3931 .MCALL $CALL,$RETURN
3932
3933 .NLIST TTM ;I WANT FAT PAPER!
3934 .LIST MC,SYM ;LIST MACRO CALLS, SYMBOL TABLE
3935 .NLIST MD,CND,ME ;DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
3936 000000 LST$$= 0 ;DEFINED TO LIST SUPERMAC EXPANSIONS
3937 163000 $SWR= 163000 ;USE THESE SYSMAC SWITCHES
3938 000001 $TN= 1 ;FIRST TEST NUMBER TO ONE(1)
3939 000000 SMACIT

```



```

3942          .SBTTL DEFINE TRAPS
3943          ;ALL ENTRIES HERE MUST HAVE A CORRESPONDING ENTRY IN THE
3944          ;TRAP TABLE "$TRPAD" (NEAR END OF PROGRAM).
3945          ;*TRAP DEFINITIONS
3946          ;
3947          ;HERE IS HOW TRAPS WORK IN THIS PROGRAM
3948          ;
3949          ;ALL TRAPS EXECUTE A "TRAP" INSTRUCTION WHICH TAKES THE PROGRAM
3950          ;TO SYMBOLIC LOCATION "$TRAP"
3951          ;
3952          ;AT $TRAP THE PROGRAM PICKS UP THE RIGHT BYTE OF THE TRAP INSTRUCTION
3953          ;AND INDEXES INTO A TABLE AT LOCATION "$TRPAD" WHICH SENDS THE PROGRAM TO
3954          ;THE SPECIFIC ROUTINE TO HANDLE THAT SPECIFIC TRAPS TASK.
3955          ;
3956          ;THE ULTIMATE DESTINATION OF A TRAP INSTRUCTION CAN BE GUESSED AT AS FOLLOWS
3957          ;
3958          ;EXAMPLE:      NOP
3959                        NOP
3960                        NOP
3961                        KERNEL          ;ENTER KERNFL MODE
3962                        NOP
3963          ;
3964          ;      ADD A DOLLAR SIGN TO THE SYMBOLIC NAME AND CHECK THE CRF FOR SOMETHING CLOSE
3965          ;      IN THIS CASE THE CRF HAS $KERNE LISTED AS 032546
3966          ;      AT LOCATION 32546 YOU FIND THE ROUTINE $KERNEL
3967          ;
3968          ;      NOTE THAT CRF SYMBOLS ARE TRUCNATED TO 6 CHARACTERS
3969          ;      SYMBOLIC NAMES GREATER THAT 6 CHARACTERS ARE USED SO I CAN
3970          ;      REMEMBER WHAT THEY MEAN!
3971          ;
3972          ;      ALL GOOD TRAP ROUTINES RETURN VIA AN "RTI" INSTRUCTION
3973          104401 TYPEIT= 104401          ;;TTY TYPEOUT ROUTINE
3974          104402 TYPOC= 104402          ;;TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3975          104403 TYPOS= 104403          ;;TYPE OCTAL NUMBER (NO LEADING ZEROS)
3976          ;TYPON= 104404          ;;TYPE OCTAL NUMBER (AS PER LAST CALL)
3977          104405 TYPDS= 104405          ;;TYPE DECIMAL NUMBER (WITH SIGN)
3978          ;TYPBN= 104406          ;;TYPE BINARY (ASCII) NUMBER
3979          ;
3980          104407 GTSWR= 104407          ;;GET SOFT-SWR SETTING
3981          104410 CKSWR= 104410          ;;TEST FOR CHANGE IN SOFT-SWR
3982          ;
3983          104411 RDCHR= 104411          ;;TTY TYPEIN CHARACTER ROUTINE
3984          104412 RDLIN= 104412          ;;TTY TYPEIN STRING ROUTINE
3985          104413 RDOCT= 104413          ;;READ AN OCTAL NUMBER FROM TTY
3986          104414 RDDEC= 104414          ;;READ A DECIMAL NUMBER FROM TTY
3987          ;
3988          104415 SAVREG= 104415          ;;SAVE R0-R5 ROUTINE
3989          104416 RESREG= 104416          ;;RESTORE R0-R5 ROUTINE
3990          ;
3991          104417 KERNEL= 104417          ;ENTER KERNEL MODE
3992          ;
3993          104420 ENERGIZE=104420          ;TURN ON MEMORY MANAGEMENT & TRAPS
3994          104421 DEENERGIZE=104421        ;TURN OFF MEMORY MANAGEMENT & TRAPS
3995          104422 KMAP= 104422          ;MAP KERNEL 1 TO 1
3996          ;
3997          104423 CACHON= 104423          ;TURN ON CACHE
3998          104424 CACHOFF=104424          ;TURN OFF CACHE

```

3999			
4000	104425	LOADCSR=104425	;LOAD CORRECT CSR
4001	104426	READCSR=104426	;READ CORRECT CSR
4002			
4003	104427	PERR01= 104427	;PROGRAM DETECTED ERROR
4004	104430	PERR02= 104430	;PROGRAM DETECTED ERROR
4005	104431	PERR03= 104431	;PROGRAM DETECTED ERROR
4006	104432	PERR04= 104432	;PROGRAM DETECTED ERROR
4007	104433	PERR07= 104433	;PROGRAM DETECTED ERROR
4008	104434	PERR10= 104434	;PROGRAM DETECTED ERROR
4009	104435	PERR11= 104435	;PROGRAM DETECTED ERROR
4010	104436	PERR12= 104436	;PROGRAM DETECTED ERROR
4011	104437	PERR13= 104437	;PROGRAM DETECTED ERROR
4012	104440	PERR14= 104440	;PROGRAM DETECTED ERROR
4013	104441	PERR15= 104441	;PROGRAM DETECTED ERROR
4014	104442	PERR16= 104442	;PROGRAM DETECTED ERROR
4015	104443	PERR17= 104443	;PROGRAM DETECTED ERROR
4016	104444	PERR20= 104444	;PROGRAM DETECTED ERROR
4017	104445	PERR21= 104445	;PROGRAM DETECTED ERROR
4018	104446	PERR22= 104446	;PROGRAM DETECTED ERROR
4019	104447	PERR23= 104447	;PROGRAM DETECTED ERROR
4020	104450	PERR24= 104450	;PROGRAM DETECTED ERROR
4021	104451	PERR25= 104451	;PROGRAM DETECTED ERROR
4022	104452	PERR26= 104452	;PROGRAM DETECTED ERROR
4023	104453	PERR27= 104453	;PROGRAM DETECTED ERROR
4024	104454	PERR30= 104454	;PROGRAM DETECTED ERROR
4025	104455	PERR31= 104455	;PROGRAM DETECTED ERROR
4026	104456	PERR32= 104456	;PROGRAM DETECTED ERROR
4027	104457	PERR33= 104457	;PROGRAM DETECTED ERROR
4028	104460	PERR34= 104460	;PROGRAM DETECTED ERROR
4029	104461	PERR35= 104461	;PROGRAM DETECTED ERROR
4030	104462	PERR36= 104462	;PROGRAM DETECTED ERROR
4031	104463	PERR37= 104463	;PROGRAM DETECTED ERROR
4032	104464	PERR40= 104464	;PROGRAM DETECTED ERROR
4033	104465	PERR41= 104465	;PROGRAM DETECTED ERROR
4034	104466	PERR42= 104466	;PROGRAM DETECTED ERROR
4035	104467	PERR43= 104467	;PROGRAM DETECTED ERROR
4036			
4037	104470	ECCDIS= 104470	;DISABLE ECC ON ALL CSR'S
4038	104471	ECC1DIS=104471	;DISABLE ECC ON 1 SELECTED CSR
4039	104472	ECCINIT=104472	;INITIALIZE ALL ECC CSR'S
4040	104473	ECC1INIT=104473	;INITIALIZE 1 SELECTED ECC CSR
4041	104474	CBCSR= 104474	;WRITE GENERATED CHECKBITS IN ALL CSR'S
4042	104475	CB1CSR= 104475	;WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
4043	104476	WASSBE= 104476	;WAS THERE A SBE ON ANY CSR?
4044	104477	WAS1SBE=104477	;WAS THERE A SBE ON 1 SELECTED CSR?
4045	104500	WASDBE= 104500	;WAS THERE A DBE ON ANY CSR?
4046	104501	WAS1DBE=104501	;WAS THERE A DBE ON 1 SELECTED CSR?
4047	104502	CLRCSR= 104502	;CLEAR ALL CSR'S
4048	104503	CLR1CSR=104503	;CLEAR 1 SELECTED CSR
4049	104504	CHKDIS= 104504	;DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
4050	104505	CHK1DIS=104505	;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
4051	104506	ENASBE= 104506	;ENABLE TRAPS ON SBE'S FROM ALL CSR'S
4052	104507	ENA1SBE=104507	;ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
4053	104510	TSTREAD=104510	;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4054	104511	INVALID=104511	;INVALIDATE BACKGROUND PATTERN ON "BANK"
4055	104512	ERRGEN =104512	;CHECK ERROR ADDRESS

CVMJAAO MSV11-J MEMORY DIAG.
DEFINE TRAPS

MACRO V05.01b Friday 15-Feb-85 16:10 Page 98-2

SEQ 0097

4056
4057

104513
104514

CBREG =104513
SYNREG =104514

;ENABLES CHECKBIT REGISTER
;ENABLES SYNDROME BIT REGISTER

```

4060          .SBTTL  DEFINE  BASIC PDP11 STUFF
4061
4062          ;*INITIAL ADDRESS OF THE STACK POINTER
4063          002000      STACK= 2000          ;;FIRST ADDRESS OF THE STACK
4064          002000      KERSTK= STACK        ;;KERNEL STACK
4065          000740      SUPSTK= 740          ;;SUPERVISOR STACK
4066          000700      USESTK= 700          ;;USER STACK
4067          104000      ERROR=EMT            ;;BASIC DEFINITION OF ERROR CALL
4068          000004      SCOPE=IOT            ;;BASIC DEFINITION OF SCOPE CALL
4069          177776      PSW= 177776          ;;PROCESSOR STATUS WORD
4070          ;STKLMT=177774                    ;;STACK LIMIT REGISTER
4071          ;PIRQ= 177772                    ;;PROGRAM INTERRUPT REQUEST REGISTER
4072          177570      DSWR= 177570          ;;HARDWARE SWITCH REGISTER
4073          177570      DDISP= 177570        ;;HARDWARE DISPLAY REGISTER
4074          177546      LKS= 177546          ;;LINE CLOCK (KW11-L) STATUS REGISTER
4075
4076          ;*MISCELLANEOUS DEFINITIONS
4077          000011      HT= 11                ;;CODE FOR HORIZONTAL TAB
4078          000012      LF= 12                ;;CODE LINE FEED
4079          000015      CR= 15                ;;CODE CARRIAGE RETURN
4080          000200      CRLF= 200              ;;CODE FOR CARRIAGE RETURN-LINE FEED
4081          000007      MFPT= 7                ;;CODE FOR PROCESSOR TYPE INSTRUCTION
4082
4083          ;*GENERAL PURPOSE REGISTER DEFINITIONS
4084          ;SP=R6                ;;STACK POINTER
4085          ;KSP=SP              ;;KERNEL STACK POINTER
4086          000006      SSP=SP                ;;SUPERVISOR STACK POINTER
4087          000006      USP=SP                ;;USER STACK POINTER
4088          ;PC=R7                ;;PROGRAM COUNTER
4089
4090          ;*"SWITCH REGISTER" SWITCH DEFINITIONS
4091          100000      SW15= 100000
4092          040000      SW14= 40000
4093          020000      SW13= 20000
4094          010000      SW12= 10000
4095          004000      SW11= 4000
4096          002000      SW10= 2000
4097          001000      SW9= 1000
4098          000400      SW8= 400
4099          000200      SW7= 200
4100          000100      SW6= 100
4101          000040      SW5= 40
4102          000020      SW4= 20
4103          000010      SW3= 10
4104          000004      SW2= 4
4105          000002      SW1= 2
4106          000001      SW0= 1
4107
4108          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
4109          100000      BIT15= 100000
4110          040000      BIT14= 40000
4111          020000      BIT13= 20000
4112          010000      BIT12= 10000
4113          004000      BIT11= 4000
4114          002000      BIT10= 2000
4115          001000      BIT9= 1000
4116          000400      BIT8= 400

```

```

4117      000200      BIT7= 200
4118      000100      BIT6= 100
4119      000040      BIT5= 40
4120      000020      BIT4= 20
4121      000010      BIT3= 10
4122      000004      BIT2= 4
4123      000002      BIT1= 2
4124      000001      BIT0= 1
4125
4126      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
4127      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
4128      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
4129      ;TBITVEC=14      ;; "T" BIT
4130      ;TRTVEC=      14      ;;TRACE TRAP
4131      ;BPTVEC=      14      ;;BREAKPOINT TRAP (BPT)
4132      000020      IOTVEC= 20     ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
4133      000024      PWRVEC= 24     ;;POWER FAIL
4134      000030      EMTVEC= 30     ;;EMULATOR TRAP (EMT) **ERROR**
4135      000034      TRAPVEC=34     ;; "TRAP" TRAP
4136      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
4137      ;TPVEC= 64      ;;TTY PRINTER VECTOR
4138      ;LKVEC= 100     ;;LINE CLOCK (KW11-L) VECTOR
4139      000114      CACHVEC=114    ;;CACHE ERROR INTERRUPT VECTOR
4140      000114      PARVEC=CACHVEC
4141      ;PIRQVEC=240     ;;PROGRAM INTERRUPT REQUEST VECTOR
4142      000250      MMVEC= 250     ;;MEMORY MANAGEMENT VECTOR
4143      ;SBTTL DEFINE CACHE REGISTERS
4144      ;MEMERR = 177744      ;;CACHE ERROR REGISTER
4145      177746      CONTRL = 177746 ;;MEMORY CONTROL REGISTER
4146      177750      MAINT = 177750  ;;MEMORY MAINTENANCE REGISTER
4147      ;HITMIS = 177752      ;;HIT MISS REGISTER "1" IMPLIES HIT IN CACHE
4148      177754      DATARG = 177754 ;;DATA REGISTER
4149
4150      ;SBTTL DEFINE CPU REGISTERS
4151      177766      CPUERR = 177766  ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
4152
4153      ;SBTTL DEFINE MEMORY MANAGEMENT REGISTERS
4154      ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
4155      177572      MMRO= 177572
4156      177574      MMR1= 177574
4157      177576      MMR2= 177576
4158      172516      MMR3= 172516
4159
4160      ;*USER "I" PAGE DESCRIPTOR REGISTERS
4161      177600      UIPDRO= 177600
4162      ;UIPDR1=      177602
4163      ;UIPDR2=      177604
4164      ;UIPDR3=      177606
4165      ;UIPDR4=      177610
4166      ;UIPDR5=      177612
4167      ;UIPDR6=      177614
4168      ;UIPDR7=      177616
4169
4170      ;*USER "D" PAGE DESCRIPTOR REGISTERS
4171      ;UDPDR0=      177620
4172      ;UDPDR1=      177622
4173      ;UDPDR2=      177624

```

```

4174      ;UDPDR3=      177626
4175      ;UDPDR4=      177630
4176      ;UDPDR5=      177632
4177      ;UDPDR6=      177634
4178      ;UDPDR7=      177636
4179
4180      ;*USER "I" PAGE ADDRESS REGISTERS
4181      FASTCITY=UIPAR0
4182      177640      UIPAR0= 177640      ;PATTERN PROGRAM SPACE
4183      177642      UIPAR1= 177642      ;PATTERN PROGRAM SPACE
4184      177644      UIPAR2= 177644      ;PATTERN PROGRAM SPACE
4185      177646      UIPAR3= 177646      ;PATTERN PROGRAM SPACE
4186      177650      UIPAR4= 177650      ;PATTERN PROGRAM SPACE
4187      177652      UIPAR5= 177652      ;PATTERN PROGRAM SPACE
4188      177654      UIPAR6= 177654      ;PATTERN PROGRAM SPACE
4189      ;UIPAR7=      177656      ;PATTERN PROGRAM SPACE
4190
4191      ;*USER "D" PAGE ADDRESS REGISTERS
4192      177660      UDPAR0= 177660      ;PATTERN PROGRAM SPACE
4193      ;UDPAR1=      177662      ;PATTERN PROGRAM SPACE
4194      ;UDPAR2=      177664      ;PATTERN PROGRAM SPACE
4195      ;UDPAR3=      177666      ;PATTERN PROGRAM SPACE
4196      ;UDPAR4=      177670      ;PATTERN PROGRAM SPACE
4197      ;UDPAR5=      177672      ;PATTERN PROGRAM SPACE
4198      ;UDPAR6=      177674      ;PATTERN PROGRAM SPACE
4199      177676      UDPAR7= 177676      ;PATTERN PROGRAM SPACE
4200
4201      ;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
4202      172200      SIPDR0= 172200
4203      ;SIPDR1=      172202
4204      ;SIPDR2=      172204
4205      ;SIPDR3=      172206
4206      ;SIPDR4=      172210
4207      ;SIPDR5=      172212
4208      ;SIPDR6=      172214
4209      ;SIPDR7=      172216
4210
4211      ;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
4212      ;SDPDR0=      172220
4213      ;SDPDR1=      172222
4214      ;SDPDR2=      172224
4215      ;SDPDR3=      172226
4216      ;SDPDR4=      172230
4217      ;SDPDR5=      172232
4218      ;SDPDR6=      172234
4219      ;SDPDR7=      172236
4220
4221      ;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
4222      172240      SIPAR0= 172240
4223      ;SIPAR1=      172242
4224      ;SIPAR2=      172244
4225      172246      SIPAR3= 172246      ;TEST AREA
4226      ;SIPAR4=      172250      ;TEST AREA
4227      172252      SIPAR5= 172252      ;TEST AREA
4228      172254      SIPAR6= 172254      ;TEST AREA
4229      ;SIPAR7=      172256
4230

```

```

4231                                     ;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
4232      172260      SDPAR0=      172260
4233      ;SDPAR1=      172262
4234      ;SDPAR2=      172264
4235      ;SDPAR3=      172266
4236      ;SDPAR4=      172270
4237      172272      SDPAR5=      172272
4238      172274      SDPAR6=      172274
4239      172276      SDPAR7=      172276
4240
4241                                     ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
4242      172300      KIPDR0=      172300
4243      ;KIPDR1=      172302
4244      ;KIPDR2=      172304
4245      ;KIPDR3=      172306
4246      ;KIPDR4=      172310
4247      ;KIPDR5=      172312
4248      ;KIPDR6=      172314
4249      ;KIPDR7=      172316
4250
4251                                     ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
4252      ;KDPDR0=      172320
4253      ;KDPDR1=      172322
4254      ;KDPDR2=      172324
4255      ;KDPDR3=      172326
4256      ;KDPDR4=      172330
4257      ;KDPDR5=      172332
4258      ;KDPDR6=      172334
4259      ;KDPDR7=      172336
4260
4261                                     ;*KERNEL "I" PAGE ADDRESS REGISTERS
4262      172340      KIPAR0=      172340
4263      ;KIPAR1=      172342
4264      ;KIPAR2=      172344
4265      ;KIPAR3=      172346
4266      172350      KIPAR4=      172350
4267      172352      KIPAR5=      172352
4268      172354      KIPAR6=      172354
4269      ;KIPAR7=      172356
4270
4271                                     ;*KERNEL "D" PAGE ADDRESS REGISTERS
4272      172360      KDPAR0=      172360
4273      ;KDPAR1=      172362
4274      ;KDPAR2=      172364
4275      ;KDPAR3=      172366
4276      ;KDPAR4=      172370
4277      ;KDPAR5=      172372
4278      172374      KDPAR6=      172374
4279      172376      KDPAR7=      172376
4280

```

```

4283          .SBTTL  DEFINE  Q-BUS MAP REGISTERS
4284          ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
4285          ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
4286          170200      MAPL0 = 170200
4287          170202      MAPH0 = 170202
4288          170204      MAPL1 = 170204
4289          ;MAPH1 = 170206
4290          ;MAPL2 = 170210
4291          ;MAPH2 = 170212
4292          ;MAPL3 = 170214
4293          ;MAPH3 = 170216
4294          ;MAPL4 = 170220
4295          ;MAPH4 = 170222
4296          ;MAPL5 = 170224
4297          ;MAPH5 = 170226
4298          ;MAPL6 = 170230
4299          ;MAPH6 = 170232
4300          ;MAPL7 = 170234
4301          ;MAPH7 = 170236
4302          ;MAPL10 = 170240
4303          ;MAPH10 = 170242
4304          ;MAPL11 = 170244
4305          ;MAPH11 = 170246
4306          ;MAPL12 = 170250
4307          ;MAPH12 = 170252
4308          ;MAPL13 = 170254
4309          ;MAPH13 = 170256
4310          ;MAPL14 = 170260
4311          ;MAPH14 = 170262
4312          ;MAPL15 = 170264
4313          ;MAPH15 = 170266
4314          ;MAPL16 = 170270
4315          ;MAPH16 = 170272
4316          ;MAPL17 = 170274
4317          ;MAPH17 = 170276
4318          ;MAPL20 = 170300
4319          ;MAPH20 = 170302
4320          ;MAPL21 = 170304
4321          ;MAPH21 = 170306
4322          ;MAPL22 = 170310
4323          ;MAPH22 = 170312
4324          ;MAPL23 = 170314
4325          ;MAPH23 = 170316
4326          ;MAPL24 = 170320
4327          ;MAPH24 = 170320
4328          ;MAPL25 = 170324
4329          ;MAPH25 = 170326
4330          ;MAPL26 = 170330
4331          ;MAPH26 = 170332
4332          ;MAPL27 = 170334
4333          ;MAPH27 = 170336
4334          ;MAPL30 = 170340
4335          ;MAPH30 = 170342
4336          ;MAPL31 = 170344
4337          ;MAPH31 = 170346
4338          ;MAPL32 = 170350
4339          ;MAPH32 = 170352

```


CVMJAAO MSV11-J MEMORY DIAG.
DEFINE Q-BUS MAP REGISTERS

MACRO V05.01b Friday 15 Feb-85 16:10 Page 102-1

SEQ 0103

4340		;MAPL33 = 170354
4341		;MAPH33 = 170356
4342		;MAPL34 = 170360
4343		;MAPH34 = 170362
4344		;MAPL35 = 170364
4345		;MAPH35 = 170366
4346		;MAPL36 = 170370
4347		;MAPH36 = 170372
4348		;MAPL37 = 170374
4349		;MAPH37 = 170376

4350			
4351			
4352		.SBTTL	DEFINE SOFTWARE SWITCH & DISPLAY REGISTERS
4353	000174	DISPREG=174	
4354	000176	SWREG= 176	
4355			

4356		.SBTTL	DEFINE CONTROL STATUS REGISTERS
4357	172100	CSRADD=172100	
4358			

4359		.SBTTL	DEFINE PARAMETERS
4360	060000	FIRST=60000	;START OF THE 16K TEST PATTERN AREA
4361	157776	LAST=157776	;END OF THE 16K TEST PATTERN AREA
4362	040000	SIZE=40000	;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)

4365
4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401

```

      .LIST MD                               ;BE NICE TO SEE MY DEFINITIONS
      .SBTTL MACRO FATAL
;***** FATAL *****
;FATAL IS USED TO REPORT FATAL ERRORS (ERRORS THAT PREVENT
;THE PROGRAM FROM CONTINUING).
;*****
      .MACRO FATAL ARG                       ;***MACRO***MACRO***MACRO***
      .NLIST
      .DSABL CRF
      .IIF DF LST$$ .LIST ME
      .ENABL CRF
      .LIST
      INC FATAL$                             ;SET FATAL INDICATOR
      ERROR +ARG
      .DSABL CRF
      .IIF DF LST$$ .NLIST ME
      .ENABL CRF
      .ENDM FATAL

      .SBTTL MACRO TYPE
      .MACRO TYPE ARG
      .NLIST
      .DSABL CRF
      .IIF DF LST$$ .LIST ME
      .ENABL CRF
      .LIST
      .IF B ARG
      TYPEIT
      .IFF
      TYPEIT ,ARG
      .ENDC
      .DSABL CRF
      .IIF DF LST$$ .NLIST ME
      .ENABL CRF
      .ENDM TYPE

```

```

4404 .SBTTL MACRO NEWTST
4405 ;***** NEWTST *****
4406 ;NEWTST IS USED AS THE FIRST INSTRUCTION OF A TEST.
4407 ;IT WILL:
4408 ;1) GENERATE A TEST NUMBER FOR THE LABEL OF THIS TEST
4409 ;2) PUT STARS BEFORE AND AFTER A MESSAGE
4410 ;ARGUMENTS
4411 ;1) ASCII -- THIS IS THE MESSAGE THAT WILL APPEAR
4412 ;           ON THE LISTING
4413 ;2) ICOUNT -- IF NON-BLANK AND BIT 11 OF $SWR = 1 IT WILL BE
4414 ;             THE NUMBER OF ITERATIONS TO MAKE ON THIS TEST
4415 ;3) RETURN -- IF NON-BLANK WILL BE THE ADDRESS TO
4416 ;             WHICH THE NEXT SCOPE STATEMENT WILL
4417 ;             LOOP BACK TO.
4418 ;4) COMAND -- IF NON-BLANK WILL BE THE FIRST
4419 ;             INSTRUCTION OF THE TEST
4420 ;             IF BLANK SCOPE WILL BE THE
4421 ;             FIRST INSTRUCTION
4422 ;*****
4423 .MACRO NEWTST ASCII,ICOUNT,RETURN,COMAND
4424 $STN=1
4425 $NWTST=0
4426 .NLIST MC
4427 .IF B <COMAND>
4428 $NEWTEST \ $TN,<ASCII>,SCOPE
4429 .IFF
4430 $NEWTEST \ $TN,<ASCII>,<COMAND>
4431 .ENDC
4432 .NLIST
4433 .LIST ME
4434 .LIST
4435 .IF NE 4000&$SWR
4436 .IF NB ICOUNT
4437 .IF LE <ICOUNT-1>
4438 MOV #1,$TIMES ;;DO 1 ITERATION
4439 .IFF
4440 MOV #ICOUNT,$TIMES ;;DO ICOUNT ITERATIONS
4441 .ENDC
4442 .ENDC
4443 .IF NB RETURN
4444 MOV #RETURN,$LPADR ;;SET SCOPE LOOP ADDRESS
4445 .ENDC
4446 .ENDC
4447 .NLIST
4448 .LIST MC
4449 .LIST
4450 .NLIST ME
4451 .ENDM NEWTST

```

```

4454 .SBTTL MACRO $$NEWTEST
4455 .MACRO $$NEWTEST A,ASC,COMND
4456 .IRP ASCI,<ASC>
4457 .IF EQ $NWTST
4458 $NWTST=1
4459 .SBTTL T'A' ASCI
4460 .NLIST
4461 .LIST ME
4462 .LIST
4463 ;*****
4464 ;*TEST A ASCI
4465 .IFF
4466 ASCI
4467 .ENDC
4468 .ENDM
4469 ;*****
4470 TST'A: COMND
4471 .NLIST ME
4472 $TN=$TN+1
4473 .ENDM $$NEWTEST
4474
4475 .SBTTL MACRO SUBTST
4476 ;***** SUBTST *****
4477 ;
4478 ;THIS MACRO WILL FORMAT A SUBTEST HEADING WITH STARS
4479 ;A .SBTTL WILL BE FORCED & .NLISTED FOR THE TABLE OF CONTENTS.
4480 ;
4481 ;ARGUMENT:
4482 ;1) TXT -- THIS IS THE MESSAGE THAT WILL APPEAR IN THE TABLE OF CONTENTS & LISTING.
4483 ;
4484 ;EXAMPLE: SUBTST <<THIS IS A FUN SUBTST>>
4485 ;
4486 ;*****
4487
4488 .MACRO SUBTST ASCII
4489 .NLIST MC
4490 $SUBTST <ASCII>
4491 .LIST MC
4492 .ENDM SUBTST
4493
4494 .SBTTL MACRO $SUBTST
4495 .MACRO $SUBTST ASC
4496 .IRP ASCI,<ASC>
4497 .SBTTL ASCI
4498 .NLIST
4499 .LIST ME
4500 .LIST
4501 ;*****
4502 ;*SUBTEST ASCI
4503 .ENDM
4504 ;*****
4505 .NLIST ME
4506 .ENDM $SUBTST

```

```

4509 .SBTTL MACRO TYP OCT
4510 ;***** TYP OCT *****
4511 ;
4512 ;TYP OCT IS USED TO CHANGE A BINARY NUMBER
4513 ; TO A 6 DIGIT OCTAL NUMBER AND TYPE IT
4514 ;
4515 ;ARGUMENTS:
4516 ;
4517 ;1) NUM THE NUMBER TO BE TYPED
4518 ;
4519 ;2) REMARK ALLOWS A COMMENT TO BE MADE
4520 ;
4521 ;ROUTINES REQUIRED
4522 ;
4523 ;1) CONVERT BINARY TO OCTAL AND TYPE (.$TYP OCT)
4524 ;
4525 ;2) TYPE AN ASCII STRING (.$TYPE)
4526 ;
4527 ;EXAMPLES:
4528 ;
4529 ;1) TYP OCT HILMT,<TYPES THE CONTENTS OF HILMT>
4530 ;2) TYP OCT #5,<TYPES ' 000005'>
4531 ;
4532 ;*****
4533
4534 .MACRO TYP OCT NUM,REMARK
4535 .NLIST
4536 .DSABL CRF
4537 .IIF DF LST$$ .LIST ME
4538 .ENABL CRF
4539 .LIST
4540 MOV NUM,-(SP) ;;SAVE NUM FOR TYPEOUT
4541 .IIF NB <REMARK>, ;;REMARK
4542 TYP OCT ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
4543 .DSABL CRF
4544 .IIF DF LST$$ .NLIST ME
4545 .ENABL CRF
4546 .ENDM TYP OCT

```

```

4549 .SBTTL MACRO TYPOCS
4550 ;***** TYPOCS *****
4551 ;
4552 ;TYPOCS IS USED TO CHANGE A BINARY NUMBER TO AN OCTAL
4553 ;NUMBER AND TYPE 1 TO 6 DIGITS
4554 ;WITH OR WITHOUT LEADING ZEROS.
4555 ;
4556 ;ARGUMENTS:
4557 ;
4558 ;1) NUM NUMBER TO BE TYPED
4559 ;
4560 ;2) REMARK ALLOWS A COMMENT TO BE MADE
4561 ;
4562 ;3) N NUMBER OF DIGITS (1 TO 6) TO BE TYPED
4563 ;
4564 ;4) Z BLANK=SUPPRESS LEADING ZEROS (TYPES SPACES)
4565 ;NON-BLANK-TYPE LEADING ZEROS
4566 ;
4567 ;ROUTINES REQUIRED
4568 ;
4569 ;1) CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
4570 ;
4571 ;2) TYPE AN ASCIZ STRING (.$TYPE)
4572 ;
4573 ;EXAMPLES:
4574 ;1) TYPOCS #12345,<TYPES "5">,1
4575 ;2) TYPOCS #004,<TYPES "04">,2,X
4576 ;3) TYPOCS #004,<TYPES " 4">,2
4577 ;
4578 ;*****
4579
4580 .MACRO TYPOCS NUM,REMARK,N,Z
4581 .NLIST
4582 .DSABL CRF
4583 .IIF DF LST$$ .LIST ME
4584 .ENABL CRF
4585 .LIST
4586 MOV NUM, -(SP) ;;SAVE NUM FOR TYPEOUT
4587 .IIF NB <REMARK>, ;;REMARK
4588 TYPOS ;;GO TYPE--OCTAL ASCII
4589 .IF NB N
4590 .BYTE N ;;TYPE N DIGIT(S)
4591 .IFF
4592 .BYTE 6 ;;TYPE 6 DIGITS
4593 .ENDC
4594 .IF NB Z
4595 .BYTE 1 ;;TYPE LEADING ZEROS
4596 .IFF
4597 .BYTE 0 ;;SUPPRESS LEADING ZEROS
4598 .ENDC
4599 .DSABL CRF
4600 .IIF DF LST$$ .NLIST ME
4601 .ENABL CRF
4602 .ENDM TYPOCS

```

```

4605 .SBTTL MACRO TYPDEC
4606 ;***** TYPDEC *****
4607 ;
4608 ;TYPDEC IS USE TO CHANGE A BINARY NUMBER TO A SIGNED
4609 ; DECIMAL NUMBER AND TYPE IT REPLACING LEADING ZERO
4610 ; WITH SPACES.
4611 ;NOTE: IF THE NUMBER IS NEGATIVE A
4612 ; MINUS SIGN WILL BE TYPED.
4613 ;
4614 ;ARGUMENTS:
4615 ;
4616 ;1) NUM NUMBER TO BE TYPED
4617 ;
4618 ;2) REMARK ALLOWS A COMMENT TO BE MADE
4619 ;
4620 ;ROUTINES REQUIRED
4621 ;
4622 ;1) CONVERT BINARY TO DECIMAL AND TYPE (.$TYPDEC)
4623 ;
4624 ;2) TYPE AN ASCII STRING (.$TYPE)
4625 ;
4626 ;EXAMPLES
4627 ;
4628 ;1) TYPDEC SIZE,<TYPE THE CONTENTS OF SIZE>
4629 ;2) TYPDEC #-10.,<TYPE A MINUS TEN>
4630 ;
4631 ;*****
4632
4633 .MACRO TYPDEC NUM,REMARK
4634 .NLIST
4635 .DSABL CRF
4636 .IIF DF LST$$ .LIST ME
4637 .ENABL CRF
4638 .LIST
4639 MOV NUM,-(SP) ;;SAVE NUM FOR TYPFOUT
4640 .IIF NB <REMARK>, ;;REMARK
4641 TYPDS ;;GO TYPE -DECIMAL ASCII WITH SIGN
4642 .DSABL CRF
4643 .IIF DF LST$$ .NLIST ME
4644 .ENABL CRF
4645 .ENDM TYPDEC

```

```

4647 .SBTTL MACRO BMOV
4648 ;***** BMOV *****
4649 ;
4650 ; THIS MACRO MOVES A BLOCK OF DATA.
4651 ;
4652 ; ARGUMENTS:
4653 ;
4654 ; 1) FROMHERE THE FIRST ADDRESS OF THE SOURCE BLOCK.
4655 ;
4656 ; 2) TOHERE THE FIRST ADDRESS OF THE DESTINATION BLOCK.
4657 ; IF BLANK THE 1ST ADDRESS OF THE USER INSTRUCTION
4658 ; PAR'S IS USED (FASTCITY).
4659 ;
4660 ; 3) SIZE THE SIZE OF THE SOURCE BLOCK.
4661 ; IF BLANK A 16 WORD TRANSFER IS ASSUMED.
4662 ; "WHY DEFAULT TO 16 WORDS?" YOU ASK!
4663 ; "BECAUSE THAT'S HOW MANY WORDS TO THE USER PAR
4664 ; REGISTERS & THAT'S WHERE I INTEND TO MOVE LOTS
4665 ; OF STUFF." I REPLY!
4666 ;
4667 ;*****
4668 .MACRO BMOV FROMHERE,TOHERE,SIZE
4669 .IF B TOHERE
4670 .NLIST
4671 .DSABL CRF
4672 .IIF DF LST$$ .LIST ME
4673 .ENABL CRF
4674 .LIST
4675 JSR R5,BLOCK1
4676 FROMHERE
4677 .DSABL CRF
4678 .IIF DF LST$$ .NLIST ME
4679 .ENABL CRF
4680 .MEXIT
4681 .ENDC
4682 .IF B SIZE
4683 .NLIST
4684 .DSABL CRF
4685 .IIF DF LST$$ .LIST ME
4686 .ENABL CRF
4687 .LIST
4688 JSR R5,BLOCK2
4689 TOHERE
4690 FROMHERE
4691 .DSABL CRF
4692 .IIF DF LST$$ .NLIST ME
4693 .ENABL CRF
4694 .MEXIT
4695 .IFF
4696 .NLIST
4697 .DSABL CRF
4698 .IIF DF LST$$ .LIST ME
4699 .ENABL CRF
4700 .LIST
4701 JSR R5,BLOCK3
4702 SIZE
4703

```


CVMJAAO MSV11-J MEMORY DIAG.
MACRO BMOV

MACRO V05.01b Friday 15 Feb-85 16:10 Page 115 1

SEQ 0111

4704
4705
4706
4707
4708
4709
4710

TOHERE
FROMHERE
.DSABL CRF
.IIF DF LST\$\$.NLIST ME
.ENABL CRF
.ENDC
.ENDM BMOV

```

4713      .SBTTL  MACRO  MAP
4714      ;***** MAP *****
4715      ;
4716      ; THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
4717      ; TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-157777)).
4718      ;
4719      ; ARGUMENTS:
4720      ;
4721      ; 1)  BANK  THE BANK OF 16K WORDS TO BE MAPPED.
4722      ;        THERE ARE 120 BANKS OF 16K WORDS
4723      ;
4724      ; EXAMPLES
4725      ;
4726      ;      MAP      LOC          ;LOCATION "LOC" CONTAINS THE # OF THE BANK TO MAP
4727      ;
4728      ;      MAP      #28.         ;BANK 34 (OCTAL) WILL BE MAPPED
4729      ;
4730      ;*****
4731
4732      .MACRO  MAP      BANK
4733      PUSH   R3
4734      .NLIST
4735      .DSABL CRF
4736      .IIF DF LST$$ .LIST  ME
4737      .ENABL CRF
4738      .LIST
4739      .IF B BANK
4740      MOV    #200,R3
4741      .IFF
4742      MOV    BANK,R3
4743      .ENDC
4744      CALL   MAPPER
4745      .DSABL CRF
4746      .IIF DF LST$$ .NLIST  ME
4747      .ENABL CRF
4748      POP    R3
4749      .ENDM  MAP

```

J9

```

4752      .SBTTL  MACRO  SUPERVISOR
4753      ;***** SUPERVISOR *****
4754      ;
4755      ; THIS MACRO SWITCHES TO SUPERVISOR MODE.
4756      ;
4757      ; ARGUEMENTS: NONE.
4758      ;
4759      ;*****
4760
4761      .MACRO  SUPERVISOR
4762      .NLIST
4763      .DSABL  CRF
4764      .IIF DF LST$$ .LIST  ME
4765      .ENABL  CRF
4766      .LIST
4767      BIS     #BIT14,PSW                ;GO TO SUPERVISOR MODE
4768      .DSABL  CRF
4769      .IIF DF LST$$ .NLIST  ME
4770      .ENABL  CRF
4771      .ENDM  SUPERVISOR
4772
4773      .SBTTL  MACRO  USER
4774      ;***** USER *****
4775      ;
4776      ; THIS MACRO SWITCHES TO USER MODE.
4777      ;
4778      ; ARGUEMENTS: NONE.
4779      ;
4780      ;*****
4781
4782      .MACRO  USER
4783      .NLIST
4784      .DSABL  CRF
4785      .IIF DF LST$$ .LIST  ME
4786      .ENABL  CRF
4787      .LIST
4788      BIS     #BIT15!BIT14,PSW          ;GO TO USER MODE
4789      .DSABL  CRF
4790      .IIF DF LST$$ .NLIST  ME
4791      .ENABL  CRF
4792      .ENDM  USER
4793

```

4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814

```
.SBTTL MACRO TESTAREA
;***** TESTAREA *****
;
; THIS MACRO SWITCHES TO THE SPECIFIED TEST MODE.
;
; ARGUMENTS: NONE.
;
;*****
```

```
.MACRO TESTAREA
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TESTAREA
```

L9

```

4817      .SBTTL  MACRO  SET4 & RES4
4818      ;***** SET4 & RES4 *****
4819      ;
4820      ; THESE MACROS SET & RESTORE VECTOR 4(TIMEOUT TRAP)
4821      ;
4822      ; IN IT'S RESTORED MODE TRAPS ARE REPORTED AS SUCH.
4823      ;
4824      ; ARGUMENTS:  LOC      ;THE LOCATION TO VECTOR TO (ONLY USED IN "SET4" NOT "RES4")
4825      ;
4826      ; I USE THE SET4 AND RES4 MACROS AROUND CODE THAT I EXPECT TO TRAP TO 4
4827      ; LIKE LOOKING FOR ALL POSSIBLE CSR'S AND ETC.  WHENEVER CODE IS NOT
4828      ; SURROUNDED BY SET4 AND RES4 THEN ANY TRAPS TO 4 WILL CAUSE AN ERROR
4829      ; PRINTOUT THAT SAYS "UNEXPFCTED TRAP TO 4" AND ALL THE ASSOCIATED REGISTER JUNK
4830      ;*****
4831
4832      .MACRO  SET4      ARG
4833      .NLIST
4834      .DSABL  CRF
4835      .IIF DF LST$$ .LIST  ME
4836      .ENABL  CRF
4837      .LIST
4838      MOV     ARG,4
4839      .DSABL  CRF
4840      .IIF DF LST$$ .NLIST  ME
4841      .ENABL  CRF
4842      .ENDM   SET4
4843
4844      .MACRO  RES4
4845      .NLIST
4846      .DSABL  CRF
4847      .IIF DF LST$$ .LIST  ME
4848      .ENABL  CRF
4849      .LIST
4850      MOV     #TIMEOUT,4
4851      CMP     #5,PROTYP      ;IS THIS AN 11/83/84 ?
4852      BNE     101$           ;BRANCH IF NOT
4853      CLR     CPUERR         ;CLEAR OUT THE CPU ERROR REGISTER BITS
4854      101$:
4855
4856      .DSABL  CRF
4857      .IIF DF LST$$ .NLIST  ME
4858      .ENABL  CRF
4859      .ENDM   RES4

```

;IS THIS AN 11/83/84 ?
;BRANCH IF NOT
;CLEAR OUT THE CPU ERROR REGISTER BITS
;THAT A EXPECTED TRAP COULD HAVE SET

```

4862          .SBTTL MACRO DLEFT
4863          ;***** DLEFT *****
4864          ;
4865          ; THIS MACRO DOES A DOUBLE WORD LEFT SHIFT
4866          ;
4867          ; ARGUMENTS:  LOC      ;THE LOCATION TO BE SHIFTED LEFT (CARRY TO LOC+2)
4868          ;
4869          ;*****
4870
4871          .MACRO DLEFT ARG
4872          .NLIST
4873          .DSABL CRF
4874          .IIF DF LST$$ .LIST ME
4875          .ENABL CRF
4876          .LIST
4877          ROL ARG
4878          ROL ARG+2
4879          .DSABL CRF
4880          .IIF DF LST$$ .NLIST ME
4881          .ENABL CRF
4882          .ENDM DLEFT
4883          .NLIST MD ;DON'T NEED TO SEE THEM ANY MORE
4884          ;*****
4885          .SBTTL ABORT ROUTINE FOR ORION UFD MODE (VMK)
4886          ;*****
4887
4888 000000 105737 056732 ABORT: TSTB $DDW7 ;TEST FOR USER FRIENDLY MODE
4889 000004 001420 BEQ ABORT2 ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
4890 000006 013737 003706 000030 MOV SAV30,30 ;RESTORE EMT LOCATION (30)
4891 000014 013737 003710 000032 MOV SAV32,32 ;RESTORE EMT PRIORITY LOCATION (32)
4892 000022 104042 EMT +42 ;GET DCA LOCATION INTO R0 FROM MONITOR
4893 000024 012760 177777 000042 MOV #-1,42(R0) ;SET A -1 INTO LOCATION DRSEERR IN MONITOR
4894 000032 013700 000042 ABORT1: MOV 42,R0 ;AND PUT THE MONITOR RETURN ADDRESS IN R0
4895 000036 005037 000042 CLR 42 ;CLEAR MONITOR RETURN FLAG
4896 000042 000137 014042 JMP $ENDAD ;RETURN TO MONITOR DO NOT PUSH STACK HERE
4897 000046 000207 ABORT2: RTS PC ;IF NOTUFD RETURN TO MAINLINE
4898
4899

```

```

4902          .SBTTL  TRAP CATCHER
4903          .=0
4904 000000 000000 000000          .WORD  0,0
4905          000177          .REPT  177          ;.WORD  .+2,HALT
4909
4910          .SBTTL  ACT11 HOOKS
4911          ;*THE HOOKS REQUIRED BY ACT11 ARE DEFINED AND SETUP BELOW:
4912          ;*   DEFINITIONS:
4913          ;*       1)LOC.46          "END OF-PASS" HOOK
4914          ;*                       =ADDRESS OF END OF PASS ROUTINE
4915          ;*                       MODIFIED BY ACT11.
4916          ;*       2)LOC.52          PROGRAM NEEDS HOOK
4917          ;*                       BIT 15=1 PROGRAM SHOULD BE POWER
4918          ;*                       FAILED WHILE RUNNING
4919          ;*                       =0 NO POWER FAIL
4920          ;*                       BIT 14=1 PROGRAM MEMORY SIZE DEPENDENT
4921          ;*                       =0 NOT MEMORY SIZE DEPENDENT
4922          ;*                       BIT 13=1 PROGRAM REQUIRES MANUAL INTERVENTION
4923          ;*                       =0 MANUAL INTERVENTION NOT REQUIRED
4924          ;*                       BITS 12-0 MUST BE ZERO'S
4925          ;*
4925          .=46
4926 000046 014042          $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN . $EOP
4927          .=52
4928 000052 000020          .WORD  BIT4          ;;2)SET LOC.52 TO INDICATE MEMORY SIZE DEPENDANT
4929          .SBTTL  APT11 HOOKS
4930          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
4931 000024 000200          200          ;;FOR APT START UP
4932          .=42
4933 000042 002000          STACK          ;SO RT11 CAN START WITH RUN COMMAND
4934          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
4935 000044 056750          $APTHDR ;;POINT TO APT HEADER BLOCK
4936          .=200
4937 000200 000437          START3: BR          START1          ;"NORMAL" START
4938 000202 000442          BR          START2          ;RESTART (SAVE ERROR ACCOUNTING)
4939          .=300
4940 000300 005037 002626          START1: CLR          RESTART
4941 000304 000137 003670          JMP          START
4942 000310          START2: SET          RESTART
4943          000310 012737 177777 002626          JMP          START          MOV  #-1,RESTART
4944          000316 000137 003670          .=STACK

```

			.SBTTL VARIABLES	INITIALIZED TO ZERO
4947			;	*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
4948			;	*USED IN THE PROGRAM.
4949			\$CMTAG:	::START OF COMMON TAGS
4950 002000			UFDSET:0	::USER FRIENDLY FLAG
4951 002000	000000		SELONLY:0	::SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
4952 002002	000000		DIAGFLAG:0	::SET FOR SHIFTING DIAGONAL TEST
4953 002004	000000		KAMIKAZE:0	::SET FOR KAMIKAZE MODE TESTING
4954 002006	000000		SKIPKAMI:0	::USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
4955 002010	000000		;	NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER
4956			\$PATMAR:.BYTE 0	::PATTERN NUMBER
4957 002012	000		\$BANK:.BYTE 0	::BANK & SIGN
4958 002013	000		\$ERFLG:.BYTE 0	::CONTAINS ERROR FLAG
4959 002014	000		\$ITEMB:.BYTE 0	::CONTAINS ITEM CONTROL BYTE
4960 002015	000		LASTERROR:.WORD 0	::NUMBER OF ERRORS ON LAST PASS
4961 002016	000000		ERRPC:.WORD 0	::CONTAINS PC OF ERROR FOR TYPEOUT
4962 002020	000000		BADPC:.WORD 0	::CONTAINS PC OF ERROR
4963 002022	000000		ERRSP:.WORD 0	::CONTAINS SP OF ERROR FOR TYPEOUT
4964 002024	000000		BADSP:.WORD 0	::CONTAINS SP OF ERROR
4965 002026	000000		ERRPSW:.WORD 0	::CONTAINS PSW OF ERROR FOR TYPEOUT
4966 002030	000000		BADPSW:.WORD 0	::CONTAINS PSW OF ERROR
4967 002032	000000		ADDRESS:.WORD 0	::CONTAINS ADDRESS OF 'BAD' DATA
4968 002034	000000		PADDRESS:.WORD 0	::ADDRESS OF PARITY ERROR
4969 002036	000000		PHYADD:.WORD 0,0	::22 BIT PHYSICAL ADDRESS
4970 002040	000000	000000	GOOD:.WORD 0	::CONTAINS 'GOOD' DATA
4971 002044	000000		GOOD2:.WORD 0	::CONTAINS 'GOOD2' DATA
4972 002046	000000		GOOD3:.WORD 0	::CONTAINS 'GOOD3' DATA
4973 002050	000000		BAD:.WORD 0	::CONTAINS 'BAD' DATA
4974 002052	000000		BAD2:.WORD 0	::CONTAINS 'BAD2' DATA
4975 002054	000000		BAD3:.WORD 0	::CONTAINS 'BAD3' DATA
4976 002056	000000		BADXOR:.WORD 0	::XOR OF GOOD & BAD = BAD BITS!
4977 002060	000000		\$AUTO:.WORD 0	::AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
4978 002062	000000		FATAL\$:.WORD 0	::FATAL ERROR INDICATOR
4979 002064	000000		SKPERR:.WORD 0	::USED TO SKIP ERROR MESSAGE IN "\$ERRGEN"
4980 002066	000000		NEMCNT:0	::NON-EXISTANT MEMORY COUNTER (HOLES)
4981 002070	000000		PARCNT:0	::PARITY ERROR COUNTER
4982 002072	000000		PATERR:0	::PATTERN ERROR COUNTER
4983 002074	000000		NOPAR:0	::NO PARITY ERROR MODE INDICATOR
4984 002076	000000		NONEM:0	::NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
4985 002100	000000		BANK:0	::MEMORY BANK UNDER TEST
4986 002102	000000		BANKINDEX:0	::USED TO INDEX INTO CONFIG TABLE
4987 002104	000000		CPUBIT:0	::CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
4988 002106	000000		MUT:0	::MEMORY UNDER TEST FLAG
4989 002110	000000		PATTERN:0	::PATTERN NUMBER UNDER TEST
4990 002112	000000		KPFLAG:.WORD 0	::BANK IS PROTECTED REGION OF ECC
4991 002114	000000		ACFLAG:.WORD 0	::BANK CAN BE ACCESSED BY THIS CPU
4992 002116	000000		MKFLAG:.WORD 0	::IF SET INDICATES MSV11-J OR MF11S-K UNDER TEST***** ?
4993 002120	000000		PFLAG:.WORD 0	::BANK IS IN PROGRAM SPACE
4994 002122	000000		RRFLAG:.WORD 0	::BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
4995 002124	000000		RLFLAG:.WORD 0	::PROGRAM IS RELOCATED FLAG
4996 002126	000000		BMFLAG:.WORD 0	::"BANK IS IDENTIFIED AS BAD MEMORY" FLAG
4997 002130	000000		EQFLAG:.WORD 0	::"BANK HAS EQB MEMORY" FLAG
4998 002132	000000		TMFLAG:.WORD 0	::"TYPE OF MEMORY TO TEST" FLAG; 0 = PARITY, 1 = ECC
4999 002134	000000		INTFLAG:.WORD 0	::"BANK IS INTERLEAVED" FLAG***** ? **
5000 002136	000000		INT64K:.WORD 0	::"BANK IS 64K INTERLEAVED" FLAG***** ? **
5001 002140	000000		PMEMFLG:.WORD 0	::"MEMORY UNDER TEST IS A MSV11-J" FLAG
5002 002142	000000		ABORTFLAG:.WORD 0	::"ABORT OCCURED" FLAG
5003 002144	000000			

5004	002146	000000	CTLKVEC: .WORD	0	;HOLDS OLD KERNAL STACK POINTER IN CASE OF CNTL/K
5005	002150	000000	CSR: .WORD	0	;DATA TO OR FROM CSR
5006	002152	000000	CSRNO: .WORD	0	;CSR ADDRESS NUMBER (4 LSB'S)
5007	002154	000000	SAVCSR: .WORD	0	;LOCATION TO SAVE CSRNO DURING FS COMMAND
5008	002156	000000	OLDCSR: .WORD	0	;OLD CSR NUMBER(USED IN INH PTR TEST)
5009					;THESE LOCATIONS STORE GPR'S DURING SUPERVISOR TESTS
5010	002160	000000	SUPDR0: .WORD	0	
5011	002162	000000	SUPDR1: .WORD	0	
5012	002164	000000	SUPDR2: .WORD	0	
5013	002166	000000	SUPDR3: .WORD	0	
5014	002170	000000	SUPDR4: .WORD	0	
5015	002172	000000	SUPDR5: .WORD	0	
5016	002174	000000	SUPDR6: .WORD	0	
5017	002176	000000	DUMMY: .WORD	0	;DUMMY LOCATION FOR ADDRESS PASSING
5018					;THESE LOCATIONS STORE GPR'S & PSW DURING DETAILED ERROR PRINTOUTS
5019	002200	000000	DETR0: .WORD	0	
5020	002202	000000	DETR1: .WORD	0	
5021	002204	000000	DETR2: .WORD	0	
5022	002206	000000	DETR3: .WORD	0	
5023	002210	000000	DETR4: .WORD	0	
5024	002212	000000	DETR5: .WORD	0	
5025	002214	000000	DETSP: .WORD	0	
5026	002216	000000	DETPSW: .WORD	0	
5027	002220	000000	DETFLAG: .WORD	0	;DETAILED REPORT FLAG
5028	002222	000000	CONTFLAG: .WORD	0	;CSR'S HAVE BEEN TESTED FLAG
5029	002224	000000	TOTCSRS: .WORD	0	;1 BIT PER EXISTING CSR, EG-
5030					;CSR 0 REPRESENTED BY BIT 15, ETC.
5031	002226	000000	CSRFIRST: .WORD	0	;FIRST ADDRESS UNDER CONTROL OF THIS CSR
5032	002230	000000	CSRLAST: .WORD	0	
5033	002232	000000	CSRFBANK: .WORD	0	
5034	002234	000000	CSRLBANK: .WORD	0	
5035	002236	000000	CSRINT: .WORD	0	
5036	002240	000000	SPLTCSR: .WORD	0	
5037	002242	000000	DATBUF: .WORD	0,0	;TWO WORD DATA BUFFER
5038	002246	000000	TSTDAT: .WORD	0,0	;TWO WORD TEST DATA
5039	002252	000000	SBEMSK: .WORD	0,0	;TWO WORD SINGLE BIT ERROR MASK
5040	002256	000000	DBEMSK: .WORD	0,0	;TWO WORD DOUBLE BIT ERROR MASK
5041	002262	000000	SUPDOADD: .WORD	0	;ADDRESS OF SUBROUTINE TO EXECUTE IN SUPERVISOR MODE
5042	002264	000	PASFLG: .BYTE	0	;LOCAL LOOP PASS CONTROL
5043	002265	000	UPPFLG: .BYTE	0	;LOCAL LOOP PASS CONTROL
5044	002266	000000	PASSNO: .WORD	0	;LOCAL LOOP PASS CONTROL
5045	002270	000000	SAV4: .WORD	0	;USED TO SAVE KERNAL PAR 5
5046	002272	000000	SAVPAR: .WORD	0	;USED TO SAVE KERNAL PAR 5
5047	002274	000000	SAVMON: .WORD	0	;XXDP MONITOR RETURN ADDRESS
5048	002276	000000	MONFLG: .WORD	0	;RETURN TO MONITOR FLAG
5049	002300	000000	REALPAT: .WORD	0	;REAL PATTERN UNDER TEST
5050	002302	000000	OLDCACHE: .WORD	0	;BACKED UP VALUE OF CACHE CONTROL REGISTER
5051	002304	000000	PARTHERE: .WORD	0	;PARITY TRAPS SOMETIMES GO TO ADDRESS STORED HERE
5052	002306	000000	FSSTACK: .WORD	0	;STACK SAVED HERE IF IN FIELD SERVICE MODE
5053	002310	000000	NEWBANK: .WORD	0	;USED FOR RELOCATION TO A NEW BANK
5054	002312	000000	SOURCE: .WORD	0	;SOURCE OF DATA WORDS FOR CHECKBIT GENERATION SUBROUTINE
5055	002314	000000	CHECK: .WORD	0	;CHECKBITS TO BE LOADED INTO CSR
5056	002316	000000	CBITS: .WORD	0	;CHECK BITS TO BE WRITTEN
5057	002320	000000	MASK: .WORD	0	;BIT MASK FOR CSR
5058	002322	000000	CSR1S: .WORD	0	;CSR ALL 1'S PATTERN
5059	002324	000000	BITNO: .WORD	0	;BIT POINTER
5060	002326	000000	PCBUMP: .WORD	0	;VALUE TO BUMP THE PC BY TO RECOVER AFTER A PARITY TRAP

;;I.L.C.:REV

5061	002330	000000		CSRINC: .WORD	0	;VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR
5062	002332	000000		CSRLOOP: .WORD	0	;LOOP CONTROL FOR CSR TESTING
5063	002334	000000		SUCCESS: .WORD	0	;FLAG SET BY SUCCESSFULL TASK OR SUBROUTINE
5064	002336	000000		ZEROS: .WORD	0	;FOR AID IN "MOV" INSTRUCTIONS
5065	002340	000000		TIME: .WORD	0	;SECONDS THAT BATTERIES SHOULD LAST
5066	002342	000000		SKIPMK: .WORD	0	;FLAG TO SKIP MKCONTROL SUBROUTINE
5067	002344	000000		NULLFLAG: .WORD	0	;SET WHEN RUNNING NULL PATTERNS
5068	002346	000000		QVFLAG: 0		;FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
5069	002350	000000		ACTFLAG: 0		;FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
5070	002352	000000		APTFLAG: 0		;FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
5071	002354	000000		XXDPCHAIN: 0		;FLAGS XXDP CHAIN MODE PROGRAMMING RULES
5072				;NOTE: THESE TWO BYTES MUST STAY TOGETHER		
5073	002356	000		\$NULL: .BYTE	0	;CONTAINS NULL CHARACTER FOR FILLS
5074	002357	000		\$FILLS: .BYTE	0	;CONTAINS # OF FILL CHARACTERS
5075	002360	000		\$TPFLG: .BYTE	0	; 'TERMINAL NOT AVAILABLE' FLAG
5076				.EVEN		
5077	002362	000000		\$ESCAPE: 0		;ESCAPE ON ERROR ADDRESS
5078	002364	000000		EVEN: 0		;USED FOR ALTERNATE DATA PATTERNS
5079	002366	000000		STRIPES: 0		;COUNTS DIAGONAL STRIPES
5080	002370	000000		COUNT: 0		;BACKED UP COPY OF STRIPES
5081	002372	000000		NOTAB: 0		;NO TABLE BEING PRINTED - NOW
5082	002374	000000		BSIZE: 0		;SIZE OF 11/45 MOS MEMORY IN K WORDS***** ? *****
5083	002376	000000		KSIZE: 0		;SIZE OF MF11S-K MEMORY IN K WORDS***** ? *****
5084	002400	000000		LSIZE: 0		;SIZE OF MSV11-L/P MEMORY IN K WORDS
5085	002402	000000		MSIZE: 0		;SIZE OF MSV11-J MEMORY IN K WORDS
5086	002404	000000		PSIZE: 0		;SIZE OF Q-BUS PARITY MEMORY IN K WORDS
5087	002406	000000		TOOMANY: 0		;FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
5088	002410	000000		READONLY: 0		;FLAG TO PATTERNS TO READ ONLY
5089	002412	000000	000000	TESTADD: 0,0		;THE ADDRESS TO RUN CSR TESTS ON
5090	002416	000000		UNITOP: 0		;HIGHEST ACCESSABLE BANK OF MEMORY THRU Q-BUS MAP
5091	002420	000000		STOPOK: 0		;FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
5092	002422	000000		APTPAR: .WORD	0	;AMOUNT OF PARITY MEMORY ACCORDING TO APT
5093	002424	000000		APTECC: .WORD	0	;AMOUNT OF ECC MEMORY ACCORDING TO APT
5094	002426	000000		NOFSMODE: 0		;FLAG TO DISABLE FIELD SERVICE MODE
5095	002430	000000		NOERROR: 0		; "THIS IS NOT AN ERROR" FLAG
5096	002432	000000		LOADBANK: 0		;BANK LOADERS ARE RELOCATED TO
5097	002434	000000		TEMP: 0		;USED FOR JUNK
5098	002436	000000		QUICK: 0		;QUICK STOP FLAG FOR APT POWER FAIL
5099	002440	000000		NOSCOPE: 0		; "NO SCOPE LOOP ALLOWED" FLAG
5100	002442	000000		FSINFLAG: 0		; "FIELD SERVICE - NO INTERNAL INTERLEAVE" FLAG
5101	002444	000000		APTSIZE: 0		;APT SIZING INFO FLAG
5102	002446	000000		FS7FLAG: 0		;TRUE WHEN IN FIELD SERVICE COMMAND 7
5103	002450	000000		CONFGERROR: 0		;CONFIGURATION ERROR FLAG
5104	002452	000000		I: 0		;USED FOR GENERAL PURPOSE INDEXING
5105	002454	000000		NO22BIT: 0		;NO 22-BIT MODE FLAG
5106	002456	000000		NOSUPER: 0		;NO SUPERVISOR MODE FLAG
5107	002460	000000		ERRADD: .WORD	0	;HOLDS THE CSR'S ERROR ADDRESS
5108	002462	000000	000000 000000	CSRINFO: 0,0,0,0,0,0,0,0,0,0		;USED TO STORE INFORMATION ABOUT THE 16
	002470	000000	000000 000000			
	002476	000000	000000			
5109	002502	000000	000000 000000		0,0,0,0,0,0,0,0,0,0	;POSSIBLE CSR'S
	002510	000000	000000 000000			
	002516	000000	000000			
5110	002522	000000		LINK1: 0		;USED TO HOLD LINKS TO PATTERNS WHICH
5111	002524	000000		LINK2: 0		;CAN EXECUTE IN THE PAR/PUR'S OR NOT
5112	002526	000000		CSRHOLD: 0		;USED TO STORE CSR VALUES FOR CSR TESTS
5113	002530	000000		KFLAG: 0		;USED TO FLAG MF11S-K MEMORY TO TESTS*****

5114	002532	000000	000000	PGMCSR: .WORD	0,0	;POINTS TO PROGRAM CSR
5115	002536	000000		INHECC: .WORD	0	;FLAGS INHIBIT ECC TESTS ON RELOCATION
5116	002540	000000		INHBANK: .WORD	0	:
5117	002542	000000		FULLREL: .WORD	0	:
5118	002544			\$CMTGE: ;*	END OF COMMON TAGS	

				SBTTL	VARIABLES	INITIALIZED TO NON ZERO
5121						
5122	002544	000401	000000	CACHKN:	401,0	;CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
5123	002550	001014		CACHKF:	1014	;CACHE CONSTANT (MOVED TO CONTRL TO TURN OFF CACHE)
5124	002552	040000		TESTMODE:	40000	;USED TO SELECT THE PROPER TEST MODE FOR A PATTERN RUN
5125	002554	000012		ERRMAX:	10	;MAX # OF ERRORS PER BANK WITH SW11
5126	002556	000177		LASTBANK:	177	;HIGHEST BANK OF MEMORY
5127	002560	177000		LASTBLOCK:	177000	;HIGHEST BANK OF MEMORY*1 (IN PAR FORMAT)
5128	002562	160000		ENDADD:	160000	;ENDING ADDRESS
5129	002564	000000		ENDFLG:	0	;END FLAG
5130	002566	000000		SWRFLG:	0	;USED TO BUMP STACK ON FIRST CALL TO GTSWR
5131	002570	000000		PASCNT:	0	;PASS COUNTER
5132	002572	000031		SOBK:	25	;SOB CONSTANT
5133	002574	002000		KSTACK:	STACK	;STACK BEGINNING
5134	002576	000001		LOADHOME:	1	;HOME BANK OF LOADERS
5135	002600	177777		WORST:	177777	;SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
5136	002602	176543		SEEDHI:	176543	;WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
5137	002604	123456		SEEDLO:	123456	;WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
5138	002606	176543		MSEEDH:	176543	;MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
5139	002610	123456		MSEEDL:	123456	;MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
5140	002612	177777		HEADER:	177777	;USED TO PRINT HEADINGS ONLY ONCE
5141	002614	177777		ONES:	177777	;FOR AID IN "MOV" INSTRUCTIONS
5142	002616	000003		FLIPLOC:	3	;COUNTER FOR FLIPPING DATA ON WORST CASE NOISE TEST
5143	002620	052525		SOFTPAT:	52525	;PATTERN FOR SOFT ERROR BACKGROUND TESTS
5144	002622	000000		\$LPADR:	.WORD 0	;CONTAINS SCOPE LOOP ADDRESS
5145	002624	000000		\$LPERR:	.WORD 0	;CONTAINS SCOPE RETURN FOR ERRORS
5146	002626	000000		RESTART:	0	;RESTART (START ADD 202) FLAG
5147	002630	000000		\$ERTTL:	.WORD 0	;CONTAINS TOTAL ERRORS
5148						
5149				;***** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER *****		
5150	002632	000377		BAKPAT:	.WORD 377	;BACKGROUND PATTERN *
5151	002634	177400		SWAPAT:	.WORD 177400	;SWAPPED BAKPAT *
5152				;*****		
5153						
5154	002636	177570		SWR:	.WORD DSWR	;ADDRESS OF SWITCH REGISTER
5155	002640	177570		DISPLAY:	.WORD DDISP	;ADDRESS OF DISPLAY REGISTER
5156	002642	177560		\$TKS:	177560	;TTY KBD STATUS
5157	002644	177562		\$TKB:	177562	;TTY KBD BUFFER
5158	002646	177564		\$TPS:	177564	;TTY PRINTER STATUS REG. ADDRESS
5159	002650	177566		\$TPB:	177566	;TTY PRINTER BUFFER REG. ADDRESS
5160	002652	012		\$FILLC:	.BYTE 12	;INSERT FILL CHARS. AFTER A "LINE FEED"
5161	002653	207	377	\$BELL:	.ASCIZ <207><377><377>	;CODE FOR BELL
	002656	000				
5162	002657	077		\$QUES:	.ASCII /?/	;QUESTION MARK
5163	002660	015		\$CRLF:	.ASCII <15>	;CARRIAGE RETURN
5164	002661	012	000	\$LF:	.ASCIZ <12>	;LINE FEED
5165					.EVEN	

```
5168 .SBTTL CONFIGURATION TABLE
5169 ;CONFIG:FIRST 16K CONFIGURATION WORDS (2 EACH)
5170 ; 2ND 16K CONFIGURATION WORDS (2 EACH)
5171 ;
5172 ; 200TH 16K CONFIGURATION WORDS (2 EACH)
5173 ;
5174 ;CONFIGURATION WORDS:
5175 ; LOW: BIT 0 ERRORS PRESENT
5176 ; BIT 1 MEMORY SUCCESSFULLY ACCESSED
5177 ; BIT 2-4 RESERVED
5178 ; BIT 5 SKIP ECC LOGIC TESTS FLAG (1=SKIP)
5179 ; BIT 6 PROTECTED REGION OF ECC MEMORY
5180 ; BIT 7 PROTECTED (PROGRAM SPACE)
5181 ; BIT 8-11 CSR CODE
5182 ; BIT 12-15 RESERVED
5183 ; HIGH: BIT 0-7 NUMBER OF ERRORS
5184 ; BIT 8-10 MEMORY TYPE
5185 ; BIT 11 RESERVED
5186 ; BIT 12 RESERVED
5187 ; BIT 13 "BACKGROUND PATTERN VALID" FLAG
5188 ; BIT 14 BANK SELECTED FOR TEST BY FIELD SERVICE MODE
5189 ; BIT 15 LOADERS HOME BANK
5190 002664 000201 CONFIG: .REPT 201
5193 003670 CONFIEND:
```

```

5195
5196 003670
5197
5198 003670
5199
5200
5201
5202 003670 013737 000030 003706 EMTSAV: MOV 30,SAV30 ;;SAVE EMULATOR ADDRESS
5203 003676 013737 000032 003710 MOV 32,SAV32 ;;SAVE EMULATOR PRIORITY LEVEL
5204 003704 000402 BR VMKOR ;;GET AROUND TAG AREA
5205 003706 000000 SAV30: .WORD 0 ;;PUT EMULATOR INFO HERE
5206 003710 000000 SAV32: .WORD 0 ;;PUT PRIORITY LOCATION HERE
5207 003712 105737 056666 VMKOR: TSTB $ENV
5208 003716 001001 BNE NORES
5209 003720 000005 RESET
5210 003722 NORES: CLEAR MONFLG ;;CLEAR RETURN TO MONITOR FLAG
5211 003722 005037 002276 CLR MONFLG
5212 003726 010637 002274 MOV SP,SAVMON ;;SAVE XXDP MONITOR RESTART ADDRESS
5213 003732 013706 002574 MOV KSTACK,SP ;;SETUP THE STACK POINTER
5214 003736 012700 002000 MOV #$CMTAG,R0 ;;FIRST LOCATION TO BE CLEARED
5215 003742 005020 1$: CLR (R0)+ ;;CLEAR MEMORY LOCATION
5216 003744 022700 002544 CMP #$CMTGE,R0 ;;DONE?
5217 003750 001374 BNE 1$ ;LOOP BACK IF NO
5218 003752 012737 000177 002556 MOV #177,LASTBANK ;RESTORE LASTBANK (THIS MUST BE DONE PRIOR TO SYSTEM SIZING)
5219 003760 SUBTST <<CLEAR NON-PROGRAM SPACE>>
5220
5221
5222 003760 012737 000001 002076 ;*****
5223 003766 005000 ;*SUBTEST CLEAR NON-PROGRAM SPACE
5224 003770 000241 ;*****
5225 003772 005520 ;THIS ATTEMPS TO GET RID OF ANY PARITY ERRORS BY WRITING INTO
5226 003774 020027 160000 ;EVERY LOCATION THAT IS NOT LOADED INTO BY THE PROGRAM OR ALLOCATED
5227 004000 103773 ;TO THE XXDP LOADERS
5228 004002 005037 002076 MOV #1,NOPAR ;PARITY ACTION = COUNT & IGNORE
5229 CLR R0
5230 2$: CLC
5231 ADC (R0)+
5232 CMP R0,#160000
5233 BLO 2$
5234 CLR NOPAR ;RESTORE DEFAULT PARITY ACTION

```

5231 004006

SUBTST <<TYPE OF SYSTEM SIZER>>

```

*****
;*****
;SUBTEST      TYPE OF SYSTEM SIZER
;*****

```

5232 004006 000401

5233 004010 000000

5234 004012 012737 004074 000004

5235 004020 012737 004032 000010

5236 004026 000007

5237

5238

5239

5240

5241 004030 000406

5242 004032 012737 047122 000034 1\$:

5243 004040 104401 071474

5244 004044 000000

5245 004046 012737 000012 000010 2\$:

5246 004054 110037 004010

5247 004060 012737 004074 000004

5248 004066 005737 177746

5249 004072 000447

5250 004074 005037 002544 3\$:

5251 004100 012737 002336 060342

5252 004106 022737 000005 004010

5253 004114 001436

5254 004116 012737 004146 000004

5255 004124 005037 172516

5256 004130 052737 000020 172516

5257 004136 032737 000020 172516

5258 004144 001005

5259 004146 005237 002454 4\$:

5260 004152 012737 000007 002556

5261 004160 012737 140000 002552 5\$:

5262 004166 005237 002456

5263 004172 005037 060212

5264 004176 005037 060352

5265 004202 005037 060214

5266 004206 005037 060354

5267 004212 022737 000005 004010 6\$:

5268 004220 001003

5269 004222 052737 000020 172516

5270 004230 012737 004254 000004 22\$:

5271 004236 005037 052314

5272 004242 005737 177766

5273 004246 012737 177777 052314

5274 004254 7\$:

BR SYSSIZ

PROTYP: .WORD 0

SYSSIZ: SET4 #3\$

MOV #3\$,4

.DSABL CRF

MOV #1\$,10

MFPT

BR 2\$

MOV #TYPE,34

TYPE MSG130

TYPEIT ,MSG130

.DSABL CRF

HALT

MOV #12,10

MOVB R0,PROTYP

SET4 #3\$

MOV #3\$,4

.DSABL CRF

TST CONTRL

BR 6\$

CLR CACHKN

MOV #ZEROS,DT14

CMP #5,PROTYP

BEQ 6\$

SET4 #4\$

MOV #4\$,4

.DSABL CRF

CLR MMR3

BIS #BIT4,MMR3

BIT #BIT4,MMR3

BNE 5\$

INC NO22BIT

MOV #7,LASTBANK

MOV #140000,TESTMODE

INC NOSUPER

CLR DT5+10

CLR DT14+10

CLR DT5+12

CLR DT14+12

CMP #5,PROTYP

BNE 22\$

BIS #BIT4,MMR3

SET4 #7\$

MOV #7\$,4

.DSABL CRF

CLR CPERRF

TST @#177766

MOV #-1,CPERRF

RES4

;SKIP OVER VARIABLE LOCATION

```

;TRAPS TO 10 = BAD PROCESSOR TYPE
;TYPE OF PROCESSOR TEST: THIS INSTRUCTION
;(AVAILABLE ON NEWER PROCESSORS ONLY) PLACES
;A CODE IN THE LOWER BYTE OF R0 THAT
;INDICATES THE PROCESSOR TYPE. 1=11/44
;3=11/23, 5=11/83/84 (Orion)

```

```

;LOAD TRAP VECTOR
;TELL THEM BAD PROCESSOR TYPE

```

```

;NO NEED TO GO ON
;RESTORE TRAPS TO 10
;MOV THE CODE TO PROTYP
;

```

```

;SEE IF CACHE REGISTER RESPONDS
;BRANCH IF CACHE AVAILABLE
;NO CACHE ON SYSTEM
;DO NOT PRINT CONTRL ERROR MESSAGES
;IS THIS A 11/83/84
;YES - BRANCH
;

```

;SEE IF THERE IS 22-BIT ADDRESSING

```

;BRANCH IF 22-BIT RELOCATION
;SET FOR NO 22 BIT ADDRESSING
;HIGHEST BANK OF MEMORY
;MAKE TESTMODE USER

```

;CLEAR SOME ERROR DATA TAGS

```

;CPU TYPE = 11/83/84
;NO - BRANCH
;SET UP 22 BIT ADDRESSING
;TRAPS GO TO 4$

```

```

;CLEAR THE FLAG
;IS THERE A CPU ERROR REGISTER?
;YES-TRAPPED
;ENABLE TRAPS TO 4

```

004254	012737	034114	000004	MOV	#TIMEOUT,4	
004262	022737	000005	004010	CMP	#5,PROTYP	;IS THIS AN 11/83/84 ?
004270	001002			BNE	101\$;BRANCH IF NOT
004272	005037	177766		CLR	CPUERR	;CLEAR OUT THE CPU ERROR REGISTER BITS
004276			101\$:			;THAT A EXPECTED TRAP COULD HAVE SET
				.DSABL	CRF	

5275

5278 004276

SUBTST <<INITIALIZE VARIABLES TO NON ZERO>>

```

*****
;*****
;SUBTEST      INITIALIZE VARIABLES TO NON ZERO
;*****

```

5279 004276

SET WORST

MOV #-1,WORST

5280 004304

012737 177777 002600

MOV #3,FLIPLOC

5281 004312

012737 000003 002616

SET HEADER

MOV #-1,HEADER

5282 004320

012737 177777 002612

MOV #176543,MSEEDH

5283 004326

012737 176543 002606

MOV #123456,MSEEDL

5284 004334

013737 123456 002610

MOV MSEEDH,SEEDHI

;PRIME THE RANDOM NUMBER GENERATOR

5285 004342

013737 002606 002602

MOV MSEEDL,SEEDLO

;BOTH HIGH AND LOW WORDS

5286 004350

012737 000377 002632

MOV #377,BAKPAT

5287 004356

012737 177400 002634

MOV #177400,SWAPAT

SUBTST <<INITIALIZE VECTORS>>

```

*****
;*****
;SUBTEST      INITIALIZE VECTORS
;*****

```

5289 004364

012737 051146 000020

MOV #SCOPE,IOTVEC

;;IOT VECTOR FOR SCOPE ROUTINE

5290 004372

012737 000340 000022

MOV #340,IOTVEC+2

;;LEVEL 7

5291 004400

012737 051502 000030

MOV #ERROR,EMTVEC

;;EMT VECTOR FOR ERROR ROUTINE

5292 004406

012737 000340 000032

MOV #340,EMTVEC+2

;;LEVEL 7

5293 004414

012737 056764 000034

MOV #TRAP,TRAPVEC

;;TRAP VECTOR FOR TRAP CALLS

5294 004422

012737 000340 000036

MOV #340,TRAPVEC+2

;;LEVEL 7

5295 004430

012737 045352 000024

MOV #PWRDN,PWRVEC

;;POWER FAILURE VECTOR

5296 004436

012737 000340 000026

MOV #340,PWRVEC+2

;;LEVEL 7

5297 004444

012737 033744 000114

MOV #PARITY,PARVEC

;GET READY FOR PARITY ERRORS

5298 004452

012737 000340 000116

MOV #340,PARVEC+2

5299 004460

012737 034140 000010

MOV #PDP1105,RESVEC

;RESERVED INSTRUCTION TRAP

5300 004466

012737 000340 000012

MOV #340,RESVEC+2

5301 004474

012737 034114 000004

MOV #TIMEOUT,ERRVEC

;SETUP TIMEOUT ERRORS

5302 004502

012737 000340 000006

MOV #340,ERRVEC+2

;SET PRIORITY OF ERROR TRAPS

5303 004510

012737 034126 000250

MOV #MMTRAP,MMVEC

;VECTOR FOR MEMORY MANAGEMENT

5304 004516

012737 000340 000252

MOV #340,MMVEC+2

5305 004524

104423

CACHON

;TURN CACHE ON

5308 004526

SUBTST <<INITIALIZE PATTERNS>>

```

*****
;*****
;SUBTEST      INITIALIZE PATTERNS
;*****

```

```

;THE APT E TABLE DETERMINES WHICH PATTERNS ARE GOING TO BE RUN.
;EACH BIT SET REPRESENTS A PATTERN TABLE ENTRY THAT IS TO BE LEFT
;ALONE (TO BE RUN).  EACH BIT CLEARED REPRESENTS A PATTERN TABLE ENTRY
;THAT IS TO BE OVERLAYED WITH THE ADDRESS OF A NULL PATTERN.

```

```

5309
5310
5311
5312
5313 004526 012700 056734
5314 004532 012001
5315 004534 012703 017014
5316 004540 012702 000020
5317 004544 004737 004644
5318 004550 012001
5319 004552 012702 000010
5320 004556 004737 004644
5321 004562 012001
5322 004564 012703 017244
5323 004570 012702 000020
5324 004574 004737 004644
5325 004600 012001
5326 004602 012702 000010
5327 004606 004737 004644
5328 004612 012001
5329 004614 012703 017430
5330 004620 012702 000020
5331 004624 004737 004644
5332 004630 012001
5333 004632 012702 000010
5334 004636 004737 004644
5335 004642 000417
5336
5337 004644

```

```

MOV    #DDW0,R0
MOV    (R0)+,R1
MOV    #MKCSRT,R3
MOV    #16.,R2
CALL   PATPLUG
MOV    (R0)+,R1
MOV    #8.,R2
CALL   PATPLUG
MOV    (R0)+,R1
MOV    #MKPAT,R3
MOV    #16.,R2
CALL   PATPLUG
MOV    (R0)+,R1
MOV    #8.,R2
CALL   PATPLUG
MOV    (R0)+,R1
MOV    #MJPAT,R3
MOV    #16.,R2
CALL   PATPLUG
MOV    (R0)+,R1
MOV    #8.,R2
CALL   PATPLUG
BR      SUBAAA

```

PATPLUG:SUBTST <<SUBR PLUG IN NULL PATTERNS>>

```

*****
;*****
;SUBTEST      SUBR      PLUG IN NULL PATTERNS
;*****

```

FOR I := #1 TO R2

```

5338 004644
      004644 012737 000001 002452
      004652
5339 004652 006001
5340 004654
      004654 103402
5341 004656 012713 024220
5342 004662
      004662
5343 004662 062703 000002
5344 004666
      004666 005237 002452
      004672 023702 002452
      004676 003765
      004700
5345 004700 000207

```

```

ROR    R1
ON.NOERROR      ;IF CARRY CLEAR

MOV    #MT0999,(R3)
END ;OF ON.ERROR

ADD    #2,R3
END ;OF FOR

```

```

MOV    #1,I
B0:;;;;;

```

BCS L0

L0:;;;;;

```

INC    I
CMP    I,R2
BLE    B0

```

E0:;;;;;

RETURN

5348 004702

SUBAAA: SUBTST <<CLEAR THE CONFIGURATION TABLE>>

;*****

;*SUBTEST CLEAR THE CONFIGURATION TABLE

;*****

;THIS ZEROS (UNLESS WE STARTED AT ADDRESS 202) THE CONFIG TABLE
;WHICH IS FULLY DISCRIBED AT LOCATION "CONFIG".

.ENABLE LSB

IF RESTART IS FALSE

TST RESTART

BNE L1

5349

5350

5351

5352 004702

004702 005737 002626

004706 001006

5353 004710

012700 002664

5354 004714

005020

5355 004716

022700 003670

5356 004722

001374

5357 004724

004724

MOV #CONFIG,R0

CLR (R0)+

CMP #CONFIEND,R0

BNE 1\$

END ;OF IF RESTART

L1:::;;

5358

5359 004724

012737 000002 002106

5360 004732

.DSABL LSB

MOV #BIT1,CPUBIT ;SET ID BIT

SUBTST <<SIZE FOR A HARDWARE SWITCH REGISTER>>

;*****

;*SUBTEST SIZE FOR A HARDWARE SWITCH REGISTER

;*****

;;IF NOT FOUND OR IT IS

;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.

.ENABL LSB

SET4 #3\$

;TRAPS TO 4 GOTO 3\$

MOV #3\$,4

.DSABL CRF

MOV #DSWR,SWR

;;SETUP FOR A HARDWARE SWITCH REGISTER

MOV #DDISP,DISPLAY

;;AND A HARDWARE DISPLAY REGISTER

IF #-1 EQ @SWR

;IF NO TRAP FROM REFERENCE TO @SWR AND @SWR = #-1

CMP #-1,@SWR

BNE L2

5361

5362

5363

5364 004732

004732 012737 004766 000004

5365 004740

012737 177570 002636

5366 004746

012737 177570 002640

5367 004754

004754 022777 177777 175654

004762 001023

5368 004764

000403

5369 004766

012716 004774

5370 004772

000002

5371 004774

004774 012737 034114 000004

005002 022737 000005 004010

005010 001002

005012 005037 177766

005016

BR 2\$

;;BRANCH IF NO TIMEOUT

3\$: MOV #2\$, (SP)

;;SET UP FOR TRAP RETURN

RTI

2\$: RES4

;RESET TRAPS TO 4 TO DEFAULT

MOV #TIMEOUT,4

CMP #5,PROTYP

;IS THIS AN 11/83/84 ?

BNE 101\$

;BRANCH IF NOT

CLR CPUERR

;CLEAR OUT THE CPU ERROR REGISTER BITS

101\$:

;THAT A EXPECTED TRAP COULD HAVE SET

.DSABL CRF

MOV #SWREG,SWR

;;POINT TO SOFTWARE SWR

MOV #DISPREG,DISPLAY

END ;OF IF #-1

L2:::;;

5372 005016

012737 000176 002636

5373 005024

012737 000174 002640

5374 005032

005032

5375

.DSABL LSB

5378 005032

SUBAAB: SUBTST <<SETUP ACT, APT, & XXDP>>

;*****
;SUBTEST SETUP ACT, APT, & XXDP
;*****

5379

5380

5381 005032 005037 056654

5382 005036 132737 000040 056667

005044 001403

5383 005046

005046 012737 177777 002360

5384 005054

005054

5385 005054

005054 132737 000200 056667

005062 001403

5386 005064

005064 012737 177777 002444

5387 005072

005072

5388 005072

005072 123727 056666 000001

005100 001023

5389 005102

005102 012737 177777 002352

005110 012737 177777 002346

005116 012737 177777 002062

005124 012737 177777 002436

5390 005132 012737 040720 000024

5391 005140 012737 056670 002636

CLR \$PASS ;CLEAR PASS COUNT
IFB #BIT5 SET.IN \$ENVM

SET \$TPFLG ;INDICATE NO TERMINAL

END ;OF IFB #BIT5

IFB #BIT7 SET.IN \$ENVM

SET APTSIZE

END ;OF IFB #BIT7

IFB \$ENV EQ #1

SET APTFLAG,QVFLAG,\$AUTO,QUICK

MOV #APTDOWN,PWRVEC
MOV #SWREG,SWR ;USE APT SWRBITB #BIT5,\$ENVM
BEQ L3

MOV #-1,\$TPFLG

L3:::~::~

BITB #BIT7,\$ENVM
BEQ L4

MOV #-1,APTSIZE

L4:::~::~

CMPB \$ENV,#1
BNE L5

MOV #-1,APTFLAG

MOV #-1,QVFLAG

MOV #-1,\$AUTO

MOV #-1,QUICK

5392	005146			ELSE	
	005146	000430			BR L6
	005150				L5: :::::
5393	005150			IF 42 NE #STACK AND 42 NE #0	
	005150	023727	000042		002000
	005156	001424			
	005160	005737	000042		
	005164	001421			
5394	005166			SET QVFLAG, \$AUTO	
	005166	012737	177777		002346
	005174	012737	177777		002062
5395	005202			IF 42 EQ #ENDAD	
	005202	023727	000042		014042
	005210	001004			
5396	005212			SET ACTFLAG	
	005212	012737	177777		002350
5397	005220			ELSE	
	005220	000403			
	005222				
5398	005222			SET XXDPCHAIN	
	005222	012737	177777		002354
5399	005230			END ;OF IF 42	
	005230				
5400	005230			END ;OF IF 42	
	005230				
5401	005230			END ;OF IFB \$ENV	
	005230				

5403 005230

SUBTST <<PROTECT PROGRAM & LOADERS>>

```

*****
;SUBTEST      PROTECT PROGRAM & LOADERS
*****

```

5404 005230 052737 000200 002664

BIS #BIT7,CONFIG

;PROTECT PROGRAM SPACE (BANK 0)

5405 005236 052737 000200 002670

BIS #BIT7,CONFIG+4

;PROTECT LOADER SPACE (BANK 1)

5406 005244 022737 014042 000042

IF #ENDAD NE 42

;NOT ACT-11?

CMP #ENDAD,42

BEQ L12

5407 005254 005737 002454

IF NO22BIT NE #0

TST NO22BIT

BEQ L13

5408 005262 012737 .77777 002276

SET MONFLG

;RETURN TO XXDP MONITOR

MOV #-1,MONFLG

5409 005270 104064

ERROR +64

;ILLEGAL PROCESSOR

5410 005272 000402

ELSE

BR L14

5411 005274 104401 071422

TYPE MSG000

L13:::;;

TYPEIT ,MSG000

;TYPE PROGRAM TITLE

.DSABL CRF

END

5412 005300

5413 005300

5414 005300

5415 005300

END ;OF IF #ENDAD

L14:::;;

L12:::;;

SUBTST <<CHECK FOR CACHE AND MEMORY MANAGEMENT>>

```

*****
;SUBTEST      CHECK FOR CACHE AND MEMORY MANAGEMENT
*****

```

5416

5417

5418 005300 012737 005500 000004

SET4 #3

MOV #3,4

.DSABL CRF

;IS THERE A CONTROL REGISTER?

5419 005306 005737 177746

TST CONTRL

5420 005312 012737 005412 000004

SET4 #1

MOV #1,4

.DSABL CRF

;IS THERE A MMR3 REGISTER?

5421 005320 005737 172516

TST MMR3

;;V SAVE OLD TIME OUT

5422 005324 013746 000004

MOV 4,-(SP)

;;V PASS ON KTJ11

5423 005330 012737 005400 000004

MOV #14,4

;;V TEST FOR 1184

5424 005336 032737 001000 177750

BIT #BIT9,@#177750

;;V ELSE REPORT 1183

5425 005344 001415

BEQ 14

;;V REPORT 1184

5426 005346 104401 071100

TYPE M1184

TYPEIT ,M1184

.DSABL CRF

;;V SET ABILITY TO CLEAR

5427 005352 052737 000400 177730 12:

BIS #BIT8,@#177730

;;V UNIBUS MEMORY ACCESS

5428 005360 042737 000077 177734

BIC #77,@#177734

;;V TO UNIBUS

5429 005366 104401 071547

TYPE NOUBMT

TYPEIT ,NOUBMT

.DSABL CRF

;;V AND REPORT NO UNIBUS

5430 005372 012637 000004

MOV (SP)+,4

;;V MEMORY ACCESS AND CONTINUE

5431 005376 000411

BR 4

;;V ALSO RESTORE OLD TIME OUT

5432 00540 012637 000004 14:

MOV (SP)+,4

; 11/83

5433 005404 104401 071112

TYPE MSG117

TYPEIT ,MSG117

5434	005410	000404			.DSABL	CRF	
5435	005412				BR	4\$	
	005412	012737	177777	002276	SET	MONFLG	;PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC
							MOV #-1,MONFLG
5436	005420	104064			ERROR	+64	;NO MEMORY MANAGEMENT
5437	005422	052737	000014	177746	4\$: BIS	#BIT2!BIT3,CONTRL	;SET CACHE DISABLE BITS
5438	005430	042737	000014	177746	BIC	#BIT2!BIT3,CONTRL	;CLEAR CACHE DISABLE BITS
5439	005436	032737	000004	177746	BIT	#BIT2,CONTRL	;IS THE BIT SET?
5440	005444	001004			BNE	7\$;BRANCH IF THE BIT IS SET
5441	005446	032737	000010	177746	BIT	#BIT3,CONTRL	;IS THE BIT SET?
5442	005454	001413			BEQ	6\$;BRANCH IF THE BIT IS SET
5443	005456				7\$: TYPE	MSG121	; CACHE BYPASSED
	005456	104401	071154		TYPEIT	,MSG121	
					.DSABL	CRF	
5444	005462	104424			CACHOFF		
5445	005464	013737	002544	002546	MOV	CACHKN,CACHKN+2	;SAVE INFO ABOUT CACHE
5446	005472	005037	002544		CLR	CACHKN	;CACHE CANNOT BE USED - IT'S BYPASSED
5447	005476	000404			BR	8\$	
5448	005500				3\$: TYPE	MSG119	
	005500	104401	071124		TYPEIT	,MSG119	
					.DSABL	CRF	
5449	005504				6\$: TYPE	MSG120	;CACHE AVAILABLE
	005504	104401	071133		TYPEIT	,MSG120	
					.DSABL	CRF	

```

5451 005510          SUBTST <<SETUP USER & SUPERVISOR STACK>>
;*****
;SUBTEST          SETUP USER & SUPERVISOR STACK
;*****
5452 005510 104421    8$: DEENERGIZE          ;TURN OFF MEMORY MANAGEMENT
5453 005512 005737 002456 TST      NOSUPER      ;IS THERE A SUPERVISOR MODE?
5454 005516 001011    BNE      5$              ;NO-SKIP SUPERVISOR SETUP.
5455
5456          ;SET PREVIOUS MODE TO SUPERVISOR
5457 005520 042737 030000 177776 BIC      #BIT13!BIT12,PSW
5458 005526 052737 010000 177776 BIS      #BIT12,PSW
5459
5460 005534          PUSH      #SUPSTK
5461 005534 012746 000740          MOV #SUPSTK,-(SP)
5462 005540 006606    MTPI      SSP
5463
5464 005542 052737 030000 177776 5$: ;SET PREVIOUS MODE TO USER
5465          BIS      #BIT13!BIT12,PSW
5466 005550          PUSH      #USESTK
5467 005550 012746 000700          MOV #USESTK,-(SP)
5468 005554 006606    MTPI      USP
5469 005556          SUBTST <<GET SOFTWARE SWITCH REGISTER IF NECESSARY>>
;*****
;SUBTEST          GET SOFTWARE SWITCH REGISTER IF NECESSARY
;*****
5470 005556          IF $AUTO IS FALSE          ;IF NOT(APT OR ACT)
5471 005556 005737 002062          TST $AUTO
5472 005562 001012          BNE L15
5473 005564          IF SWR EQ #SWREG          ;IF SOFTWARE SWITCH REG SELECTED
5474 005564 023727 002636 000176          CMP SWR,#SWREG
5475 005572 001006          BNE L16
5476 005574          SET SWRFLG          ;;SET FLG TO BUMP STACK
5477 005574 012737 177777 002566          MOV #-1,SWRFLG
5478 005602 104407          GTSWR          ;;GET SOFT-SWR SETTINGS
5479 005604 005037 002566          CLR SWRFLG          ;;CLEAR IT FOR REST OF PROGRAM
5480 005610          END ;OF IF SWR
5481 005610          L16:::
5482 005610          END ;OF IF $AUTO          L15:::
5483
5484          SUBTST <<GET MEMORY MANAGEMENT READY>>
;*****
;SUBTEST          GET MEMORY MANAGEMENT READY
;*****
5485 005610 104422    KMAP          ;MAP KERNEL SPACE 1 TO 1
5486 005612          MAP          ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1
5487 005612 010346          MOV R3,-(SP)
5488 005614 012703 000200          MOV #200,R3
5489 005620 004737 035716          CALL MAPPER
5490          .DSABL CRF
5491 005624 012603          MOV (SP)+,R3
5492 005626 104420          ENERGIZE          ;TURN ON MEMORY MANAGEMENT

```


5484 005630

005630 000004

5485
5486
5487
5488
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502
5503 005632 005005
5504 005634 005000
5505 005636 012703 172100
5506 005642 012737 000001 002076
5507 005650
005650 012737 006024 000004

5508 005656
005656
5509 005656 005713
5510 005660 052705 000001
5511 005664 005004
5512 005666 042760 000006 002462
5513 005674 052760 000030 002462
5514 005702 005013
5515 005704
005704 012713 020000
5516 005710
005710 032713 020000
005714 001403
5517 005716 052760 000001 002462
5518 005724
005724
5519 005724 005013
5520 005726 004737 006220
5521 005732
005732 026027 002462 000030
005740 100004
5522 005742 016037 002462 002052
5523 005750 104021
5524 005752
005752
5525 005752 062700 000002
5526 005756 062703 000002

```

NEWST <<BIT TEST OF ALL CSR'S>>
;*****
;*TEST 1      BIT TEST OF ALL CSR'S
;*****
TST1:  SCOPE
;* THE FIRST PART OF THE CONFIGURATION ANALYSIS DOES THE FOLLOWING:
;*   1) FINDS WHICH CSR'S RESPOND, AND PUTS THEM INTO THE CSR INFORMATION
;*      TABLE, AND STORES ANOTHER BIT FOR "TOTCSRS".
;*   2) TESTS THE CSR BITS COMMON TO ALL CSR'S.
;*   3) FIGURES OUT IF THE MODULE IS A ECC, OR PARITY MEMORY
;*   4) TESTS THE BITS PARTICULAR TO THAT TYPE OF CSR.
;*   5) IF ANY BITS TEST BAD IN THE CSR UNDER TEST, THE CSR OK BIT IN THE
;*      CSR INFORMATION TABLE IS CLEARED.
;* THE INFORMATION BITS ONE THROUGH THREE FORM A CODE WHICH GIVES THE TYPE
;* OF CSR:
;*
;*      TYPE      ECC      BIT0
;*
;*      MSV11-L/P      0      PARITY
;*      MSV11-J        1      ECC
;*
;* THIS MEMORY CODE WILL BE USED IN THE SECOND PART OF THIS ANALYSIS
;*
CLR      R5          ;R5 IS THE TOTAL CSR NUMBER
CLR      R0          ;R0 IS A TABLE INDEX
MOV      #CSRADD,R3  ;R3 HAS THE CSR ADDRESS
MOV      #1,NOPAR    ;IGNORE PARITY ERRORS
SET4     #CSRBMP
MOV      #CSRBMP,4
.DSABL   CRF
REPEAT

B1:;;;;;
TST (R3)            ;DOES THIS CSR RESPOND???
BIS #1,R5           ;MARK IT IN CSR MAP
CLR      R4          ;CLEAR THE LAST CSR INDICATOR
BIC #6,CSRINFO(R0)  ;CLEAR UNUSED BITS
BIS #BIT4:BIT3,CSRINFO(R0) ;YES-MARK IT IN CSR INFORMATION TABLE
CLR (R3)            ;CLEAR THE CSR UNDER TEST
LET (R3) := #BIT13  ;IS THIS AN ECC MEMORY???
MOV #BIT13,(R3)
IF #BIT13 SET.IN (R3);IS BIT 13 SET
BIT #BIT13,(R3)
BEQ L17
BIS #BIT0,CSRINFO(R0);MARK IT IN THE TABLE AS BEING A ECC MEMORY
END
;
L17:;;;;;
CLR (R3)            ;CLEAR CSR UNDER TEST
CALL RWCSR          ;BIT TEST OF ALL BITS IN CSR'S
IF CSRINFO(R0) MI #30 ;DO WE HAVE A LEGAL CONFIGURATION?
CMP CSRINFO(R0),#30
BPL L20
MOV CSRINFO(R0),BAD ;MOVE IN BAD DATA
ERROR +21
;
END
;
L20:;;;;;
NXTCSR:  ADD #2,R0    ;GO TO NEXT CSR
          ADD #2,R3    ;GO TO NEXT CSR

```

```

5527 005762 006305          ASL R5          ;SHIFT CSR MAP
5528 005764          ON.ERROR        ;IS THERE A CSR 0
          005764 103001          ;
5529 005766 005204          INC R4          ;YES SET CSR PRESENT FLAG
5530 005770          END              ;
          005770          L21:::;
5531 005770          UNTIL R0 EQ #40      ;UNTIL ALL CSR'S ARE DONE
          005770 020027 000040          CMP R0,#40
          005774 001330          BNE B1
          005776          E1:::;
5532 005776 006005          ROR R5          ;RESYNC R5
5533 006000 005704          TST R4          ;WAS THERE A CSR 0?
5534 006002          RNE 22$           ;BRANCH IF NOT EQUAL
          006002 001402          BEQ 22$
5535 006004 052705 100000      BIS #BIT15,R5 ;YES SET IT IN CSR TABLE
5536 006010          22$: LET TOTCSRS := R5 ;STORE CSR MAP IN TOTCSRS
          006010 010537 002224          MOV R5,TOTCSRS
5537 006014 004737 006034          CALL CSRMAP ;PRINT CSR MAP
5538 006020          JUMPTO CTEST        ;
          006020 000137 006714          JMP CTEST
5539
5540 006024 062706 000004      CSRBMP: ADD #4,SP ;FIX STACK POINTER FOR NON-EXISTANT CSR TRAP
5541 006030 000137 005752      JMP NXTCSR- ;GO ON TO NEXT CSR
5542

```

5544 006034

CSRMAP: SUBTST <<PRINT CSR REGISTER MAP>>

```

;*****
;SUBTEST PRINT CSR REGISTER MAP
;*****

```

5545 006034 005000
 5546 006036 104401 071370
 006036

```

CLR R0 ;CLEAR CSR INFO POINTER
TYPE MSG008 ;PRINT TITLE

```

5547 006042 104401 066064
 006042

```

TYPEIT ,MSG008
.DSABL CRF
TYPE MSG016 ;PRINT CSR NUMBERS
TYPEIT ,MSG016
.DSABL CRF

```

5548 006046 005001

5549 006050

```

CLR R1
REPEAT

```

5550 006050 010102
 5551 006052 022702 000011
 5552 006056 100002

```

MOV R1,R2 ;
CMP #9.,R2 ;
BPL 1$ ;JUMP AROUND NEXT INSTRUCTION

```

5553 006060 062702 000007
 5554 006064 062702 000060
 5555 006070 110237 066062

1\$:

```

ADD #7,R2
ADD #60,R2 ;MAKE IT ASCII

```

5556 006074 104401 066062
 006074

```

MOVB R2,MSG015
TYPE MSG015

```

5557 006100 104401 066060
 006100

```

TYPEIT ,MSG015
.DSABL CRF
TYPE MSG014 ;TYPE SINGLE SPACE
TYPEIT ,MSG014
.DSABL CRF

```

5558 006104 005201

5559 006106

```

INC R1
UNTIL R1 EQ #16.

```

006106 020127 000020
 006112 001356
 006114

```

CMP R1,#16.
BNE B2

```

5560 006114 104401 065544
 006114

```

TYPE MSG009 ;TYPE MEMTYPE
TYPEIT ,MSG009
.DSABL CRF
REPEAT

```

E2:::~::~

5561 006120

006120

5562 006120

006120

006124

5563 006126

006126

5564 006136

5565 006144

006144

006146

5566 006146

5567 006154

006154

5568 006154

006154

006156

5569 006156

5570 006164

006164

5571 006164

006164

104401 066062

```

IF CSRINFO(R0) NE #0 ;IS CSR NONEXSISTANT???

```

B3:::~::~

IF #BIT0 SET.IN CSRINFO(R0)

```

TST CSRINFO(R0)
BEQ L24

```

```

BIT #BIT0,CSRINFO(R0)
BEQ L25

```

```

MOVB #'E,MSG015 ;IT IS A MSV11-J
ELSE

```

BR L26

L25:::~::~

MOVB #'P,MSG015 ;IT IS A MSV11-L/P

END

L26:::~::~

ELSE

BR L27

L24:::~::~

MOVB #' ,MSG015

END

L27:::~::~

```

TYPE MSG015
TYPEIT ,MSG015

```

;TYPE MEMORY TYPE

CVMJAAO MSV11-J MEMORY DIAG.
PRINT CSR REGISTER MAP

MACRO V05.01b Friday 15-Feb 85 16:10 Page 147-1

SEQ 0138

5572	006170			.DSABL CRF		
	006170	104401	066060	TYPE MSG014		;TYPE SPACE
				TYPEIT ,MSG014		
				.DSABL CRF		
5573	006174	000240		NOP		
5574	006176	062700	000002	ADD #2,R0		;POINT TO NEXT ENTRY
5575	006202			UNTIL R0 EQ #40		
	006202	020027	000040			
	006206	001344				
	006210					
5576	006210			TYPE MSG129		
	006210	104401	071471	TYPEIT ,MSG129		
				.DSABL CRF		
5577	006214	000207		RETURN		
5578	006216	000000		TRACE: .WORD 0		

CMP R0,#40
BNE B3
E3:::;;

5580 006220

SUBTST <<READ AND WRITE ALL CSR BITS>>

;*****
;SUBTEST READ AND WRITE ALL CSR BITS
;*****; THIS ROUTINE "RWCSR" CHECK TO SEE THAT THE CSR CAN BEWRITTEN ON CORRECTLY
; BY WRITING AND CHECKING FOR THE FOLLOWING PATTERNS:; 1-ZEROS
; 2-ONES
; 3-SHIFTING A ONE THROUGH A FIELD OF ZEROS
; 4-SHIFTING A ZEROS THROUGH A FIELD OF ONES

;RWCSR:

PUSH R4,R5,UIPARO

;SAVE R4,R5, AND UIPARO ON STACK

MOV R4, (SP)

MOV R5, (SP)

MOV UIPARO, -(SP)

LET R5 := R0

;GET CSR NUMBER FOR POSSIBLE ERROR

MOV R0,R5

ASR R5

LET CSRNO := R5

MOV R5,CSRNO

LET ADDRESS := R3

;GET ADDRESS FOR POSSIBLE ERROR

MOV R3,ADDRESS

IF #BIT0 SET.IN CSRINFO(R0) ;WHAT KIND OF MEMORY IS THIS???

;GET BIT MASKS FOR D

BIT #BIT0,CSRINFO(R0)

BEQ L31

LET R5 := #017740 ;MASK FOR MSV11-J

MOV #017740,R5

ELSE

;IT IS A MSV11-L/P

BR L32

LET R5 := #070032 ;MASK FOR MSV11-L/P

MOV #070032,R5

END

LET CSR1S := #177777 ;SET CSR1S TO ALL ONES

MOV #177777,CSR1S

BIC R5,CSR1S

;CLEAR BITS FOR GOOD DATA

LET (R3) := #0

;0 ---->CSR

CLR (R3)

LET R4 := (R3)

;MASK OUT UNWANTED BITS

MOV (R3),R4

BIC R5,R4

IF R4 NE #0

;DO WE HAVE A CORRECT READ

TST R4

BEQ L33

LET GOOD := #0

;GOOD DATA=0'S

CLR GOOD

LET CSR := R4

;BAD DATA=CSR

MOV R4,CSR

ERROR +35

;BIT SET ERROR

BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT

END

LET (R3) := CSR1S

;ONES >(R3)

L33:::;;

5581
5582
5583
5584
5585
5586
5587
5588
5589
5590 006220
5591 006220
006220 010446
006222 010546
006224 013746 177640
5592 006230
006230 010005
5593 006232 006205
5594 006234
006234 010537 002152
5595 006240
006240 010337 002034
5596 006244
006244 032760 000001 002462
006252 001403
5597 006254
006254 012705 017740
5598 006260
006260 000402
006262
5599 006262
006262 012705 070032
5600 006266
006266
5601 006266
006266 012737 177777 002322
5602 006274 040537 002322
5603 006300
006300 005013
5604 006302
006302 011304
5605 006304 040504
5606 006306
006306 005704
006310 001410
5607 006312
006312 005037 002044
5608 006316
006316 010437 002150
5609 006322 104035
5610 006324 042760 000010 002462
5611 006332
006332
5612 006332

5613	006332	013713	002322	LET R4 := (R3)	;MASK OUT CORRECT FIELD	MOV CSR15,(R3)
	006336	011304				MOV (R3),R4
5614	006340	005013		CLR (R3)	;CLEAR OUT CSR	
5615	006342	040504		BIC R5,R4		
5616	006344			IF R4 NE CSR15	;WAS PATTERN WRITTEN CORRECTLY?	
	006344	020437	002322			CMP R4,CSR15
	006350	001411				BEQ L34
5617	006352			LET GOOD := CSR15	;GOOD DATA = ALL LEGAL BITS SET	IN CSR
	006352	013737	002322 002044	LET CSR := R4	;BAD DATA=CSR	MOV CSR15,GOOD
5618	006360					MOV R4,CSR
	006360	010437	002150	ERROR +10	;BIT CLEAR ERROR	
5619	006364	104010		BIC #BIT3,CSRINFO(R0)	;CLEAR CSR OK BIT	
5620	006366	042760	000010 002462	END		
5621	006374					L34:::;:::;
	006374			LET PASFLG := #0	;SET UP LOOP COUNTER	
5622	006374					CLR PASFLG
5623	006400	005037	002264	REPEAT	;REPEAT WITH A FIELD OF 1'S THROUGH 0'S	
	006400					B4:::;:::;
5624						0'S THROUGH 1'S
5625	006400	005237	002264	INC PASFLG	;INCREMENT LOOP COUNTER	
5626	006404			LET UIPARO := #-1	;USE USER PAR FOR BIT COUNTER	
	006404	012737	177777 177640			MOV #-1,UIPARO
5627	006412			IF PASFLG EQ #1	;PASS 1	
	006412	023727	002264 000001			CMP PASFLG,#1
	006420	001003				BNE L35
5628	006422			LET R2 := #1	;1-- -->FIELD OF ZEROS	
	006422	012702	000001			MOV #1,R2
5629	006426			ELSE	;PASS 2	
	006426	000402				BR L36
	006430					L35:::;:::;
5630	006430			LET R2 := #177776	;0----->FIELD OF ONES	
	006430	012702	177776			MOV #177776,R2
5631	006434			END		
	006434					L36:::;:::;
5632	006434			REPEAT	;DO BITS 0-4 AND 13-15	
	006434					B5:::;:::;
5633	006434	005237	177640	INC UIPARO	;INCREMENT BIT POINTER	
5634	006440			IF PASFLG EQ #1 AND #BIT0 OFF.IN CSRINFO(R0)		
	006440	023727	002264 000001			CMP PASFLG,#1
	006446	001006				BNE L37
	006450	032760	000001 002462			BIT #BIT0,CSRINFO(R0)
	006456	001002				BNE L37
5635	006460	042702	040000	BIC #BIT14,R2	;IF THIS IS PASS 1 ON A MSV11-L/P,CLEAR BIT 14	
5636	006464			END		
	006464					L37:::;:::;
5637	006464			IF PASFLG EQ #2 AND #BIT0 OFF.IN CSRINFO(R0)		
	006464	023727	002264 000002			CMP PASFLG,#2
	006472	001006				BNE L40
	006474	032760	000001 002462			BIT #BIT0,CSRINFO(R0)
	006502	001002				BNE L40
5638	006504	042702	040004	BIC #BIT14!BIT2,R2	;IF THIS IS PASS 2 ON A MSV11-L/P, CLEAR ECRSD BIT AND	
5639	006510			END		
	006510					L40:::;:::;
5640	006510			LET (R3) := R2	;WRITE DATA	
	006510	010213				MOV R2,(R3)

```
5641 006512          LET R1 := R2          ;GET GOOD DATA AND MASK IT OUT
      006512 010201          ;MOV R2,R1
5642 006514 040501    BIC R5,R1          ;GET GOOD DATA
5643 006516          LET R4 := (R3)        ;GET DATA THAT IS READ
      006516 011304          ;MOV (R3),R4
5644 006520 040504    BIC R5,R4          ;MASK OUT CSR BITS
5645 006522          IF R1 NE R4          ;IS DATA CORRECT???
      006522 020104          ;CMP R1,R4
      006524 001416          ;BEQ L41
5646 006526          LET GOOD := R1        ;BAD DATA = CSR CONTENTS
      006526 010137 002044    ;MOV R1,GOOD
5647 006532          LET CSR := R4         ;GET GOOD DATA
      006532 010437 002150    ;MOV R4,CSR
5648 006536          IF PASFLG EQ #1      ;SELECT ERROR DEPENDING ON PASS
      006536 023727 002264 000001    ;CMP PASFLG,#1
      006544 001002          ;BNE L42
5649 006546 104035    ERROR +35          ;BIT SET ERROR
5650 006550          ELSE                ;PASS 2
      006550 000401          ;BR L43
      006552          L42:;;;;;;;;
5651 006552 104010    ERROR +10          ;BIT CLEAR ERROR
5652 006554          END
      006554          L43:;;;;;;;;
5653 006554 042760 000010 002462    BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
5654 006562          END
      006562          L41:;;;;;;;;
5655 006562          IF PASFLG EQ #1      ;GET DATA FOR NEXT LOOP
      006562 023727 002264 000001    ;CMP PASFLG,#1
      006570 001002          ;BNE L44
5656 006572 006302    ASL R2            ;SHIFT 1 ACROSS 0'S
5657 006574          ELSE
      006574 000402          ;BR L45
      006576          L44:;;;;;;;;
5658 006576 000261    SEC                ;SET CARRY
5659 006600 006102    ROL R2            ;ROTATE A 0 ACROSS A FIELD OF ONES
5660 006602          END
      006602          L45:;;;;;;;;
5661 006602          UNTIL UIPARO EQ #15. ;UNTIL ALL BITS ARE DONE
      006602 023727 177640 000017    ;CMP UIPARO,#15.
      006610 001311          ;BNE B5
      006612          E5:;;;;;;;;
5662 006612          UNTIL PASFLG EQ #2   ;DONE WITH 2 PASSES
      006612 023727 002264 000002    ;CMP PASFLG,#2
      006620 001267          ;BNE B4
      006622          E4:;;;;;;;;
5663 006622          IF #BIT0 SET.IN CSRINFO(R0) THEN JUMPTO DONE ;IF MSV11-L/P DO ONE LAST WRITE
      006622 032760 000001 002462    ;BIT #BIT0,CSRINFO(R0)
      006630 001402          ;BEQ L50
      006632 000137 006700          ;JMP DONE
      006636          L50:;;;;;;;;
5664 006636          LET (R3) := #140005  ;WRITE ONES TO CSR WITH ECSRRD BIT ENABLED
      006636 012713 140005          ;MOV #140005,(R3)
5665 006642          LET R2 := (R3)       ;READ CSR FOR CORRECT BITS
      006642 011302          ;MOV (R3),R2
5666 006644 042702 037772    BIC #37772,R2 ;CLEAR UNWANTED BITS
5667 006650          IF R2 NE #140005     ;WAS WRITE CORRECT
      006650 020227 140005          ;CMP R2,#140005
```

```

5668 006654 001411
5668 006656 012737 140005 002044
5669 006664 010237 002150
5670 006670 104010
5671 006672 042760 000010 002462
5672 006700
5673 006700 005013
5674 006702 012637 177640
5675 006712 000207

LET GOOD := #140005 ;GOOD DATA
LET CSR := R2 ;BAD DATA
ERROR +10 ;BIT CLEAR ERROR
BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT!
END
DONE: LET (R3) := #0 ;CLEAR OUT CSR
POP UIPAR0,R5,R4 ;RESTORE UIPAR0,R4, AND R5
RETURN

BEQ L51
MOV #140005,GOOD
MOV R2,CSR
L51:;;;;;
CLR (R3)
MOV (SP)+,UIPAR0
MOV (SP)+,R5
MOV (SP)+,R4

```


;THE FOLLOWING ROUTINE DETERMINES WHICH CSR CONTROLS PROGRAM SPACE

;CTEST: CACHOFF

5678									
5679									
5680	006714	104424							
5681	006716	012737	177777	002532					
5682	006724	012737	002000	172350					
5683	006732	012701	002412						
5684	006736	012737	100000	002412					
5685	006744	012737	100002	002414					
5686	006752	005000							
5687	006754	005037	002152						
5688	006760	013703	002224						
5689	006764	000240							
5690	006766	006303			4\$:				
5691	006770	103407							
5692	006772	062700	000002		1\$:				
5693	006776	010037	002152						
5694	007002	005703							
5695	007004	001465							
5696	007006	000767							
5697	007010	000240			2\$:				
5698	007012	000241							
5699	007014	032760	000001	002462					
5700	007022	001014							
5701	007024	052760	000004	172100					
5702	007032	012771	123456	000000					
5703	007040	012771	123456	000002					
5704	007046	005060	172100						
5705	007052	000414							
5706	007054	012760	000000	172100	5\$:				
5707	007062	012771	123456	000000					
5708	007070	012771	123456	000002					
5709	007076	012760	020006	172100					
5710	007104	005771	000000		6\$:				
5711	007110	016004	172100						
5712	007114	032760	000001	002462					
5713	007122	001003							
5714	007124	005704							
5715	007126	100412							
5716	007130	000720							

PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

5735 007232

SUBTST <<CLEAR ALL MEMORY SPACE FROM BANK 2 ON>>

```

*****
;SUBTEST      CLEAR ALL MEMORY SPACE FROM BANK 2 ON
*****

```

```

;THIS ROUTINE CLEARS ALL MEMORY SPACE BEGINNING AT ADDRESS 200,000 AND
;CONTINUES UNTIL THERE IS NO MEMORY LEFT. IT SHOULD CLEAR ANY PARITY ERRORS
;CREATED BY THE LAST ROUTINE, AND CLEAN UP ANY JUNK LEFT HANGING AROUND IN
;HIGHER MEMORY.
;

```

5742 007232

CLRMEM: SET4 #CLREX ;NONEM TRAPS GO TO CLREX

007232 012737 007342 000004

MOV #CLREX,4

.DSABL CRF

CLR TRACE

5743 007240 005037 006216

5744 007244 012737 000001 002076

MOV #1,NOPAR

;IGNORE PARITY ERRORS

5745 007252 012737 002000 172350

MOV #2000,KIPAR4

;SET UP MAP TO START AT BANK 2

5746 007260 012701 100C00

MOV #100000,R1

;R1 MAPS TO KIPAR4

5747 007264 020127 117776

1\$: CMP R1,#117776

;WHOLE 16K BANK DONE?

5748 007270 001003

BNE 2\$

;KEEP GOING IF NOT

5749 007272 012737 177777 006216

MOV #-1,TRACE

;USE TRACE FLAG TO FLAG END OF BANK

5750 007300 005021

2\$: CLR (R1)+

;CLEAR CONTENTS & INCREMENT

5751 007302 005737 006216

TST TRACE

;EOB FLAG SET?

5752 007306 001001

BNE 3\$

;GO TO NEXT BANK IF SO

5753 007310 000765

BR 1\$

5754 007312 062737 000200 172350

3\$: ADD #200,KIPAR4

;SET MAP FOR NEXT BANK

5755 007320 022737 177600 172350

CMP #177600,KIPAR4

;ARE WE AT THE PERIPHERAL PAGE

5756 007326 001405

BEQ CLREX

;YES-GO ON

5757 007330 005037 006216

CLR TRACE

;RESET FLAG

5758 007334 012701 100000

MOV #100000,R1

;RESET R1

5759 007340 000751

BR 1\$

;CLEAR NEXT BANK

5760 007342 000240

CLREX: NOP

5761 007344 005037 006216

CLR TRACE

5762 007350

RES4

007350 012737 034114 000004

MOV #TIMEOUT,4

007356 022737 000005 004010

CMP #5,PROTYP

;IS THIS AN 11/83/84 ?

007364 001002

BNE 101\$

;BRANCH IF NOT

007366 005037 177766

101\$: CLR CPUERR

;CLEAR OUT THE CPU ERROR REGISTER BITS

007372

;THAT A EXPECTED TRAP COULD HAVE SET

.DSABL CRF

5765 007372

ANA2: SUBTST <<MATCH ALL CSR'S WITH MEMORY>>

;*****
;SUBTEST MATCH ALL CSR'S WITH MEMORY;*****
;* THE SECOND PART OF THE ANALYSIS MATCHES UP THE CSR'S WITH THE MEMORY, AND
;* INSTALLS ALL THE INFORMATION FOUND IN THE CONFIGURATION TABLE. FOR ECC,
;* THIS IS DONE BY TAKING EACH CSR FOUND IN THE PREVIOUS SECTION SEQUENTIALLY
;* AND CHECKING THROUGH ALL OF MEMORY, ONE BANK AT A TIME, TO SEE WHICH BANKS
;* RESPOND TO THE CSR IN QUESTION. THE FIRST DOUBLE WORD PAIR IN EACH BANK IS
;* WRITTEN WITH DATA AND DIAGNOSTIC CHECK MODE SET IN THE CSR AND ARE CHECKED
;* FOR EACH BANK THROUGH USE OF TESTADD AND KERNEL INSTRUCTION PAGE ADDRESS
;* REGISTERS 4 AND 5. IF WE GET THE PROPER CHECKBITS BACK, WE HAVE A MATCH.
;* IF NOT, THE ROUTINE CHECKS FOR SINGLE OR DOUBLE BIT ERRORS.
;* IF ONE OR THE OTHER IS FOUND, THE ERROR ADDRESS IS CHECKED
;* TO SEE IF IT IS THAT BANK. IF IT IS, WE HAVE A MATCH. AT THE END OF EACH
;* BANK PASS, FOR EACH CSR PASS, THE PROGRAM COMES UP WITH A NUMBER, STORED IN
;* "I", WHICH DENOTES THE FOLLOWING:;*
;* I MEMORY DESCRIPTION
;* -
;* 0 NON-EXISTANT MEMORY
;* 1 MSV11-L/P MEMORY
;* 2 MSV11-J MEMORY
;*;* NOTE THAT PARITY MEMORY WRITES WRONG PARITY TO THE DOUBLE WORDS, THEN LOOKS
;* FOR THE PARITY ERROR BIT TO BE SET. IF THE BIT IS SET, WE HAVE A MATCH.
;*

5790 007372 012737 010444 000004

SET4 #100\$;NE MEMORY TRAPS GO TO 100\$

MOV #100\$,4

.DSABL CRF

CLR CHECK

MOV #TESTADD,R1

MOV TOTCSRS,R3

CLR R0

CLR R5

TST NO22BIT

BEQ 7\$

CLR LASTBLOCK

BR 1\$

CMP #177,LASTBANK

BEQ 1\$

MOV LASTBANK,R2

INC R2

ASH #9,R2

MOV R2,LASTBLOCK

MOV #4,R2

CLR ENDFLG

MOV #1000,KIPAR4

MOV #1000,KIPAR5

ASL R3

BCS 3\$

ADD #2,R0

MOV R0,CSRNO

TST R3

BNE 2\$

MOV #1000,KIPAR4

;CLEAR CHECK

;SET UP THE VIRTUAL ADDR. POINTER

;MOVE CSR MAP INTO R3

;CLEAR THE CSR POINTER

;CLEAR THE PROGRAM CSR STATUS POINTER

;DO WE HAVE 22 BIT ADDRESSING ?

;BRANCH IF WE DO

;ADJUST LASTBLOCK INDICATOR FOR 124K MACHINE

;BRANCH OVER NEXT PIECE OF CODE

;IS THERE Q-BUS MEMORY ABOVE 17776000?

;BRANCH IF NOT

;SET UP A NEW LAST BLOCK INDICATOR

;R2 IS INDEX FOR CONFIG TABLE

;CLEAR END OF MEMORY FLAG

;SET KIPAR4 FOR BANK 1

;SET KIPAR5 FOR BANK 1

;DOES THIS CSR EXIST?

;BRANCH IF IT DOES EXIST

;INCREMENT THE CSR POINTER

;STORE IT IN CSRNO ALSO

;ARE THERE ANY MORE CSR'S TO DO?

;BRANCH IF ALL CSRS NOT DONE

;RESTORE KIPAR4

5791 007400 005037 002314

5792 007404 012701 002412

5793 007410 013703 002224

5794 007414 005000

5795 007416 005005

5796 007420 005737 002454

5797 007424 001403

5798 007426 005037 002560

5799 007432 000413

5800 007434 022737 000177 002556 7\$:

5801 007442 001407

5802 007444 013702 002556

5803 007450 005202

5804 007452 072227 000011

5805 007456 010237 002560

5806 007462 012702 000004 1\$:

5807 007466 005037 002564

5808 007472 012737 001000 172350

5809 007500 012737 001000 172352

5810 007506 006303 2\$:

5811 007510 103420

5812 007512 062700 000002

5813 007516 010037 002152

5814 007522 005703

5815 007524 001370

5816 007526 012737 001000 172350

5817	007534	012737	001200	172352	MOV	#1200,KIPAR5	;RESTORE KIPAR5
5818	007542	013706	002574		MOV	KSTACK,SP	;RESTORE STACK
5819	007546	000137	010454		JMP	SUBAAS	;JUMP TO SUBAAS IF ALL CSR'S ARE DONE
5820	007552	010037	002152	3\$:	MOV	R0,CSRNO	;MAKE SURE CSRNO IS UPDATED
5821	007556	104424		13\$:	CACHOFF		;TURN THE CACHE OFF
5822	007560	000240			NOP		
5823	007562	012737	100000	002412	45\$:	MOV	#100000,TESTADD
5824	007570	012737	120002	002414		MOV	#120002,TESTADD+2
5825	007576	032762	000040	002664		BIT	#BIT5,CONFIG(R2)
5826	007604	001402			BEQ	43\$;IS THIS A BANK TO SKIP ECC/LOGIC TESTS?
5827	007606	000137	010342		JMP	6\$;NO - BRANCH AROUND NEXT INSTRUCTION
5828	007612	005037	002452		43\$:	CLR	I
5829	007616	005771	000000		4\$:	TST	@(R1)
5830	007622	005237	002452			INC	I
5831	007626				PUSH	@(R1),@2(R1)	;MEMORY PRESENT
	007626	017146	000000				;SAVE THE LOCATIONS UNDER TEST
	007632	017146	000002				MOV @2(R1),-(SP)
5832	007636	032760	000001	002462	BIT	#BIT0,CSRINFO(R0)	;IS THIS PARITY MEMORY?
5833	007644	001014			BNE	34\$;NO - BRANCH
5834	007646	052760	000004	172100	BIS	#BIT2,CSRADD(R0)	;SET WRITE WRONG PARITY
5835	007654	012771	123456	000000	MOV	#123456,@(R1)	;SET THE FIRST LOCATION UNDER TEST
5836	007662	012771	123456	000002	MOV	#123456,@2(R1)	;SET THE SECOND LUT
5837	007670	005060	172100		CLR	CSRADD(R0)	;CLEAR THE CSR
5838	007674	000411			BR	41\$;TEST LOCATIONS
5839	007676	012771	123456	000000	34\$:	MOV	#123456,@(R1)
5840	007704	012771	123456	000002		MOV	#123456,@2(R1)
5841	007712	104503			CLR1CSR		;SET THE SECOND LUT
5842	007714	104475			CB1CSR		;RESET CSR
5843	007716	000240			NOP		;SET DIAG. CHECK MODE IN CSR UNDER TEST
5844	007720	005771	000000		41\$:	TST	@(R1)
5845	007724	104426			READCSR		;DEBUG AID
5846	007726	000240			NOP		;READ THE FIRST LUT TO WRITE CKBITS. INTO CSR
5847	007730	013704	002150		MOV	CSR,R4	;READ THE CSR UNDER TEST
5848	007734	000240			NOP		;DEBUG AID
5849	007736	010437	002434		MOV	R4,TEMP	;GET THE CHECKBITS FROM THE CSR
5850	007742	104503			CLR1CSR		;DEBUG AID
5851	007744				POP	@2(R1),@2(R1)	;SAVE IN TEMP FOR LATER
	007744	012671	000002				;RESET CSR
	007750	012671	000000				;RESTORE LOCATIONS UNDER TEST
5852	007754	032760	000001	002462	BIT	#BIT0,CSRINFO(R0)	MOV (SP)+,@2(R1)
5853	007762	001004			BNE	42\$	MOV (SP)+,@2(R1)
5854	007764	005704			TST	R4	;IS THIS PARITY MEMORY?
5855	007766	100420			BMI	25\$;NO - BRANCH
5856	007770	000137	010342		JMP	6\$;DID WE GET A PARITY ERROR?
5857	007774	072427	177773		42\$:	ASH	#-5,R4
5858	010000	042704	177700		BIC	#1C77,R4	;YES - FILL IN CONFIG TABLE
5859	010004	012737	000040	002316	MOV	#40,CBITS	;NO - JUMP TO END OF BANK
5860	010012	000240			NOP		;MANIPULATE THE CSR BITS
5861	010014	023704	002316		77\$:	CMP	CBITS,R4
5862	010020	000240			NOP		;INTO A USABLE FORM.
5863	010022	001402			BEQ	25\$;MSV11-J CHECK BITS
5864	010024	000137	010170		JMP	22\$;DEBBUGING AIDE
5865					*		;DO THE CHECKBITS COMPARE TO WHAT WAS WRITTEN?
5866					*		;DEBBUG AIDE
5867					*		;BRANCH IF THERE IS A MATCH
5868	010030	010004			25\$:	MOV	R0,R4
5869	010032	000240			NOP		;ELSE BRANCH IF NOT THE SAME
							;GET THE CSR NUMBER

5870	010034	006204				ASR	R4	;SET IT UP FOR USE IN THE
5871	010036	000304				SWAB	R4	;CONFIGURATION TABLE.
5872	010040	042704	170377			BIC	#170377,R4	;CLEAR OFF EXTRANEIOUS BITS
5873	010044	050462	002664	15\$:		BIS	R4,CONFIG(R2)	;PUT CSR NUMBER IN CONFIG. TABLE
5874	010050	016004	002462			MOV	CSRINFO(R0),R4	;GET MEMORY TYPE
5875	010054	042704	177770			BIC	#1C7,R4	;CLEAR OFF THE EXTRANEIOUS BITS
5876	010060	000304				SWAB	R4	;MOVE INTO PROPER POSITION
5877	010062	050462	002666			BIS	R4,CONFIG+2(R2)	;SET IT INTO THE CONFIG TABLE
5878						;		
5879						;		
5880						;		
5881	010066	022737	001000	172350	24\$:	CMP	#1000,KIPAR4	;IS THIS BANK 1 ?
5882	010074	001402				BEQ	30\$;BRANCH IF TRUE
5883	010076	000137	010342			JMP	6\$;ELSE JUMP TO END OF THIS BANK
5884	010102	032737	100020	002434	30\$:	BIT	#BIT15!BIT4,TEMP	;WAS THERE A SBE OR DBE?
5885	010110	001417				BEQ	10\$;BRANCH IF NOT
5886	010112	013704	002434			MOV	TEMP,R4	;GET CSR CONTENTS
5887	010116	072427	177767			ASH	#-9.,R4	;MAKE ERROR ADDRESS INTO BANK #
5888	010122	022704	000001			CMP	#1,R4	;ERROR IN BANKS 0 OR 1?
5889	010126	003010				BGT	10\$;BRANCH IF NOT
5890	010130	052762	000001	002664		BIS	#BIT0,CONFIG(R2)	;SET ERROR FLAG IN CONFIG TABLE
5891	010136	105262	002666			INCB	CONFIG+2(R2)	;ADD ONE TO BANK ERROR COUNT
5892	010142					SET	CONFGERROR	;PRINT CONFIG TABLE
	010142	012737	177777	002450				MOV # 1,CONFGERROR
5893	010150	053737	002670	002664	10\$:	BIS	CONFIG+4,CONFIG	;SET UP INFORMATION IN BANK ZERO
5894	010156	053737	002672	002666		BIS	CONFIG+6,CONFIG+2	
5895	010164	000240				NOP		;DEBUG AID
5896	010166	000465				BR	6\$;BRANCH
5897						;		
5898						;		
5899						;		
5900	010170	032737	100020	002150	22\$:	BIT	#BIT15!BIT4,CSR	;SBE OR DBE FLAGS SET?
5901	010176	001001				BNE	8\$;BRANCH IF TRUE
5902	010200	000460				BR	6\$;CHECK TO SEE IF IT IS MSV11 J
5903	010202	013704	002152		8\$:	MOV	CSRNO,R4	;GET CSRNO
5904	010206	042764	000006	172100		BIC	#6,CSRADD(R4)	;TURN OFF DIAG CHECK & ECC DISABLE
5905	010214					PUSH	R0,R1	;SAVE R0 & R1
	010214	010046						MOV R0, (SP)
	010216	010146						MOV R1,-(SP)
5906	010220	016401	172100			MOV	CSRADD(R4),R1	;GET CSR INFORMATION
5907	010224	072127	177773			ASH	#-5,R1	;SET UP ERROR ADDRESS
5908	010230	042701	177600			BIC	#1C177,R1	
5909	010234	052764	040000	172100		BIS	#BIT14,CSRADD(R4)	;GET EXTENDED ERROR ADDRESS BITS
5910	010242	016400	172100			MOV	CSRADD(R4),R0	;READ FROM CSR
5911	010246	042764	040000	172100		BIC	#BIT14,CSRADD(R4)	;TURN OFF EQB BIT
5912	010254	042700	177037			BIC	#1C740,R0	;SET UP EXTENDED BITS
5913	010260	006300				ASL	R0	
5914	010262	006300				ASL	R0	
5915	010264	060001				ADD	R0,R1	;SET UP TOTAL ERROR ADDRESS
5916	010266	010104			27\$:	MOV	R1,R4	;SAVE IN R4
5917	010270					POP	R1,R0	;RESTORE R0 & R1
	010270	012601						MOV (SP)+,R1
	010272	012600						MOV (SP)+,R0
5918	010274	072427	000005			ASH	#5,R4	;SET ERROR ADDRESS UP IN PAR NOTATION
5919	010300	020437	172350			CMP	R4,KIPAR4	;DOES IT EQUAL KIPAR4?
5920	010304	001001				BNE	28\$;BRANCH IF FALSE
5921	010306	000403				BR	35\$;YES - MARK INFO IN CONFIG TABLE

5922	010310	020437	172352	28\$:	CMP	R4,KIPAR5	;DOES IT EQUAL KIPAR5?
5923	010314	001012			BNE	6\$;BRANCH IF FALSE
5924	010316	052762	000001	002664	35\$:	BIS	#BIT0,CONFIG(R2)
5925	010324	105262	002666		INCB	CONFIG+2(R2)	;SET BANK ERROR FLAG
5926	010330				SET	CONFGERROR	;INCREMENT BANK ERROR COUNTER
	010330	012737	177777	002450			;PRINT CONFIG TABLE
5927	010336	000137	010030		JMP	25\$	MOV #-1,CONFGERROR
5928					;		;YES MARK INFO IN CONFIG TABLE
5929					;		
5930					;		
5931	010342	104503		6\$:	CLR1CSR		;CLEAR THE CSR UNDER TEST
5932	010344	005737	002564		TST	ENDFLG	;ARE WE AT TOP OF MEMORY?
5933	010350	001021			BNE	70\$;IF SO THEN EXIT
5934	010352	062702	000004		ADD	#4,R2	;UPDATE CONFIGURATION POINTER
5935	010356	062737	001000	172350	ADD	#1000,KIPAR4	;UPDATE KIPAR4 TO NEXT BANK
5936	010364	013737	172350	172352	MOV	KIPAR4,KIPAR5	;AND UPDATE KIPAR4
5937	010372	022737	177000	172350	CMP	#177000,KIPAR4	;ARE WE AT BANK 177
5938	010400	001005			BNE	70\$;BRANCH IF NOT
5939	010402				SET	ENDFLG	;WERE AT LAST BANK
	010402	012737	177777	002564			MOV #-1,ENDFLG
5940	010410	000137	007562		JMP	45\$	
5941	010414	023737	002560	172350	70\$:	CMP	LASTBLOCK,KIPAR4
5942	010422	101402			BLOS	19\$;HAVE WE DONE THE WHOLE MEMORY SPACE?
5943	010424	000137	007562		JMP	45\$;BRANCH IF DONE ;R-C
5944	010430	062700	000002	19\$:	ADD	#2,R0	;JUMP IF NOT DONE
5945	010434	000240			NOP		;INCREMENT CSR POINTER
5946	010436	104423			CACHON		;DEBUG AID
5947	010440	000137	007462		JMP	1\$;TURN ON THE CACHE
5948							;JUMP TO TRY NEXT CSR
5949	010444	062706	000004	100\$:	ADD	#4,SP	;RESTORE STACK ;R-C
5950	010450	000137	010342		JMP	6\$;GO TO END OF BANK ROUTINE ;R-C

5952 010454 104423
5953 010456 104472
5954 010460

SUBAAS: CACHON ;MAKE SURE THE CACHE IS ON
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
NEWST <<TEST BANK 0 ACCESSES>>

;TEST 2 TEST BANK 0 ACCESSES

010460 000004

5955
5956
5957
5958
5959
5960
5961 010462 005037 002072
5962 010466 012737 000001 002076
5963 010474 005037 002070
5964 010500 012737 000001 002100
5965 010506
010506 012737 034050 000004

TST2: SCOPE
;THIS DOES A "TST" INSTRUCTION ON EVERY LOCATION IN BANK #0 TO SEE
;IF IT GETS ANY PARITY TRAPS.
;SINCE EVERY LOCATION IS EITHER LOADED OR WRITTEN INTO BY THE PROGRAM
;PRIOR TO THIS POINT - THEN A PARITY ERROR IMPLIES THAT THERE IS A
;HARDWARE FAILURE IN THE MEMORY.
;THESE ERRORS ARE COUNTED AND A FATAL ACTION IS TAKEN
CLR PARCNT ;CLEAR PARITY ERROR COUNTER
MOV #1,NOPAR ;SET THE NO PARITY ERROR FLAG
CLR NEMCNT ;CLEAR NON-EXISTANT MEMORY ERROR COUNTER
MOV #1,NONEM ;SET THE NON-EXISTANT MEMORY ERROR MODE TO COUNT
SET4 #NONEXIST ;TRAPS TO 4 GOTO NONEXIST
MOV #NONEXIST,4
.DSABL CRF
CLR RO
MOV #SIZE,R1
CACHOFF ;TURN CACHE OFF
1\$: TST (R0)+ ;SEE IF I CAN DO A READ ACCESS WITHOUT A PARITY TRAP
SOB R1,1\$
CACHON ;TURN CACHE ON
;SEE IF ANY FAILURES
TST PARCNT ;ANY PARITY ERRORS?
BEQ 2\$;NO - SKIP
FATAL 3
INC FATAL\$;SET FATAL INDICATOR
ERROR +3
.DSABL CRF
2\$: TST NEMCNT ;ANY NON EXISTANT MEMORY (HOLES)?
BEQ 3\$;SKIP IF EQUAL
SUB #2,ADDRESS ;UPDATE 1ST ADDRESS FAILURE FROM AUTO INCREMENT #
FATAL 4
INC FATAL\$;SET FATAL INDICATOR
ERROR +4
.DSABL CRF
3\$: BIS CPUBIT,CONFIG ;SET CORRECT ACCESSED BIT ON BANK 0
RES4 ;RESET TRAPS TO 4 TO DEFAULT
MOV #TIMEOUT,4
CMP #5,PROTYP ;IS THIS AN 11/83/84 ?
BNE 101\$;BRANCH IF NOT
CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
101\$: ;THAT A EXPECTED TRAP COULD HAVE SET
.DSABL CRF

5982
5983 010620

SUBTST <<ENABLE ECC FOR CORRECT TRAPS>>

;SUBTEST ENABLE ECC FOR CORRECT TRAPS

IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE

5984 010620
010620 032777 000001 172010
010626 001003

BIT #SWO,@SWR
BNE L52

CVMJAAO MSV11-J MEMORY DIAG.
ENABLE ECC FOR CORRECT TRAPS

MACRO V05.01b Friday 15-Feb 85 16:10 Page 154-1

SEQ 0151

```
010630 005737 002350
010634 001402
010636
5985 010636 104506
5986 010640
010640 000401
010642
5987 010642 104472
5988 010644
010644

ENASBE
ELSE

ECCINIT
END ;OF IF #SWO

TST ACTFLAG
BEQ L53
L52:;;;;;;
;TRAP ON SINGLE BIT ERRORS
BR L54
L53:;;;;;;
;TRAP ON DOUBLE BIT ERRORS (NORMAL)
L54:;;;;;;
```

5991 010644

010644 000004

5992
5993
5994
5995
5996
5997
5998
5999 010646 005037 002102
6000 010652 012737 000001 002076
6001 010660 012737 000002 002100
6002 010666 012737 034050 000004

6003 010674 012737 011420 002522
6004 010702 012737 011422 002524
6005 010710 005237 002102
6006 010714 023737 002556 002102
6007 010722 103451
6008 010724 013701 002102
6009 010730 006301
6010 010732 006301
6011 010734 010137 002104
6012 010740 005037 002074
6013 010744 005037 002072
6014 010750 005037 002070
6015 010754
010754 010346
010756 013703 002102
010762 004737 035716

010766 012603
6016 010770 105761 002664
6017 010774 100542
6018 010776 012777 000207 171516 WARN1:
6019 011004 012700 060000
6020 011010 010004
6021 011012 012701 040000
6022 011016 010103
6023 011020 005002
6024 011022 104424
6025 011024
011024 053737 002552 177776

6026 011032 004737 011414
6027 011036 104417
6028 011040 104423
6029 011042 000240
6030 011044 000416
6031 011046 005037 002102
6032 011052
011052 012737 034114 000004
011060 022737 000005 004010

```
NEWST <<TEST BANKS 1 200 (OCTAL) FOR ZEROS & ONES>>
;*****
;*TEST 3      TEST BANKS 1 200 (OCTAL) FOR ZEROS & ONES
;*****
TST3:  SCOPE
      ;EACH BANK IS TESTED FOR EXISTANCE AND IF IT EXISTS
      ;THEN IT IS TESTED FOR ZEROS & ONES.
      ;EXCEPT -
      ;      PROTECTED BANKS (WHERE THE PROGRAM IS) ARE ONLY TESTED BY
      ;      "TST" INSTRUCTIONS LIKE BANK #0
      ;ANY BAD BANKS ARE LOGGED IN THE CONFIGURATION TABLE.
      ;THIS ROUTINE IS ONLY DOING A SMART SIZE  NOT ACTUAL TESTING!
      CLR      BANK
      MOV      #1,NOPAR          ;SET NO PARITY ERROR FLAG
      MOV      #2,NONEM         ;SET NON-EXISTANT MEMORY MODE TO EXIT TEST LOOP
      SET4     #NONEXIST        ;TRAPS TO 4 GOTO NONEXIST
      MOV      #NONEXIST,4
      .DSABL   CRF
      MOV      #MTST3+4,LINK1    ;SET UP LINKS
      MOV      #MTST3+6,LINK2
TAG9$: INC      BANK
      CMP      LASTBANK,BANK    ;DONE?
      BLO      TAG2$           ;YES - SKIP TO NEXT TEST
      MOV      BANK,R1
      ASL      R1
      ASL      R1               ;BANK * 4
      MOV      R1,BANKINDEX
      CLR      PATERR          ;CLEAR PATTERN ERROR COUNTER
      CLR      PARCNT          ;CLEAR PARITY ERROR COUNTER
      CLR      NEMCNT          ;CLEAR NON-EXISTANT MEMORY COUNTER (HOLES)
      MAP      BANK            ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
                                MOV R3, (SP)
                                MOV (SP)+,R3
      TSTB     CONFIG(R1)      ;IS THIS BANK PROTECTED?
      BMI      TSTBANK         ;YES - GO TEST BANK SPECIAL
      MOV      #207,@LINK1     ;PUT "RETURN" INSTRUCTION AFTER WRITE ROUTINE
      MOV      #FIRST,R0
      MOV      R0,R4
      MOV      #SIZE,R1
      MOV      R1,R3
      CLR      R2
      CACHOFF
      TESTAREA
      BIS      TESTMODE,PSW    ;ENTER SUPERVISOR MODE
                                ;GO TO SYSTEM TEST MODE
      .DSABL   CRF
      CALL     MTST3
      2$:      KERNEL
      CACHON
      NOP
      BR       TAG3$           ;SKIP NEXT INSTRUCTION
TAG2$: CLR      BANK
      RES4
      MOV      #TIMEOUT,4      ;RESET TRAPS TO 4 TO DEFAULT
      CMP      #5,PROTP
                                ;IS THIS AN 11/83/84 ?
```

PC	OP	OP2	OP3	OP4	OP5	OP6	OP7	OP8	OP9	OP10	OP11	OP12	OP13	OP14	OP15	OP16	OP17	OP18	OP19	OP20	OP21	OP22	OP23	OP24	OP25	OP26	OP27	OP28	OP29	OP30	OP31	OP32	OP33	OP34	OP35	OP36	OP37	OP38	OP39	OP40	OP41	OP42	OP43	OP44	OP45	OP46	OP47	OP48	OP49	OP50	OP51	OP52	OP53	OP54	OP55	OP56	OP57	OP58	OP59	OP60	OP61	OP62	OP63	OP64	OP65	OP66	OP67	OP68	OP69	OP70	OP71	OP72	OP73	OP74	OP75	OP76	OP77	OP78	OP79	OP80	OP81	OP82	OP83	OP84	OP85	OP86	OP87	OP88	OP89	OP90	OP91	OP92	OP93	OP94	OP95	OP96	OP97	OP98	OP99	OP100	OP101	OP102	OP103	OP104	OP105	OP106	OP107	OP108	OP109	OP110	OP111	OP112	OP113	OP114	OP115	OP116	OP117	OP118	OP119	OP120	OP121	OP122	OP123	OP124	OP125	OP126	OP127	OP128	OP129	OP130	OP131	OP132	OP133	OP134	OP135	OP136	OP137	OP138	OP139	OP140	OP141	OP142	OP143	OP144	OP145	OP146	OP147	OP148	OP149	OP150	OP151	OP152	OP153	OP154	OP155	OP156	OP157	OP158	OP159	OP160	OP161	OP162	OP163	OP164	OP165	OP166	OP167	OP168	OP169	OP170	OP171	OP172	OP173	OP174	OP175	OP176	OP177	OP178	OP179	OP180	OP181	OP182	OP183	OP184	OP185	OP186	OP187	OP188	OP189	OP190	OP191	OP192	OP193	OP194	OP195	OP196	OP197	OP198	OP199	OP200	OP201	OP202	OP203	OP204	OP205	OP206	OP207	OP208	OP209	OP210	OP211	OP212	OP213	OP214	OP215	OP216	OP217	OP218	OP219	OP220	OP221	OP222	OP223	OP224	OP225	OP226	OP227	OP228	OP229	OP230	OP231	OP232	OP233	OP234	OP235	OP236	OP237	OP238	OP239	OP240	OP241	OP242	OP243	OP244	OP245	OP246	OP247	OP248	OP249	OP250	OP251	OP252	OP253	OP254	OP255	OP256	OP257	OP258	OP259	OP260	OP261	OP262	OP263	OP264	OP265	OP266	OP267	OP268	OP269	OP270	OP271	OP272	OP273	OP274	OP275	OP276	OP277	OP278	OP279	OP280	OP281	OP282	OP283	OP284	OP285	OP286	OP287	OP288	OP289	OP290	OP291	OP292	OP293	OP294	OP295	OP296	OP297	OP298	OP299	OP300	OP301	OP302	OP303	OP304	OP305	OP306	OP307	OP308	OP309	OP310	OP311	OP312	OP313	OP314	OP315	OP316	OP317	OP318	OP319	OP320	OP321	OP322	OP323	OP324	OP325	OP326	OP327	OP328	OP329	OP330	OP331	OP332	OP333	OP334	OP335	OP336	OP337	OP338	OP339	OP340	OP341	OP342	OP343	OP344	OP345	OP346	OP347	OP348	OP349	OP350	OP351	OP352	OP353	OP354	OP355	OP356	OP357	OP358	OP359	OP360	OP361	OP362	OP363	OP364	OP365	OP366	OP367	OP368	OP369	OP370	OP371	OP372	OP373	OP374	OP375	OP376	OP377	OP378	OP379	OP380	OP381	OP382	OP383	OP384	OP385	OP386	OP387	OP388	OP389	OP390	OP391	OP392	OP393	OP394	OP395	OP396	OP397	OP398	OP399	OP400	OP401	OP402	OP403	OP404	OP405	OP406	OP407	OP408	OP409	OP410	OP411	OP412	OP413	OP414	OP415	OP416	OP417	OP418	OP419	OP420	OP421	OP422	OP423	OP424	OP425	OP426	OP427	OP428	OP429	OP430	OP431	OP432	OP433	OP434	OP435	OP436	OP437	OP438	OP439	OP440	OP441	OP442	OP443	OP444	OP445	OP446	OP447	OP448	OP449	OP450	OP451	OP452	OP453	OP454	OP455	OP456	OP457	OP458	OP459	OP460	OP461	OP462	OP463	OP464	OP465	
----	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--

```

6079 011322 104424          CACHOFF          ;TURN CACHE OFF
6080 011324          TESTAREA          ;ENTER TEST MODE
      011324 053737 002552 177776      BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
      .DSABL CRF
6081 011332 005720          4$: TST (R0)+
6082 011334 077102          SOB R1,4$
6083 011336 104417          KERNEL
6084 011340 104423          CACHON          ;ENTER KERNEL MODE
6085 011342 012737 000002 002100      MOV #2,NONEM          ;TURN CACHE ON
6086 011350          POP R1          ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
      011350 012601          MOV (SP)+,R1
6087 011352          IF PARCNT NE #0          TST PARCNT
      011352 005737 002072          BEQ L55
      011356 001406          ;ERROR BANK
6088 011360 052761 000001 002664      BIS #BIT0,CONFIG(R1)
6089 011366          SET CFGERR          MOV #-1,CFGERR
      011366 012737 177777 002450      END ;OF IF PARCNT
6090 011374          IF NEMCNT EQ #0          L55:;;;;;
      011374          TST NEMCNT
6091 011374          005737 002070          BNE L56
      011400 001003
6092 011402 053761 002106 002664      BIS CPUBIT,CONFIG(R1)          ;ACCESSED BANK
6093 011410          END ;OF IF NEMCNT          L56:;;;;;
      011410
6094 011410 000137 010710          JMP TAG9$
6095 011414 010220          MTST3: MOV R2,(R0)+          ;WRITE MEMORY TO PATTERN
6096 011416 077102          SOB R1,MTST3          ;DO TILL DONE
6097 011420 000240          NOP          ;THIS IS EITHER A NOP OR RETURN
6098 011422 012401          2$: MOV (R4)+,R1          ;READ DATA
6099 011424 020102          CMP R1,R2          ;DOES IT MATCH ?
6100 011426 001402          BEQ 3$          ;
6101 011430 005237 002074          INC PATERR          ;IF NOT COUNT ERRORS
6102 011434 077306          3$: SOB R3,2$          ;ANY WAY DO ALL MEMORY
6103 011436 000207          RETURN          ;DONE....RETURN

```

6105 011440

SUBAAI: SUBTST <<FIND SHADOW INHIBIT MODE POINTERS>>

;*****

;*SUBTEST FIND SHADOW INHIBIT MODE POINTERS

;*****

;* THIS SECTION LOOKS FOR INTERLEAVED MSV11-J MEMORIES AND FIGURES OUT

;* WHERE THE SHADOW INHIBIT MODE POINTERS ARE LOCATED. THESE AREAS

;* ARE THEN MARKED AS PROGRAM SPACE.

6106

6107

6108

6109 011440 005037 002102

6110 011444 004737 040052

6111 011450 013700 002104

6112 011454

011454 005737 002116

011460 001414

011462 005737 002136

011466 001411

6113 011470 062702 000040

6114 011474 062737 000020 002102

6115 011502 052760 000200 002664

6116 011510

011510 000402

011512

6117 011512 005237 002102

6118 011516

011516

6119 011516 023737 002556 002102

6120 011524 002347

CLR BANK ;RESET BANK TO ZERO

SHADL1: CALL EXBANK ;SET BANK PARAMETERS

MOV BANKINDEX,R0

IF ACFLAG IS TRUE AND INTFLAG IS TRUE

TST ACFLAG

BEQ L57

TST INTFLAG

BEQ L57

ADD #40,R2

;POINT TO BANKINDEX + 8

ADD #20,BANK

;POINT TO BANK + 16

BIS #BIT7,CONFIG(R0)

;MAKE NEW BANK PROGRAM SPACE

ELSE

BR L60

L57:;;;;;

INC BANK

;GO TO NEXT BANK

END; OF IF ACFLAG

L60:;;;;;

CMP LASTBANK,BANK

;HAVE WE DONE ALL THE BANKS?

BGE SHADL1

;BRANCH IF NOT

6123 011526

NEWST <<ECC INHIBIT MODE POINTER TEST>>

;*****
;TEST 4 ECC INHIBIT MODE POINTER TEST
;*****

TST4: SCOPE

;THE MSV11-J OR MF11S-K INHIBIT ECC DISABLE AND DIAGNOSTIC CHECK MODE
;ON THE BOTTOM FIRST OR SECOND 16K WORDS CONTROLLED BY A CSR. THIS
;IS CONSIDERED TO BE A PROTECTED BANK BY THE PROGRAM. IT MAY BE
;QUITE COMPLEX TO DETERMINE ON A GIVEN SYSTEM CONFIGURATION WHICH
;BANKS CAN BE PROTECTED;

;SO

;THIS ROUTINE ATTEMPS TO CREATE A DOUBLE BIT ERROR IN ADDRESS 0 & 2
;OF EVERY ECC BANK. ECC HARDWARE WILL PREVENT THIS FROM HAPPENING
;IN PROTECTED BANKS WHICH SHOULD ALWAYS INCLUDE BANK ZERO - WHERE
;THE PROGRAM IS.

;

;WARNING:!!!!!!!!!!

; IN CASE OF HARDWARE FAILURE IT IS COMMON THAT A DOUBLE BIT ERROR
; WILL BE CREATED ON THE KERNEL STACK & "CRASH" THE DIAGNOSTIC
;DURING THIS ROUTINE. YOUR ONLY CLUE IS THAT YOU CAN GET AS FAR AS
;THIS ROUTINE BUT NOT PAST IT!

CACHOFF

;TURN CACHE OFF

MOV #1,OLDCSR

FOR BANK := #0 TO LASTBANK

CLR BANK

B6:; ; ; ; ;

;SET UP VIRT ADDR POINTER

MOV #FIRST,R1

CALL EXBANK

MOV BANKINDEX,R0

IF ACFLAG IS TRUE

TST ACFLAG

BEQ L61

IF MKFLAG IS TRUE

TST MKFLAG

BEQ L62

IF SKIPMK IS FALSE

TST SKIPMK

BNE L63

MOV #2,R3

;SET INDEX COUNTER

MOVB CONFIG+1(R0),R2

ASL R2

BIC #C36,R2

MOV R2,CSRNO

IF CSRNO NE OLDCSR

CMP CSRNO,OLDCSR

BEQ L64

MOV CSRNO,OLDCSR

IF PFLAG IS FALSE

TST PFLAG

BNE L65

BIS #BIT6,CONFIG(R0)

END; OF IF PFLAG

L65:; ; ; ; ;

CALL IMPTEST

END; OF IF CSRNO

L64:; ; ; ; ;

011526 000004

6124

6125

6126

6127

6128

6129

6130

6131

6132

6133

6134

6135

6136

6137

6138

6139

6140

6141 011530 104424

6142 011532 012737 177777 002156

6143 011540

011540 005037 002102

011544

6144 011544 012701 060000

6145 011550 004737 040052

6146 011554 013700 002104

6147 011560

011560 005737 002116

011564 001436

6148 011566

011566 005737 002120

011572 001433

6149 011574

011574 005737 002342

011600 001030

6150 011602 012703 000002

6151 011606 116002 002665

6152 011612 006302

6153 011614 042702 177741

6154 011620 010237 002152

6155 011624

011624 023737 002152 002156

011632 001413

6156 011634 013737 002152 002156

6157 011642

011642 005737 002122

011646 001003

6158 011650 052760 000100 002664

6159 011656

011656

6160 011656 004737 011750

6161 011662

011662

```

6162 011662          END; OF IF SKIPMK          L63:~~~~~
        011662
6163 011662          END; OF IF MKFLAG          L62:~~~~~
        011662
6164 011662          END; OF IF ACFLAG          L61:~~~~~
        011662
6165 011662          END; OF FOR BANK
        011662 005237 002102          INC BANK
        011666 023737 002102 002556          CMP BANK, LASTBANK
        011674 003723          BLE B6
        011676          E6:~~~~~
6166 011676          MAP          ;MAP TEST SPACE TO BANK 0
        011676 010346          MOV R3, -(SP)
        011700 012703 000200          MOV #200, R3
        011704 004737 035716          CALL MAPPER
          .DSABL CRF
          MOV (SP), R3
6167 011710 012603          CLR BANK
        011712 005037 002102          IF #SW0 SET. IN @SWR OR ACTFLAG IS TRUE
6168 011716 032777 000001 170712          BIT #SW0, @SWR
        011724 001003          BNE L66
        011726 005737 002350          TST ACTFLAG
        011732 001402          BEQ L67
        011734          L66:~~~~~
6169 011734 104506          ENASBE          ;TRAP ON SINGLE BIT ERRORS
6170 011736          ELSE
        011736 000401          BR L70
        011740          L67:~~~~~
6171 011740 104472          ECCINIT          ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
6172 011742          END; OF IF #SW0
        011742          L70:~~~~~
6173 011742 104423          CACHON          ;TURN THE CACHE BACK ON
6174 011744 000137 012172          JMP SUBAAR          ;JUMP OVER THE SUBROUTINE

```

```

6176 011750 005004      IMPTEST:CLR      R4
6177 011752      MAP BANK                      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      011752 010346      MOV      BANK,R3      MOV R3, (SP)
      011754 013703 002102  CALL      MAPPER
      011760 004737 035716  .DSABL CRF

      011764 012603      MOV (SP),R3
6178 011766 005005      CLR      R5
6179 011770 012737 020000 002150  MOV      #BIT13,CSR
6180 011776      TESTAREA                      ;ENTER TEST MODE
      011776 053737 002552 177776  BIS      TESTMODE,PSW      ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      PUSH      (R1)                      ;SAVE TEST LOCATION
      MOV (R1),-(SP)
6181 012004      ADD      R3,R1              ;INDEX TO NEXT LOCATION
      012004 011146      PUSH      (R1)      ;SAVE TEST LOCATION
6182 012006 060301      MOV (R1),-(SP)
6183 012010      CHK1DIS                      ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
      012010 011146      MOV      R4,(R1)      ;WRITE CHECKBITS (ALL ZEROS)
6184 012012 104505      SUB      R3,R1
6185 012014 010411      MOV      R4,(R1)
6186 012016 160301      CLR1CSR
6187 012020 010411      TST      (R1)
6188 012022 104503      WAS1DBE
6189 012024 005711      ;CLEAR CSR
6190 012026 104501      ;READ CHECKBITS INTO REAL CSR
6191      ;WAS THERE A DOUBLE BIT ERROR
6192      ;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
6193
6194 012030      ON.NOERROR ;1
      012030 103435      BCS L71
6195 012032 012737 020000 002150  MOV      #BIT13,CSR
6196 012040 104505      CHK1DIS
6197 012042 013711 002614  MOV      ONES,(R1)
6198 012046 104503      ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6199 012050 005711      CLR1CSR
6200 012052 104501      ;CLEAR CSR
6201 012054      TST      (R1)
6202 012056 103423      WAS1DBE
6203 012060 012737 023140 002150  ON.NOERROR ;2
6204 012066 104505      CBREG
6205 012070 010411      MOV      #23140,CSR
6206 012072 104503      CHK1DIS
6207 012074 005711      MOV      R4,(R1)
6208 012076 104501      CLR1CSR
6209 012100      TST      (R1)
6210 012102 103411      WAS1DBE
6211 012104 012737 023604 002150  ON.NOERROR ;3
6212 012112 104505      CBREG
6213 012114 010411      MOV      #23604,CSR
6214 012116 104503      CHK1DIS
6215 012120 005711      MOV      R4,(R1)
6216 012122 104501      CLR1CSR
6217 012124      TST      (R1)
6218 012124      WAS1DBE
      012124      END ;OF ON.NOERROR ;3
      012124      END ;OF ON.NOERROR ;2

```

L73::::

L72::::


```

6219 012124      END ;OF ON.NOERROR ;1
      012124
6220 012124      ON.ERROR
      012124 103001
6221 012126      INC R5
      005205      ;IDENTIFY AS BAD BANK
6222 012130      END ;OF ON.ERROR
      012130
      L71:;;;;;
      BCC L74
6223 012130 104471 ECC1DIS
      012132 010411 MOV R4,(R1)
      060301 ADD R3,R1
      010411 MOV R4,(R1)
      104503 CLR1CSR
      005705 TST R5
      001405 BEQ 1$
      050560 BIS R5,CONFIG(R0)
      105260 INCB CONFIG+2(R0)
      104036 ERROR +36
      1$: POP (R1)
      ;RESTORE TEST LOCATION (2ND WORD)
      MOV (SP)+,(R1)
6234 012160 012611 SUB R3,R1
      160301 POP (R1)
      ;GO BACK TO FIRST WORD
      ;RESTORE TEST LOCATION (1ST WORD)
      MOV (SP)+,(R1)
6235 012164 012611
      012164 012611
6236 012166 104417
      012170 000207
      KERNEL
      RETURN
6238
6239 012172      SUBAAR: SET STOPOK
      012172 012737 177777 002420
      ;PROGRAM CAN NOW BE HALTED
      MOV #-1,STOPOK

```

6242 012200

SUBTST <<LEGAL CONFIGURATION CHECK>>

```

*****
;SUBTEST      LEGAL CONFIGURATION CHECK
*****

```

6243 012200 012700 000020

6244 012204 012701 002462

6245 012210 005021

6246 012212 077002

6247 012214

012214 005037 002102

012220

6248 012220 004737 040052

6249 012224 013700 002104

6250

6251 012230

012230 005737 002116

012234 001444

6252 012236 116003 002665

6253 012242 042703 177760

6254 012246 006303

6255 012250 005263 002462

6256 012254

012254 005737 002120

012260 001432

6257

6258 012262

012262

6259 012262

012262 005737 002136

012266 001423

6260 012270 116003 002665

6261 012274 010304

6262 012276 042703 177760

6263 012302 072427 177774

6264 012306 042704 177760

6265 012312

012312 020304

012314 001007

6266 012316 042760 014000 002666

6267 012324 042760 170000 002664

6268 012332

012332 000405

6269 012334

012334

6270 012334

012334 000401

012336

6271 012336

012336 000403

6272 012340

012340

6273 012340

012340 012737 177777 002450

6274 012346

012346

6275 012346

012346

```

MOV      #16.,R0
MOV      #CSRINFO,R1
1$:      CLR      (R1)+
        SOB      R0,1$
        FOR BANK := #0 TO LASTBANK

```

```

CALL      EXBANK
MOV      BANKINDEX,R0

```

IF ACFLAG IS TRUE

```

MOVB      CONFIG+1(R0),R3
BIC      #+C17,R3
ASL      R3
INC      CSRINFO(R3)
IF MKFLAG IS TRUE

```

```

;MAKE SURE THAT EACH BANK HAS NO MORE THAN 2 CSRS
BEGIN      LEGALCSR

```

IF INTFLAG IS TRUE

```

MOVB      CONFIG+1(R0),R3
MOV      R3,R4
BIC      #+C17,R3
ASH      #-4,R4
BIC      #+C17,R4
IF R3 EQ R4

```

```

BIC      #BIT11!BIT12,CONFIG+2(R0)
BIC      #170000,CONFIG(R0)
LEAVE      LEGALCSR

```

END; OF IF R3

ELSE

LEAVE LEGALCSR

END; OF IF INTFLAG

SET CONFGERROR

END LEGALCSR

END ;OF IF MKFLAG

CLR BANK

B7:::~::~:

```

TST ACFLAG
BEQ L75

```

```

TST MKFLAG
BEQ L76

```

B10:::~::~:

```

TST INTFLAG
BEQ L77

```

```

CMP R3,R4
BNE L100

```

BR E10

L100:::~::~:

BR L101

L77:::~::~:

BR E10

L101:::~::~:

MOV #-1,CONFGERROR

E10:::~::~:

L76:::~::~:

```

6276 012346                                END ;OF IF ACFLAG
      012346
6277 012346                                END; OF FOR BANK
      012346 005237 002102
      012352 023737 002102 002556
      012360 003717
      012362
6278 012362                                PUSH    R5,R0
      012362 010546
      012364 010046
6279 012366 005000                                CLR     R0
6280 012370 005001                                CLR     R1
6281 012372 005005                                CLR     R5
6282 012374 005037 012602                                CLR     MBERR
6283 012400 022761 000177 002462 2$:          CMP     #177,CSRINFO(R1)
6284 012406 002043                                BGE     5$
6285 012410 022761 000010 002462            CMP     #10,CSRINFO(R1)
6286 012416 002003                                BGE     3$
6287 012420 004737 012724                                CALL    ILLCSR
6288 012424 000434                                BR       5$
6289 012426 016005 002664 3$:              MOV     CONFIG(R0),R5
6290 012432 032705 000002                                BIT     #BIT1,R5
6291 012436 001415                                BEQ     4$
6292 012440 042705 170377                                BIC     #C7400,R5
6293 012444 072527 177771                                ASH     #-7,R5
6294 012450 020501                                CMP     R5,R1
6295 012452 001007                                BNE     4$
6296 012454 032760 010000 002666            BIT     #BIT12,CONFIG+2(P0)
6297 012462 001003                                BNE     4$
6298 012464 012737 000001 012602            MOV     #1,MBERR
6299 012472 062700 000004 4$:              ADD     #4,R0
6300 012476 022700 000340                                CMP     #340,R0
6301 012502 001351                                BNE     3$
6302 012504 005737 012602                                TST     MBERR
6303 012510 001402                                BEQ     5$
6304 012512 004737 012724                                CALL    ILLCSR
6305 012516 005000 5$:                    CLR     R0
6306 012520 005037 012602                                CLR     MBERR
6307 012524 062701 000002                                ADD     #2,R1
6308 012530 022701 000040                                CMP     #40,R1
6309 012534 001321                                BNE     2$
6310 012536                                POP      R0,R5
      012536 012600
      012540 012605
6311 012542 005037 012602                                CLR     MBERR
6312 012546 012700 000774                                MOV     #774,R0
6313 012552 032760 000002 002664 6$:          BIT     #BIT1,CONFIG(R0)
6314 012560 001003                                BNE     7$
6315 012562 162700 000004                                SUB     #4,R0
6316 012566 000771                                BR       6$
6317 012570 006200 7$:                    ASR     R0
6318 012572 006200                                ASR     R0
6319 012574 010037 002556                                MOV     R0,LASTBANK
6320 012600 000402                                BR       SKUJ
6321 012602 000000                                MBERR:  .WORD 0
6322 012604 000000                                PHEBE:  .WORD 0
6323 012606 005000                                SKUJ:   CLR     R0

```

L75:::~::~

```

INC BANK
CMP BANK, LASTBANK
BLE B7

```

E7:::~::~

```

;SAVE CONTENTS OF R5, R0
MOV R5, -(SP)
MOV R0, -(SP)

```

;CLEAR REGISTERS

```

;CLEAR ERROR INDICATOR
;IS CURRENT CSR <= 177
;BRANCH IF SO
;IS CURRENT CSR < 10
;BRANCH IF SO
;CALL ERROR ROUTINE
;TRY NEXT CSR
;MOVE LOW WORD TO R5
;DOES MEMORY EXIST HERE?
;BRANCH IF NOT
;ISOLATE CSR NUMBER IN
;REGISTER 5
;IS IT THE CURRENT CSR?
;TRY NEXT WORD OF CONFIG IF NOT
;IS IT INTERLEAVED?
;BRANCH IF SO
;SET ERROR INDICATOR
;UPDATE CONFIG COUNTER
;CONFIG TABLE ALL DONE?
;BRANCH IF NOT
;ERRORS FOUND?
;TRY NEXT CSR IF NOT
;CALL ERROR ROUTINE
;REINITIALIZE CONFIG COUNTER
;CLEAR ERROR INDICATOR
;UPDATE CSR COUNTER
;ALL CSR'S DONE?
;BRANCH IF NOT
;RESTORE REGISTERS

```

```

MOV (SP)+, R0
MOV (SP)+, R5

```

;RESET ERROR INDICATOR

```

;INDEX TO TOP OF CONFIG TABLE ;R-C
;MEMORY PRESENT? ;R-C
;BRANCH IF SO ;R-C
;TRY NEXT LOWER ENTRY IN CONFIG TABLE ;R-C

```

```

;DIVIDE INDEX BY 4 TO GET BANK #;R-C
;STORE IN LASTBANK ;R-C

```

```

;SAVE SPACE FOR ERROR INDICATOR
;SAVE SPACE FOR ODD BOUNDARY INTERLEAVED INDICATOR
;CLEAR CONFIG COUNTER

```

6324	012610	005037	C12604		CLR	PHEBE	;CLEAR COUNTER
6325	012614	032760	000002	002664	1\$:	BIT	#BIT1,CONFIG(R0)
6326	012622	001431			BEQ	3\$;IS THERE MEMORY PRESENT?
6327	012624	032760	010000	002666		BEQ	;BRANCH IF NOT
6328	012632	001005			BIT	#BIT12,CONFIG+2(R0)	;IS IT INTERLEAVED?
6329	012634	005237	012604		BNE	2\$;BRANCH IF SO
6330	012640	062700	000004		INC	PHEBE	;INCREMENT COUNTER
6331	012644	000763			ADD	#4,R0	;INCREMENT CONFIG COUNTER
6332	012646	023727	012604	000010	BR	1\$;TRY NEXT BANK
6333	012654	001417			CMP	PHEBE,#10	;IS THE COUNTER EQUAL TO...
6334	012656	023727	012604	000030	BEQ	4\$;ONE OF THE SPECIAL VALUES.
6335	012664	001413			CMP	PHEBE,#30	;IF IT IS...
6336	012666	023727	012604	000050	BEQ	4\$;BRANCH TO 4\$
6337	012674	001407			CMP	PHEBE,#50	
6338	012676	023727	012604	000070	BEQ	4\$	
6339	012704	001403			CMP	PHEBE,#70	
6340	012706	005037	012604		BEQ	4\$	
6341	012712	000403			CLR	PHEBE	;CLEAR INDICATOR
6342	012714	012737	000001	012604	BR	5\$	
6343	012722	000421			MOV	#1,PHEBE	;SET INDICATOR
6344	012724	010102			BR	SUBAAP	;BRANCH TO NEXT SUBTEST
6345	012726	006202			ILLCSR: MOV	R1,R2	;R2 HAS CSR NUMBER
6346	012730	022702	000012		ASR	R2	;MAKE ACCEPTABLE FOR PRINTING
6347	012734	100002			CMP	#10.,R2	
6348	012736	062702	000007		BPL	1\$	
6349	012742	062702	000060		ADD	#7,R2	
6350	012746	110237	071210		ADD	#60,R2	
6351	012752			1\$:	MOVB	R2,MSGA122	;PUT NUMBER INTO ERROR MESSAGE
	012752	104401	071174		TYPE	MSG122	
					TYPEIT	,MSG122	
					.DSABL	CRF	
6352	012756				SET	CONFGERROR	
	012756	012737	177777	002450			MOV #-1,CONFGERROR
6353	012764	000207			RETURN		

6382	013132	005237	002416				INC UNITOP
	013136					END; OF IF CPUBIT	
6383	013136					END; OF FOR R1	L105:::~::~
	013136	062701	000004				ADD #4,R1
	013142	020127	000200				CMP #1, #200
	013146	003765					BLE B13
	013150						E13:::~::~
6384	013150	006337	002416			ASL UNITOP	
6385	013154	006337	002416			ASL UNITOP	
6386	013160	006337	002416			ASL UNITOP	
6387	013164	006337	002416			ASL UNITOP	
6388	013170					IF I LT UNITOP THEN LET I := UNITOP	
	013170	023737	002452	002416			CMP I,UNITOP
	013176	002003					BGE L106
	013200	013737	002416	002452			MOV UNITOP,I
	013206						L106:::~::~
6389	013206					TYPE \$CRLF	
	013206	104401	002660			TYPEIT , \$CRLF	
						.DSABL CRF	
6390	013212	005737	002400	2\$:		TST LSIZE	
6391	013216	001414				BEQ 3\$	
6392	013220	022737	004000	002400		CMP #2048.,LSIZE	
6393	013226	001003				BNE 12\$	
6394	013230	162737	000004	002400		SUB #4,LSIZE	; SUBTRACT 4K FOR THE I/O PAGE
6395	013236					TYPDEC LSIZE	
	013236	013746	002400	12\$:		MOV LSIZE, -(SP)	::SAVE LSIZE FOR TYPEOUT
	013242	104405				TYPDS	::GO TYPE--DECIMAL ASCII WITH SIGN
						.DSABL CRF	
6396	013244					TYPE MSG112	
	013244	104401	071025			TYPEIT ,MSG112	
						.DSABL CRF	
6397	013250	005737	002402	3\$:		TST MSIZE	
6398	013254	001414				BEQ 5\$	
6399	013256	022737	004000	002402		CMP #2048.,MSIZE	
6400	013264	001003				BNE 13\$	

Address	Instruction	Comments
6401	013266 162737 000004 002402	
6402	013274 134: SUB #4,MSIZE ; SUBTRACT 4K FOR THE I/O PAGE	
	013274 013746 002402 TYPDEC MSIZE	
	013300 104405 MOV MSIZE,-(SP) ;;SAVE MSIZE FOR TYPEOUT	
		;;GO TYPE--DECIMAL ASCII WITH SIGN
6403	013302 .DSABL CRF	
	013302 104401 071057 TYPE MSG113	
		TYPEIT ,MSG113
		.DSABL CRF
6404	013306 022737 004000 002452 54: CMP #2048.,I	
6405	013314 001003 BNE 64	
6406	013316 162737 000004 002452 SUB #4,I ; SUBTRACT 4K FOR THE I/O PAGE	
6407	013324 64: TYPDEC I	
	013324 013746 002452 MOV I,-(SP) ;;SAVE I FOR TYPEOUT	
	013330 104405 TYPDS ;;;GO TYPE--DECIMAL ASCII WITH SIGN	
		.DSABL CRF
6408	013332 TYPE MSG070	
	013332 104401 070127 TYPEIT ,MSG070	
		.DSABL CRF
6409	013336 IF #SW6 OFF.IN @SWR	
	013336 032777 000100 167272	
	013344 001002	
6410	013346 004737 032722	
6411	013352 CALL PCONFIG	
	013352 END; OF IF #SW6	
		BIT #SW6,@SWR
		BNE L107
		L107:;;;;;

6414 013352

SUBTST <<CHECK APT SIZING>>

;SUBTEST CHECK APT SIZING

IF APTFLAG IS TRUE AND APTSIZE IS TRUE

6415 013352 005737 002352

TST APTFLAG
BEQ L110
TST APTSIZE
BEQ L110

013356 001475 002444

6416 013366 005037 002434

6417 013372 012700 056676

6418 013376 005002

CLR TEMP
MOV #MAMS1,R0
FOR R2 := #0 TO #4

CLR R2

013400

B14:::;;

6419 013400 105760 000001

IFB 1(R0) NE #0

TSTB 1(R0)
BEQ L111

013404 001441

6420 013406 111001

MOVB (R0),R1
BIC #177400,R1
IF 2(R0) LT #0

6421 013410 042701 177400

6422 013414 005760 000002

TST 2(R0)
BGE L112

013420 002002

6423 013422 000261

SEC
ELSE

6424 013424 000401

BR L113

013426

L112:::;;

6425 013426 000241

CLC
END ;OF IF 2(R0)

6426 013430

L113:::;;

013430

6427 013430 006101

ROL R1
INC R1
ASL R1
ASL R1
ASL R1
ASL R1
SUB TEMP,R1
MOV R1,TEMP
IFB 1(R0) EQ #3

;TO COMPENSATE FOR 4 BANKS BEING (0-3)

6428 013432 005201

6429 013434 006301

6430 013436 006301

6431 013440 006301

6432 013442 006301

6433 013444 163701 002434

6434 013450 010137 002434

6435 013454

CMPB 1(R0),#3
BNE L114

013454 126027 000001 000003

013462 001002

6436 013464 060137 002422

ADD R1,APTPAR
END ;OF IFB 1(R0)

6437 013470

L114:::;;

6438 013470

IFB 1(R0) EQ #4

CMPB 1(R0),#4
BNE L115

013470 126027 000001 000004

013476 001002

6439 013500 060137 002424

ADD R1,APTECC
END ;OF IFB 1(R0)

6440 013504

L115:::;;

013504

6441 013504 062700 000004

ADD #4,R0
END ;OF IFB 1(R0)

6442 013510

L111:::;;

013510

6443 013510

END ;OF FOR R2

INC R2
CMP R2,#4
BLE B14

013510 005202

013512 020227 000004

013516 003730

013520

E14:::;;


```
6444 013520      IF APTPAR NE LSIZE OR APTECC NE MSIZE OR APTECC NE PSIZE
      013520      023737 002422 002400      CMP APTPAR,LSIZE
      013526      001010      BNE L116
      013530      023737 002424 002402      CMP APTECC,MSIZE
      013536      001004      BNE L116
      013540      023737 002424 002404      CMP APTECC,PSIZE
      013546      001401      BEQ L117
      013550      L116:::
6445 013550      104046      ERROR      +46
6446 013552      END ;OF IF APTPAR
      013552      L117:::
6447 013552      END ;OF IF APTFLAG
      013552      L110:::
```

6449 013552

LOOP: NEWTST <<DIAGNOSTIC MODE DISPATCH ROUTINE>>

```

;*****
;TEST 5      DIAGNOSTIC MODE DISPATCH ROUTINE
;*****

```

TST5: SCOPE

CLR CONTFLAG

MOV @SWR,R0

;GET SWITCHES

BIC #C16,R0

;MASK TO ONLY MODE BITS

CALL @DISPTBL(R0)

;DISPATCH TO ROUTINE THROUGH NEXT TABLE

JMP MEMDONE

;GO TO NEXT TEST

DISPTBL:BAFPAF

;MODE 0;BANKS FORWARD, PATTERNS FORWARD

BAFPAF

;MODE 1;BANKS FORWARD, PATTERNS REVERSE

BAWPAF

;MODE 2;BANKS WORST FIRST, PATTERNS FORWARD

BAWPAF

;MODE 3;BANKS WORST FIRST, PATTERNS REVERSE

PAFBAF

;MODE 4;PATTERNS FORWARD, BANKS FORWARD

PAFBAW

;MODE 5;PATTERNS FORWARD, BANKS WORST FIRST

PARBAF

;MODE 6;PATTERNS REVERSE, BANKS FORWARD

PARBAW

;MODE 7;PATTERNS REVERSE, BANKS WORST FIRST

MEMDONE:CALL DOBACK

;CHECK BACKGROUND PATTERN

NEWTST<<UNIQUE BANK TEST>>

```

;*****
;TEST 6      UNIQUE BANK TEST
;*****

```

TST6: SCOPE

;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA

;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)

IF SELONLY IS FALSE

TST SELONLY

BNE L120

SET HEADER,MUT

MOV #-1,HEADER

MOV #-1,MUT

CALL MT0027

SET HEADER

MOV #-1,HEADER

CLR MUT

END ;OF IF SELONLY

L120:::::

CALL DOBACK

;RESTORE BACKGROUND PATTERN

FLUSH: SUBTST <<FLUSH OUT DBE'S>>

```

;*****
;SUBTEST    FLUSH OUT DBE'S
;*****

```

CALL MT0030

```

013552 000004
6450 013554 005037 002222
6451 013560 017700 167052
6452 013564 042700 177761
6453 013570 004770 013600
6454 013574 000137 013620
6455 013600 014262
6456 013602 014370
6457 013604 014476
6458 013606 014626
6459 013610 014756
6460 013612 015106
6461 013614 015260
6462 013616 015410
6463
6464 013620 004737 014162
6465
6466 013624

```

```

013624 000004
6467
6468
6469 013626 005737 002002
013626 001015
6470 013634 012737 177777 002612
013634 012737 177777 002110
013642 012737 177777 002110
6471 013650 004737 021610
6472 013654 012737 177777 002612
013654 005037 002110
6473 013662 005037 002110
6474 013666
013666
6475 013666 004737 014162
6479
6480 013672

```

6481 013672 004737 022200

```

6484          .SBTTL  END OF PASS ROUTINE
6485          ;*****
6486          ;*INCREMENT THE PASS NUMBER ($PASS)
6487          ;*INDICATE END-OF-PROGRAM AFTER EACH PASSES THRU THE PROGRAM
6488          ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
6489          ;*IF THERES A MONITOR GO TO IT
6490          ;*IF THERE ISN'T JUMP TO LOOP
6491 013676 005037 002442      $EOP:  CLR      FSINFLAG
6492 013702 012700 002666      MOV      #CONFIG+2,R0      ;MOVE 2ND WORD OF CONFIG TO R0
6493 013706 042710 020000      1$:   BIC      #BIT13,(R0)      ;CLEAR BACKGROUND VALID BIT
6494 013712 062700 000004      ADD      #4,R0      ;INCREMENT TO NEXT BANK
6495 013716 020027 003620      CMP      R0,#3620      ;DONE?
6496 013722 003771          BLE      1$      ;NO - BRANCH
6497 013724 013737 002630 002016      MOV      $ERTTL,LASTERROR
6498 013732 005237 056654      INC      $PASS      ;;INCREMENT THE PASS NUMBER
6499 013736 042737 100000 056654      BIC      #100000,$PASS      ;;DON'T ALLOW A NEG. NUMBER
6500 013744          TYPE      MSG077      ;;TYPE "END PASS #"
        013744 104401 070251      TYPEIT  ,MSG077
        .DSABL  CRF
        IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE OR $PASS EQ #1
6501 013750          BIT      #SW11,@SWR
        013750 032777 004000 166660      BNE      L121
        013756 001007          TST      QVFLAG
        013760 005737 002346          BNE      L121
        013764 001004          CMP      $PASS,#1
        013766 023727 056654 000001      BNE      L122
        013774 001004          L121:;;;;;
        013776          TYPE      MSG035      ;QV
6502 013776          TYPEIT  ,MSG035
        013776 104401 067273      .DSABL  CRF
        014002 005037 002346      CLR      QVFLAG
6503 014006          END ;OF IF SW11
6504 014006          L122:;;;;;
6505 014006          TYPDEC  $PASS
        014006 013746 056654      MOV      $PASS,-(SP)      ;;SAVE $PASS FOR TYPEOUT
        014012 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
        .DSABL  CRF
6506 014014 013700 000042      MOV      42,R0      ;;GET MONITOR ADDRESS
6507 014020 001456          BEQ      $DOAGAIN      ;;BRANCH IF NO MONITOR
6508 014022 022700 002000      $ZAP42: CMP      #STACK,R0      ;ARE WE UNDER RT11
6509 014026 001453          BEQ      $DOAGAIN      ;YES - BRANCH
6510          ;WE ARE UNDER (HEAVEN HELP US) XXDP!
6511 014030          PUSH      R0
        014030 010046          MOV      R0,-(SP)
6512 014032 004737 040636      CALL      SHUTUP
6513 014036          POP      R0
        014036 012600          MOV      (SP)+,R0
6514 014040 000005          RESET          ;;CLEAR THE WORLD
6515 014042 004710      $ENDAD: CALL      (R0)      ;;GO TO MONITOR
6516 014044 000240          NOP          ;;SAVE ROOM
6517 014046 000240          NOP          ;;FOR
6518 014050 000240          NOP          ;;ACT11
6519 014052      $DOAGN: ;UNDO SHUTUP STUFF
6520          ;      RESTORE STACK
6521          ;      ENERGIZE Q-BUS MAP & 22 BIT ADDRESSING
6522          ;      ENERGIZE MEMORY MANAGEMENT
6523          ;      PUT LOADERS BACK HOME

```

Address	Hex	Hex	Hex	Label	Comment	Comment
6524	014052	013706	002574		MOV KSTACK, SP	
6525	014056	005737	002454		TST NO22BIT	:IS THIS AN 11/83.11/23 B OR 11/23?
6526	014062	001003			BNE 1\$	
6527	014064	052737	000060	172516	BIS #BIT5:BIT4,MMR3	
6528	014072	104420		1\$:	ENERGIZE	:TURN ON MEMORY MANAGEMENT
6529	014074	013700	002576		MOV LOADHOME, R0	:DESTINATION BANK
6530	014100	012701	000001		MOV #1, R1	:SOURCE BANK
6531	014104	004737	037442		CALL BANKMOV	
6532	014110				IF APTFLAG IS TRUE	
	014110	005737	002352			TST APTFLAG
	014114	001420				BEQ L123
6533	014116				IF \$USWR EQ \$PASS	
	014116	023737	056672	056654		CMP \$USWR, \$PASS
	014124	001014				BNE L124
6534	014126	012701	000050	APTHANG:	MOV #50, R1	
6535	014132	077001		2\$:	SOB R0, 2\$	
6536	014134	062737	000001	056656	ADD #1, \$DEVCT	
6537	014142	005537	056660		ADC \$UNIT	
6538	014146	077107			SOB R1, 2\$	
6539	014150	005237	056654		INC \$PASS	
6540	014154	000764			BR APTHANG	
6541	014156				END ;OF IF \$USWR	
	014156					L124:::~::~
6542	014156				END ;OF IF APTFLAG	
	014156					L123:::~::~
6543	014156	000137	013552	\$DOAGAIN:	JMP LOOP	:RETURN

6546 014162

DOBACK: SUBST <<WRITE BACKGROUND PATTERNS>>

;*****
;*SUBTEST WRITE BACKGROUND PATTERNS
;*****

6547 014162 005037 002112

6548 014166 005037 002102

014166 005037 002102

014172

6549 014172 004737 040052

6550 014176 005737 002116

014176 005737 002116

014202 001420

014204 005737 002124

014210 001015

6551 014212

014212 012737 177777 002612

014220 012737 177777 002110

6552 014226 004737 017104

6553 014232 005037 002110

6554 014236

014236 012737 177777 002612

6555 014244

014244

6556 014244

014244 005237 002102

014250 023737 002102 002556

014256 003745

014260

6557 014260 000207

CLR PATTERN
FOR BANK := #0 TO LASTBANKCALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE

SET HEADER,MUT

CALL MKTEST
CLR MUT
SET HEADER

END ;OF IF ACFLAG

END ;OF FOR BANK

RETURN

CLR BANK
B15:;:;:;:TST ACFLAG
BEQ L125
TST RRFLAG
BNE L125MOV #-1,HEADER
MOV #-1,MUT

;CALL MJTEST WOULD ALSO WORK

MOV #-1,HEADER

L125:;:;:;:

INC BANK
CMP BANK, LASTBANK
BLE B15

E15:;:;:;:

6560
6561
6562 014262

6563 014262 005037 002102
6564
6565 014266 004737 040052
6566 014272 005737 002116
6567 014276 001412
6568 014300 005737 002124
6569 014304 001007
6570 014306 005037 002112
6571
6572 014312 004737 015562
6573
6574 014316 004737 040474
6575 014322 001373
6576
6577 014324 005037 002222
6578 014330 004737 040520
6579 014334 002354
6580
6581 014336 005737 002126
6582 014342 001401
6583 014344 000207
6584 014346 004737 036402
6585 014352
014352 103001
014354 000207
014356

6586
6587 014356 004737 014262
6588 014362 004737 037214
6589 014366 000207

.SBTTL MTEST MODES

```
BAFPAF: SUBTST <<BANKS FORWARD,PATTERNS FORWARD    **RECURSIVE**>>
;*****
;*SUBTEST    BANKS FORWARD,PATTERNS FORWARD    **RECURSIVE**
;*****
        CLR    BANK                ;SET BANK TO 0
        ;START OF BANK LOOP
1$:      CALL   EXBANK              ;EXAMINE BANK
        TST    ACFLAG              ;CAN WE ACCESS THIS BANK?
        BEQ    4$                  ;NO - GO TO BANK LOOP TERMINATION
        TST    RRFLAG              ;RELOCATION REQUIRED?
        BNE    4$                  ;YES - GO TO BANK LOOP TERMINATION
        CLR    PATTERN            ;SET PATTERN TO 0
        ;START OF PATTERN LOOP
2$:      CALL   MTEST              ;GO TEST CORRECT MEMORY
        ;TERMINATION OF PATTERN LOOP
        CALL   INCPAT              ;GO SEE IF THIS IS THE LAST PATTERN
        BNE    2$                  ;NO LOOP ON THIS PATTERN
        ;TERMINATION OF BANK LOOP
4$:      CLR    CONFLAG
        CALL   INCBNK              ;NEXT HIGHER BANK
        BGE    1$                  ;IF NOT DONE - LOOP ON THIS BANK
        ;END OF LOOPS
        TST    RLFLAG              ;HAVE WE BEEN RELOCATED?
        BEQ    5$                  ;NO SKIP
        RETURN                     ;YES - RETURN
5$:      CALL   RELOCATE            ;MOVE & MAP PROGRAM
        ON.ERROR THEN $RETURN

;***NOTE** RECURSIVE CALL
CALL    BAFPAF                    ;CALL SELF
CALL    UNRELOCATE                ;UNMOVE & UNMAP PROGRAM
RETURN
```

BCC L126
RTS PC
L126:::;::;

6592 014370

BAFPAR: SUBTST <<BANKS FORWARD,PATTERNS REVERSE **RECURSIVE***>>
;*****
;*SUBTEST BANKS FORWARD,PATTERNS REVERSE **RECURSIVE**
;*****

6593 014370 005037 002102

6594

6595 014374 004737 040052

6596 014400 005737 002116

6597 014404 001412

6598 014406 005737 002124

6599 014412 001007

6600 014414 004737 040510

6601

6602 014420 004737 015562

6603

6604 014424 005337 002112

6605 014430 100373

6606

6607 014432 005037 002222

6608 014436 004737 040520

6609 014442 002354

6610

6611 014444 005737 002126

6612 014450 001401

6613 014452 000207

6614 014454 004737 036402

6615 014460

014460 103001

014462 000207

014464

6616

6617 014464 004737 014370

6618 014470 004737 037214

6619 014474 000207

CLR BANK ;SET BANK TO 0
;START OF BANK LOOP
1\$: CALL EXBANK ;EXAMINE BANK
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4\$;NO GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4\$;YES - GO TO BANK LOOP TERMINATION
CALL SETPAT ;SET HIGH PATTERN FOR CORRECT MEMORY
;START OF PATTERN LOOP
2\$: CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF PATTERN LOOP
DEC PATTERN ;IS THIS THE LAST PATTERN?
BPL 2\$;NO LOOP ON THIS PATTERN
;TERMINATION OF BANK LOOP
4\$: CLR CONFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 1\$;IF NOT DONE - LOOP ON THIS BANK
;END OF LOOPS
TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 5\$;NO - SKIP
RETURN ;YES - RETURN
5\$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN \$RETURN

BCC L127
RTS PC
L127:::;;

;**NOTE** RECURSIVE CALL
CALL BAFPAR ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN

6622 014476

```
BAWPAF: SUBTST <<BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**>>
;*****
;SUBTEST      BANKS WORST FIRST,PATTERNS FORWARD **RECURSIVE**
;*****
```

6623 014476 005037 002102

CLR BANK ;SET BANK TO 0

6624

;START OF BANK LOOP

6625 014502 004737 040052

1\$: CALL EXBANK ;EXAMINE BANK

6626 014506 005737 002116

TST ACFLAG ;CAN WE ACCESS THIS BANK?

6627 014512 001415

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

6628 014514 005737 002130

TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)?

6629 014520 001412

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

6630 014522 005737 002124

TST RRFLAG ;RELOCATION REQUIRED?

6631 014526 001007

BNE 4\$;YES - GO TO BANK LOOP TERMINATION

6632 014530 005037 002112

CLR PATTERN ;SET PATTERN TO 0

6633

;START OF PATTERN LOOP

6634 014534 004737 015562

2\$: CALL MTEST ;GO TEST CORRECT MEMORY

6635

;TERMINATION OF PATTERN LOOP

6636 014540 004737 040474

CALL INCPAT ;GO SEE IF THIS IS THE LAST PATTERN

6637 014544 001373

BNE 2\$;NO - LOOP ON THIS PATTERN

6638

;TERMINATION OF BANK LOOP

6639 014546 005037 002222

4\$: CLR CONFLAG

6640 014552 004737 040520

CALL INCBNK ;NEXT HIGHER BANK

6641 014556 002351

BGE 1\$;IF NOT DONE - LOOP ON THIS BANK

6642

;END OF LOOPS

6643 014560 005137 002600

COM WORST ;IS THIS AN EVEN NUMBERED PASS?

6644 014564 001003

BNE 5\$;YES - SKIP

6645

;**NOTE** RECURSIVE CALL

6646 014566 004737 014476

CALL BAWPAF ;CALL SELF

6647 014572 000207

RETURN

6648 014574 005737 002126

5\$: TST RLFLAG ;HAVE WE BEEN RELOCATED?

6649 014600 001401

BEQ 6\$;NO - SKIP

6650 014602 000207

RETURN ;YES - RETURN

6651 014604 004737 036402

6\$: CALL RELOCATE ;MOVE & MAP PROGRAM

6652 014610

ON.ERROR THEN \$RETURN

014610 103001

BCC L130

014612 000207

RTS PC

014614

L130:::;::;

6653

;**NOTE** RECURSIVE CALL

6654 014614 004737 014476

CALL BAWPAF ;CALL SELF

6655 014620 004737 037214

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

6656 014624 000207

RETURN

6659 014626

BAWPAR: SUBTST <<BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE***>>
 ;*****
 ;*SUBTEST BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**
 ;*****

6660 014626 005037 002102

6661

6662 014632 004737 040052

6663 014636 005737 002116

6664 014642 001415

6665 014644 005737 002130

6666 014650 001412

6667 014652 005737 002124

6668 014656 001007

6669 014660 004737 040510

6670

6671 014664 004737 015562

6672

6673 014670 005337 002112

6674 014674 100373

6675

6676 014676 005037 002222

6677 014702 004737 040520

6678 014706 002351

6679

6680 014710 005137 002600

6681 014714 001003

6682

6683 014716 004737 014626

6684 014722 000207

6685 014724 005737 002126

6686 014730 001401

6687 014732 000207

6688 014734 004737 036402

6689 014740

014740 103001

014742 000207

014744

6690

6691 014744 004737 014626

6692 014750 004737 037214

6693 014754 000207

CLR BANK ;SET BANK TO 0
 ;START OF BANK LOOP
 1\$: CALL EXBANK ;EXAMINE BANK
 TST ACFLAG ;CAN WE ACCESS THIS BANK?
 BEQ 4\$;NO - GO TO BANK LOOP TERMINATION
 TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)
 BEQ 4\$;NO - GO TO BANK LOOP TERMINATION
 TST RRFLAG ;RELOCATION REQUIRED?
 BNE 4\$;YES - GO TO BANK LOOP TERMINATION
 CALL SETPAT ;SET HIGH PATTERN FOR CORRECT MEMORY
 ;START OF PATTERN LOOP
 2\$: CALL MTEST ;GO TEST CORRECT MEMORY
 ;TERMINATION OF PATTERN LOOP
 DEC PATTERN ;IS THIS THE LAST PATTERN?
 BPL 2\$;NO - LOOP ON THIS PATTERN
 ;TERMINATION OF BANK LOOP
 4\$: CLR CONTFLAG
 CALL INCBNK ;NEXT HIGHER BANK
 BGE 1\$;IF NOT DONE - LOOP ON THIS BANK
 ;END OF LOOPS
 COM WORST ;IS THIS AN EVEN NUMBERED PASS?
 BNE 5\$;YES - SKIP
 ;**NOTE** RECURSIVE CALL
 CALL BAWPAR ;CALL SELF
 RETURN
 5\$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
 BEQ 6\$;NC SKIP
 RETURN ;YES - RETURN
 6\$: CALL RELOCATE ;MOVE & MAP PROGRAM
 ON.ERROR THEN \$RETURN

BCC L131

RTS PC

L131:::;::;

;**NOTE** RECURSIVE CALL
 CALL BAWPAR ;CALL SELF
 CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
 RETURN

6696 014756

```

PAFBAF: SUBTST <<PATTERNS FORWARD,BANKS FORWARD      **RECURSIVE**>>
;*****
;*SUBTEST      PATTERNS FORWARD,BANKS FORWARD      **RECURSIVE**
;*****

```

6697 014756 005037 002112

6698

6699 014762 005037 002102

6700

6701 014766 004737 040052

6702 014772 004737 040456

6703 014776 001010

6704 015000 005737 002116

6705 015004 001405

6706 015006 005737 002124

6707 015012 001002

6708 015014 004737 015562

6709

6710 015020 005037 002222

6711 015024 004737 040520

6712 015030 002356

6713

6714 015032 004737 040474

6715 015036 001351

6716

6717 015040 005137 002134

6718

6719 015044 001403

6720

6721 015046 004737 014756

6722 015052 000207

6723 015054 005737 002126

6724 015060 001401

6725 015062 000207

6726 015064 004737 036402

6727 015070

015070 103001

015072 000207

015074

6728

6729 015074 004737 014756

6730 015100 004737 037214

6731 015104 000207

```

CLR      PATTERN      ;SET PATTERN TO 0
;START OF PATTERN LOOP
1$: CLR      BANK      ;SET BANK TO 0
;START OF BANK LOOP
2$: CALL     EXBANK     ;EXAMINE BANK
CALL     BANKOK        ;CORRECT MEMORY FOR THIS BANK?
BNE      4$            ;NO - GO TO BANK LOOP TERMINATOR
TST      ACFLAG        ;CAN WE ACCESS THIS BANK?
BEQ      4$            ;NO - GO TO BANK LOOP TERMINATION
TST      RRFLAG        ;RELOCATION REQUIRED?
BNE      4$            ;YES - GO TO BANK LOOP TERMINATION
CALL     MTEST         ;GO TEST CORRECT MEMORY
;TERMINATION OF BANK LOOP
4$: CLR      CONTFLAG
CALL     INCBNK        ;NEXT HIGHER BANK
BGE      2$            ;IF NOT DONE - LOOP ON THIS BANK
;TERMINATION OF PATTERN LOOP
CALL     INCRPT        ;NEXT HIGHER PATTERN
BNE      1$            ;OK - LOOP; ELSE CONTINUE
;END OF LOOPS
COM      TMFLAG        ;COMPLEMENT TYPE OF MEMORY
;IS THIS AN EVEN NUMBER PASS?
BEQ      5$            ;YES - SKIP
;**NOTE** RECURSIVE CALL
CALL     PAFBAF        ;CALL SELF
RETURN
5$: TST      RLFLAG     ;HAVE WE BEEN RELOCATED?
BEQ      6$            ;NO - SKIP
RETURN   ;YES - RETURN
6$: CALL     RELOCATE   ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN

```

BCC L132

RTS PC

L132:::~::~

```

;**NOTE** RECURSIVE CALL
CALL     PAFBAF        ;CALL SELF
CALL     UNRELOCATE    ;UNMOVE & UNMAP PROGRAM
RETURN

```

6734 015106

PAFBAW: SUBST <<PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**>>

;*****
;SUBTEST PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**
;*****

6735 015106 005037 002112

CLR PATTERN ;SET PATTERN TO 0

6736

;START OF PATTERN LOOP

6737 015112 005037 002102

1\$: CLR BANK ;SET BANK TO 0

6738

;START OF BANK LOOP

6739 015116 004737 040052

2\$: CALL EXBANK ;EXAMINE BANK

6740 015122 004737 040456

CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?

6741 015126 001013

BNE 4\$;NO - GO TO BANK LOOP TERMINATOR

6742 015130 005737 002116

TST ACFLAG ;CAN WE ACCESS THIS BANK?

6743 015134 001410

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

6744 015136 005737 002130

TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)

6745 015142 001405

BEQ 4\$;NO - GO TO BANK LOOP TERMINATION

6746 015144 005737 002124

TST RRFLAG ;RELOCATION REQUIRED?

6747 015150 001002

BNE 4\$;YES - GO TO BANK LOOP TERMINATION

6748 015152 004737 015562

CALL MTEST ;GO TEST CORRECT MEMORY

6749

;TERMINATION OF BANK LOOP

6750 015156 005037 002222

4\$: CLR CONFLAG

6751 015162 004737 040520

CALL INCBNK ;NEXT HIGHER BANK

6752 015166 002353

BGE 2\$;IF NOT DONE - LOOP ON THIS BANK

6753

;TERMINATION OF PATTERN LOOP

6754 015170 004737 040474

CALL INCRPT ;NEXT HIGHER PATTERN

6755 015174 001346

BNE 1\$;OK - LOOP; ELSE CONTINUE

6756

;END OF LOOPS

6757 015176 005137 002134

COM TMFLAG

6758

;COMPLEMENT TYPE OF MEMORY

6759 015202 001403

BEQ 5\$;IS THIS AN EVEN NUMBER PASS?

6760

;**NOTE** RECURSIVE CALL

6761 015204 004737 015106

CALL PAFBAW ;CALL SELF

6762 015210 000207

RETURN

6763 015212 005137 002600

5\$: COM WORST ;4TH PASS?

6764 015216 001003

BNE 6\$;YES - SKIP

6765

;**NOTE** RECURSIVE CALL

6766 015220 004737 015106

CALL PAFBAW ;CALL SELF

6767 015224 000207

RETURN

6768 015226 005737 002126

6\$: TST RLFLAG ;HAVE WE BEEN RELOCATED?

6769 015232 001401

BEQ 7\$;NO - SKIP

6770 015234 000207

RETURN ;YES - RETURN

6771 015236 004737 036402

7\$: CALL RELOCATE ;MOVE & MAP PROGRAM

6772 015242

ON.ERROR THEN \$RETURN

015242 103001

BCC L133

015244 000207

RTS PC

015246

L133:::;;

6773

;**NOTE** RECURSIVE CALL

6774 015246 004737 015106

CALL PAFBAW ;CALL SELF

6775 015252 004737 037214

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

6776 015256 000207

RETURN

6779 015260

```

PARBAF: SUBTST <<PATTERNS REVERSE,BANKS FORWARD **RECURSIVE**>>
;*****
;*SUBTEST PATTERNS REVERSE,BANKS FORWARD **RECURSIVE**
;*****
        CALL HIPAT ;SET HIGHEST PATTERNS
        ;START OF PATTERN LOOP
1$: CLR BANK ;SET BANK TO 0
        ;START OF BANK LOOP
2$: CALL EXBANK ;EXAMINE BANK
        CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
        BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
        TST ACFLAG ;CAN WE ACCESS THIS BANK?
        BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
        TST RRFLAG ;RELOCATION REQUIRED?
        BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
        CALL MTEST ;GO TEST CORRECT MEMORY
        ;TERMINATION OF BANK LOOP
4$: CLR CONTFLAG
        CALL INCBNK ;NEXT HIGHER BANK
        BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
        ;TERMINATION OF PATTERN LOOP
        DEC PATTERN ;NEXT LOWER PATTERN
        BPL 1$ ;OK - LOOP; ELSE CONTINUE
        ;END OF LOOPS
        COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
        ;IS THIS AN EVEN NUMBER PASS?
        BEQ 5$ ;YES - SKIP
        ;**NOTE** RECURSIVE CALL
        CALL PARBAF ;CALL SELF
        RETURN
5$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
        BEQ 6$ ;NO SKIP
        RETURN ;YES - RETURN
6$: CALL RELOCATE ;MOVE & MAP PROGRAM
        ON.ERROR THEN $RETURN

;*****
;**NOTE** RECURSIVE CALL
CALL PARBAF ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN

```

BCC L134
RTS PC

L134:::~::~

```

6780 015260 004737 040510
6781
6782 015264 005037 002102
6783
6784 015270 004737 040052
6785 015274 004737 040456
6786 015300 001010
6787 015302 005737 002116
6788 015306 001405
6789 015310 005737 002124
6790 015314 001002
6791 015316 004737 015562
6792
6793 015322 005037 002222
6794 015326 004737 040520
6795 015332 002356
6796
6797 015334 005337 002112
6798 015340 100351
6799
6800 015342 005137 002134
6801
6802 015346 001403
6803
6804 015350 004737 015260
6805 015354 000207
6806 015356 005737 002126
6807 015362 001401
6808 015364 000207
6809 015366 004737 036402
6810 015372
        015372 103001
        015374 000207
        015376
6811
6812 015376 004737 015260
6813 015402 004737 037214
6814 015406 000207

```

6817 015410

```

PARBAW: SUBTST <<PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**>>
;*****
;SUBTEST PATTERNS REVERSE,BANKS WORST FIRST **RECURSIVE**
;*****
CALL HIPAT ;SET HIGHEST PATTERN
;START OF PATTERN LOOP
1$: CLR BANK ;SET BANK TO 0
;START OF BANK LOOP
2$: CALL EXBANK ;EXAMINE BANK
CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF BANK LOOP
4$: CLR CONFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
;TERMINATION OF PATTERN LOOP
DEC PATTERN ;NEXT LOWER PATTERN
BPL 1$ ;OK - LOOP; ELSE CONTINUE
;END OF LOOPS
COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
BEQ 5$ ;IS THIS AN EVEN NUMBER PASS?
;YES - SKIP
;***NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
RETURN
5$: COM WORST ;4TH PASS?
BNE 6$ ;YES - SKIP
;***NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
RETURN
6$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 7$ ;NO - SKIP
RETURN ;YES - RETURN
7$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN

;***NOTE** RECURSIVE CALL
CALL PARBAW ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN

```

BCC L135
RTS PC

L135:::~::~

```

6818 015410 004737 040510
6819
6820 015414 005037 002102
6821
6822 015420 004737 040052
6823 015424 004737 040456
6824 015430 001013
6825 015432 005737 002116
6826 015436 001410
6827 015440 005737 002130
6828 015444 001405
6829 015446 005737 002124
6830 015452 001002
6831 015454 004737 015562
6832
6833 015460 005037 002222
6834 015464 004737 040520
6835 015470 002353
6836
6837 015472 005337 002112
6838 015476 100346
6839
6840 015500 005137 002134
6841
6842 015504 001403
6843
6844 015506 004737 015410
6845 015512 000207
6846 015514 005137 002600
6847 015520 001003
6848
6849 015522 004737 015410
6850 015526 000207
6851 015530 005737 002126
6852 015534 001401
6853 015536 000207
6854 015540 004737 036402
6855 015544
    015544 103001
    015546 000207
    015550
6856
6857 015550 004737 015410
6858 015554 004737 037214
6859 015560 000207

```

6862 015562

MTEST: SUBTST <<SUBR SETUP MEMORY TEST>>

 ;*****
 ;*SUBTEST SUBR SETUP MEMORY TEST
 ;*****

6863 015562

SET HEADER

;INITIALIZE HEADER MESSAGE TYPEOUT

015562 012737 177777 002612

MOV #-1,HEADER

6864 015570

SET MUT

;INDICATE THERE IS A MEMORY UNDER TEST

015570 012737 177777 002110

MOV #-1,MUT

6865 015576

CLR PASFLG

6866 015602

TST MKFLAG

;ECC?

6867 015606

BEQ MT1

;NO - SKIP

6868 015610

BEGIN HOLDLOOP

015610

B16:::;;

6869 015610

IF CONTFLAG IS TRUE THEN LEAVE HOLDLOOP

TST CONTFLAG

015610 005737 002222

BNE E16

015614 001005

6870 015616

IF SKIPMK IS FALSE

TST SKIPMK

015616 005737 002342

BNE L136

015622 00100~

6871 015624

CALL MKCONTROL

6872 015630

END; OF IF SKIPMK

L136:::;;

015630

6873 015630

END HOLDLOOP

E16:::;;

6874 015630 004737 017104

CALL MKTEST

;YES - DO ECC TESTS

6875 015634

BR MT2

6876 015636

MT1:

CALL MJTEST

;DO PARITY TESTS

6877 015642

MT2:

CLR MUT

;NOW - NO MEMORY UNDER TEST

6878 015646

SET HEADER

;ALLOW HEADERS NORMAL

015646 012737 177777 002612

MOV #-1,HEADER

6879 015654

000207

RETURN

6882 015656

MKCONTROL:SUBTST <<SUBR TEST ECC CSR LOGIC DISPATCH>>

```

*****
;SUBTEST SUBR TEST ECC CSR LOGIC DISPATCH
*****

```

6883

6884

6885

```

6886 015656 005737 002002
      015656 001401
      015662 000207
      015664
      015666

```

6887

```

      015666 005737 002536
      015672 001401
      015674 000207
      015676

```

6888

```

      015676 013746 002102
      015702 010046
      015704 010146
      015706 010246
      015710 010346

```

6889 015712 012737 060000 002232

6890 015720 012737 157776 002234

6891 015726 005037 002236

6892 015732 005037 002240

6893 015736 005037 002332

6894 015742 013700 002104

6895 015746 016001 002664

6896 015752 000301

6897 015754 042701 177760

6898 015760 006301

6899 015762 010137 002526

6900 015766 005737 002136

6901 015772 001421

6902 015774 005237 002240

6903 016000 012737 120000 002234

6904 016006 005237 002332

6905 016012 005237 002236

6906 016016 016001 002664

6907 016022 072127 177775

6908 016026 042701 160777

6909 016032 050137 002526

6910 016036 005003

6911 016040 116337 002526 002152

6912 016046 042737 177741 002152

6913 016054

016054 005037 016476

016060

6914 016060 013737 002232 002226

016060

016066

6915 016066

016066 010346

016070 013703 002102

016074 004737 035716

```

;THE NEXT TWO MODULES SOLVE THE PROBLEM OF
;HOW TO RUN THE CSR TESTS ON EACH ECC MEMORY

```

IF SELONLY IS TRUE THEN \$RETURN

```

TST SELONLY
BEQ L137
RTS PC

```

L137:::~::~

IF INHECC IS TRUE THEN \$RETURN

```

TST INHECC
BEQ L140
RTS PC

```

L140:::~::~

PUSH BANK,R0,R1,R2,R3

```

MOV BANK,-(SP)
MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)
MOV R3,-(SP)

```

MOV #FIRST,CSRFBANK

;SET FIRST TEST ADDRESS TO FIRST ADDR.

MOV #LAST,CSRLBANK

CLR CSRINT

CLR SPLTCSR

CLR CSRLOOP

; AND ZERO THE LOOP COUNTER

MOV BANKINDEX,R0

;GET THE BANK INDEX

MOV CONFIG(R0),R1

;GET CSR NUMBER

SWAB R1

BIC #C17,R1

ASL R1

MOV R1,CSRHOLD

;STORE IN THE LOW BYTE

TST INTFLAG

;IS THIS BANK INTERLEAVED?

BEQ 1\$

;BRANCH IF NOT INTERLEAVED

INC SPLTCSR

MOV #120000,CSRLBANK

INC CSRLOOP

;WE MUST LOOP TWICE FOR AN INTERLEAVED BANK

INC CSRINT

MOV CONFIG(R0),R1

;GET THE INTERLEAVE CSR NUMBER

ASH #-3,R1

BIC #C17000,R1

BIS R1,CSRHOLD

;STORE IT IN CSRHOLD'S UPPER BYTE

CLR R3

1\$:

MKLOOP: MOV CSRHOLD(R3),CSRNO

;CLEAR ANY UNNECESSARY BITS

BIC #C36,CSRNO

FOR MKCNT := #0 TO CSRINT

CLR MKCNT

B17:::~::~

FOR CSRFIRST := CSRFBANK TO CSRLBANK BY #4000

MOV CSRFBANK,CSRFIRST

B20:::~::~

MAP BANK

;MAP TEST SPACE TO BANK

MOV R3,(SP)

MOV BANK,R3

CALL MAPPER

.DSABL CRF

```

016100 012603
6916 016102 104511
6917 016104
016104
6918 016104 005037 002334
6919 016110
016110 005737 002116
016114 001503
016116 005737 002124
016122 001100
6920 016124 013737 002226 002230
6921 016132 062737 004000 002230
6922 016140
016140 013737 002226 002412
016146
6923 016146 013737 002412 002414
6924 016154 005737 002240
6925 016160 001404
6926 016162 062737 040000 002414
6927 016170 000403
6928 016172 062737 000002 002414
6929 016200 004737 016500
6930 016204
016204 103440
6931 016206 104424
6932 016210 005037 002076
6933 016214
016214 005037 002452
016220
6934 016220
016220 012737 177777 002612
6935 016226 005037 002264
6936 016232
016232 013700 002452
6937 016236
016236 010346
6938 016240 010637 002146
6939 016244 162737 000002 002146
6940 016252 004737 017004
6941 016256
016256 012603
6942 016260
016260 005237 002452
016264 023727 002452 000027
016272 003752
016274
6943 016274 104423
6944 016276
016276 012737 177777 002334
6945 016304
016304 000407
6946 016306
016306
6947 016306
016306 062737 000004 002412
016314 023737 002412 002230

```

INVALIDATE
BEGIN CSRSTUFF

MOV (SP)+,R3
;INVALIDATE BACKGROUND PATTERN

B21:::~::~

CLR SUCCESS
IF ACFLAG IS TRUE AND RRFLAG IS FALSE

TST ACFLAG
BEQ L141
TST RRFLAG
BNE L141

MOV CSRFIRST,CSRLAST
ADD #4000,CSRLAST
FOR TESTADD := CSRFIRST TO CSRLAST BY #4

MOV CSRFIRST,TESTADD

B22:::~::~

MOV TESTADD,TESTADD+2
TST SPLTCSR
BEQ 1\$
ADD #40000,TESTADD+2
BR 2\$
1\$: ADD #2,TESTADD+2
2\$: CALL SBETEST
ON.NOERROR

CACHOFF
CLR NOPAR
FOR I := #0 TO #27

BCS L142
;TURN CACHE OFF
;INDICATE PARITY ACTION

CLR I

B23:::~::~

SET HEADER

MOV #-1,HEADER

CLR PASFLG
LET R0 := I

PUSH R3

MOV I,R0
;SAVE LOOP COUNTER
MOV R3,-(SP)
;SAVE VECTOR IN CSR OF +K

MOV SP,CTLKVEC
SUB #2,CTLKVEC
CALL CSRCASE
POP R3

;RESTORE LOOP COUNTER
MOV (SP)+,R3

END ;OF FOR I

INC I
CMP I,#27
BLE B23

E23:::~::~

CACHON
SET SUCCESS

;TURN CACHE ON

MOV #-1,SUCCESS

LEAVE CSRSTUFF

BR E21

END ;OF ON.NOERROR

L142:::~::~

END ;OF FOR TESTADD

ADD #4,TESTADD
CMP TESTADD,CSRLAST

	016322	003711							BLE B22
	016324								E22:::~::~:
6948	016324					END ;OF IF			L141:::~::~:
6949	016324					END CSRSTUFF			E21:::~::~:
	016324					IF SUCCESS IS FALSE			TST SUCCESS
6950	016324	005737	002334						BNE L143
	016330	001012							
6951	016332					TYPE MSGA34			
	016332	104401	067207			TYPEIT ,MSGA34			
						.DSABL CRF			
6952	016336					TYPOCS BANK,<TYPES BANK NUMBER>,3			
	016336	013746	002102			MOV BANK,-(SP)	::SAVE BANK FOR TYPEOUT		
							::TYPES BANK NUMBER		
	016342	104403				TYPOS	::GO TYPE--OCTAL ASCII		
	016344	003				.BYTE 3	::TYPE 3 DIGIT(S)		
	016345	000				.BYTE 0	::SUPPRESS LEADING ZEROS		
						.DSABL CRF			
6953	016346					TYPE MSGB34			
	016346	104401	067245			TYPEIT ,MSGB34			
						.DSABL CRF			
6954	016352	004737	050446			CALL PERBNK			
6955	016356					END ;OF IF SUCCESS			
	016356								L143:::~::~:
6956	016356					END; OF FOR CSRFIRST			
	016356	062737	004000	002226					ADD #4000,CSRFIRST
	016364	023737	002226	002234					CMP CSRFIRST,CSRFBANK
	016372	003635							BLE B20
	016374								E20:::~::~:
6957	016374	005237	002240			INC SPLTCSR			
6958	016400					END; OF FOR MKCNT			
	016400	005237	016476						INC MKCNT
	016404	023737	016476	002236					CMP MKCNT,CSRINT
	016412	00362?							BLE B17
	016414								E17:::~::~:
6959	016414	062737	000002	002232		ADD #2,CSRFBANK			
6960	016422	012737	000001	002240		MO' #1,SPLTCSR			
6961	016430	005203				I R3			
6962	016432	020337	002332			CM+ R3,CSRLOOP			
6963	016436	003002				BGT 1\$			
6964	016440	000137	016040			JMP MKLOOP			
6965	016444	104472			1\$:	ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)			
6966	016446					SET CONFLAG			
	016446	012737	177777	002222					MOV #-1,CONFLAG
6967	016454	005037	002240			CLR SPLTCSR			
6968	016460					POP R3,R2,R1,R0,BANK			
	016460	012603							MOV (SP)+,R3
	016462	012602							MOV (SP)+,R2
	016464	012601							MOV (SP)+,R1
	016466	012600							MOV (SP)+,R0
	016470	012637	002102						MOV (SP)+,BANK
6969	016474	000207				RETURN			
6970	016476	000000			MKCNT:	.WORD 0			;COUNTER FOR MKLOOP

6973 016500

SBETEST:SUBTST <<CHECK FOR SBE FREE LOCATIONS>>

;*****
;*SUBTEST CHECK FOR SBE FREE LOCATIONS
;*****

;IN ORDER TO DETERMINE IF A LOCATION IS SBE FREE I DO THIS

;WRITE ZEROS WITH ECC DISABLE
;READ ZEROS BACK
;IF NOT ZEROS THEN RETURN ERROR;WRITE ZEROS WITH ECC ENABLED BUT TRAPS DISABLED
;READ ZEROS BACK
;IF NOT ZEROS THEN RETURN ERROR;TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
;IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR;COMPLIMENT ZEROS TO ONES WITH ECC DISABLE
;READ ONES BACK
;IF NOT ONES THEN RETURN ERROR;WRITE 100,100000,00000 (CHECKBITS COMPLIMENT OF BEFORE)
; WITH ECC ENABLED BUT TRAPS DISABLED
;TEST THE LOCATION FROM THE PAR'S (WITH NO PROGRAM FETCHES)
;IF THERE WERE ANY SBE'S OR DBE'S THEN RETURN ERROR

;IF NONE OF THE ABOVE FORCES A RETURN ERROR THEN RETURN NO.ERROR

.ENABL LSB
PUSH R0,R1,R4

;PUSH R0,R1,R4 ONTO STACK

MOV R0,-(SP)

MOV R1,-(SP)

MOV R4,-(SP)

6998 016500

016500 010046

016502 010146

016504 010446

6999 016506 013701 002412

7000 016512 013704 002414

7001 016516 053737 002552 177776

MOV TESTADD,R1

MOV TESTADD+2,R4

TESTAREA

BIS TESTMODE,PSW

.DSABL CRF

CACHOFF

ECC1DIS

CLEAR (R1),(R4)

;ENTER TEST MODE

;GO TO SYSTEM TEST MODE

;TURN CACHE OFF

;DISABLE ECC ON 1 SELECTED CSR

CLR (R1)

CLR (R4)

7002 016524 104424

7003 016526 104471

7004 016530

016530 005011

016532 005014

7005 016534 005711

7006 016536 001107

7007 016540 005714

7008 016542 001105

7009

7010 016544 104503

7011 016546

016546 005011

016550 005014

7012 016552 005711

7013 016554 001100

7014 016556 005714

7015 016560 001076

7016

7017 016562 104510

CLR1CSR

CLEAR (R1),(R4)

;CLEAR 1 SELECTED CSR

CLR (R1)

CLR (R4)

TST (R1)

BNE SBENT

TST (R4)

BNE SBENT

TSTREAD

;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)

```

7018 016564      032737 100020 002150      IF #BIT15:BIT4 SET.IN CSR
      016564      001415
      016572
7019 016574      012737 177777 002066      SET SKPERR                      ;DISABLE ERRGEN'S ERROR
      016574      104512
      016602      013700 002460      ERRGEN
7021 016604      072027 177774      MOV ERRADD,R0
7022 016610      042700 177600      ASH #-4,R0
7023 016614      023700 002102      BIC #+C177,R0
7024 016620      001454
      016620      002102      IF BANK EQ R0 THEN GOTO SBENT
      016624      001454
      016626
7025 016626      104471      ECC1DIS                      L144:;;;;;;
      016626      005111      ;DISABLE ECC ON 1 SELECTED CSR
7026 016630      005114
7027 016632      023711 002614      COM (R1)
7028 016634      001046      COM (R4)
7029 016640      023714 002614      CMP ONES,(R1)
7030 016642      001043      BNE SBENT
7031 016646      001043      CMP ONES,(R4)
7032 016650      005011      BNE SBENT
7033 016652      012714 100000      CLR1CSR                      ;CLEAR 1 SELECTED CSR
7034 016654      005711      CLR (R1)
7035 016660      001035      MOV #BIT15,(R4)
7036 016662      022714 100000      TST (R1)
7037 016664      001032      BNE SBENT
7038 016670      001032      CMP #BIT15,(R4)
7039 016672      001032      BNE SBENT
7040 016674
7041 016674      032737 100020 002150      TSTREAD                      ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
7042 016674      001415      IF #BIT15:BIT4 SET.IN CSR
      016702
7043 016704      012737 177777 002066      SET SKPERR                      ;DISABLE ERRGEN'S ERROR
      016704      104512
      016712      013700 002460      ERRGEN
7044 016714      072027 177774      MOV ERRADD,R0
7045 016720      042700 177600      ASH #-4,R0
7046 016724      023700 002102      BIC #+C177,R0
7047 016730      001410      IF BANK EQ R0 THEN GOTO SBENT
7048 016734      001410
      016736
7049 016736
7050 016736
      016736
7051 016736      104417      KERNEL                      ;ENTER KERNEL MODE
7052 016740      104473      ECC1INIT                      ;INITIALIZE 1 SELECTED CSR
7053 016742      104423      CACHON                      ;TURN CACHE ON
7054 016744      012604      POP R4,R1,R0                      ;POP R0,R1 & R4 FROM STACK
      016744      012601
      016750      012600
      016752
7055 016752      000241
      016754      000207
7056 016752
7057 016754

```

7058	016756	104503	SBENT: CLR1CSR	;CLEAR 1 SELECTED CSR	
7059	016760		CLEAR (R1),(R4)		
	016760	005011			CLR (R1)
	016762	005014			CLR (R4)
7060	016764	104417	KERNEL	;ENTER KERNEL MODE	
7061	016766	104473	ECC1INIT	;INITIALIZE 1 SELECTED CSR	
7062	016770	104423	CACHON	;TURN CACHE ON	
7063	016772		POP R4,R1,R0	;POP R0,R1 & R4 FROM STACK	
	016772	012604			MOV (SP)+,R4
	016774	012601			MOV (SP)+,R1
	016776	012600			MOV (SP)+,R0
7064	017000		\$RETURN ERROR		
	017000	000261			SEC
	017002	000207			RTS PC
7065			.DSABL LSB		

7068 017004

CSRCASE:SUBST <<CSR PATTERN CASE STATEMENT>>

```

;*****
;SUBTEST      CSR PATTERN CASE STATEMENT
;*****
CASE R0

```

7069 017004

017004 010046

017006 006316

017010 004737 017074

MOV R0,-(SP)

ASL @SP

JSR PC,L146

7070

7071

7072

7073

7074 017014 020222

7075

7076

7077

7078

7079 017016 024024

7080 017020 020354

7081 017022 024104

7082 017024 023560

7083 017026 020456

7084 017030 023622

7085 017032 023660

7086 017034 023740

7087 017036 023774

7088 017040 024134

7089 017042 024164

7090 017044 020320

7091 017046 024220

7092 017050 024220

7093 017052 024220

7094 017054 024220

7095 017056 024220

7096 017060 024220

7097 017062 024220

7098 017064 024220

7099 017066 024220

7100 017070 024220

7101 017072 024220

7102 017074

017074

017074 062616

017076 013646

017100 004736

7103 017102 000207

```

;WARNING IF YOU CHANGE THIS TABLE ALSO
;CHANGE "$DDWO" - "$DDWS" (THE PATTERN BIT MAP)

```

MKCSRT:

;PAT
MT0006TIME
;<1 SECDESCRIPTION
INITIAL DATA TEST

MSV11-J ECC TESTS

MT0044

; 5 SEC

SHIFTING 1/0'S THROUGH CHECK BITS

MT0014

; 1 SEC

BASIC DOUBLE ERROR TEST

MT0045

;<1 SEC

SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST

MT0036

; 1 SEC

CORRECTION CODE TEST

MT0020

; 1 SEC

SYNDROMES TO CSR ON SINGLE BIT ERROR TEST

MT0037

;<1 SEC

ECC DISABLE TEST

MT0041

; 1 SEC

ADDRESS TO CSR ON DOUBLE BIT ERROR

MT0042

;<1 SEC

EXTENDED ADDRESS TO CSR ON ERROR TEST

MT0043

;<1 SEC

WRITE BYTE CLEARS SBE TEST

MT0046

; 1 SEC

CHECK SINGLE BIT ERRORS WITH ECC DISABLED TEST

MT0047

;<1 SEC

NO CSR UPDATE ON SBE WITH EXSISTING DBE

MT0010

;<1 SEC

BYTE ADDRESSING TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

MT0999

; 0 SEC

NULL TEST

END ;OF CASE R0

L146:::~::~

ADD (SP)+,@SP

MOV @SP)+,-(SP)

JSR PC,@SP)+

RETURN

```

7106 017104      MKTEST: SUBTST <<SUBR ECC TEST DISPATCH>>
;*****
;*SUBTEST      SUBR      ECC TEST DISPATCH
;*****
IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE

7107 017104      BIT #SWO,@SWR
017104 032777 000001 163524      BNE L147
017112 001003      TST ACTFLAG
017114 005737 002350      BEQ L150
017120 001402
017122
7108 017122      ECCDIS      L147:;;;;;
104470      ;DISABLE ERROR CORRECTION
7109 017124      ELSE
017124 000401      BR L151
017126      L150:;;;;;
7110 017126 104502      CLRCSR      ;CLEAR ALL CSR'S
7111 017130      END ;OF IF
017130      L151:;;;;;
7112 017130 012737 000002 002076      MOV #2,NOPAR      ;INDICATE PARITY ACTION
7113 017136 012737 000002 002326      MOV #2,PCBUMP      ;TRAPS ADD 2 TO PC
7114 017144 013700 002112      MOV PATTERN,R0      ;GET PATTERN NUMBER
7115 017150 006300      ASL R0      ;MAKE IT A WORD ADDRESS
7116 017152      IF MKPAT(R0) NE #MT0034 AND MKPAT(R0) NE #MT0999
017152 026027 017244 023346      CMP MKPAT(R0),#MT0034
017160 001405      BEQ L152
017162 026027 017244 024220      CMP MKPAT(R0),#MT0999
017170 001401      BEQ L152
017172 104511      INVALIDATE      ;INVALIDATE BACKGROUND PATTERN ON "BANK"
7117 017172      END ;OF IF MKPAT(R0)
7118 017174
017174      L152:;;;;;
7119 017174 010637 002146      MOV SP,CTLKVEC      ;SAVE VECTOR IN CASE OF 'K'
7120 017200 162737 000002 002146      SUB #2,CTLKVEC
7121 017206 004770 017244      CALL @MKPAT(R0)      ;INDEX OFF TABLE
7122 017212      IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
017212 032777 000001 163416      BIT #SWO,@SWR
017220 001003      BNE L153
017222 005737 002350      TST ACTFLAG
017226 001402      BEQ L154
017230      L153:;;;;;
7123 017230 104506      ENASBE      ;TRAP ON SINGLE BIT ERRORS
7124 017232      ELSE
017232 000401      BR L155
017234      L154:;;;;;
7125 017234 104472      ECCINIT      ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7126 017236      END ;OF IF #SWO
017236      L155:;;;;;
7127 017236 005037 002076      CLR NOPAR      ;INDICATE PARITY ACTION
7128 017242 000207      RETURN
7129
7130      ;WARNING IF YOU CHANGE THIS TABLE ALSO
7131      ;CHANGE "$DDW0" - "$DDW5" (THE PATTERN BIT MAP)
7132      ;PAT TIME DISCRPTION
7133 017244      — MKPAT: ;NOTE MT0034 MUST BE FIRST & LAST
7134 017244 023346      MT0034 ;<1 SEC ;SOFT ERROR - BACKGROUND PATTERN TEST
7135 017246 020434      MT0017 ;<1 SEC ;HOLDING 1'S & 0'S TEST
7136 017250 020256      MT0007 ;<1 SEC ;ADDRESS BIT TEST
7137 017252 017544      MT0001 ;<1 SEC ;ADDRESS TEST
7138 017254 017640      MT0002 ;<1 SEC ;COMPLEMENT ADDRESS TEST

```

7139 017256 020070	MT0004 : 1 SEC	;ROTATING ZEROS TEST
7140 017260 020140	MT0005 : 1 SEC	;ROTATING ONES TEST
7141 017262 020536	MT0021 : 1 SEC	;MARCHING 0'S & 1'S TEST
7142 017264 021026	MT0022 :10 SEC	;REFRESH & SHIFTING DIAGONAL TEST
7143 017266 021332	MT0026 :<1 SEC	;RANDOM DATA TEST
7144 017270 021124	MT0024 :20 SEC	;FAST GALLOPING PATTERN TEST
7145 017272 022454	MT0031 : 3 SEC	;SOB-A-LONG TEST
7146 017274 022644	MT0032 :<1 SEC	;WRITE RECOVERY TEST
7147 017276 023160	MT0033 :35 SEC	;BRANCH GOBBLE TEST
7148 017300 023346	MT0034 :<1 SEC	;SOFT ERROR - BACKGROUND PATTERN TEST
7149	;NOTE MT0034 MUST BE FIRST & LAST	
7150 017302 024220	MT0999 : 0 SEC	;NULL TEST
7151 017304 024220	MT0999 : 0 SEC	;NULL TEST
7152 017306 024220	MT0999 : 0 SEC	;NULL TEST
7153 017310 024220	MT0999 : 0 SEC	;NULL TEST
7154 017312 024220	MT0999 : 0 SEC	;NULL TEST
7155 017314 024220	MT0999 : 0 SEC	;NULL TEST
7156 017316 024220	MT0999 : 0 SEC	;NULL TEST
7157 017320 024220	MT0999 : 0 SEC	;NULL TEST
7158 017322 024220	MT0999 : 0 SEC	;NULL TEST

7161 017324

MJTEST: SUBTST <<SUBR PARITY TEST DISPATCH>>

```

;*****
;SUBTEST      SUBR      PARITY TEST DISPATCH
;*****

```

```

7162 017324 012737 000002 002076
7163 017332 012737 000002 002326
7164 017340 012737 060000 002412
7165 017346 012737 060002 002414
7166 017354 013700 002112
7167 017360 006300
7168 017362

```

```

      MOV      #2,NOPAR      ;INDICATE PARITY ACTION
      MOV      #2,PCBUMP     ;TRAPS ADD 2 TO PC
      MOV      #FIRST,TESTADD
      MOV      #FIRST+2,TESTADD+2
      MOV      PATTERN,RO     ;GET PATTERN NUMBER
      ASL      RO             ;MAKE IT A WORD ADDRESS
      IF MJPAT(RO) NE #MT0034 AND MJPAT(RO) NE #MT0999

```

```

      CMP MJPAT(RO),#MT0034
      BEQ L156
      CMP MJPAT(RO),#MT0999
      BEQ L156

```

```

      017362 026027 017430 023346
      017370 001405
      017372 026027 017430 024220
      017400 001401
7169 017402 104511
7170 017404

```

```

      INVALIDATE      ;INVALIDATE BACKGROUND PATTERN ON "BANK"
      END ;OF IF MJPAT(RO)

```

```

      017404
7171 017404 010637 002146
7172 017410 162737 000002 002146
7173 017416 004770 017430
7174 017422 005037 002076
7175 017426 000207
7176
7177
7178
7179
7180

```

```

      MOV      SP,CTLKVEC      ;SAVE VECTOR IN CASE OF "K"
      SUB      #2,CTLKVEC
      CALL     @MJPAT(RO)      ;INDEX OFF TABLE
      CLR      NOPAR          ;INDICATE PARITY ACTION
      RETURN

```

```

;WARNING IF YOU CHANGE THIS TABLE ALSO
;CHANGE '$DDW0' - '$DDW5' (THE PATTERN BIT MAP)

```

```

7181 017430
7182 017430 023346
7183 017432 020222
7184 017434 020434
7185 017436 020256
7186 017440 017544
7187 017442 017640
7188 017444 017754
7189 017446 020070
7190 017450 020140
7191 017452 020536
7192 017454 023446
7193 017456 021026
7194 017460 021060
7195 017462 021332
7196 017464 021124
7197 017466 022454
7198 017470 022644
7199 017472 023160
7200 017474 023346
7201
7202 017476 024220
7203 017500 024220
7204 017502 024220
7205 017504 024220
7206 017506 024220

```

```

      ;PAT      TIME      DISCRIPTION
MJPAT: ;NOTE MT0034 MUST BE FIRST & LAST
      MT0034 ;<1 SEC      ;SOFT ERROR - BACKGROUND PATTERN TEST
      MT0006 ;<1 SEC      ;INITIAL DATA TEST
      MT0017 ;<1 SEC      ;HOLDING 1'S & 0'S TEST
      MT0007 ;<1 SEC      ;ADDRESS BIT TEST
      MT0001 ;<1 SEC      ;ADDRESS TEST
      MT0002 ;<1 SEC      ;COMPLEMENT ADDRESS TEST
      MT0003 ; 1 SEC      ;3 XOR 9 WORST CASE NOISE TEST
      MT0004 ; 1 SEC      ;ROTATING ZEROS TEST
      MT0005 ; 1 SEC      ;ROTATING ONES TEST
      MT0021 ; 1 SEC      ;MARCHING 0'S & 1'S TEST
      MT0035 ;<1 SEC      ;WORSE CASE NOISE PARITY TEST
      MT0022 ;10 SEC      ;REFRESH TEST
      MT0023 ;10 SEC      ;SHIFTING DIAGONAL TEST
      MT0026 ;<1 SEC      ;RANDOM DATA TEST
      MT0024 ;20 SEC      ;FAST GALLOPING PATTERN TEST
      MT0031 ; 3 SEC      ;SOB-A-LONG TEST
      MT0032 ;<1 SEC      ;WRITE RECOVERY TEST
      MT0033 ;35 SEC      ;BRANCH GOBBLE TEST
      MT0034 ;<1 SEC      ;SOFT ERROR - BACKGROUND PATTERN TEST
      ;NOTE MT0034 MUST BE FIRST & LAST
      MT0999 ; 0 SEC      ;NULL TEST
      MT0999 ; 0 SEC      ;NULL TEST
      MT0999 ; 0 SEC      ;NULL TEST
      MT0999 ; 0 SEC      ;NULL TEST
      MT0999 ; 0 SEC      ;NULL TEST

```


7208
7209
7210
7211 017510

.SBTTL PATTERNS

.SBTTL MEMORY TEST SETUP ROUTINES

MT0000: SUBTST <<MT0000 SETUP DATA PATTERN TEST>>

;*****

;*SUBTEST MT0000 SETUP DATA PATTERN TEST

;*****

CLR REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

MOV #FIRST,R0

MOV #SIZE,R1

CALL REGCOPY

MOV #MTP000,SUPDOADD ;ELSE DO PATTERN IN MAIN MEMORY

CALL SUPD03

RETURN

MT0001: SUBTST <<MT0001 SETUP ADDRESS TEST>>

;*****

;*SUBTEST MT0001 SETUP ADDRESS TEST

;*****

MOV #1,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

MOV #FIRST,R0

MOV #SIZE,R1

TST NOSUPER

BNE 2\$

CMP SIPAR5,SIPAR6

BNE 4\$

BR 3\$

CMP UIPAR5,UIPAR6

BNE 4\$

MOV #30000,R1

CLR R2

CALL REGCOPY

MOV #MTP001,SUPDOADD ;SET UP CALLING ADDRESS

CALL SUPD03

RETURN

7212 017510 005037 002300
7213 017514 012700 060000
7214 017520 012701 040000
7215 017524 004737 032470
7216 017530 012737 024640 002262
7217 017536 004737 024446
7218 017542 000207
7219 017544

7220 017544 012737 000001 002300
7221 017552 012700 060000
7222 017556 012701 040000
7223 017562 005737 002456
7224 017566 001005
7225 017570 023737 172252 172254
7226 017576 001007
7227 017600 000404
7228 017602 023737 177652 177654 2\$:
7229 017610 001002
7230 017612 012701 030000 3\$:
7231 017616 005002 4\$:
7232 017620 004737 032470
7233 017624 012737 024664 002262
7234 017632 004737 024446
7235 017636 000207

7236 017640

MT0002: SUBTST <<MT0002 SETUP COMPLEMENT ADDRESS TEST>>

;*****
;SUBTEST MT0002 SETUP COMPLEMENT ADDRESS TEST
;*****

7237	017640	012737	000002	002300	MOV	#2,REALPAT	;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7238	017646	012700	160000		MOV	#LAST+2,R0	
7239	017652	012701	040000		MOV	#SIZE,R1	
7240	017656	012704	060000		MOV	#FIRST,R4	
7241	017662	012705	100001		MOV	#100001,R5	
7242	017666	005737	002456		TST	NOSUPER	
7243	017672	001005			BNE	2\$	
7244	017674	023737	172252	172254	CMP	SIPAR5,SIPAR6	
7245	017702	001013			BNE	4\$	
7246	017704	000404			BR	3\$	
7247	017706	023737	177652	177654	2\$: CMP	UIPAR5,UIPAR6	
7248	017714	001006			BNE	4\$	
7249	017716	012701	030000		3\$: MOV	#30000,R1	
7250	017722	012700	140000		MOV	#140000,R0	
7251	017726	012705	120001		MOV	#120001,R5	
7252	017732	012702	000001		4\$: MOV	#1,R2	
7253	017736	010103			MOV	R1,R3	
7254	017740	012737	024716	002262	MOV	#MTP002,SUPDOADD	;SET UP CALLING ADDRESS
7255	017746	004737	024446		CALL	SUPD03	
7256	017752	000207			RETURN		

7259 017754

MT0003: SUBTST <<MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST>>

```

*****
;SUBTEST      MT0003  SETUP 3 XOR 9 WORST CASE NOISE TEST
*****
IF EQFLAG IS TRUE THEN $RETURN

```

```

7260 017754 005737 002132
      017754 001401
      017760 000207
      017762
      017764

```

```

7261 017764 012737 000003 002300
7262 017772 005037 002326
7263 017776 004737 032500
7264 020002 012701 060000
7265 020006 012703 020000
7266 020012 072327 177770
7267 020016 012702 000004
7268 020022 012705 000100
7269 020026 104415
7270 020030 012737 024750 002262
7271 020036 004737 024446
7272 020042 104416
7273 020044 012737 025010 002262
7274 020052 004737 024462
7275 020056 022737 000003 002616
7276
7277 020064 001344
7278 020066 000207
7279
7280 020070

```

```

      MOV      #3,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      CLR      PCBUMP          ;TRAPS DO NOT ADD TO PC
      CALL     FLIPWARN        ;SETUP WARNING CONSTANTS & R2
      MOV      #FIRST,R1      ;R1 --- STARTING ADDRESS
      MOV      #20000,R3
      ASH      #-8.,R3        ;R3 --- R3 / 256.
      MOV      #4,R2          ;SMALL LOOP SIZE
      MOV      #64.,R5        ;MEDIUM LOOP SIZE
      SAVREG
      MOV      #MTPA03,SUPDOADD
      CALL     SUPD03          ;DO IT IN MAIN MEMORY
      RESREG
      MOV      #MTPB03,SUPDOADD
      CALL     SUPD04
      CMP      #3,FLIPL0C      ;DONE WITH 4 PATTERNS
      BNE      1$              ;[(0,177777):(177777,0):(401,177777):(177777,401)]?
      RETURN                  ;NO - LOOP

```

MT0004: SUBTST <<MT0004 SETUP ROTATING ZEROS TEST>>

```

*****
;SUBTEST      MT0004  SETUP ROTATING ZEROS TEST
*****

```

```

7281 020070 012737 000004 002300
7282 020076 012737 000004 002326
7283 020104 013702 002614
7284 020110 004737 032630
7285 020114 012700 060000
7286 020120 012701 040000
7287 020124 012737 025106 002262
7288 020132 004737 024462
7289 020136 000207
7290 020140

```

```

      MOV      #4,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #4,PCBUMP        ;TRAPS ADD 4 TO PC
      MOV      ONES,R2
      CALL     BACKGND          ;WRITE BACKGROUND OF ONES
      MOV      #FIRST,R0
      MOV      #SIZE,R1
      MOV      #MTPA04,SUPDOADD ;SET UP LINKS
      CALL     SUPD04
      RETURN

```

MT0005: SUBTST <<MT0005 SETUP ROTATING ONES TEST>>

```

*****
;SUBTEST      MT0005  SETUP ROTATING ONES TEST
*****

```

```

7291 020140 012737 000005 002300
7292 020146 012737 000004 002326
7293 020154 005002
7294 020156 004737 032630
7295 020162 012700 060000
7296 020166 012701 040000
7297 020172 012737 025162 002262
7298 020200 012737 025176 025160
7299 020206 004737 024462
7300 020212 012737 025122 025160
7301 020220 000207

```

```

      MOV      #5,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      MOV      #4,PCBUMP        ;TRAPS ADD 4 TO PC
      CLR      R2
      CALL     BACKGND          ;WRITE BACKGROUND OF ZEROS
      MOV      #FIRST,R0
      MOV      #SIZE,R1
      MOV      #MTP005,SUPDOADD ;SET UP LINKS
      MOV      #MTP005*14,MTPB04*16
      CALL     SUPD04
      MOV      #MTPA04*14,MTPB04*16 ;RESET TEST'S ORIGINAL VALUE
      RETURN

```

7304 020222

MT0006: SUBTST <<MT0006 SETUP INITIAL DATA TEST>>

;*****

;*SUBTEST MT0006 SETUP INITIAL DATA TEST

;*****

7305 020222 012737 000006 002300

MOV #6,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

7306 020230 012737 000004 002326

MOV #4,PCBUMP ;TRAPS ADD 4 TO PC

7307 020236 012701 002412

MOV #TESTADD,R1

7308 020242 012737 025216 002262

MOV #MTP006,SUPDOADD

7309 020250 004737 024446

CALL SUPD03

;DO IT IN SUPERVISOR MODE

7310 020254 000207

RETURN

7311 020256

MT0007: SUBTST <<MT0007 SETUP ADDRESS BIT TEST>>

;*****

;*SUBTEST MT0007 SETUP ADDRESS BIT TEST

;*****

7312 020256 012737 000007 002300

MOV #7,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

7313 020264 005002

CLR R2

7314 020266 004737 032630

CALL BACKGND ;OF ZEROS

7315 020272 012701 060000

MOV #FIRST,R1

7316 020276 012702 000001

MOV #1,R2

7317 020302 050201

BIS R2,R1

7318 020304 012737 025416 002262

MOV #MTP007,SUPDOADD

7319 020312 004737 024446

CALL SUPD03

;DO IT IN SUPERVISOR MODE

7320 020316 000207

RETURN

7321 020320

MT0010: SUBTST <<MT0010 SETUP BYTE ADDRESSING TEST>>

;*****

;*SUBTEST MT0010 SETUP BYTE ADDRESSING TEST

;*****

7322 020320 012737 000010 002300

MOV #10,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

7323 020326 012737 000004 002326

MOV #4,PCBUMP ;TRAPS ADD 4 TO PC

7324 020334 013704 002412

MOV TESTADD,R4

7325 020340 012737 025516 002262

MOV #MTP010,SUPDOADD

7326 020346 004737 024446

CALL SUPD03

;DO IT IN SUPERVISOR MODE

7327 020352 000207

RETURN

7330 020354

MT0014: SUBTST <<MT0014 SETUP BASIC DOUBLE BIT ERROR TEST>>

;*****
;SUBTEST MT0014 SETUP BASIC DOUBLE BIT ERROR TEST
;*****

IF ACTFLAG TS TRUE OR APTFLAG IS TRUE

7331 020354 005737 002350
020360 001003
020362 005737 002352
020366 001404
020370TST ACTFLAG
BNE L160
TST APTFLAG
BEQ L161

L160:::;;

7332 020370
020370 005737 056654
020374 001401
020376 000207
020400

IF \$PASS NE #0 THEN \$RETURN

TST \$PASS
BEQ L162
RTS PC

L162:::;;

7333 020400
020400
7334 020400 012737 000014 002300
7335 020406 004737 037740
7336 020412

END; OF IF ACTFLAG

L161:::;;

MOV #14,REALPAT
CALL MAPKERNAL
LET R1 := #100000;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
;MAP KERNAL SPACE
;SETUP TEST ADDRESS020412 012701 100000
7337 020416 004737 032674
7338 020422 004737 025624
7339 020426 004737 040026
7340 020432 000207CALL GETCSR
CALL MTP014
CALL UNMAP
RETURNMOV #100000,R1
;GET CSR INFO FROM CONFIGURATION TABLE
;DO BASIC DOUBLE BIT ERROR TEST
;UNMAP KERNAL SPACE

B16

VMJAA0 MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15-Feb 85 16:10 Page 208
MT0014 SETUP BASIC DOUBLE BIT ERROR TEST

SEQ 0196

7343
7344 020434

MT0017: SUBTST <<MT0017 SETUP HOLDING 1'S & 0'S>>
;*****
;SUBTEST MT0017 SETUP HOLDING 1'S & 0'S
;*****
7345 020434 012737 000017 002300 MOV #17,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7346 020442 012737 026050 002262 MOV #MTP017,SUPDOADD
7347 020450 004737 024446 CALL SUPD03 ;DO IT IN SUPERVISOR MODE
7348 020454 000207 RETURN

7351 020456

```

MT0020: SUBTST <<MT0020      SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR>>
;*****
;SUBTEST      MT0020  SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR
;*****
      IF ACTFLAG IS TRUE OR APTFLAG IS TRUE

```

7352 020456

```

020456 005737 002350
020462 001003
020464 005737 002352
020470 001404
020472

```

```

      TST ACTFLAG
      BNE L163
      TST APTFLAG
      BEQ L164

```

L163.::::;;

7353 020472

```

020472 005737 056654
020476 001401
020500 000207
020502

```

IF \$PASS NE #0 THEN \$RETURN

```

      TST $PASS
      BEQ L165
      RTS PC

```

L165.::::;;

7354 020502

020502

END; OF IF ACTFLAG

L164.::::;;

7355 020502 012737 000020 002300

7356 020510 004737 037740

7357 020514 012701 100000

7358 020520 004737 032674

7359 020524 004737 026126

7360 020530 004737 040026

7361 020534 000207

```

MOV      #20,REALPAT
CALL MAPKERNAL
LET R1 := #100000

```

```

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
;MAP KERNAL SPACE
;SETUP TEST ADDRESS

```

```

CALL GETCSR
CALL MTP020
CALL UNMAP
RETURN

```

```

      MOV #100000,R1
;GET CSR INFO FROM CONFIGURATION TABLE
;DO SYNDROMES TO CSR ON SINGLE ERROR TEST
;UNMAP KERNAL SPACE

```

7363 020536

MT0021: SUBTST <<MT0021 SETUP MARCHING 0'S & 1'S TEST>>

;*****

;*SUBTEST MT0021 SETUP MARCHING 0'S & 1'S TEST

;*****

SET NOSCOPE

MOV #-1,NOSCOPE

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

7364	020536	012737	177777	002440		
7365	020544	012737	000021	002300		
7366	020552	013702	002632			
7367	020556	004737	032630			
7368	020562	010203				
7369	020564	000303				
7370	020566	012701	160000			
7371	020572	010105				
7372	020574	012704	060000			
7373	020600	022737	000005	004010		
7374	020606	001450				
7375	020610	022737	000003	004010		
7376	020616	001407				
7377	020620	022737	000007	002102		
7378	020626	001003				
7379	020630	012701	140000			
7380	020634	010105				
7381	020636	022737	000177	002102	3\$:	
7382	020644	001003				
7383	020646	012701	140000			
7384	020652	010105				
7385	020654	012737	026442	002262	5\$:	
7386	020662	004737	024446			
7387	020666	012737	026472	002262		
7388	020674	004737	024462			
7389	020700	010401				
7390	020702	012737	026526	002262		
7391	020710	004737	024462			
7392	020714	012737	026562	002262		
7393	020722	004737	024462			
7394	020726	000434				
7395	020730	022737	000177	002102	1\$:	
7396	020736	001003				
7397	020740	012701	140000			
7398	020744	010105				
7399	020746	012737	026442	002262	4\$:	
7400	020754	004737	024446			
7401	020760	012737	026472	002262		
7402	020766	004737	024462			
7403	020772	010401				
7404	020774	012737	026526	002262		
7405	021002	004737	024462			
7406	021006	012737	026562	002262		
7407	021014	004737	024462			
7408	021020	005037	002440		2\$:	
7409	021024	000207				

MOV #21,REALPAT

MOV BAKPAT,R2

CALL BACKGND

MOV R2,R3

SWAB R3

MOV #LAST+2,R1

MOV R1,R5

MOV #FIRST,R4

CMP #5,PROTYP

BEQ 1\$

CMP #3,PROTYP

BEQ 3\$

CMP #7,BANK

BNE 3\$

MOV #140000,R1

MOV R1,R5

CMP #177,BANK

BNE 5\$

MOV #140000,R1

MOV R1,R5

MOV #MTPA21,SUPDOADD

CALL SUPD03

MOV #MTPB21,SUPDOADD

CALL SUPD04

MOV R4,R1

MOV #MTPC21,SUPDOADD

CALL SUPD04

MOV #MTPD21,SUPDOADD

CALL SUPD04

BR 2\$

CMP #177,BANK

BNE 4\$

MOV #140000,R1

MOV R1,R5

MOV #MTPA21,SUPDOADD

CALL SUPD03

MOV #MTPB21,SUPDOADD

CALL SUPD04

MOV R4,R1

MOV #MTPC21,SUPDOADD

CALL SUPD04

MOV #MTPD21,SUPDOADD

CALL SUPD04

CLR NOSCOPE

RETURN

;IS THIS AN 11/83

;BRANCH IF IT IS

;IS THIS AN 11/24?

;BRANCH IF SO

7411 021026

MT0022: SUBTST <<MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST>>

;*****

;*SUBTEST MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST

;*****

7412 021026 004737 024234

7413 021032

021032 103001

021034 000207

021036

7414 021036 012737 000022 002300

7415 021044 012737 026612 002262

7416 021052 004737 024446

7417 021056 000207

7418

7419 021060

CALL KAMITEST

ON.ERROR THEN \$RETURN

;CHECK FOR KAMIKAZE MODE

;IF NOT IN KAMIKAZE MODE RETURN

BCC L166

RTS PC

L166:::~::~

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

MOV #22,REALPAT

MOV #MTP022,SUPDOADD

CALL SUPD03

RETURN

;DO IT IN SUPERVISOR MODE

MT0023: SUBTST <<MT0023 SHIFTING DIAGONAL TEST>>

;*****

;*SUBTEST MT0023 SHIFTING DIAGONAL TEST

;*****

7420 021060 004737 024234

7421 021064

021064 103001

021066 000207

021070

7422 021070 012737 000023 002300

7423 021076 012737 026612 002262

7424 021104

021104 012737 177777 002004

7425 021112 004737 024446

7426 021116 005037 002004

7427 021122 000207

CALL KAMITEST

ON.ERROR THEN \$RETURN

;CHECK FOR KAMIKAZE MODE

;IF NOT IN KAMIKAZE MODE RETURN

BCC L167

RTS PC

L167:::~::~

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

MOV #23,REALPAT

MOV #MTP022,SUPDOADD

SET DIAGFLAG

;IDENTIFY DIAGONAL TEST TO MTP022

MOV #-1,DIAGFLAG

CALL SUPD03

CLR DIAGFLAG

RETURN

;DO IT IN SUPERVISOR MODE

7429 021124

MT0024: SUBTST <<MT0024 SETUP FAST GALLOPING PATTERN TEST>>

;*****
;SUBTEST MT0024 SETUP FAST GALLOPING PATTERN TEST
;*****

7430 021124 004737 024234

7431 021130

021130 103001

021132 000207

021134

7432 021134

021134 012737 177777 002440

7433 021142 012737 000024 002300

7434 021150 013702 002632

7435 021154 004737 032630

7436 021160 010203

7437 021162 010304

7438 021164 000304

7439 021166 012701 060000

7440 021172 012705 157776

7441 021176 022737 000005 004010

7442 021204 001426

7443 021206 022737 000003 004010

7444 021214 001406

7445 021216 022737 000007 002102

7446 021224 001002

7447 021226 012705 137776

7448 021232 022737 000177 002102 3\$:

7449 021240 001003

7450 021242 012701 137776

7451 021246 010105

7452 021250 104415 7\$:

7453 021252 012737 027326 002262

7454 021260 000412

7455 021262 022737 000177 002556 1\$:

7456 021270 001002

7457 021272 012705 137776

7458 021276 104415 4\$:

7459 021300 012737 027326 002262

7460 021306 004737 024462 2\$:

7461

7462 021312 104416

7463 021314 000302

7464 021316 000303

7465 021320 004737 024462

7466 021324 005037 002440

7467 021330 000207

CALL KAMITEST

ON.ERROR THEN \$RETURN

;CHECK FOR KAMIKAZE MODE

;IF NOT IN KAMIKAZE MODE RETURN

BCC L170

RTS PC

L170:::;;

SET NOSCOPE

MOV #1,NOSCOPE

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

MOV #24,REALPAT

MOV BAKPAT,R2

CALL BACKGND

MOV R2,R3

MOV R3,R4

SWAB R4

MOV #FIRST,R1

MOV #LAST,R5

CMP #5,PROTYP

BEQ 1\$

CMP #3,PROTYP

BEQ 3\$

CMP #7,BANK

BNE 3\$

MOV #137776,R5

CMP #177,BANK

BNE 7\$

MOV #137776,R1

MOV R1,R5

SAVREG

MOV #MTPB24,SUPDOADD

BR 2\$

CMP #177,LASTBANK

BNE 4\$

MOV #137776,R5

SAVREG

MOV #MTPB24,SUPDOADD

CALL SUPD04

;DO IT AGAIN FOR COMPLEMENT DATA

RESREG

SWAB R2

SWAB R3

CALL SUPD04

CLR NOSCOPE

RETURN

7470 021332

MT0026: SUBTST <<MT0026 SETUP RANDOM DATA TEST>>

;*****
;SUBTEST MT0026 SETUP RANDOM DATA TEST
;*****

7471	021332	012737	000026	002300	MOV	#26,REALPAT	
7472	021340	005037	002326		CLR	PCBUMP	
7473	021344	013703	002604		MOV	SEEDLO,R3	;TRAPS DO NOT ADD TO THE PC
7474	021350	013702	002602		MOV	SEEDHI,R2	;INITIALIZE RANDOM NUMBERS
7475	021354	010305			MOV	R3,R5	
7476	021356	010204			MOV	R2,R4	
7477	021360	012701	060000		MOV	#FIRST,R1	
7478	021364	012700	020000		MOV	#SIZE/2,R0	
7479	021370	022737	000005	004010	CMP	#5,PROTYP	;DO WE HAVE AN 11/83 ?
7480	021376	001445			BEQ	1\$;BRANCH IF WE DO
7481	021400	022737	000003	004010	CMP	#3,PROTYP	;11/24?
7482	021406	001406			BEQ	3\$;BRANCH IF SO
7483	021410	022737	000007	002102	CMP	#7,BANK	
7484	021416	001002			BNE	3\$	
7485	021420	012700	014000		MOV	#14000,R0	
7486	021424	022737	000177	002102	CMP	#177,BANK	
7487	021432	001002			BNE	7\$	
7488	021434	012700	014000		MOV	#14000,R0	
7489	021440	104415			SAVREG		
7490	021442	012737	027364	027464	MOV	#MTPA26+4,MTPD26+14	
7491	021450	012737	027360	002262	MOV	#MTPA26,SUPDOADD	
7492	021456	004737	024446		CALL	SUPD03	
7493	021462	005037	027410		CLR	RANODD	;FOR ERROR REPORTING
7494	021466	012737	027400	027464	MOV	#MTPB26+4,MTPD26+14	;SET UP NEXT LINK
7495	021474	012737	027374	002262	MOV	#MTPB26,SUPDOADD	
7496	021502	104416			RESREG		
7497	021504	004737	024446		CALL	SUPD03	
7498	021510	000432			BR	2\$	
7499	021512	022737	000177	002102	CMP	#177,BANK	
7500	021520	001002			BNE	4\$	
7501	021522	012700	014000		MOV	#14000,R0	
7502	021526	104415			SAVREG		
7503	021530	012737	027364	027464	MOV	#MTPA26+4,MTPD26+14	
7504	021536	012737	027360	002262	MOV	#MTPA26,SUPDOADD	
7505	021544	004737	024446		CALL	SUPD03	
7506	021550	005037	027410		CLR	RANODD	;FOR ERROR REPORTING
7507	021554	012737	027400	027464	MOV	#MTPB26+4,MTPD26+14	;SET UP NEXT LINK
7508	021562	012737	027374	002262	MOV	#MTPB26,SUPDOADD	
7509	021570	104416			RESREG		
7510	021572	004737	024446		CALL	SUPD03	
7511	021576	010337	002604		MOV	R3,SEEDLO	;UPDATE FOR NEW RANDOM NUMBERS
7512	021602	010237	002602		MOV	R2,SEEDHI	
7513	021606	000207			RETURN		

```
7516 021610 MT0027: SUBTST <<MT0027      UNIQUE BANK TEST>>
;*****
;SUBTEST      MT0027  UNIQUE BANK TEST
;*****
;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
MOV      #27,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLRCSR      ;CLEAR CSRS
MOV      #SUPD03,LINK1    ;SET UP LINK
SET NOFSMODE

7517
7518
7519 021610 012737 000027 002300
7520 021616 104502
7521 021620 012737 024446 002522
7522 021626 012737 177777 002426
7523 021634 012737 000001 002452
7524 021642
7525 021646 004737 040052
7526 021652
7527 021652 005737 002116
7528 021656 001436
7529 021660 005737 002124
7530 021664 001033
7531 021666 104511
7532 021670
7533 021670 013702 002102
7534 021674 012700 060000
7535 021700 010004
7536 021702 012701 040000
7537 021706 010103
7538 021710 023727 002452 000001
7539 021716 001005
7540 021720 012737 027704 002262
7541 021726 004777 160570
7542 021732
7543 021732
7544 021732 023727 002452 000002
7545 021740 001005
7546 021742 012737 027712 002262
7547 021750 004737 024446
7548 021754
7549 021754
7550 021754
7551 021754 005237 002102
7552 021760 023737 002102 002556
7553 021766 003727
7554 021770
7555 021770 005237 002452
7556 021774 023727 002452 000002
7557 022002 003717
7558 022004
7559 022004

STAR27: FOR I := #1 TO #2

FOR BANK := #0 TO LASTBANK

CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE

INVALIDATE
LET R2 := BANK

MOV      #FIRST,R0
MOV      R0,R4
MOV      #SIZE,R1
MOV      R1,R3
IF I EQ #1

MOV      #MTP034,SUPDOADD
CALL     @LINK1
END ;OF IF

IF I EQ #2

MOV      #MTP034+6,SUPDOADD
CALL     SUPD03
END ;OF IF

END ;OF IF

END ;OF FOR BANK

END ;OF FOR I

IF FS7FLAG IS TRUE
```

```

022004 005737 002446
022010 001403
7545 022012 005037 002426
7546 022016 000207
7547 022020
022020
7548 022020
022020 012737 000001 002452
022026
7549 022026
022026 013737 002556 002102
022034
7550 022034 004737 040052
7551 022040
022040 005737 002116
022044 001436
022046 005737 002124
022052 001033
7552 022054
022054 013702 002102
7553 022060 005102
7554 022062 012700 060000
7555 022066 010004
7556 022070 012701 040000
7557 022074 010103
7558 022076
022076 023727 002452 000001
022104 001005
7559 022106 012737 027704 002262
7560 022114 004777 160402
7561 022120
022120
7562 022120
022120 023727 002452 000002
022126 001005
7563 022130 012737 027712 002262
7564 022136 004737 024446
7565 022142
022142
7566 022142
022142
7567 022142
022142 005337 002102
022146 023727 002102 000000
022154 002327
022156
7568 022156
022156 005237 002452
022162 023727 002452 000002
022170 003716
022172
7569 022172 005037 002426
7570 022176 000207

```

```

CLR NOFSMODE
RETURN
END ;OF IF FS7FLAG

```

```

FOR I := #1 TO #2

```

```

FOR BANK := LASTBANK DOWNT0 #0

```

```

CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE

```

```

LET R2 := BANK

```

```

COM      R2
MOV      #FIRST,R0
MOV      R0,R4
MOV      #SIZE,R1
MOV      R1,R3
IF I EQ #1

```

```

MOV      #MTP034,SUPDOADD
CALL     @LINK1
END ;OF IF

```

```

IF I EQ #2

```

```

MOV      #MTP034+6,SUPDOADD
CALL     SUPD03
END ;OF IF

```

```

END ;OF IF

```

```

END ;OF FOR BANK

```

```

END ;OF FOR I

```

```

CLR NOFSMODE
RETURN

```

```

TST FS7FLAG
BEQ L174

```

```

L174:::

```

```

MOV #1,I
B26:::

```

```

MOV LASTBANK,BANK
B27:::

```

```

TST ACFLAG
BEQ L175
TST RRFLAG
BNE L175

```

```

MOV BANK,R2

```

```

CMP I,#1
BNE L176

```

```

L176:::

```

```

CMP I,#2
BNE L177

```

```

L177:::

```

```

L175:::

```

```

DEC BANK
CMP BANK,#0
BGE B27
E27:::

```

```

INC I
CMP I,#2
BLE B26
E26:::

```

7573 022200

MT0030: SUBTST <<MT0030 SETUP FLUSH OUT DBE'S TEST>>

;*****
;*SUBTEST MT0030 SETUP FLUSH OUT DBE'S TEST
;*****

7574 022200 005037 002264

7575 022204 012737 177777 002542

7576 022212 012737 000030 002300

7577 022220 012737 000001 002076

7578 022226 012737 024446 002522

7579 022234 012737 027466 002262

7580 022242 104470

7581 022244 012737 177777 002426

022252 012737 177777 002440

7582 022260 005037 002102

022264 004737 040052

7583 022264 004737 040052

7584 022270 005737 002120

022274 001414

7585 022276 005737 002116

022302 001411

022304 005737 002124

022310 001006

7586 022312 012701 040000

7587 022316 012700 060000

7588 022322 004777 160174

7589 022326 005237 002102

022332 023737 002102 002556

022340 003751

7592 022342 005737 002264

022346 001032

7593 022350 012737 177777 002264

7594 022356 104502

7595 022360 004737 036402

7596 022364 103010

7597 022366 104472

7598 022370 005037 002426

022374 005037 002440

022400 005037 002542

7599 022404 000207

7600 022406 013737 002310 002102

7601 022406 013737 002310 002102

CLR PASFLG

SET FULLREL

MTA030: MOV #30,REALPAT

MOV #1,NOPAR

MOV #SUPD03,LINK1

MOV #MTP030,SUPDOADD

ECCDIS

SET NOFSMODE,NOSCOPE

FOR BANK := #0 TO LASTBANK

CALL EXBANK

IF MKFLAG IS TRUE

IF ACFLAG IS TRUE AND RRFLAG IS FALSE

MOV #SIZE,R1

MOV #FIRST,R0

CALL @LINK1

END ;OF IF ACFLAG

END ;OF IF MKFLAG

END ;OF FOR

SET PASFLG

CLRCR

CALL RELOCATE

ON.ERROR

ECCINIT

CLEAR

;TRAP ON DOUBLE BIT ERRORS (NORMAL)

NOFSMODE,NOSCOPE,FULLREL

RETURN

END ;OF ON.ERROR

MOV NEWBANK,BANK

MOV #-1,FULLREL

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

;INDICATE COUNT PARITY ERRORS

;DISABLE ERROR CORRECTION

MOV #-1,NOFSMODE

MOV #-1,NOSCOPE

CLR BANK

B30:::;:

TST MKFLAG

BEQ L200

TST ACFLAG

BEQ L201

TST RRFLAG

BNE L201

L201:::;:

L200:::;:

INC BANK

CMP BANK, LASTBANK

BLE B30

E30:::;:

TST PASFLG

BNE L202

MOV #-1,PASFLG

;CLEAR CSRS

BCC L203

CLR NOFSMODE

CLR NOSCOPE

CLR FULLREL

L203:::;:

K16

CVMJAAO MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15 Feb 85 16:10 Page 219-1
MT0030 SETUP FLUSH OUT DBE'S TEST

SEQ 0205

7602	022414	004737	040052	CALL	EXBANK		
7603	022420	004737	022212	CALL	MTA030		
7604	022424	104472		ECCINIT		;TRAP ON DOUBLE BIT ERRORS (NORMAL)	
7605	022426	004737	037214	CALL	UNRELOCATE		
7606	022432	000207		RETURN			
7607	022434			END ;OF IF PASFLG			
	022434					L202:::~::~	
7608	022434	104472		ECCINIT		;TRAP ON DOUBLE BIT ERRORS (NORMAL)	
7609	022436			CLEAR	NOFSMODE,NOSCOPE,FULLREL		
	022436	005037	002426			CLR	NOFSMODE
	022442	005037	002440			CLR	NOSCOPE
	022446	005037	002542			CLR	FULLREL
7610	022452	000207		RETURN			

```

7613 022454      MT0031: SUBTST <<MT0031      SETUP SOB-A-LONG TEST>>
;*****
;SUBTEST      MT0031      SETUP SOB-A-LONG TEST
;*****
7614 022454      004737      024234      CALL      KAMITEST      ;CHECK FOR KAMIKAZE MODE
7615 022460      103001      CN.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
      022462      000207      ;*****
      022464      ;*****
7616 022464      012737      177777      002440      SET      NOSCOPE      ;*****
      022464      012737      000031      002300      MOV      #31,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7617 022472      012737      000031      002300      CLR      NOPAR      ;SETUP PARITY ACTION
7618 022500      005037      002076      MAP      BANK      ;MAP FIRST SO BLOCK MOVE WORKS
7619 022504      010346      MOV      R3,-(SP)
      022506      013703      002102      MOV      BANK,R3
      022512      004737      035716      CALL      MAPPER
      .DSABL      CRF
      022516      012603      MOV      (SP)+,R3
7620 022520      053737      002552      177776      TESTAREA      ;ENTER TEST MODE
      .DSABL      CRF      ;GO TO SYSTEM TEST MODE
7621 022526      004537      041024      BMOV      MTP031,FIRST,SOBLENGTH/2
      022526      000027      JSR      R5,BLOCK3
      022532      060000      SOBLENGTH/2
      022534      027476      FIRST
      022536      027476      MTP031
      .DSABL      CRF
7622 022540      104417      KERNEL      ;ENTER KERNEL MODE
7623 022542      013702      002572      MOV      SOBK,R2
7624 022546      010200      MOV      R2,R0
7625 022550      012701      100776      MOV      #100776,R1
7626 022554      012705      060056      MOV      #FIRST+SOBLENGTH,R5
7627 022560      012737      060002      002262      MOV      #FIRST+2,SUPDOADD
7628 022566      012737      160000      002522      MOV      #LAST+2,LINK1
7629 022574      005737      002456      TST      NOSUPER
7630 022600      001005      BNE      1$
7631 022602      023737      172252      172254      CMP      SIPAR5,SIPAR6
7632 022610      001405      BEQ      2$
7633 022612      000407      BR      3$
7634 022614      023737      177652      177654      1$:      CMP      UIPAR5,UIPAR6
7635 022622      001003      BNE      3$
7636 022624      012737      140000      002522      2$:      MOV      #140000,LINK1
7637 022632      004737      024462      3$:      CALL      SUPD04
7638 022636      005037      002440      CLR      NOSCOPE
7639 022642      000207      RETURN

```


7642 022644

MT0032: SUBTST <<MT0032 SETUP WRITE RECOVERY TEST>>

 ;SUBTEST MT0032 SETUP WRITE RECOVERY TEST

7643 022644 004737 024234

7644 022650

022650 103001

022652 000207

022654

7645 022654

022654 012737 177777 002440

7646 022662 012737 000032 002300

7647 022670 005037 002076

7648 022674

022674 010346

022676 013703 002102

022702 004737 035716

CALL KAMITEST
 ON.ERROR THEN \$RETURN

;CHECK FOR KAMIKAZE MODE
 ;IF NOT IN KAMIKAZE MODE RETURN
 BCC L205
 RTS PC

L205:::~::~

SET NOSCOPE

MOV #32,REALPAT
 CLR NOPAR
 MAP BANK

MOV #-1,NOSCOPE
 ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 ;SETUP PARITY ACTION
 ;MAP FIRST SO THAT THE BLOCK MOVE WORKS
 MOV R3,-(SP)

MOV BANK,R3
 CALL MAPPER
 .DSABL CRF

MOV (SP)+,R3

7649 022706 012603

7650 022710 012700 010247

7651 022714 012701 177667

7652 022720 012702 020000

7653 022724 010237 002522

7654 022730 012703 060000

7655 022734 012704 160000

7656 022740 005037 002524

7657 022744 005737 002456

7658 022750 001005

7659 022752 023737 172252 172254

7660 022760 001405

7661 022762 000415

7662 022764 023737 177652 177654 1\$:

7663 022772 001011

7664 022774 012704 140000 2\$:

7665 023000 012702 014000

7666 023004 010237 002522

7667 023010 012737 000001 002524

7668

7669 023016 053737 002552 177776 3\$:

023016 053737 002552 177776

TESTAREA
 BIS TESTMODE,PSW
 .DSABL CRF

;ENTER TEST MODE
 ;GO TO SYSTEM TEST MODE

;MOVE TEST TO MEMORY UNDER TEST

7670 023024 010023

7671 023026 010144

7672 023030 077203

7673

7674 023032 022737 000005 004010

7675 023040 001003

7676

7677 023042

023042 004537 040770

023046 027554

MOV R0,(R3)+
 MOV R1,-(R4)
 SOB R2,4\$

CMP #5,PROTYP
 BNE 5\$

;MOVE LAST PART OF TEST TO FASTCITY

BMOV MTP032
 JSR R5,BLOCK1
 MTP032
 .DSABL CRF

7678 023050 104417

7679

7680 023052 012702 005141

7681 023056 012700 023156

5\$: KERNEL

;ENTER KERNEL MODE

MOV #5141,R2
 MOV #10\$,R0

;OP CODE OF INSTRUCTION "COM (R1)
 ;ADDRESS TO RETURN TO IN R0

7682	023062	012701	160000		MOV	#LAST.2,R1			
7683	023066	012737	060000	002262	MOV	#FIRST,SUPDOADD			:TOP OF BANK
7684	023074	005737	002524		TST	LINK2			
7685	023100	001402			BEQ	6\$			
7686	023102	012701	140000		MOV	#140000,R1			
7687	023106	004737	024462	6\$:	CALL	SUPD04			
7688	023112	012703	020000		MOV	#SIZE/2,R3			
7689	023116	012705	000110		MOV	#110,R5			
7690	023122	012704	060000		MOV	#FIRST,R4			
7691	023126	005737	002524		TST	LINK2			
7692	023132	001402			BEQ	7\$			
7693	023134	012703	014000		MOV	#14000,R3			
7694	023140	012737	027554	002262 7\$:	MOV	#MTP032,SUPDOADD			
7695	023146	004737	024462		CALL	SUPD04			
7696	023152	005037	002440	9\$:	CLR	NOSCOPE			
7697	023156	000207		10\$:	RETURN				
7698									
7699									

;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0032
;ALSO A RETURN FROM THE "CALL SUPD04" ABOVE

7702 023160

MT0033: SUBTST <<MT0033 SETUP BRANCH GOBBLE TEST>>

;*****

;SUBTEST MT0033 SETUP BRANCH GOBBLE TEST

;*****

7703 023160 004737 024234

7704 023164

023164 103001

023166 000207

023170

7705 023170

023170 012737 177777 002440

7706 023176 012737 000033 002300

7707 023204 005037 002076

7708 023210

023210 010346

023212 013703 002102

023216 004737 035716

023222 012603

7709

7710 023224

023224 053737 002552 177776

7711 023232

023232 004537 041024

023236 000037

023240 060000

023242 027606

7712 023244 104417

7713

7714 023246 012705 060076

7715 023252 012737 060004 002262

7716 023260 012701 060002

7717 023264 012702 060003

7718 023270 012737 160000 002522

7719 023276 005737 002456

7720 023302 001005

7721 023304 023737 172252 172254

7722 023312 001405

7723 023314 000407

7724 023316 023737 177652 177654 1\$:

7725 023324 001003

7726 023326 012737 140000 002522 2\$:

7727

7728 023334 004737 024462 3\$:

7729 023340 005037 002440

7730 023344 000207

7731

7732 023346

CALL KAMITEST
ON.ERROR THEN \$RETURN;CHECK FOR KAMIKAZE MODE
;IF NOT IN KAMIKAZE MODE RETURNBCC L206
RTS PC

L206:::~::~

SET NOSCOPE

MOV #33,REALPAT
CLR NOPAR
MAP BANKMOV #-1,NOSCOPE
;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
;SETUP PARITY ACTION
;MAP FIRST SO THAT BLOCK MOVE WORKS
MOV R3,-(SP)MOV BANK,R3
CALL MAPPER
.DSABL CRF

MOV (SP)+,R3

TESTAREA ;ENTER TEST MODE

BIS TESTMODE,PSW
.DSABL CRF

;GO TO SYSTEM TEST MODE

BMOV MTP033,FIRST,GBLENGTH/2

JSR R5,BLOCK3

GBLENGTH/2

FIRST

MTP033

.DSABL CRF

KERNEL ;ENTER KERNEL MODE

MOV #FIRST+GBLENGTH,R5

MOV #FIRST+4,SUPDOADD

MOV #FIRST+2,R1

MOV #FIRST+3,R2

MOV #LAST+2,LINK1

TST NOSUPER

BNE 1\$

CMP SIPAR5,SIPAR6

BEQ 2\$

BR 3\$

CMP UIPAR5,UIPAR6

BNE 3\$

MOV #140000,LINK1

CALL SUPDO4

CLR NOSCOPE

RETURN

MT0034: SUBTST <<MT0034 SOFT ERROR - BACKGROUND PATTERN TEST>>

;*****

;SUBTEST MT0034 SOFT ERROR - BACKGROUND PATTERN TEST

;*****

7733 023346 012737 000034 002300

7734 023354 012700 060000

7735 023360 012701 040000

7736 023364 013702 002620

MOV #34,REALPAT
MOV #FIRST,R0
MOV #SIZE,R1
MOV SOFTPA~,R2

```

7737 023370 010103      MOV      R1,R3
7738 023372 013705 002104  MOV      BANKINDEX,R5
7739 023376 010004      MOV      R0,R4
7740 023400      IF #BIT13 SET.IN CONFIG+2(R5)
    023400 032765 020000 002666      BIT #BIT13,CONFIG+2(R5)
    023406 001406      BEQ L207
7741      ;BACKGROUND PATTERN IS VALID
7742 023410 012737 027712 002262  MOV      #MTP034+6,SUPDOADD
7743 023416 004737 024446      CALL SUPD03      ;READ IT
7744 023422      ELSE
    023422 000410      BR L210
    023424      L207:::;
7745      ;BACKGROUND PATTERN HAS BEEN INVALIDATED
7746 023424 012737 027704 002262  MOV      #MTP034,SUPDOADD
7747 023432 004737 024446      CALL SUPD03
7748 023436 052765 020000 002666  BIS      #BIT13,CONFIG+2(R5)      ;VALIDATE IT
7749 023444      END ;OF IF #BIT13
    023444      L210:::;
7750 023444 000207      RETURN
7751
7752 023446      MT0035: SUBTST <<MT0035      SETUP WORST CASE NOISE PARITY TEST>>
;*****
;SUBTEST      MT0035      SETUP WORST CASE NOISE PARITY TEST
;*****
7753 023446 012737 000035 002300      MOV      #35,REALPAT      ;SET UP TEST NUMBER FOR DISPLAY
7754 023454 013703 002104      MOV      BANKINDEX,R3
7755 023460 016301 002664      MOV      CONFIG(R3),R1
7756 023464 000301      SWAB      R1
7757 023466 042701 177760      BIC      #C17,R1
7758 023472 006301      ASL      R1
7759 023474 010137 002152      MOV      R1,CSRNO
7760 023500 023737 002152 002532  CMP      CSRNO,PGMCSR
7761 023506 001001      BNE      1$
7762 023510 000207      RETURN
7763 023512 012702 052524      1$: MOV      #52524,R2
7764 023516 004737 032630      CALL BACKGND      ;WRITE BACKGROUND OF ALMOST ALT. 1'S AND 0'S
7765 023522 012737 027730 002262  MOV      #MTP035,SUPDOADD
7766 023530 004737 024446      CALL SUPD03
7767 023534      IF QVFLAG IS TRUE THEN $RETURN
    023534 005737 002346      TST QVFLAG
    023540 001401      BEQ L211
    023542 000207      RTS PC
    023544      L211:::;
7768 023544 005102      COM      R2
7769 023546 004737 032630      CALL BACKGND      ;WRITE COMPLEMENT PATTERN INTO MUT
7770 023552 004737 024462      CALL SUPD04
7771 023556 000207      RETURN

```

E1

CVMJAA0 MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15 Feb 85 16:10 Page 226
 MTC035 SETUP WORST CASE NOISE PARITY TEST

SEQ 0211

7773 023560

MT0036: SUBTST <<MT0036 SETUP CORRECTION CODE TEST>>

;*****'

,*SUBTEST MT0036 SETUP CORRECTION CODE TEST

;*****

7774 023560 012737 000036 002300
 7775 023566 004737 032674
 7776 023572 005037 002264
 7777 023576 005000
 7778 023600 012701 100000
 7779 023604 004737 037740
 7780 023610 004737 030072
 7781 023614 004737 040026
 7782 023620 000207

MOV #36,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
 CALL GETCSR ;GET CSR INFO FROM CONFIG TABLE
 CLR PASFLG ;CLEAR LOOP COUNTER
 CLR R0 ;GET TEST DATA
 MOV #100000,R1 ;GET FIRST ADDRESS IN BANK
 CALL MAPKERNAL ;MAP KIPARS AND 6 TO BANK
 CALL MTP036 ;EXECUTE TEST
 CALL UNMAP ;REMAP KERNAL SPACE
 RETURN

7784 023622

MT0037: SUBTST <<MT0037 SETUP ECC DISABLE TEST>>

;*****

;*SUBTEST MT0037 SETUP ECC DISABLE TEST

;*****

7785 023622 012737 000037 002300
7786 023630 012701 100000
7787 023634 005000
7788 023636 004737 037740
7789 023642 004737 032674
7790 023646 004737 030316
7791 023652 004737 040026
7792 023656 000207

MOV #37,REALPAT
MOV #100000,R1
CLR R0
CALL MAPKERNAL
CALL GETCSR
CALL MTP037
CALL UNMAP
RETURN

;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY
;SET UP TEST ADDRESS
;CLEAR DATA TO BE WRITTEN
;MAP THIS TEST TO KERNEL SPACE
;GET CSRINFO FROM CONFIG TABLE
;CHECK ECC DISABLE
;REMAP KERNEL SPACE

7794 023660

MT0041: SUBTST <<MT0041 SETUP ADDRESS TO CSR ON DOUBLE BIT ERROR TEST>>

;*****

;SUBTEST MT0041 SETUP ADDRESS TO CSR ON DOUBLE BIT ERROR TEST

;*****

7795 023660 012737 000041 002300

MOV #41,REALPAT

;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY

7796 023666 004737 032674

CALL GETCSR

;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE

7797 023672

LET SUPDOADD := #MTP041

;SET UP TEST ADDRESS

7798 023700

LET R1 := #FIRST

;SET UP FIRST ADDRESS

7799 023704

IF BANK EQ #177

;ARE WE AT BANK 177?

7799 023704

CMP BANK,#177

7799 023704

BNE L212

7800 023712

LET PASCNT := #12.

MOV #12.,PASCNT

7800 023714

ELSE

7801 023722

LET PASCNT :- #16.

7801 023722

7802 023724

7802 023724

7803 023732

7804 023732

7805 023736

CALL SUPD03

;EXECUTE ADDRESS TO CSR TEST IN SUPVISIOR MODE

RETURN

;

7807 023740

MT0042: SUBTST <<MT0042 SETUP EXTENDED Q BUS ADDRESS TO CSR TEST>>

;*****

;*SUBTEST MT0042 SETUP EXTENDED Q-BUS ADDRESS TO CSR TEST

;*****

7808 023740 012737 000042 002300
7809 023746 012701 100000
7810 023752 004737 037740
7811 023756 004737 032674
7812 023762 004737 030542
7813 023766 004737 040026
7814 023772 000207

MOV #42,REALPAT
MOV #100000,R1
CALL MAPKERNAL
CALL GETCSR
CALL MTP042
CALL UNMAP
RETURN

;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY
;SET UP TEST ADDRESS
;MAP TO KERNEL SPACE
;SET UP CSRINFO FROM CONFIGURATION TABLE
;CHECK EXTENDED Q-BUS ADDRESS TO CSR
;REMAP KERNEL SPACE


```

7816 023774      MT0043: SUBTST <<MT0043      SETUP WRITE BYTE CLEARS SBE TEST>>
;*****
;*SUBTEST      MT0043  SETUP WRITE BYTE CLEARS SBE TEST
;*****
7817 023774 012737 000043 002300      MOV    #43,REALPAT      ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7818 024002 004737 037740      CALL    MAPKERNAL      ;MAP TO KERNEL SPACE
7819 024006      LET R1 := #100000      ;SET UP TEST ADDRESS
      024006 012701 100000      MOV    #100000,R1
7820 024012 004737 031030      CALL MTP043      ;PERFORM WRITE BYTE TEST
7821 024016 004737 040026      CALL UNMAP      ;REMAP KERNEL SPACE
7822 024022 000207      RETURN
7823 024024      MT0044: SUBTST <<MT0044      SETUP SHIFTING 1/0'S THROUGH THE CHECK BITS TEST>>
;*****
;*SUBTEST      MT0044  SETUP SHIFTING 1/0'S THROUGH THE CHECK BITS TEST
;*****
7824 024024 012737 000044 002300      MOV    #44,REALPAT      ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7825 024032 004737 032674      CALL    GETCSR      ;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE
7826 024036      LET SUPDOADD := #MTP044      ;SET UP TEST ADDRESS      ;:IL
      024036 012737 031224 002262      MOV    #MTP044,SUPDOADD
7827 024044      LET R1 := #FIRST      ;SET UP FIRST ADDRESS      ;:IL
      024044 012701 060000      MOV    #FIRST,R1
7828 024050      IF BANK EQ #177      ;ARE WE AT BANK 177?
      024050 023727 002102 000177      CMP    BANK,#177
      024056 001004      BNE    L214
7829 024060      LET ENDADD := #120000      MOV    #120000,ENDADD
      024060 012737 120000 002562      ELSE
7830 024066      BR    L215
      024066 000403      L214:::;
      024070      LET ENDADD := #160000      MOV    #160000,ENDADD
7831 024070 012737 160000 002562      END
7832 024076      CALL SUPD03      ;EXECUTE ADDRESS TO CSR TEST IN SUPVISIOR MODE
      024076 004737 024446      L215:::;
7834 024102 000207      RETURN
7835 024104      MT0045: SUBTST <<MT0045      SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR>>
;*****
;*SUBTEST      MT0045  SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR
;*****
7836 024104 012737 000045 002300      MOV    #45,REALPAT      ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7837 024112 004737 037740      CALL    MAPKERNAL      ;MAP TO KERNEL SPACE
7838 024116      LET R1 := #100000      ;SET UP TEST ADDRESS
      024116 012701 100000      MOV    #100000,R1
7839 024122 004737 031546      CALL MTP045      ;PERFORM SYNDROMES TO CSR ON DOUBLE BIT ERROR
7840 024126 004737 040026      CALL UNMAP      ;REMAP KERNEL SPACE
7841 024132 000207      RETURN
7842 024134      MT0046: SUBTST <<MT0046      SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET>>
;*****
;*SUBTEST      MT0046  SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET
;*****
7843 024134 012737 000046 002300      MOV    #46,REALPAT      ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7844 024142 004737 037740      CALL    MAPKERNAL      ;MAP TO KERNEL SPACE
7845 024146      LET R1 := #100000      ;SET UP TEST ADDRESS
      024146 012701 100000      MOV    #100000,R1
7846 024152 004737 031734      CALL MTP046      ;PERFORM TRAPS DETECTED ON SBE WITH ECC DISABLED TE
7847 024156 004737 040026      CALL UNMAP      ;REMAP KERNEL SPACE
7848 024162 000207      RETURN

```

7849 024164

MT0047: SUBTST <<MT0047 SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST>>

;*****

;SUBTEST MT0047 SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST

;*****

7850 024164 012737 000047 002300

MOV #47,REALPAT

;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY

7851 024172 004737 037740

CALL MAPKERNAL

;MAP TO KERNEL SPACE

7852 024176

LET R1 := #100000

;SET UP TEST ADDRESS

024176 012701 100000

MOV #100000,R1

7853 024202

LET R2 := #120000

; " " SECOND TEST ADDRESS

024202 012702 120000

MOV #120000,R2

7854 024206 004737 032274

CALL MTP047

;PERFORM NO UPDATE TO CSR ON SBE WITH DBE

7855 024212 004737 040026

CALL UNMAP

;REMAP KERNEL SPACE

7856 024216 000207

RETURN

```

7859 024220      MT0999: SUBTST <<MT0999      SETUP NULL TEST>>
;*****
;*SUBTEST      MT0999  SETUP NULL TEST
;*****
7860 024220      005037  002300      CLR      REALPAT
7861 024224      012737  177777  002344      SET      NULLFLAG
7862 024232      000207      RETURN      MOV    #-1,NULLFLAG
7863
7864 024234      KAMITEST:SUBTST <<CHECK FOR KAMIKAZE MODE>>
;*****
;*SUBTEST      CHECK FOR KAMIKAZE MODE
;*****
7865 024234      005737  002006      IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE
024234      001006      TST KAMIKAZE
024240      005737  002350      BNE L216
024246      001003      TST ACTFLAG
024250      005737  002352      BNE L216
024254      001403      TST APTFLAG
024256      BEQ L217
7866 024256      $RETURN NOERROR      ;RUN THE TEST      L216:::;
024256      000241      CLC
024260      000207      RTS PC
7867 024262      ELSE
024262      000402      BR L220
024264      L217:::;
7868 024264      $RETURN ERROR      ;DON'T RUN THE TEST
024264      000261      SEC
024266      000207      RTS PC
7869 024270      END ;OF IF KAMIKAZE
024270      L220:::;

```

7872 024270

SUPD01: SUBTST <<SUBR EXECUTE PATTERN IN SUPERVISOR>>

;*****

;*SUBTEST SUBR EXECUTE PATTERN IN SUPERVISOR

;*****

MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
MOV R3,-(SP)MOV BANK,R3
CALL MAPPER
.DSABL CRF

MOV (SP)+,R3

SUPD02: CALL GETDIS
PUSH \$LPERR,\$LPADRMOV \$LPERR,-(SP)
MOV \$LPADR,(SP)MOV R0,SUPDR0
MOV #SUPDR1,R0
MOV R1,(R0)+
MOV R2,(R0)+
MOV R3,(R0)+
MOV R4,(R0)+
MOV R5,(R0)+
MOV SP,(R0)+
MOV SUPDR0,R0
MOV #TAG4\$, \$LPADR
MOV \$LPADR,\$LPERR
TAG4\$: MOV #SUPDR6+2,R0
MOV -(R0),SP
MOV -(R0),R5
MOV -(R0),R4
MOV -(R0),R3
MOV -(R0),R2
MOV -(R0),R1
MOV -(R0),R0SUPERVISOR ;ENTER SUPERVISOR MODE
BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
.DSABL CRF

MOV #SUPSTK,SSP

CACHOFF ;TURN CACHE OFF
CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
CACHON ;TURN CACHE ON
KERNEL ;ENTER KERNEL MODESCOPE
POP \$LPADR,\$LPERRMOV (SP)+,\$LPADR
MOV (SP)+,\$LPERR

RETURN

7873	024270	010346		
	024270	013703	002102	
	024272	004737	035716	
	024276			
	024302	012603		
7874	024304	004737	051430	
7875	024310			
	024310	013746	002624	
	024314	013746	002622	
7876	024320	010037	002160	
7877	024324	012700	002162	
7878	024330	010120		
7879	024332	010220		
7880	024334	010320		
7881	024336	010420		
7882	024340	010520		
7883	024342	010620		
7884	024344	013700	002160	
7885	024350	012737	024364	002622
7886	024356	013737	002622	002624
7887	024364	012700	002176	
7888	024370	014006		
7889	024372	014005		
7890	024374	014004		
7891	024376	014003		
7892	024400	014002		
7893	024402	014001		
7894	024404	014000		
7895	024406			
	024406	052737	040000	177776
7896	024414	012706	000740	
7897	024420	104424		
7898	024422	004737	177640	
7899	024426	104423		
7900	024430	104417		
7901	024432	000004		
7902	024434			
	024434	012637	002622	
	024440	012637	002624	
7903	024444	000207		

```

7906 024446          SUPD03: MAP      BANK      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      024446 010346          MOV      BANK,R3      MOV R3,-(SP)
      024450 013703 002102      CALL    MAPPER
      024454 004737 035716      .DSABL CRF

      024460 012603          MOV (SP)+,R3
7907 024462 004737 051430      SUPD04: CALL  GETDIS
7908 024466          PUSH    $LPERR,$LPADR
      024466 013746 002624          MOV $LPERR,(SP)
      024472 013746 002622          MOV $LPADR,-(SP)
7909 024476 010037 002160      MOV      R0,SUPDR0
7910 024502 012700 002162      MOV      #SUPDR1,R0
7911 024506 010120          MOV      R1,(R0)+
7912 024510 010220          MOV      R2,(R0)+
7913 024512 010320          MOV      R3,(R0)+
7914 024514 010420          MOV      R4,(R0)+
7915 024516 010520          MOV      R5,(R0)+
7916 024520 010620          MOV      SP,(R0)+
7917 024522 013700 002160      MOV      SUPDR0,R0
7918 024526 012737 024542 002622      MOV      #TBG4$, $LPADR
7919 024534 013737 002622 002624      MOV      $LPADR,$LPERR
7920 024542 012700 002176      TBG4$: MOV      #SUPDR6+2,R0
7921 024546 014006          MOV      -(R0),SP
7922 024550 014005          MOV      -(R0),R5
7923 024552 014004          MOV      -(R0),R4
7924 024554 014003          MOV      -(R0),R3
7925 024556 014002          MOV      -(R0),R2
7926 024560 014001          MOV      -(R0),R1
7927 024562 014000          MOV      -(R0),R0
7928 024564          TESTAREA
      024564 053737 002552 177776      BIS      TESTMODE,PSW      ;ENTER SUPERVISOR MODE
                                          ;GO TO SYSTEM TEST MODE
                                          .DSABL CRF
7929 024572 005737 002456      TST      NOSUPER
7930 024576 001403          BEQ      1$
7931 024600 012706 000700      MOV      #USESTK,USP
7932 024604 000402          BR      2$
7933 024606 012706 000740      1$: MOV      #SUPSTK,SSP
7934 024612 104424          2$: CACHOFF
7935 024614 004777 155442      CALL    @SUPDOADD      ;TURN CACHE OFF
7936 024620 104423          CACHON      ;TURN CACHE ON
7937 024622 104417          KERNEL      ;ENTER KERNEL MODE
7938 024624 000004          SCOPE
7939 024626          POP      $LPADR,$LPERR
      024626 012637 002622          MOV (SP)+,$LPADR
      024632 012637 002624          MOV (SP)+,$LPERR
7940 024636 000207          RETURN

```

7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953 024640

7954 024640 010220
7955 024642 077102
7956 024644 000240
7957 024646 012401
7958 024650 020102
7959 024652 001402
7960 024654 104430
7961 024656 000240
7962 024660 077306
7963 024662 000207
7964 024664

7965 024664 010220
7966 024666 062702 000002
7967 024672 077104
7968 024674 000240
7969 024676 012400
7970 024700 020005
7971 024702 001401
7972 024704 104427
7973 024706 062705 000002
7974 024712 077307
7975 024714 000207
7976 024716

7977 024716 010540
7978 024720 062705 000002
7979 024724 077104
7980 024726 000240
7981 024730 162702 000002
7982 024734 012401
7983 024736 020102
7984 024740 001401
7985 024742 104430
7986 024744 077307
7987 024746 000207

```
.SBTTL MEMORY TEST PATTERN ROUTINES
;*****
; PATTERN REGISTER CONVENTIONS
; R0 FIRST ADDRESS OF PATTERN (FIRST, LAST, 2, ETC)
; R1 NUMBER OF ADDRESSES IN PATTERN (SIZE)
; R2 DATA FOR PATTERN (ONES, 52525, ETC)
; R3 COPY OF R1 (IF NECESSARY)
; R4 COPY OF R0 (IF NECESSARY)
; R5 COPY OF R2 (IF NECESSARY)
;*****
MTP000: SUBTST <<MTP000 BASIC DATA TEST>>
;*****
; *SUBTEST MTP000 BASIC DATA TEST
;*****
1$: MOV R2, (R0)+ ;V177640
   SOB R1, MTP000 ;V177642
   NOP ;V177644
2$: MOV (R4)+, R1 ;V177646
   CMP R1, R2 ;V177650
   BEQ 3$ ;V177652
   PERR02 ;V177654
   NOP ;V177656
3$: SOB R3, 2$ ;V177660
   RETURN ;V177662
MTP001: SUBTST <<MTP001 ADDRESS TEST>>
;*****
; *SUBTEST MTP001 ADDRESS TEST
;*****
3$: MOV R2, (R0)+ ;V177640
   ADD #2, R2 ;V177642
   SOB R1, 3$ ;V177646
   NOP ;V177650
1$: MOV (R4)+, R0 ;V177652
   CMP R0, R5 ;V177654
   BEQ 2$ ;V177656
   PERR01 ;V177660
2$: ADD #2, R5 ;V177662
   SOB R3, 1$ ;V177666
   RETURN ;V177672
MTP002: SUBTST <<MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)>>
;*****
; *SUBTEST MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
;*****
3$: MOV R5, -(R0) ;V177640
   ADD #2, R5 ;V177642
   SOB R1, 3$ ;V177646
   NOP ;V177650
1$: SUB #2, R2 ;V177652
   MOV (R4)+, R1 ;V177656
   CMP R1, R2 ;V177660
   BEQ 2$ ;V177662
   PERR02 ;V177664
2$: SOB R3, 1$ ;V177666
   RETURN ;V177670
```

7990 024750

```

MTPA03: SUBTST <<MTPA03      3 XOR 9 WORST CASE NOISE TEST (WRITE)>>
;*****
;*SUBTEST      MTPA03  3 XOR 9 WORST CASE NOISE TEST (WRITE)
;*****
;R1 = ADDRESS
;R2 = SMALL LOOP CONSTANT
;R3 = NUM OF ADD TO TEST (LARGE LOOP)
;R4 = GOOD DATA
;R5 = MEDIUM LOOP CONSTANT
.ENABL  LSB
1$:     MOV      R4,(R1)+      ;V177640
        MOV      R4,(R1)+      ;V177642
        SOB      R2,1$        ;V177644
        COM      R4            ;V177646
        BIS      (PC)+,R4      ;V177650
WARN2:  401      ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
        MOV      #4,R2        ;V177654
        SOB      R5,1$        ;V177660
        COM      R4            ;V177662
        BIS      (PC)+,R4      ;V177664
WARN3:  401      ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
        MOV      #64.,R5      ;V177670
        SOB      R3,1$        ;V177674
        RETURN    ;V177676
        .DSABL  LSB

```

```

7991
7992
7993
7994
7995
7996
7997 024750 010421
7998 024752 010421
7999 024754 077203
8000 024756 005104
8001 024760 052704
8002 024762 000401
8003 024764 012702 000004
8004 024770 077511
8005 024772 005104
8006 024774 052704
8007 024776 000401
8008 025000 012705 000100
8009 025004 077317
8010 025006 000207
8011
8012
8013 025010

```

```

MTPB03: SUBTST <<MTPB03      3 XOR 9 WORST CASE NOISE TEST (READ)>>
;*****
;*SUBTEST      MTPB03  3 XOR 9 WORST CASE NOISE TEST (READ)
;*****
.ENABL  LSB
1$:     JMP      @MTPC03      ;V177640      GO TO V172360
        SOB      R2,1$        ;V177644
        COM      R4            ;V177646
        BIS      (PC)+,R4      ;V177650
WARN4:  401      ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
        MOV      #4,R2        ;V177654
        SOB      R5,1$        ;V177660
        COM      R4            ;V177662
        BIS      (PC)+,R4      ;V177664
WARN5:  401      ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
        MOV      #64.,R5      ;V177670
        SOB      R3,1$        ;V177674
        RETURN    ;V177676
        .DSABL  LSB

```

```

8014
8015 025010 000137 025050
8016 025014 077203
8017 025016 005104
8018 025020 052704
8019 025022 000401
8020 025024 012702 000004
8021 025030 077511
8022 025032 005104
8023 025034 052704
8024 025036 000401
8025 025040 012705 000100
8026 025044 077317
8027 025046 000207
8028

```

8031 025050

MTPC03: SUBTST <<MTPC03 TEST DATA SUBPROGRAM>>

;*****

;*SUBTEST MTPC03 TEST DATA SUBPROGRAM

;*****

8032 025050 020421

CMP R4,(R1);V172360

8033 025052 001401

BEQ 1\$;V172362

8034 025054 104431

PERR03;V172364

8035 025056 005141

1\$: COM -(R1);V172366

8036 025060 005111

COM (R1);V172370

8037 025062 000137 025066

JMP @#MTPD03;V172372 GO TO V172260

8038

8039 025066

MTPD03: SUBTST <<MTPD03 TEST DATA SUBSUBPROGRAM>>

;*****

;*SUBTEST MTPD03 TEST DATA SUBSUBPROGRAM

;*****

8040 025066 020421

CMP R4,(R1);V172260

8041 025070 001401

BEQ 1\$;V172262

8042 025072 104431

PERR03;V172264

8043 025074 005127

1\$: COM (PC);V172266

8044 025076 000000

O;V172270

8045 025100 001363

BNE MTPC03;V172272 GO TO V172360

8046 025102 000137 025014

JMP @#MTPB03+4;V172274 GO TO V177644

8049 025106

MTPA04: SUBTST <<MTPA04 ROTATING ZEROS TEST>>

;*****
;SUBTEST MTPA04 ROTATING ZEROS TEST
;*****

8050 025106 012705 000010
 8051 025112 010504
 8052 025114 000241
 8053 025116 000137 025142
 8054 025122 016004 177776
 8055 025126 103402
 8056 025130 020204
 8057 025132 001401
 8058 025134 104432
 8059 025136 077115
 8060 025140 000207
 8061
 8062 025142

1\$: MOV #8.,R5 ;V177640
 MOV R5,R4 ;V177644
 CLC ;V177646
 JMP @#MTPB04 ;V177650
 MOV -2(R0),R4 ;V177654
 BCS 2\$;V177660
 CMP R2,R4 ;V177662
 BEQ 3\$;V177664
 2\$: PERR04 ;V177666
 3\$: SOB R1,1\$;V177670
 RETURN ;V177672

MTPB04: SUBTST <<MTPB04 SUBR ROTATING BIT>>

;*****
;SUBTEST MTPB04 SUBR ROTATING BIT
;*****

8063 025142 106110
 8064 025144 077502
 8065 025146 106120
 8066 025150 106110
 8067 025152 077402
 8068 025154 106120
 8069 025156 000137 025122
 8070
 8071 025162

1\$: ROLB (R0) ;V172360
 SOB R5,1\$;V172362
 ROLB (R0)+ ;V172364
 2\$: ROLB (R0) ;V172366
 SOB R4,2\$;V172370
 ROLB (R0)+ ;V172372
 JMP @#MTPA04+14 ;V172374

MTP005: SUBTST <<MTP005 ROTATION ONES TEST>>

;*****
;SUBTEST MTP005 ROTATION ONES TEST
;*****

8072 025162 012705 000010
 8073 025166 010504
 8074 025170 000261
 8075 025172 000137 025142
 8076 025176 016004 177776
 8077 025202 103002
 8078 025204 020204
 8079 025206 001401
 8080 025210 104432
 8081 025212 077115
 8082 025214 000207

1\$: MOV #8.,R5 ;V177640
 MOV R5,R4 ;V177644
 SEC ;V177646
 JMP @#MTPB04 ;V177650
 MOV -2(R0),R4 ;V177654
 BCC 2\$;V177660
 CMP R2,R4 ;V177662
 BEQ 3\$;V177664
 2\$: PERR04 ;V177666
 3\$: SOB R1,1\$;V177670
 RETURN ;V177672

IF THIS HAPPENS THE GOOD & BAD MATCH

8085 025216

MTP006: SUBTST <<MTP006 INITIAL DATA TEST>>

```

*****
;SUBTEST      MTP006  INITIAL DATA TEST
*****

```

```

8086
8087
8088 025216 012737 000001 002242      MOV    #1,DATBUF      ;SET THE FIRST TEST BIT
8089 025224 005037 002244              CLR    DATBUF+2      ;CLEAR 2ND WORD
8090 025230 013771 002242 000000 1$:  MOV    DATBUF,@(R1)    ;WRITE TEST WORD 1
8091 025236 013771 002244 000002      MOV    DATBUF+2,@2(R1) ;AND TEST WORD 2
8092 025244 017102 000000              MOV    @2(R1),R2
8093 025250 023702 002242              CMP    DATBUF,R2      ;NOW READ THEM
8094 025254 001401                      BEQ    2$            ;BR IF FIRST 16 OK
8095 025256 104433                      PERR07           ;ERROR TRAP
8096
8097 025260 017102 000002 2$:          MOV    @2(R1),R2
8098 025264 023702 002244              CMP    DATBUF+2,R2    ;NOW READ SECOND WORD
8099 025270 001401                      BEQ    3$            ;BR IF OK
8100 025272 104434                      PERR10           ;ERROR TRAP
8101
8102 025274 005737 002244 3$:          TST    DATBUF+2      ;HAS LAST BIT BEEN TESTED ?
8103 025300 100405                      BMI    4$            ;MINUS MEANS BIT 31
8104 025302                      DLEFT  DATBUF      ;NO, SHIFT TEST BIT LEFT
      025302 006137 002242              ROL    DATBUF
      025306 006137 002244              ROL    DATBUF+2
      .DSABL CRF
8105 025312 000746                      BR     1$            ;GO WRITE NEW TEST DATA
8106
8107 025314 012737 177776 002242 4$:  MOV    #177776,DATBUF ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
8108 025322 012737 177777 002244      MOV    #-1,DATBUF+2 ;PUT A 0 IN BIT 0
8109 025330 013771 002242 000000 5$:  MOV    DATBUF,@(R1)    ;AND 1'S IN ALL OTHERS
8110 025336 013771 002244 000002      MOV    DATBUF+2,@2(R1) ;WRITE THE DATA
8111 025344 017102 000000              MOV    @2(R1),R2      ;2 WORDS WORTH
8112 025350 023702 002242              CMP    DATBUF,R2      ;NOW READ FIRST WORD
8113 025354 001401                      BEQ    6$            ;BR IF OK
8114 025356 104433                      PERR07
8115
8116 025360 017102 000002 6$:          MOV    @2(R1),R2
8117 025364 023702 002244              CMP    DATBUF+2,R2    ;NOW, READ SECOND WORD
8118 025370 001401                      BEQ    7$            ;BR IF OK
8119 025372 104434                      PERR10
8120
8121 025374 005737 002244 7$:          TST    DATBUF+2      ;TESTED BIT 31 YET?
8122 025400 100005                      BPL    8$            ;BR IF YES, WE'RE DONE
8123 025402                      DLEFT  DATBUF
      025402 006137 002242              ROL    DATBUF
      025406 006137 002244              ROL    DATBUF+2
      .DSABL CRF
8124 025412 000746                      BR     5$            ;KEEP GOING
8125 025414 000207 8$:              BR     RETURN

```

8128 025416

MTP007: SUBTST <<MTP007 ADDRESS BIT TEST>>

```

;*****
;SUBTEST      MTP007 ADDRESS BIT TEST
;*****

```

```

;
;      THIS TEST CHECKS TO SEE THAT EACH ADDRESS
;      BIT IN EACH 16K BANK CAN BE ASSERTED UNIQUELY.
;      IT CHECKS FOR ADDRESS BITS THAT MAY BE STUCK
;      HIGH, STUCK LOW OR STUCK TOGETHER.

```

```

8129
8130
8131
8132
8133 025416 111100      MOVB      (R1),R0
8134 025420 105700      TSTB      R0          ;READ AND COMPARE FOR ZEROS
8135 025422 001401      BEQ        1$          ;BR IF OK
8136 025424 104435      PERR11
8137
8138 025426 105111      1$:      COMB      (R1)          ;COMPLEMENT THE BYTE
8139 025430 111100      MOVB      (R1),R0
8140 025432 105700      TSTB      R0          ;READ FOR NON ZEROS
8141 025434 001001      BNE        2$          ;BR IF OK
8142 025436 104436      PERR12
8143
8144 025440 040201      2$:      BIC        R2,R1          ;MASK OFF THE ASSERTED BIT
8145 025442 006302      ASL        R2          ;SHIFT R2 FOR NEXT BIT
8146 025444 050201      BIS        R2,R1          ;SET THE NEW BIT INTO R1
8147 025446 011100      MOV        (R1),R0
8148 025450 005700      TST        R0          ;READ THE NEW ADDRESS
8149 025452 001401      BEQ        3$          ;READ FOR ZEROS
8150 025454 104437      PERR13
8151
8152 025456 005111      3$:      COM        (R1)          ;COMPL THE WORD
8153 025460 011100      MOV        (R1),R0
8154 025462 005700      TST        R0          ;READ IT AGAIN
8155 025464 001001      BNE        4$          ;
8156 025466 104440      PERR14
8157
8158 025470 022702 100000 4$:      CMP        #100000,R2
8159 025474 001407      BEQ        5$
8160 025476 022702 010000      CMP        #10000,R2          ;CHECK FOR MSB IN 4K BANK
8161 025502 001356      BNE        2$          ;NOT LAST BIT, BRANCH
8162 025504 006302      ASL        R2
8163 025506 012701 160000      MOV        #160000,R1
8164 025512 000752      BR        2$
8165 025514 000207      5$:      RETURN

```

8168 025516

MTP010: SUBTST <<MTP010 BYTE ADDRESSING TEST>>

```

;*****
;*SUBTEST      MTP010  BYTE ADDRESSING TEST
;*****

```

```

8169
8170
8171 025516 010402
8172 025520 010403
8173 025522 062702 000004
8174 025526 012713 177777
8175 025532 012763 177777 000002
8176 025540 105013
8177 025542 010401
8178 025544 020201
8179 025546 001420
8180 025550 020301
8181 025552 001007
8182 025554 111100
8183
8184 025556 022700 000000
8185 025562 001401
8186 025564 104435
8187
8188 025566 005201
8189 025570 000765
8190 025572 111100
8191 025574 122700 177777
8192 025600 001401
8193 025602 104436
8194
8195 025604 005201
8196 025606 000756
8197 025610 112713 177777
8198 025614 005203
8199 025616 020302
8200 025620 001347
8201 025622 000207
8202

```

```

MOV R4,R2
MOV R4,R3
ADD #4,R2
MOV #1,(R3)
MOV #-1,2(R3)
1$: CLRB (R3)
MOV R4,R1
2$: CMP R2,R1
BEQ 6$
CMP R3,R1
BNE 4$
MOVB (R1),R0
;WARNING IF YOU OPTOMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
CMP #0,R0
BEQ 3$
PERR11
3$: INC R1
BR 2$
4$: MOVB (R1),R0
CMPB #-1,R0
BEQ 5$
PERR12
5$: INC R1
BR 2$
6$: MOVB #-1,(R3)
INC R3
CMP R3,R2
BNE 1$
RETURN

```

```

;TEST 3 TH'S TEST CHECKS FOR PROPER
;BYTE ADDRESSING WITH ECC DISABLED
;R4 HAS LOWEST ADDRESS
;PUT IT IN R3 ALSO
;POINT R2 TO LAST BYTE +1
;WRITE ALL ONES IN
;THE 4 TEST BYTES
;CLEAR A BYTE
;INITIALIZE R1 FOR EACH PASS
;IF EQUAL, JUST READ LAST BYTE
;BR IF EQUAL
;IS THIS THE BYTE OF ZEROS
;BR IF NOT
;NEXT BYTE
;RETURN
;ITS NOT THE BYTE OF 0'S, READ 1'S
;MOVE TO NEXT BYTE
;RESTORE 1'S TO BYTE JUST TESTED
;INC TO NEXT BYTE
;WAS THAT JUST THE LAST ONE?
;BR IF NO

```

8205 025624

MTP014: SUBTST <<MTP014 BASIC DOUBLE BIT ERROR TEST>>

```

;*****
;SUBTEST      MTP014  BASIC DOUBLE BIT ERROR TEST
;*****

```

```

; THIS TEST CHECKS THAT A DOUBLE ERROR WILL BE DETECTED
; A BYTE WRITE WITH A DOUBLE ERROR ON A MSV11-P
; WILL BE ABOTRED.
;

```

8206								
8207								
8208								
8209								
8210								
8211	025624	104424				CACHOFF	;TURN OFF CACHE	
8212	025626					LET PARCNT := #0	;CLEAR PARCNT	
	025626	005037	002072					CLR PARCNT
8213	025632					LET NOPAR := #1	;SET PARITY ACTION	
	025632	012737	000001	002076				MOV #1,NOPAR
8214	025640					LET ADDRESS := #FIRST	;SET ADDRESS FOR ERROR REPORT	
	025640	012737	060000	002034				MOV #FIRST,ADDRESS
8215	025646	104513				CBREG	;ENABLE CHECK/SYNDROME BIT REGISTER	
8216	025650					LET CSR := #3145	;DBE CHECK BITS FOR CSR	
	025650	012737	003145	002150				MOV #3145,CSR
8217	025656	104425				LOADCSR	;WRITE DBE CHECK BITS TO CSR	
8218	025660					LET GOOD := #103145	;GOOD DATA	
	025660	012737	103145	002044				MOV #103145,GOOD
8219	025666					LET (R1) := #0	;WRITE ZEROS AND DBL ERROR CHK BITS A=0	
	025666	005011						CLR (R1)
8220	025670	005711				TST (R1)	;READ A=0 TO GET DOUBLE BIT ERROR	
8221	025672					IF PARCNT NE #1	;WAS BUSPBL ASSERTED???	
	025672	023727	002072	000001				CMP PARCNT,#1
	025700	001401						BEQ L221
8222	025702	104055				ERROR +55	;ERROR CALL ; MISSED EXPECTED TRAP	
8223	025704					END		
	025704							L221:::~::~
8224	025704	104426				READCSR	;READ CSR FOR CORRECT CHECK BITS AND DBE INDICATOR	
8225	025706	042737	020000	002150		BIC #BIT13,CSR	;CLEAR INHIBIT MODE POINTER FROM DATA IF IT EXSISTS1	
8226	025714					IF CSR NE GOOD THEN	;CHECK IF DOUBLE ERROR BIT IS SET	
	025714	023737	002150	002044				CMP CSR,GOOD
	025722	001407						BEQ L222
8227	025724					SET HEADER		
	025724	012737	177777	002612				MOV #-1,HEADER
8228	025732					LET BAD := CSR	;BAD DATA	
	025732	013737	002150	002052				MOV CSR,BAD
8229	025740	104065				ERROR +65		
8230	025742					END		
	025742							L222:::~::~
8231	025742	104473				ECC1INIT	;ENABLE BUSPBL	
8232	025744	005037	002266			CLR PASSNO	;CLEAR LOOP COUNTER	
8233	025750					REPEAT		
	025750							B31:::~::~
8234	025750	104473				ECC1INIT	;ENABLE BUSPBL	
8235	025752	005237	002266			INC PASSNO	;INCREMENT LOOP COUNTER	
8236	025756	005037	002072			CLR PARCNT	;CLEAR PARITY ACTION COUNTER	
8237	025762					LET (R1) :B= #377	;WRITE BYTE SHOULD BE ABORTED	
	025762	112711	000377					MOVB #377,(R1)
8238	025766	105711				TSTB (R1)	;READ R1 TO SEE IF IT IS STILL 0	
8239	025770					IF PARCNT NE #1	;WAS WRITE ABORTED???	
	025770	023727	002072	000001				CMP PARCNT,#1
	025776	001411						BEQ L223
8240	026000					SET HEADER		

SEQ 0228

PC	Instruction	PC	Instruction	PC	Instruction	PC	Instruction
8241	026000	012737	177777	002612			MOV #1,HEADER
	026006				LET GOOD := #0	;GOOD DATA	
8242	026006	005037	002044				CLR GOOD
	026012				LET BAD := #377	;BAD DATA	
8243	026012	012737	000377	002052			MOV #377,BAD
	026020	104056			ERROR +56	:	
8244	026022				END	:	
	026022						L223:::;;;
8245	026022	005201			INC R1	;AND REPEAT ON HIGH BYTE	
8246	026024				UNTIL PASSNO EQ #2	:	
	026024	023727	002266	000002			CMP PASSNO,#2
	026032	001346					BNE B31
	026034						E31:::;;;
8247	026034	005041			CLR -(R1)	;CLEAR LUT	
8248	026036	104503			CLR1CSR	;CLEAR CSR	
8249	026040	005037	002072		CLR PARCNT	;CLEAR PARITY TRAP COUNTER	
8250	026044	104423			CACHON	;TURN ON CACHE	
8251	026046	000207			RETURN	:	
8252							

8255 026050

8256
8257
8258
8259
8260
8261
8262 026050 012701 060000
8263 026054 010104
8264 026056 012705 160000
8265 026062 012700 000377
8266 026066 010003
8267 026070 000303
8268 026072 110021
8269 026074 110321
8270 026076 020105
8271 026100 103774
8272
8273 026102 014102
8274 026104 020002
8275
8276 026106 001401
8277 026110 104446
8278
8279 026112 020104
8280 026114 101372
8281 026116 000303
8282 026120 000300
8283 026122 001763
8284
8285 026124 000207
8287

```

MTP017- SUBTST <<MTP017      HOLDING 1'S & 0'S TEST>>
;*****
;SUBTEST      MTP017  HOLDING 1'S & 0'S TEST
;*****
;*(1)  THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
;*      OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
;*      OF 000377 AND READING IT
;*(2)  MEMORY IS WRITTEN USING A BYTE AT A TIME
;*(3)  STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
;NOTE:  THIS TEST WRITES BYTES & READS WORDS
MOV     #FIRST,R1
MOV     R1,R4
MOV     #LAST+2,R5
MOV     #377,R0      ;GET THE PATTERN INTO R0
MOV     R0,R3
SWAB    R3
1$:     MOV     R0,(R1)+    ;WRITE A BYTE
        MOV     R3,(R1)+    ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT.1
        CMP     R1,R5      ;COMPARE TEST LOC TO TOP + 2
        BLO     1$         ;BRANCH IF LOWER

2$:     MOV     -(R1),R2
        CMP     R0,R2      ;TEST THE MEMORY TO SEE IF IT CONTAINS
                                ;THE WORD STORED IN BAKPAT
        BEQ     3$
        PERR22

3$:     CMP     R1,R4      ;KEEP ON TESTII , THE MEMORY UNTIL
        BHI     2$         ;R1 EQUALS THE LOWEST ADDRESS
        SWAB    R3         ;CHANGE THE DATA PATTERN
        SWAB    R0
        BEQ     1$         ;IF THE DATA PATTERN DOES NOT HAVE LOW
                                ;BYTE =0 THEN FALL THRU

        RETURN

```

8289 026126

```
MTP020: SUBTST <<MTP020      SYNDROMES TO CSR ON SINGLE BIT ERROR TEST>>
;*****
;*SUBTEST      MTP020  SYNDROMES TO CSR ON SINGLE BIT ERROR TEST
;*****
;
;  THIS TEST CHECKS TO SEE IF THE SINGLE BIT ERRORS CAUSE THE SBE
;  BIT IN THE CSR TO BE SET AND CORRECT SYNDROME BITS ARE GENERATED FOR
;  ALL 16 DATA BITS.
;
;CACHOFF      ;TURN OFF CACHE
CLR R0        ;CLEAR DATA
CLRB PASFLG   ;CLEAR PASFLG
CBREG         ;ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT       ;
;
;          B32:;;;;;
;          ;INCREMENT LOOP COUNTER
;          INCB PASFLG
;          ;INDEX TO SINGLE BIT ERROR TABLE
;          MOV #-1,R4
;          ;CLEAR INNER LOOP COUNTER
;          CLR BITNO
;          ;SELECT DATA TO BE CORRECTED BY PASSNO
;          IFB PASFLG EQ #1
;          ;DATA-;BIT TO BE CORRECTED IS A ONE
;          ;          CMPB PASFLG,#1
;          ;          BNE L225
;          ;          MOV #1,R5
;          ;
;          ;          BR L226
;          ;          L225:;;;;;
;          ;DATA=177776;BIT TO BE CORRECTED IS A ZERO
;          ;          MOV #177776,R5
;          ;
;          ;          L226:;;;;;
;          ;
;          ;          B33:;;;;;
;          ;INCREMENT BIT POINTER
;          ;POINT TO NEXT SET OF CHECK BITS
;          ;          INC R4
;          ;GET NEXT SET OF CHECK BITS
;          ;          MOVB PTABLE(R4),R2
;          ;SHIFT TO LINE UP IN CSR
;          ASH #5,R2
;          ;ENABLE DIAG MODE
;          BIS #BIT2,R2
;          ;GET CHECK BITS TO BE WRITTEN
;          LET CSR := R2
;          ;LOAD CSR WITH DATA
;          LOADCSR
;          ;WRITE DATA TO TEST ADDRESS
;          LET (R1) := R0
;          ;          MOV R0,(R1)
;          ;CLEAR CSR
;          CLR1CSR
;          ;CORRECT SBE
;          TST (R1)
;          ;READ CSR FOR CORRECT SBE BIT AND SYNDROMES
;          READCSR
;          ;CLEAR ALL BUT SBE INDICATOR
;          BIC #C20,CSR
;          ;WAS DATA CORRECTED???
;          IF CSR NE #20
;          ;          CMP CSR,#20
;          ;          BEQ L227
;          ;
;          ;          MOV #20,GOOD
;          ;
;          LET GOOD := #20
;          ;
;          LET BAD := CSR
;          ;
```

```
8290
8291
8292
8293
8294
8295 026126 104424
8296 026130 005000
8297 026132 105037 002264
8298 026136 104513
8299 026140
      026140
8300 026140
      026140 105237 002264
8301 026144
      026144 012704 177777
8302 026150
      026150 005037 002324
8303 026154
      026154 123727 002264 000001
      026162 001003
8304 026164
      026164 012705 000001
8305 026170
      026170 000402
      026172
8306 026172
      026172 012705 177776
8307 026176
      026176
8308 026176
      026176
8309 026176 005237 002324
8310 026202
      026202 005204
8311 026204
      026204 116402 030276
8312 026210 072227 000005
8313 026214 052702 000004
8314 026220
      026220 010237 002150
8315 026224 104425
8316 026226
      026226 010011
8317 026230 104503
8318 026232 005711
8319 026234 104426
8320 026236 042737 177757 002150
8321 026244
      026244 023727 002150 000020
      026252 001407
8322 026254
      026254 012737 000020 002044
8323 026262
```



```

      026262 013737 002150 002052
8324 026270 104060
8325 026272
      026272
8326 026272 104514
8327 026274 104426
8328 026276 042737 174033 002150
8329 026304
      026304 116403 026422
8330 026310 072327 000005
8331 026314 052703 000004
8332 026320
      026320 020337 002150
      026324 001411
8333 026326
      026326 012737 177777 002612
8334 026334
      026334 010337 002044
8335 026340
      026340 013737 002150 002052
8336 026346 104042
8337 026350
      026350
8338 026350 005011
8339 026352
      026352 123727 002264 000001
      026360 001002
8340 026362 006305
8341 026364
      026364 000402
      026366
8342 026366 000261
8343 026370 006105
8344 026372
      026372
8345 026372
      026372 023727 002324 000020
      026400 001276
      026402
8346 026402 005100
8347 026404
      026404 123727 002264 000002
      026412 001252
      026414
8348 026414 104503
8349 026416 104423
8350 026420 000207
8351
8352
8353
8354 026422 016 013 023
      026425 025 026 031
      026430 032 034 043
      026433 045 046 051
      026436 052 054 061
      026441 064
8355

      ERROR +60
      END
      SYNREG
      READCSR
      BIC #1C3744,CSR
      LET R3 :B= SBESYN(R4)
      ASH #5,R3
      BIS #BIT2,R3
      IF R3 NE CSR
      SET HEADER
      LET GOOD := R3
      LET BAD := CSR
      ERROR +42
      END
      CLR (R1)
      IFB PASFLG EQ #1
      ASL R5
      ELSE
      SEC
      ROL R5
      END
      UNTIL BITNO EQ #16.
      COM R0
      UNTILB PASFLG EQ #2
      CLR1CSR
      CACHON
      RETURN
      MSV11-P SINGLE BIT ERROR SYNDROME BIT TABLE
      SBESYN: .BYTE 16,13,23,25,26,31,32,34,43,45,46,51,52,54,61,64

```

```

      MOV CSR,BAD
      ;NO ERROR
      ;
      L227:;;;;;
      ;ENABLE SYNDROME BIT REGISTER
      ;GET SYNDROMES FROM CSR
      ;MASK SYNDROME BITS
      ;GET GOOD SYNDROMES
      MOVB SBESYN(R4),R3
      ;SHIFT INTO POSITION
      ;SET DIAG MODE IN DATA
      ;DO SYNDROME BITS AGREE
      CMP R3,CSR
      BEQ L230
      MOV #-1,HEADER
      MOV R3,GOOD
      MOV CSR,BAD
      ;
      L230:;;;;;
      ;CLEAR LUT
      ;SHIFT NEW DATA DEPENDING ON PASFLG
      CMPB PASFLG,#1
      BNE L231
      ;SHIFT BITNO TO THE LEFT
      ;
      BR L232
      L231:;;;;;
      ;SET CARRY BIT AND.....
      ;ROTATE LEFT
      ;
      L232:;;;;;
      ;UNTIL ALL BITS ARE DONE
      CMP BITNO,#16.
      BNE B33
      E33:;;;;;
      ;COMPLEMENT DATA AND REPEAT
      ;UNTIL 2 PASSES ARE COMPLETE!
      CMPB PASFLG,#2
      BNE B32
      E32:;;;;;
      ;CLEAR CSR
      ;TURN CACHE

```

8357 026442

MTPA21: SUBTST <<MTPA21 MARCHING 1'S & 0'S PATTERN TEST>>

;*****
;*SUBTEST MTPA21 MARCHING 1'S & 0'S PATTERN TEST
;*****

;READ,BYTESWAP-MODIFY,READ,DOWN

8358
8359 026442 014100
8360 026444 020200
8361 026446 001401
8362 026450 1044431\$: MOV -(R1),R0 ;V177640
CMP R2,R0 ;V177642
BEQ 2\$;V177644
PERR17 ;V1776468363
8364 026452 000311
8365 026454 011100
8366 026456 020300
8367 026460 001401
8368 026462 1044442\$: SWAB (R1) ;V177650
MOV (R1),R0 ;V177652
CMP R3,R0 ;V177654
BEQ 3\$;V177656
PERR20 ;V1776608369
8370 026464 020401
8371 026466 001365
8372 026470 0002073\$: CMP R4,R1 ;V177662 ;DONE?
BNE 1\$;V177664 ;NO - LOOP
RETURN ;V177666 ;YES - RETURN8373
8374 026472
8375 026472 011100
8376 026474 020300
8377 026476 001401
8378 026500 104444

MTPB21: ;READ,BYTESWAP-MODIFY,READ,UP

1\$: MOV (R1),R0 ;V177640
CMP R3,R0 ;V177642
BEQ 2\$;V177644
PERR20 ;V1776468379
8380 026502 000311
8381 026504 011100
8382 026506 020200
8383 026510 001401
8384 026512 1044432\$: SWAB (R1) ;V177650
MOV (R1),R0 ;V177652
CMP R2,R0 ;V177654
BEQ 3\$;V177656
PERR17 ;V1776608385
8386 026514 062701 000002
8387 026520 020501
8388 026522 001363
8389 026524 0002073\$: ADD #2,R1 ;V177662
CMP R5,R1 ;V177666 ;DONE?
BNE 1\$;V177670 ;NO - LOOP
RETURN ;V177672 ;YES - RETURN8390
8391 026526
8392 026526 011100
8393 026530 020200
8394 026532 001401
8395 026534 104443

MTPC21: ;READ,BYTESWAP-MODIFY,READ,UP

1\$: MOV (R1),R0 ;V177640
CMP R2,R0 ;V177642
BEQ 2\$;V177644
PERR17 ;V1776468396
8397 026536 000311
8398 026540 011100
8399 026542 020300
8400 026544 001401
8401 026546 1044442\$: SWAB (R1) ;V177650
MOV (R1),R0 ;V177652
CMP R3,R0 ;V177654
BEQ 3\$;V177656
PERR20 ;V1776608402
8403 026550 062701 000002
8404 026554 020501
8405 026556 001363
8406 026560 0002073\$: ADD #2,R1 ;V177662
CMP R5,R1 ;V177666 ;DONE?
BNE 1\$;V177670 ;NO - LOOP
RETURN ;V177672 ;YES - RETURN

8409	026562		MTPD21: ;READ,BYTESWAP-MODIFY,READ,DOWN			
8410	026562	014100	1\$:	MOV	-(R1),R0	;V177640
8411	026564	020300		CMP	R3,R0	;V177642
8412	026566	001401		BEQ	2\$;V177644
8413	026570	104444		PERR20		;V177646
8414						
8415	026572	000311	2\$:	SWAB	(R1)	;V177650
8416	026574	011100		MOV	(R1),R0	;V177652
8417	026576	020200		CMP	R2,R0	;V177654
8418	026600	001401		BEQ	3\$;V177656
8419	026602	104443		PERR17		;V177660
8420						
8421	026604	020401	3\$:	CMP	R4,R1	;V177662 ;DONE?
8422	026606	001365		BNE	1\$;V177664 ;NO - LOOP
8423	026610	000207		RETURN		;V177666 ;YES - RETURN
8424						

```

8427 026612      MTP022: SUBTST  <<MTP022      REFRESH & SHIFTING DIAGONAL TEST>>
;*****
;SUBTEST      MTP022  REFRESH & SHIFTING DIAGONAL TEST
;*****
8428      ;(1)  WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
8429      ;(2)  IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
8430      ;(3)  WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
8431      ;      (WITH CACHE OFF).
8432      KDIAG=8.      ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
8433 026612 000010      FOR EVEN := #1 TO #2      ;FOR DATA & COMPLEMENT DATA
      026612 012737 000001 002364      MOV #1,EVEN
      026620      B34:;;;;;
8434 026620      IF EVEN EQ #1
      026620 023727 002364 000001      CMP EVEN,#1
      026626 001005      BNE L235
8435 026630      LET R2 := ZEROS      MOV ZEROS,R2
8436 026630 013702 002336      LET R3 := ONES      MOV ONES,R3
      026634 013703 002614      MOV ONES,R3
8437 026640      ELSE
      026640 000404      BR L236
      026642      L235:;;;;;
8438 026642      LET R2 := ONES      MOV ONES,R2
      026642 013702 002614      LET R3 := ZEROS      MOV ZEROS,R3
8439 026646 013703 002336      END ;OF IF EVEN
8440 026652      FOR STRIPES := #0 TO #KDIAG 1      ;FOR THE NUMBER OF STRIPES
      026652 005037 002366      L236:;;;;;      CLR STRIPES
      026656      B35:;;;;;
8442
8443      ;WRITE LOOP
8444 026656 104423      CACHON      ;TURN CACHE ON
8445 026660      LET COUNT := STRIPES
8446 026660 013737 002366 002370      MOV STRIPES,COUNT
      026666 012701 060000      LET R1 := #FIRST      MOV #FIRST,R1
8447 026672      WHILE R1 LOS #LAST
      026672      B36:;;;;;
      026672 020127 157776      CMP R1,#LAST
      026676 101032      BHI L237
8448 026700      IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
      026700 005737 002370      TST COUNT
      026704 002003      BGE L240
      026706 012737 000007 002370      MOV #KDIAG-1,COUNT
      026714      L240:;;;;;
8449 026714      IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
      026714 032701 000374      BIT #374,R1
      026720 001002      BNE L241
      026722 005337 002370      DEC COUNT
      026726      L241:;;;;;
8450 026726      IF COUNT NE #0
      026726 005737 002370      TST COUNT
      026732 001404      BEQ L242
8451 026734      LET (R1) := R2
      026734 010211      MOV R2,(R1)

```

8452	026736			LET 2(R1) := R2		MOV R2,2(R1)
	026736	010261	000002			
8453	026742			ELSE		BR L243
	026742	000403				L242:::~::~
	026744					
8454	026744			LET (R1) := R3		MOV R3,(R1)
	026744	010311				
8455	026746			LET 2(R1) := R3		MOV R3,2(R1)
	026746	010361	000002			
8456	026752			END ;OF IF COUNT		
	026752					L243:::~::~
8457	026752			LET COUNT := COUNT - #1		DEC COUNT
	026752	005337	002370			
8458	026756			LET R1 := R1 + #4		ADD #4,R1
	026756	062701	000004			
8459	026762			END ;OF WHILE		BR B36
	026762	000743				L237:::~::~
	026764					E36:::~::~
	026764					
8460				;END OF WRITE LOOP		
8461						
8462	026764			IF DIAGFLAG IS FALSE THEN \$CALL REFRESH		TST DIAGFLAG
	026764	005737	002004			BNE L244
	026770	001002				JSR PC,REFRESH
	026772	004737	027166			
	026776					L244:::~::~
8463				;READ LOOP		
8464	026776			LET COUNT := STRIPES		MOV STRIPES,COUNT
	026776	013737	002366 002370			
8465	027004			LET R1 := #FIRST		MOV #FIRST,R1
	027004	012701	060000			
8466	027010	104424		CACHOFF	;TURN CACHE OFF	

8468	027012			WHILE R1 LOS #LAST		
	027012				837:::~::~	
	027012	020127	157776			CMP R1,#LAST
	027016	101046				BHI L245
8469	027020			IF COUNT LT #0 THEN LET COUNT := #KDIAG-1		TST COUNT
	027020	005737	002370			BGE L246
	027024	002003				MOV #KDIAG-1,COUNT
	027026	012737	000007 002370		L246:::~::~	
	027034				- #1	
8470	027034			IF #374 OFF.IN R1 THEN LET COUNT := COUNT		BIT #374,R1
	027034	032701	000374			BNE L247
	027040	001002				DEC COUNT
	027042	005337	002370		L247:::~::~	
	027046					
8471	027046			IF COUNT NE #0		TST COUNT
	027046	005737	002370			BEQ L250
	027052	001412				
8472	027054			LET R0 := (R1)		MOV (R1),R0
	027054	011100				
8473	027056			IF R2 NE R0		
	027056	020200				CMP R2,R0
	027060	001401				BEQ L251
8474	027062	104443		PERR17		
8475	027064			END ;OF IF R2		
	027064				L251:::~::~	
8476	027064			LET R0 := 2(R1)		MOV 2(R1),R0
	027064	016100	000002			
8477	027070			IF R2 NE R0		
	027070	020200				CMP R2,R0
	027072	001401				BEQ L252
8478	027074	104443		PERR17		
8479	027076			END ;OF IF R2		
	027076				L252:::~::~	
8480	027076			ELSE		BR L253
	027076	000411			L250:::~::~	
	027100					
8481	027100			LET R0 := (R1)		MOV (R1),R0
	027100	011100				
8482	027102			IF R3 NE R0		
	027102	020300				CMP R3,R0
	027104	001401				BEQ L254
8483	027106	104444		PERR20		
8484	027110			END ;OF IF R3		
	027110				L254:::~::~	
8485	027110			LET R0 := 2(R1)		MOV 2(R1),R0
	027110	016100	000002			
8486	027114			IF R3 NE R0		
	027114	020300				CMP R3,R0
	027116	001401				BEQ L255
8487	027120	104444		PERR20		
8488	027122			END ;OF IF R3		
	027122				L255:::~::~	
8489	027122			END ;OF IF COUNT		
	027122				L253:::~::~	
8490	027122			LET COUNT := COUNT - #1		
	027122	005337	002370			DEC COUNT
8491	027126			LET R1 := R1 + #4		

```

      027126 062701 000004
8492 027132      END ;OF WHILE
      027132 000727
      027134
      027134
8493      ;END OF READ LOOP
8494
8495 027134      END ;OF FOR STRIPES
      027134 005237 002366
      027140 023727 002366 000007
      027146 003643
      027150
8496 027150      END ;OF FOR EVEN
      027150 005237 002364
      027154 023727 002364 000002
      027162 003616
      027164
8497 027164 000207      RETURN
8498
8499 027166 REFRESH:SUBTST <<SUBR REFRESH DELAY>>
      ;*****
      ;*SUBTEST      SUBR      REFRESH DELAY
      ;*****
      ;DISTURB EACH ROW FOR > 3.2 MS
      FOR RO := #FIRST TO #FIRST+374 BY #4
      MOV #FIRST,RO
      B40:;;;;;
      ADD #4,RO
      CMP RO,#FIRST+374
      BLE B40
      E40:;;;;;
      MOV #FIRST+BIT14,RO
      B41:;;;;;
      CMP RO,#LAST+BIT14+374
      BHI L256
      ADD #4,RO
      BR B41
      L256:;;;;;
      E41:;;;;;
      027166 012700 060000
      027172
8502 027172 004737 027236      CALL REFSUB
8503 027176      END ;OF FOR RO
      027176 062700 000004
      027202 020027 060374
      027206 003771
      027210
8504 027210      LET RO := #FIRST+BIT14
      027210 012700 120000
      027214      WHILE RO LOS #LAST+BIT14+374
      027214 020027 020372
      027220 101005
8506 027222 004737 027236      CALL REFSUB
8507 027226      LET RO := RO + #4
      027226 062700 000004
8508 027232      END ;OF WHILE
      027232 000770
      027234
      027234
8509 027234 000207
8510 027236 012704 000640
8511 027242 062700 000002
8512 027246 005140
8513 027250 005120
8514 027252 005110
8515 027254 005110
8516 027256 077405
8517 027260 162700 000002
8518 027264 000207
      RETURN
REFSUB: MOV      #640,R4
      ADD      #2,RO
1$:      COM      -(RO)
      COM      (RO)+
      COM      (RO)
      COM      (RO)
      SOB      R4,1$
      SUB      #2,RO
      RETURN

```

8521 027266

MTPA24: SUBTST <<MTPA24 FAST GALLOPING PATTERN TEST>>

```

*****
;SUBTEST      MTPA24 FAST GALLOPING PATTERN TEST
*****

```

8522
8523
8524
8525
8526
8527
8528
8529
8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8540
8541
8542
8543
8544
8545
8546

```

;THE TOTAL TEST (INCLUDING SETUP) IS AS FOLLOWS
;(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
;*      STORED AT LOCATION BAKPAT
;(2) TEST BEGINS AT LOWEST LOCATION BEING TESTED
;*      (LETS NAME IT 'A')
;(3) LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
;(4) SWAPS BYTES FOR LOCATION 'A'.
;(5) READS 'A', READS 'B'
;(6) 'B' = 'B'+400 (ADDS 64 DOUBLE WORDS TO 'B')
;(7) REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
;(8) END OF THE BANK A+2
;(9) REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
;(10) AFTER EXECUTING THE TEST DATA IS COMPLEMENTED
;*      AND STEPS 1-9 ARE REPEATED
;REGISTERS ARE USED AS FOLLOWS
;R0      TEST DATA
;R1      'A'
;R2      'B'
;R3      BAKPAT
;R4      SWAPAT
;R5      LAST

```

;NOTE THE PATTERN STARTS AT MTPB24!!!!!!!!!!!!!!

;UIPAR'S

8547 027266 011100	1\$: MOV (R1),R0	;V177640	;READ 'A'
8548 027270 020004	CMP R0,R4	;V177642	;CHECK 'A'
8549 027272 001401	BEQ 2\$;V177644	;BR IF OK
8550 027274 104447	PERR23	;V177646	;REPORT ERROR
8551			
8552 027276 011200	2\$: MOV (R2),R0	;V177650	;READ 'B'
8553 027300 020003	CMP R0,R3	;V177652	;CHECK 'B'
8554 027302 001401	BEQ 3\$;V177654	;BR IF OK
8555 027304 104450	PERR24	;V177656	;REPORT ERROR
8556			
8557 027306 062702 000400	3\$: ADD #400,R2	;V177660	;BUMP 'B'
8558 027312 020205	CMP R2,R5	;V177664	;AT END YET?
8559 027314 101764	BLOS 1\$;V177666	;BR IF NO
8560			
8561 027316 062701 000002	ADD #2,R1	;V177670	;BUMP 'A'
8562 027322 000137 027326	JMP @MTPB24	;V177674	;GOTO V177260

8565 027326

MTPB24: SUBTST <<MTPB24 FAST GALLOP PART B>>

;*****
;*SUBTEST MTPB24 FAST GALLOP PART B
;*****8566
8567 027326 010411
8568 027330 020105
8569 027332 001001
8570 027334 000207
8571 027336 000137 027342
8572
8573 027342;SDPAR'S
MOV R4,(R1) ;V172260 ;WRITE 'A'
CMP R1,R5 ;V172262 ;DONE?
BNE 1\$;V172264 ;BR IF NO
RETURN ;V172266 ;YES RETURN
1\$: JMP @#MTPC24 ;V172270 ;GOTO V172360

MTPC24: SUBTST <<MTPC24 FAST GALLOP PART C>>

;*****
;*SUBTEST MTPC24 FAST GALLOP PART C
;*****8574
8575 027342 010102
8576 027344 011100
8577 027346 020004
8578 027350 001401
8579 027352 104447
8580 027354 000137 027306
8581;KDPAR'S
MOV R1,R2 ;V172360 ;RESET 'B' <- 'A'
MOV (R1),R0 ;V172362 ;READ 'A'
CMP R0,R4 ;V172364 ;CHECK 'A'
BEQ 1\$;V172366 ;BR IF OK
PERR23 ;V172370 ;REPORT ERROR
1\$: JMP @#MTPA24+20 ;V172372 ;GOTO V177660

3584 027360

8585 027360 000137 027430
8586 027364 010221
8587 027366 010321
8588 027370 077005
8589 027372 000207
8590
8591 027374

8592
8593
8594 027374 000137 027430
8595 027400 020221
8596 027402 001401
8597 027404 104451
8598 027406 005127
8599 027410 000000
8600 027412 020321
8601 027414 001401
8602 027416 104451
8603 027420 005167 177764
8604 027424 077015
8605 027426 000207
8606
8607
8608
8609 027430

8610
8611
8612
8613
8614
8615 027430 073427 000007
8616 027434 060305
8617 027436 005504
8618 027440 060204
8619 027442 062705 001057
8620 027446 000240
8621
8622 027450

8623 027450 005504
8624 027452 062704 047401
8625 027456 010503
8626 027460 010402
8627 027462 000137 027364

```

MTPA26: SUBTST <<MTPA26      RANDOM DATA (WRITE)>>
;*****
;*SUBTEST      MTPA26  RANDOM DATA (WRITE)
;*****
1$:      JMP      @#MTPC26      ;V177640      GOTO V172360
        MOV      R2,(R1)+      ;V177644
        MOV      R3,(R1)+      ;V177646
        SOB      R0,1$         ;V177650
        RETURN                     ;V177652

MTPB26: SUBTST <<MTPB26      RANDOM DATA (READ)>>
;*****
;*SUBTEST      MTPB26  RANDOM DATA (READ)
;*****
        .DSABL   AMA
        .ENABL   LSB
1$:      JMP      @#MTPC26      ;V177640      GOTO V172360
        CMP      R2,(R1)+      ;V177644
        BEQ      2$           ;V177646
        PERR25                     ;V177650
2$:      COM      (PC)+         ;V177652
        RANODD: 0              ;V177654      FOR ERROR REPORTING
        CMP      R3,(R1)+      ;V177656
        BEQ      3$           ;V177660
        PERR25                     ;V177662
3$:      COM      RANODD        ;V177664
        SOB      R0,1$         ;V177670
        RETURN                     ;V177672
        .DSABL   LSB
        .ENABL   AMA

MTPC26: SUBTST <<RANDOM NUMBER SUBPROGRAM>>
;*****
;*SUBTEST      RANDOM NUMBER SUBPROGRAM
;*****
        ;CALLER MUST SETUP
        ;      MOV      SEEDLO,R3
        ;      MOV      SEEDHI,R2
        ;      MOV      R3,R5
        ;      MOV      R2,R4
        ASHC     #7,R4          ;V172360
        ADD      R3,R5          ;V172364
        ADC      R4             ;V172366
        ADD      R2,R4          ;V172370
        ADD      #1057,R5       ;V172372
        NOP                          ;V172376      GOTO V172260

MTPD26: SUBTST <<RANDOM NUMBER SUBSUBPROGRAM>>
;*****
;*SUBTEST      RANDOM NUMBER SUBSUBPROGRAM
;*****
        ADC      R4             ;V172260
        ADD      #47401,R4      ;V172262
        MOV      R5,R3          ;V172266
        MOV      R4,R2          ;V172270
        JMP      @#MTPA26+4     ;V172272      GOTO V177644

```

8630 027466

MTP030: SUBTST <<MT0030 FLUSH OUT DBE'S>>

;*****
;SUBTEST MT0030 FLUSH OUT DBE'S
;*****

8631 027466 011002

1\$: MOV (R0),R2 ;V177640

8632 027470 010220

MOV R2,(R0)+ ;V177642

8633 027472 077103

SOB R1,1\$;V177644

8634 027474 000207

RETURN ;V177646

8635

8636 027476

MTP031: SUBTST <<MTP031 SOB-A-LONG TEST>>

;*****
;SUBTEST MTP031 SOB-A-LONG TEST
;*****

8637

8638 027476 000000

.DSABL AMA

8639 027500 077001

1\$: SOB R0,1\$;MOVE TERMINATOR

8640 027502 005167 177772

COM 1\$;SOB TILL R0 UNDERFLOWS

8641 027506 020167 177766

CMP R1,1\$;WRITE COMPLEMENT OF SOB

8642 027512 001403

BEQ 2\$;READ & CHECK FOR NOT "SOB R0,DOT"

8643 027514 104454

PERR30 ;OK - SKIP

8644 027516 010167 177756

MOV R1,1\$

8645 027522 005167 177752

2\$: COM 1\$;CORRECT SOB INSTRUCTION

8646 027526 010200

MOV R2,R0 ;REINITIALIZE SOB CONSTANT

8647

8648 027530 010503

;UPDATE MOVE REGISTERS

8649 027532 005725

MOV R5,R3

8650 027534 010504

TST (R5)+

8651 027536 020537 002522

MOV R5,R4 ;BUMP (SAFELY) BY 2

8652 027542 001001

CMP R5,@LINK1

8653 027544 000207

BNE 3\$;DONE?

8654

RETURN ;NO - SKIP

8655 027546 014344

3\$: MOV -(R3),-(R4) ;YES

8656 027550 001376

BNE 3\$

8657 027552 000752

BR 1\$

8658 000056

SOBLENGTH=.-MTP031

8659

.ENABL AMA

8687 027554

MTP032: SUBTST <<MTP032 WRITE RECOVERY TEST>>

;*****

;*SUBTEST MTP032 WRITE RECOVERY TEST

;*****

;THE TEST ACTUALLY EXECUTED ALREADY IN THE MEMORY UNDER TEST.

;THIS CODE INSURES THAT IT CHANGED MEMORY TO HAVE

;1/2 BANK OF #5141 WHICH IS A "COM -(R1)" INSTRUCTION AND

;1/2 BANK OF #110 WHICH IS A "JMP (R0)" INSTRUCTION.

8688					
8689					
8690					
8691					
8692					
8693	027554	012401	1\$:	MOV (R4)+,R1	;V177640 ;GET DATA FROM LOWER 1/2 BANK
8694	027556	020102		CMP R1,R2	;V177642 ;IS IT #5141?
8695	027560	001401		BEQ 2\$;V177644 ;YES - SKIP
8696	027562	104430		PERR02	;V177646 ;NO - TAKE ERROR TRAP
8697	027564	077305	2\$:	SOB R3,1\$;V177650 ;LOOP FOR 1/2 BANK
8698	027566	013703		MOV @LINK1,R3	;V177652 ;RESTORE LOOP SIZE
8699	027572	012400	3\$:	MOV (R4)+,R0	;V177656 ;GET DATA FROM UPPER 1/2 BANK
8700	027574	020005		CMP R0,R5	;V177660 ;IS IT #110?
8701	027576	001401		BEQ 4\$;V177662 ;YES - SKIP
8702	027600	104427		PERR01	;V177664 ;NO- TAKE ERROR TRAP
8703	027602	077305	4\$:	SOB R3,3\$;V177666 ;LOOP FOR 1/2 BANK
8704	027604	000207		RETURN	

002522

8707 027606

8708
8709 027606 000000
8710 027610 000000
8711 027612 000261
8712 027614 105511
8713 027616 100402
8714 027620 105212
8715 027622 000773
8716
8717
8718 027624 102401
8719 027626 104461
8720
8721 027630 000242
8722 027632 105212
8723 027634 103402
8724 027636 102001
8725 027640 100401
8726 027642 104461
8727
8728
8729
8730 027644 010701
8731 027646 162701 000036
8732 027652 010102
8733 027654 005202
8734
8735
8736 027656 010503
8737 027660 005725
8738 027662 010504
8739
8740
8741 027664 020537 002522
8742 027670 001001
8743 027672 000207
8744
8745
8746 027674 014344
8747 027676 001376
8748 027700 005011
8749 027702 000743
8750 000076
8751

```
MTP033: SUBTST <<MTP033      BRANCH COBBLE TEST>>
;*****
;SUBTEST      MTP033  BRANCH GOBBLE TEST
;*****
      .DSABL  AMA
      0
BGTEST. 0      ;MOVE TERMINATOR
      0      ;TEST WORD (TWO BYTES)
BRGOBB: SEC      ;SET CARRY (TO BE ADDED TO "BGTEST")
      ADCB      (R1)      ;INCREMENT LOW BYTE OF "BGTEST"
      BMI      1$      ;BRANCH WHEN BIT7 IS SET
      INCB      (R2)      ;INCREMENT HIGH BYTE OF "BGTEST"
      BR      BRGOBB      ;LOOP 128 TIMES

      ;NOW CHECK FOR CORRECT CONDITION CODES
1$:   BVS      2$      ;BR IF V-BIT SET (SHOULD BE)
      PERR35      ;NO - REPORT ERROR AND ABORT TEST
      ;COND CODES NOT EQUAL TO 1010
2$:   CLV
      INCB      (R2)      ;CLEAR V-BIT
      BCS      3$      ;INCREMENT HIGH BYTE OF "BGTEST" ONCE MORE
      BVC      3$      ;BR IF C-BIT SET (SHOULD NOT BE)
      BMI      4$      ;BR IF V-BIT CLEAR (SHOULD NOT BE)
      PERR35      ;BR IF N-BIT SET (SHOULD BE)
      ;NO - REPORT ERROR AND ABORT TEST
      ;COND CODES NOT EQUAL TO 1010

      ;UPDATE TEST POINTERS
4$:   MOV      PC,R1
5$:   SUB      #5$-BGTEST,R1
      MOV      R1,R2
      INC      R2

      ;UPDATE MOVE REGISTERS
      MOV      R5,R3
      TST      (R5)+
      MOV      R5,R4      ;BUMP (SAFELY) BY 2

      ;DONE?
      CMP      R5,#LINK1
      BNE      6$
      RETURN      ;DONE?
      ;NO - SKIP
      ;YES - RETURN

      ;MOVE CODE 1 LOCATION
6$:   MOV      -(R3),-(R4)
      BNE      6$
      CLR      (R1)
      BR      BRGOBB      ;CLEAR TEST WORD "BGTEST"
      ;RUN MOVED CODE AGAIN

GBLENGTH=.-MTP033
      .ENABL  AMA
```

8753 027704

```
MTP034: SUBTST <<MTP034      SOFT ERROR - BACKGROUND PATTERN TEST>>
;*****
;SUBTEST      MTP034  SOFT ERROR - BACKGROUND PATTERN TEST
;*****
1$:      MOV      R2,(R0)+      ;V177640
        SOB      R1,MTP034      ;V177642
        RETURN      ;V177644
2$:      MOV      (R4)+,R1      ;V177646
        CMP      R1,R2          ;V177650
        BEQ      3$            ;V177652
        PERR02      ;V177654
        NOP          ;V177656
3$:      SOB      R3,2$          ;V177660
        RETURN      ;V177662
```

8754 027704 010220
8755 027706 077102
8756 027710 000207
8757 027712 012401
8758 027714 020102
8759 027716 001402
8760 027720 104430
8761 027722 000240
8762 027724 077306
8763 027726 000207

8765 027730

```

MTP035:SUBTST  <<MTP035      WORST CASE NOISE PARITY TEST>>
;*****
;SUBTEST      MTP035  WORST CASE NOISE PARITY TEST
;*****

```

8766 027730 012737 000003 002076

MOV #3,NOPAR ;SET PARITY TRAPS TO RETURN TO "PARTHERE"

8767

8768 027736 012700 060000

FOR RO := #FIRST TO #LAST BY #4000

MOV #FIRST,RO

8769 027742 012737 000005 002150

MOV #BIT2!BIT0,CSR ;SET WRITE WRONG PARITY & PAR. TRAPS INTO CSR

8770 027750 104425

LOADCSR

8771 027752 012737 030006 002304

MOV #1\$,PARTHERE

8772 027760 011010

MOV (R0),(R0) ;WWP TEST LOCATION

8773 027762 005710

TST (R0)

8774 027764 010037 002034

MOV R0,ADDRESS

8775 027770 104050

ERROR +50

8776 027772 004737 050446

CALL PERBNK

8777 027776 032763 002000 002666

BIT #BIT10,CONFIG+2(R3)

8778 030004 001002

BNE 2\$

8779 030006 104426

1\$: READCSR

8780 030010 104512

ERRGEN

8781

8782 030012 104503

2\$: CLR1CSR

8783 030014 011010

MOV (R0),(R0)

8784 030016 012737 000001 002150

;CLEAR WRONG PARITY IN MEMORY

8785 030024 104425

MOV #BIT0,CSR

8786 030026 012737 030040 002304

LOADCSR

8787 030034 005710

MOV #3\$,PARTHERE

8788 030036 000405

TST (R0)

8789 030040 010037 002034

BR 4\$

8790 030044 104050

3\$: MOV R0,ADDRESS

8791 030046 004737 050446

ERROR +50

8792 030052

CALL PERBNK

030052 062700 004000

4\$: END; OF FOR

ADD #4000,RO

030056 020027 157776

CMP RO,#LAST

030062 003727

BLE B42

030064

E42:::~::~:

8793

8794 030064 005037 002076

CLR NOPAR

;RESET PARITY TRAP ACTION

8795 030070 000207

RETURN

8797 030072

```

MTP036: SUBTST      <<MTP036      CORRECTION CODE TEST>>
;*****
;SUBTEST      MTP036  CORRECTION CODE TEST
;*****
;
; THIS TEST CHECKS TO SEE THAT EACH BIT OF A DATA WORD
; CAN BE CORRECTED INDIVIDUALLY FROM A ZERO TO A ONE AND
; VISA VERSA.
CACHOFF           ;TURN OFF CACHE
CLR8 PASFLG       ;CLEAR PASFLG
CBREG             ;ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT           ;
;
;B43:;;;;;
;INCREMENT LOOP COUNTER      INCB PASFLG
;INDEX TO SINGLE BIT ERROR TABLE
;MOV #1,R4
;CLEAR INNER LOOP COUNTER   CLR BITNO
;SELECT DATA TO BE CORRECTED BY PASSNO
;CMPB PASFLG,#1
;BNE L257
;DATA=0;BIT TO BE CORRECTED IS A ONE
;MOV #1,R5
;
;BR L260
;
;L257:;;;;;
;DATA=177776;BIT TO BE CORRECTED IS A ZERO
;MOV #177776,R5
;
;L260:;;;;;
;
;B44:;;;;;
;
;INCREMENT BIT POINTER
;POINT TO NEXT SET OF CHECK BITS
;INC R4
;GET NEXT SET OF CHECK BITS
;MOVB PTABLE(R4),R2
;SHIFT TO LINE UP IN CSR
;ENABLE DIAG MODE
;GET CHECK BITS TO BE WRITTEN
;MOV R2,CSR
;LOAD CSR WITH DATA
;WRITE DATA TO TEST ADDRESS
;MOV R0,(R1)
;CORRECT 3BE
;WAS DATA CORRECTED???
;CMP (R1),R5
;BEQ L261
;MOV #60000,ADDRESS
;MOV R2,CHECK
;MOV R5,TSTDAT
;
;LET ADDRESS := #60000
;LET CHECK := R2
;LET TSTDAT := R5
;LET TSTDAT+2 := (R1)

```

```

8798
8799
8800
8801
8802 030072 104424
8803 030074 105037 002264
8804 030100 104513
8805 030102
      030102
8806 030102
      030102 105237 002264
8807 030106
      030106 012704 177777
8808 030112
      030112 005037 002324
8809 030116
      030116 123727 002264 000001
      030124 001003
8810 030126
      030126 012705 000001
8811 030132
      030132 000402
      030134
8812 030134
      030134 012705 177776
8813 030140
      030140
8814 030140
      030140
8815
8816 030140 005237 002324
8817 030144
      030144 005204
8818 030146
      030146 116402 030276
8819 030152 072227 000005
8820 030156 052702 000004
8821 030162
      030162 010237 002150
8822 030166 104425
8823 030170
      030170 010011
8824 030172 005711
8825 030174
      030174 021105
      030176 001412
8826 030200
      030200 012737 060000 002034
8827 030206
      030206 010237 002314
8828 030212
      030212 010537 002246
8829 030216

```



```

:          MSV11-P SINGLE BIT ERROR CHECK BIT TABLE
:
:TABLE: .BYTE 2,7,37,31,32,25,26,20,57,51,52,45,46,40,75,70

```

8850 030316

MTP037: SUBTST <<MTP037 CHECK ECC DISABLE TEST>>

```

*****
;SUBTEST      MTP037  CHECK ECC DISABLE TEST
*****

```

8851

8852

8853

8854

8855 030316 104424

8856 030320 005037 002044

8857 030324 005037 002314

8858 030330 104475

8859 030332 012737 000100 002314

8860 030340 104475

8861 030342 005011

8862 030344 005711

8863 030350 011137 002052

8864 030354 012737 060000 002034

8865 030362 104037

8866 030364

8867 030364 104423

8868 030366 000207

```

; THIS TEST CHECKS THAT ECC CAN BE DISABLED AND THAT
; NO CORRECTION TAKES PLACE WITH ECC DISABLED.

```

CACHOFF ;TURN OFF CACHE

LET GOOD := #0 ;GOOD DATA FOR ERROR PRINT OUT

LET CHECK := #0 ;CLEAR CHECK BIT FIELD CLR GOOD

CB1CSR ;ENABLE SYNDROME/CHECK BIT REGISTER CLR CHECK

LET CHECK := #100 ;SBE CHECK BITS

CB1CSR ;WRITE CHECK BITS TO CB REGISTER MOV #100,CHECK

LET (R1) := #0 ;WRITE CHECK BITS TO MEMORY

IF (R1) NE #0 ;WAS CORRECTION MADE???? CLR (R1)

LET BAD := (R1) ;YES IT WAS.....ERROR TST (R1)

LET ADDRESS := #60000 ; BEQ L266

ERROR +37 ;

END ;

CACHON ;TURN ON CACHE L266:::~::~

RETURN ;

8871 030370

```

MTP041: SUBTST <<MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST>>
;*****
;SUBTEST MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
;*****
;
; THIS TEST CHECKS TO SEE IF THE CORRECT ADDRESS APPEARS
; IN CSR BITS 5-11 ON A DOUBLE ERROR.
;
      LET R4 := BANK          ;GET STARTING BANK NUMBER
                                MOV BANK,R4
      ASH #9.,R4              ;SHIFT INTO POSTION TO MATCH ADDRESS IN CSR
      BIC #1C7740,R4          ;CLEAR OFF EXTRANEQUS BITS
      LET R0 := #-40          ;INIT CSR ADDRESS TO 0 - 1K (BIT 5 = 1K ADD.)
                                MOV #-40,R0
      LET R1 := #FIRST #4000 ;GET LOW ADDRESS IN BANK
                                MOV #FIRST,R1
                                SUB #4000,R1
      LET PASFLG :B= #0       ;INIT PASFLG
                                CLRB PASFLG
      CLR CHECK                ;CLEAR CHECK BIT FIELD TO BE LOADED
      CB1CSR                   ;ENABLE CHECK/SYNDROME BIT REGISTER
      REPEAT
                                B45:;;;;;
      INCB PASFLG              ;INC LOOP COUNTER
      LET R0 := R0 + #40       ;INC CSR ADDRESS TO BE EXPECTED
                                ADD #40,R0
      LET R1 := R1 + #4000
                                ADD #4000,R1
      LET CHECK := #1340       ;DOUBLE ERROR CHECK BITS
                                MOV #1340,CHECK
      CB1CSR                   ;WRITE DOUBLE ERROR CHECK BITS
      LET (R1) := #0           ;WRITE DATA AND D.E. CHK BITS AT A=0
                                CLR (R1)
      CLR1CSR                  ;CLEAR CSR
      TST (R1)                 ;READ ADDRESS TO GET DOUBLE ERROR
      READCSR                  ;READ CSR FOR CORRECT ADDRESS
      LET R5 := CSR
                                MOV CSR,R5
      BIC #1C7740,R5
      LET R2 := R0
                                ;GET CORRECT ADDRESS
                                MOV R0,R2
      ADD R4,R2                ;ADD STARTING BANK TO DOUBLE BIT ADDRESS
      NOP                      ;DEBUG AIDE
      IF R2 NE R5              ;DO ADDRESSES AGREE?
                                CMP R2,R5
                                BEQ L267
                                LET BAD := R2
                                ;
                                MOV R2,BAD
                                LET GOOD := R5
                                ;
                                MOV R5,GOOD
      PERR31                   ;NO ERROR
      END                      ;
                                L267:;;;;;
      LET (R1) := #0
                                ;
                                CLR (R1)
      CB1CSR                   ;ENABLE CHECK/SYNDROME BIT REGISTER
      UNTILB PASFLG EQ PASCNT ;DO 16K AT A TIME

```

```

8872
8873
8874
8875
8876 030370 013704 002102
      030370 072427 000011
8877 030374 072427 000011
8878 030400 042704 170037
8879 030404 012700 177740
      030404 012700 177740
8880 030410 012701 060000
      030410 012701 060000
      030414 162701 004000
8881 030420 105037 002264
      030420 105037 002264
8882 030424 005037 002314
8883 030430 104475
8884 030432
      030432 105237 002264
8885 030432 105237 002264
8886 030436 062700 000040
      030436 062700 000040
8887 030442 062701 004000
      030442 062701 004000
8888 030446 012737 001340 002314
      030446 012737 001340 002314
8889 030454 104475
8890 030456 005011
      030456 005011
8891 030460 104503
8892 030462 005711
8893 030464 104426
8894 030466 013705 002150
      030466 013705 002150
8895 030472 042705 170037
8896 030476 010002
      030476 010002
8897 030500 060402
8898 030502 000240
8899 030504 020205
      030504 020205
      030506 001405
8900 030510 010237 002052
      030510 010237 002052
8901 030514 010537 002044
      030514 010537 002044
8902 030520 104455
8903 030522
      030522
8904 030522 005011
      030522 005011
8905 030524 104475
8906 030526

```

E4

CVMJAA0 MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15-Feb-85 16:10 Page 280-1
MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST

SEQ 0250

CMPB PASFLG,PASCT
BNE B45
E45:::~::~

030526 123737 002264 002570
030534 001336
030536
8907 030536 104503
8908 030540 000207
8909

CLR1CSR
RETURN

;
;

8912 030542

MTP042: SUBST <<MTP042 EXTENDED ADDRESS TO CSR ON ERROR TEST>>

;*****
;SUBTEST MTP042 EXTENDED ADDRESS TO CSR ON ERROR TEST
;*****

8913

8914

8915

8916

8917

8918 030542 104424

8919 030544

030544 013704 002102

8920 030550

030550 023727 002102 000177

030556 001004

8921 030560

030560 012737 000001 002570

8922 030566

030566 000403

030570

8923 030570

030570 012737 000002 002570

8924 030576

030576

8925 030576 042704 177607

8926 030602 072427 000002

8927 030606 052704 040000

8928 030612 062737 000400 172352

8929 030620

030620 105037 002264

8930 030624

030624 013705 002102

8931 030630 042705 177770

8932 030634 072527 000011

8933 030640 052705 000020

8934 030644 104513

8935 030646

030646

8936 030646 105237 002264

8937 030652

030652 012737 000104 002150

8938 030660 104425

8939 030662

030662 005011

8940 030664 104503

8941 030666 005711

8942 030670 104426

8943 030672 042737 020000 002150

8944 030700

030700 023705 002150

030704 001406

8945 030706

030706 013737 002150 002052

8946 030714

030714 010537 002044

8947 030720 104023

8948 030722

CACHOFF

LET R4 := BANK

IF BANK EQ #177

LET PASCNT := #1

ELSE

LET PASCNT := #2

END

BIC #C170,R4

ASH #2,R4

BIS #BIT14,R4

ADD #400,KIPAR5

LET PASFLG := #0

LET R5 := BANK

BIC #C7,R5

ASH #9,R5

BIS #BIT4,R5

CBREG

REPEAT

INCB PASFLG

LET CSR := #104

LOADCSR

LET (R1) := #0

CLR1CSR

TST (R1)

READCSR

BIC #BIT13,CSR

IF CSR NE R5 THEN

LET BAD := CSR

LET GOOD := R5

ERROR +23

END

;TURN OFF CACHE MEMORY

;GET BANK NUMBER TO FIGURE OUT EXTENDED ADDRESS

MOV BANK,R4

CMP BANK,#177

BNE L271

MOV #1,PASCNT

BR L272

L271:::;;

MOV #2,PASCNT

L272:::;;

;CLEAR OFF LOWER BITS

;SHIFT TO LINE UP WITH CSR

;SET EXTENDED ADDRESS BIT

;SET UP PAR TO POINT TO TOP OF A BANK

;INIT LOOP COUNTER

CLRB PASFLG

;R5 GETS THE BANK NUMBER

MOV BANK,R5

;CLEAR ALL BUT THE LOWER BITS

;ROTATE INTO POSTION

;SET UP SBE INDICATOR ;;DATA TO BE EXPECTED

;ENABLE CHECK/SYNDROME BIT REGISTER

B46:::;;

;INCR LOOP COUNTER

;WRITE CHECK BITS TO CSR WITH DIAG MODE

MOV #104,CSR

;LOAD CSR WITH DATA

;WRT ZEROS AT A=0 AND SINGLE ERROR BITS

CLR (R1)

;CLEAR CSR

;READ A=0;DATA BIT 0 SHOULD BE CORRECTED TO A 1

;READ CSR FOR DATA

;CLEAR POSSIBLE INHIBIT MODE IN DATA "CSR"

;HAS SINGLE ERROR BITS SET IN CSR?

CMP CSR,R5

BEQ L273

MOV CSR,BAD

MOV R5,GOOD

```

      030722
8949 030722      012737 040000 002150      LET CSR := #40000 ;WRITE EQB BIT TO CSR      L273:;;;;;
      030722      104425      MOV #40000,CSR
8950 030730      104426
8951 030732      104426
8952 030734      042737 020000 002150      LOADCSR ;
      READCSR ;READ FOR CORRECT EXTENDED Q-BUS ADDRESS
8953 030742      023704 002150      BIC #BIT13,CSR ;CLEAR INHIBIT MODE POINTER IN DATA
      030746      001411      IF CSR NE R4 THEN ;READ EQB ADDRESS
      030750      LET BAD : CSR ;      CMP CSR,R4
      030750      013737 002150 002052      BEQ L274
8954 030750      013737 002150 002052      MOV CSR,BAD
8955 030756      010437 002044      LET GOOD :- R4 ;      MOV CSR,BAD
      030756      010437 002044      SET HEADER ;      MOV R4,GOOD
8956 030762      012737 177777 002612      ERROR +23 ;      MOV #-1,HEADER
8957 030770      104023      END ;
8958 030772      030772      LET (R1) := #0 ;CLEAR LUT      L274:;;;;;
8959 030772      005011      LET R1 := #137776 ;SET UP NEW ADDRESS      CLR (R1)
8960 030774      012701 137776      ADD #740,R5 ;ADD TO GET NEW ADDRESS      MOV #137776,R1
      030774      062705 000740      BIS #BIT9,R4 ;SET BIT 9 SINCE WE ARE ASSERTING A1 ON PASS2
8961 031000      052704 001000      CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
8962 031004      104513      UNTILB PASFLG EQ PASCNT ;LOOP 2 TIMES
8963 031010      123737 002264 002570      CMPB PASFLG,PASCNT
      031012      001312      BNE B46
      031022
8965 031022      104503      CLR1CSR ;CLEAR CSR      E46:;;;;;
8966 031024      104423      CACHON ;TURN ON CACHE
8967 031026      000207      RETURN

```

8969 031030

MTP043: SUBTST <<MTP043 WRITE BYTE CLEARS SINGLE BIT ERROR TEST>>

;*****
;SUBTEST MTP043 WRITE BYTE CLEARS SINGLE BIT ERROR TEST
;*****

```

8970
8971
8972
8973
8974
8975 031030 104424
8976 031032 104513
8977 031034 105037 002264
8978 031040
      031040 010102
      031042 005202
8979 031044
      031044 012704 000001
8980 031050
      031050
8981 031050 105237 002264
8982 031054
      031054 012737 000604 002150
8983 031062 104425
8984 031064
      031064 010411
8985 031066 104503
8986 031070
      031070 112712 000377
8987 031074 104426
8988 031076 042737 177757 002150
8989 031104
      031104 023727 002150 000020
      031112 001407
8990 031114
      031114 012737 000020 002044
8991 031122
      031122 013737 002150 002052
8992 031130 104060
8993 031132
      031132
8994 031132 104513
8995 031134 005711
8996 031136 104426
8997 031140 042737 174037 002150
8998 031146
      031146 023727 002150 000300
      031154 001412
8999 031156
      031156 012737 177777 002612
9000 031164
      031164 012737 000300 002044
9001 031172
      031172 013737 002150 002052
9002 031200 104061
9003 031202
      031202
9004 031202 005302

```

```

;
; THIS TEST CHECKS TO SEE IF A SINGLE BIT ERROR WILL BE CORRECTED DURING
; THE READ PORTION OF A WRITE BYTE AND THAT THE CORRECT CHECK BITS WILL
; BE GENERATED ON A WRITE.
;
CACHOFF ;TURN OFF CACHE
CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
CLRB PASFLG ;CLEAR LOOP COUNTER
LET R2 := R1 + #1 ;R2 POINTS TO HIGH BYTE

MOV R1,R2
INC R2

LET R4 := #1 ;INITIAL DATA = 1
MOV #1,R4

REPEAT ;
      B47:;;;;;
      INCB PASFLG ;INCREMENT LOOP COUNTER
      LET CSR := #604 ;WRITE CHECK BITS CORRESPONDING TO DATA OF 0
                        MOV #604,CSR
      LOADCSR ;WRITE CSR
      LET (R1) := R4 ;WRITE DATA OF 1 CREATING A SINGLE BIT ERROR
                        MOV R4,(R1)
      CLR1CSR ;WRITE CSR TO NORMAL MODE
      LET (R2) := #377 ;WRITE BYTE OF WORD
                        MOV B #377,(R2)
      READCSR ;READ CSR
      BIC #1C20,CSR ;SEE IF SBE INDICATOR IS SET
      IF CSR NE #20 ;IS SBE SET????
                        CMP CSR,#20
                        BEQ L276
      LET GOOD := #20 ;
                        MOV #20,GOOD
      LET BAD := CSR ;
                        MOV CSR,BAD
      ERROR +60 ;
      END ;
                        L276:;;;;;
      CBREG ;WRITE CSR TO DIAG MODE
      TST (R1) ;READ SAO FOR CORRECT CHECK BITS
      READCSR ;READ CSR
      BIC #1C3740,CSR ;MASK OUT CHECK BIT FIELD
      IF CSR NE #300 ;WERE CORRECT CHECK BITS GENERATED????
                        CMP CSR,#300
                        BEQ L277
      SET HEADER ;
                        MOV #-1,HEADER
      LET GOOD := #300 ;
                        MOV #300,GOOD
      LET BAD := CSR ;
                        MOV CSR,BAD
      ERROR +61 ;
      END ;
                        L277:;;;;;
      DEC R2 ;POINT TO HIGH BYTE AND REPEAT

```

9005	031204			LET R4 := #400	;BIT 0 OF HIGH BYTE	
	031204	012704	000400			MOV #400,R4
9006	031210			UNTILB PASFLG EQ #2	;DO HIGH AND LOW BYTE	
	031210	123727	002264			CMPB PASFLG,#2
	031216	001314				BNE B47
	031220					E47:::;::;::
9007	031220	104423		CACHON	;TURN ON CACHE	
9008	031222	000207		RETURN	;	


```

9010 031224      MTP044: SUBTST <<MTP044      SHIFTING CHECK BITS THROUGH THE CSR TEST>>
;*****
;*SUBTEST      MTP044      SHIFTING CHECK BITS THROUGH THE CSR TEST
;*****
; THIS TEST CHECKS THE ABILITY TO READ AND WRITE CHECKBITS INTO MEMORY
; BY SHIFTING A ONE BIT THROUGH A FIELD OF ZEROS. THE CSR IS READ FOR THE
; CORRECT PATTERNS. THE TEST IS THEN REPEATED ON A ZERO BIT THROUGH A
; FIELD OF ALL ONES.
;
; REPEAT
; ILC;REV B
;*****
9011
9012
9013
9014
9015
9016
9017 031224      CACHOFF
9018 031224      104424      LET PASFLG :B= #0
9019 031226      105037      002264      ;TURN OFF CACHE
9020 031232      012705      174037      ;INIT PASFLG
9021 031236      104475      ;CHECK BIT MASK FOR CSR
9022 031240      012702      000046      ;CLR PASFLG
9023 031244      ;ENABLE CHECK/SYNDROME BIT REGISTER
9024 031244      105237      002264      ;SET UP INITIAL CSR DATA
9025
9026
9027 031250      005037      002266      ;MOV #174037,R5
9028 031254      ;REPEAT
9029 031254      005237      002266      ;B51:;;;;;
9030 031260      010204      ;INC LOOP COUNTER
9031 031262      010237      002150      INCB PASFLG
9032 031266      104425      ;CHK BITS = 1
9033 031270      005011      ;DISABLE ECC;DIAG CHK SET
9034 031272      005105      ;INIT PASSNO(INNER LOOP COUNTER)
9035 031274      010546      CLR PASSNO
9036 031276      040416      REPEAT
9037 031300      040504      LET PASSNO := PASSNO + #1
9038 031302      052604      ;INC LOOP COUNTER
9039 031304      010437      002150      INC PASSNO
9040 031310      104425      LET R4 := R2
9041 031312      104426      ;COPY R2 TO R4
9042 031314      013703      002150      MOV R2,R4
9043 031320      042703      020000      LET CSR := R2
9044 031324      020304      ;GET CSR DATA TO BE WRITTEN
9045 031330      010137      002034      MOV R2,CSR
9046 031334      ;WRITE SBE CHECK BITS TO CSR
;WRITE DATA AND CHECK BITS AT A=0
;CLR (R1)
;COMPLEMENT MASK
;SAVE R5 ON STACK
;CREATE AN XOR FUNCTION
;
;
;MOV R4,CSR
;LOAD CSR WITH COMPLEMENT CHECK BITS
;READ CSR FOR COMPLEMENT CHECK BITS
;COPY CSR DATA TO R3
;MOV CSR,R3
;CLEAR ANY POSSIBLE INHIBIT MODE POINTER
;READ CSR FOR PROPER CHECK BITS
;CMP R3,R4
;BEQ L301
;
;LET ADDRESS := R1
;
;MOV R1,ADDRESS
;
;LET GOOD := R4
;

```

9047	031334	010437	002044		LET BAD := R3	MOV R4,GOOD
9048	031340	010337	002052		SET HEADER	MOV R3,BAD
9049	031344	012737	177777	002612	PERR37	MOV #-1,HEADER
9050	031352	104463			END	::ILC::REVB
9051	031354	005105			COM R5	L301:::~::~
9052	031356	005711			TST (R1)	;COMPLEMENT MASK
9053	031360	000240			NOP	;READ CHECK BITS AT A=0 INTO CSR
9054	031362	104426			READCSR	;READ CSR FOR CORRECT CHECK BITS
9055	031364	040537	002150		BIC R5,CSR	;MASK OUT CHECK BIT FIELD
9056	031370	010204			LET R4 := R2	;GET CHECK BITS THAT WERE WRITTEN
9057	031372	040504			BIC R5,R4	MOV R2,R4
9058	031374				IF R4 NE CSR	;MASK OUT CHECK BIT FIELD
9059	031400	001413				;ARE CHECK BITS THE SAME?
9060	031402	010437	002044		LET GOOD := R4	CMP R4,CSR
9061	031406	013737	002150	002052	LET BAD := CSR	BEQ L302
9062	031414	010137	002034		LET ADDRESS := R1	MOV R4,GOOD
9063	031420	012737	177777	002612	SET HEADER	MOV CSR,BAD
9064	031426	104464			PERR40	MOV R1,ADDRESS
9065	031430	040502			END	MOV #-1,HEADER
9066	031432	123727	002264	000001	BIC R5,R2	::ILC::REVB
9067	031440	001002			IFB PASFLG EQ #1	L302:::~::~
9068	031442	006302			ASL R2	;SHIFT CHECK BITS AND CREATE NEW DATA FOR CSR
9069	031444	000413			ELSE	;SELECT FUNCTION
9070	031446	005105			COM R5	CMPB PASFLG,#1
9071	031450	010546			MOV R5,-(SP)	BNE L303
9072	031452	040216			BIC R2,(SP)	;DO A FIELD OF ZEROS---->ONES
9073	031454	040502			BIC R5,R2	;SHIFT CHECK BITS
9074	031456	052602			BIS (SP)+,R2	;DO A FIELD OF ONES ---->ZEROS
9075	031460	006302			ASL R2	BR L304
9076	031462	010546			MOV R5,-(SP)	L303:::~::~
9077	031464	040216			BIC R2,(SP)	;TAKE OUT CHECK BIT FIELD
9078	031466	040502			BIC R5,R2	;SHIFT CHECK BITS
9079	031470	052602			BIS (SP)+,R2	;PUT BACK CHECK BIT FIELD
9080	031472	005105			COM R5	;COMPLEMENT DATA PATTERN
9081	031474				END	L304:::~::~
9082	031474	062702	000006		LET R2 := R2 + #6	;ADD 6 SO THAT WRITE ON CSR WILL ENABLE DIAG MODE
9083	031500	123727	002266	000006	UNTILB PASSNO EQ #6	ADD #6,R2
	031506	001262				;DO ALL CHECK BITS
						CMPB PASSNO,#6
						BNE B52

9084	031510			LET R2 := #3706	E52:::;;	
	031510	012702	003706		;REPEAT WITH FIELD OF ONES	
9085	031514			UNTILB PASFLG EQ #2	MOV #3706,R2	
	031514	123727	002264		CMPB PASFLG,#2	
	031522	001250			BNE B51	
	031524				E51:::;;	
9086	031524	104503		CLR1CSR	:	
9087	031526	005011		CLR (R1)		
9088	031530	062701	000100	ADD #100,R1	;ALL 256 TO ADDRESS	;ILC;;REVB
9089	031534			UNTIL R1 EQ ENDADD	:	;ILC;;REVB
	031534	020137	002562			CMP R1,ENDADD
	031540	001231				BNE B50
	031542				E50:::;;	
9090	031542	104423		CACHON	;TURN ON CACHE	
9091	031544	000207		RETURN	:	

9093 031546

```

MTP045: SUBTST <<MTP045      SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST>>
;*****
;SUBTEST      MTP045  SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST
;*****
;
;  THIS TEST CHECKS TO SEE IF THE DOUBLE BIT ERROR INDICATOR IS SET
;  ON A DOUBLE BIT ERROR AND THE CORRECT SYNDROMES ARE LATCHED INTO THE
;  CSR.  THIS TEST IS THEN REPEATED WITH MULTIPLE ERROR CHECK/SYNDROME BITS
;
;CACHOFF      ;TURN OFF CACHE
CBREG         ;ENABLE CHECK/SYNDROME BIT REGISTER
LET PASSNO := #0      ;CLEAR LOOP COUNTER          CLR PASSNO

9102 031556 005037 002266      LET GOOD := #3744      ;GOOD DATA          MOV #3744,GOOD
9103 031564 012737 003744 002044      LET CSR := #3144      ;DBE CHECK BITS FOR CSR      MOV #3144,CSR
9104 031572 012737 003144 002150      REPEAT
;
;          B53:;;;;;
9105 031572 005237 002266      INC PASSNO      ;
9106 031576 104425      LOADCSR      ;WRITE DBE CHECK BITS TO CSR
9107 031600 005011      LET (R1) := #0      ;WRITE ZEROS AND DBL ERROR CHK BITS A=0      CLR (R1)
9108 031602 104503      CLR1CSR      ;CLEAR CSR OUT
9109 031604 005711      TST (R1)      ;READ A=0 TO GET DOUBLE BIT ERROR
9110 031606 104426      READCSR      ;WAS UNCORRECCABLE ERROR BIT SET???
9111 031610 032737 100000 002150      IF #BIT15 OFF.IN CSR ;
;          BIT #BIT15,CSR
;          BNE L310
9112 031620 012737 177777 002612      SET HEADER      ;
;          MOV #-1,HEADER
9113 031626 013737 002150 002052      LET BAD := CSR      ;
;          MOV CSR,BAD
9114 031634 104063      ERROR +63      ;BIT NOT SET
9115 031636 031636      END      ;
;          L310:;;;;;
9116 031636 104514      SYNREG      ;ENABLE SYNDROME BIT REGISTER
9117 031640 104426      READCSR      ;READ CSR FOR CORRECT SYNDROME BITS
9118 031642 000240      NOP      ;DEBUG AIDE
9119 031644 042737 174033 002150      BIC #+C3744,CSR      ;MASK SYNDROMES OUT
9120 031652 023737 002150 002044      IF CSR NE GOOD THEN ;CHECK IF DOUBLE ERROR BIT IS SET
;          CMP CSR,GOOD
;          BEQ L311
9121 031662 013737 002150 002052      LET BAD := CSR      ;BAD DATA          MOV CSR,BAD
9122 031670 012737 177777 002612      SET HEADER      ;
;          MOV #-1,HEADER
9123 031676 104042      ERROR +42      ;
9124 031700 031700      END      ;
;          L311:;;;;;
9125 031700 005011      CLR (R1)      ;CLEAR LUT
9126 031702 012737 003604 002044      LET GOOD := #3604      ;REPEAT WITH MULTIPLE ERROR SYNDROMES      MOV #3604,GOOD
9127 031710 012737 003004 002150      LET CSR := #3004      ;MULTIPLE ERROR CHECK BITS      MOV #3004,CSR
9128 031716 023727 002266 000002      UNTIL PASSNO EQ #2      ;          CMP PASSNO,#2

```

N4

CVMJAAO MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15 Feb 85 16:10 Page 285-1
MTP045 SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST

SEQ 0259

031724 001322
031726
9129 031726 104503
9130 031730 104423
9131 031732 000207

CLR1CSR
CACHON
RETURN

:
:

BNE B53
E53: : : : : :

9133 031734

MTP046: SUBTST <<MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED>>

;*****
;*SUBTEST MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED
;*****

9134

9135

9136

9137

9138

9139 031734 005037 002266

9140 031740 104424

9141 031742

031742

9142 031742 005237 002266

9143 031746 005000

9144 031750 105037 002264

9145 031754 104513

9146 031756

031756

9147 031756 105237 002264

9148 031762 012704 177777

9149 031766 012737 000001 002076

9150 031774 005037 002324

9151 032000 123727 002264 000001

9152 032006 001003

9153 032010 012705 000001

9154 032014 000402

9155 032016

9156 032016 012705 177776

9157 032022

9158 032022 005037 002072

9159 032026 005237 002324

9160 032032 005204

9161 032034 116402 030276

9162 032040 072227 000005

9163 032044 052702 000006

9164 032050 010237 002150

9165 032054 104425

9166 032056 010011

9167 032060

CLR PASSNO
CACHOFF
REPEAT

LET PASSNO := PASSNO + #1

CLR R0
CLRB PASFLG
CBREG
REPEAT

LET PASFLG := PASFLG + #1

LET R4 := #-1

LET NOPAR := #1

LET BITNO := #0

IFB PASFLG EQ #1

LET R5 := #1

ELSE

LET R5 := #177776

END

REPEAT

LET PARCNT := #0

INC BITNO

LET R4 := R4 + #1

LET R2 := PTABLE(R4)

ASH #5,R2

BIS #BIT2:BIT1,R2

LET CSR := R2

LOADCSR

LET (R1) := R0

IF PASSNO EQ #1

;CLEAR OUTER LOOP COUNTER
;TURN OFF CACHE

B54:;;;;;

INC PASSNO

;CLEAR DATA

;CLEAR PASFLG

;ENABLE CHECK/SYNDROME BIT REGISTER

B55:;;;;;

;INCREMENT LOOP COUNTER

INCB PASFLG

;INDEX TO SINGLE BIT ERROR TABLE

MOV #-1,R4

;ENABLE PARITY ACTION

MOV #1,NOPAR

;CLEAR INNER LOOP COUNTER

CLR BITNO

;SELECT DATA TO BE CORRECTED BY PASSNO

CMPB PASFLG,#1

BNE L313

;DATA=0;BIT TO BE CORRECTED IS A ONE

MOV #1,R5

BR L314

L313:;;;;;

;DATA=177776;BIT TO BE CORRECTED IS A ZERO

MOV #177776,R5

L314:;;;;;

B56:;;;;;

;CLEAR PARITY COUNTER

CLR PARCNT

;INCREMENT BIT POINTER

;POINT TO NEXT SET OF CHECK BITS

INC R4

;GET NEXT SET OF CHECK BITS

MOVB PTABLE(R4),R2

;SHIFT TO LINE UP IN CSR

;ENABLE DIAG MODE

;GET CHECK BITS TO BE WRITTEN

MOV R2,CSR

;LOAD CSR WITH DATA

;WRITE DATA TO TEST ADDRESS

MOV R0,(R1)

;WRITE CSR

SEQ 0261

Address	Instruction	Comment
032060	023727	002266 000001
032066	001002	
9167 032070	104471	
9168 032072	000401	
032072		
032074		
9169 032074	104507	
9170 032076		
032076		
9171 032076	005711	
9172 032100	004737	032220
9173 032104	104426	
9174 032106		
032106	032737	100000 002150
032114	001007	
9175 032116		
032116	013737	002150 002052
9176 032124		
032124	012737	177777 002612
9177 032132	104045	
9178 032134		
032134		
9179 032134	104503	
9180 032136	005011	
9181 032140		
032140	123727	002264 000001
032146	001002	
9182 032150	006305	
9183 032152		
032152	000402	
032154		
9184 032154	000261	
9185 032156	006105	
9186 032160		
032160		
9187 032160		
032160	023727	002324 000020
032166	001315	
032170		
9188 032170	005100	
9189 032172		
032172	123727	002264 000002
032200	001266	
032202		
9190 032202		
032202	023727	002266 000002
032210	001254	
032212		
9191 032212	104503	
9192 032214	104423	
9193 032216	000207	
9194		
9195		
9196 032220		CHKTRP: IF PASSNO EQ #1
032220	023727	002266 000001
032226	001011	
9197 032230		

Address	Op-Code	Register	Constant	Label	Instruction	Comment
9198	032230	023727	002072	000001		CMP PARCNT, #1
	032236	001004				BNE L326
	032240				SET HEADER	
9199	032240	012737	177777	002612		MOV # -1, HEADER
9200	032246	104057			ERROR +57	
	032250				END	
9201	032250				ELSE	
	032250	000410				BR L327
9202	032252				IF PARCNT NE #1	
	032252	023727	002072	000001		CMP PARCNT, #1
9203	032260	001404				BEQ L330
	032262				SET HEADER	
9204	032262	012737	177777	002612		MOV # -1, HEADER
9205	032270	104064			ERROR +64	
	032272				END	
9206	032272				END	
9207	032272	000207			RETURN	

9209 032274

MTP047: SUBTST <<MTP047 NO CSR UPDATE ON SBE WITH EXSISTING DBE>>
 ;*****
 ;*SUBTEST MTP047 NO CSR UPDATE ON SBE WITH EXSISTING DBE
 ;*****

9210									
9211									
9212									
9213									
9214									
9215	032274	104424							
9216	032276								
	032276	013704	002102						
9217	032302	072427	000011						
9218	032306	042704	170037						
9219	032312	052704	100000						
9220	032316	104513							
9221	032320								
	032320	012737	003144	002150					
9222	032326	104425							
9223	032330								
	032330	005011							
9224	032332								
	032332	012737	000104	002150					
9225	032340	104425							
9226	032342								
	032342	005012							
9227	032344	104503							
9228	032346	005711							
9229	032350	104426							
9230	032352	042737	020000	002150					
9231	032360								
	032360	023704	002150						
	032364	001411							
9232	032366								
	032366	013737	002150	002052					
9233	032374								
	032374	010437	002044						
9234	032400								
	032400	012737	177777	002612					
9235	032406	104063							
9236	032410								
	032410								
9237	032410	052704	000020						
9238	032414	005712							
9239	032416	104426							
9240	032420	042737	020000	002150					
9241	032426								
	032426	023704	002150						
	032432	001411							
9242	032434								
	032434	013737	002150	002052					
9243	032442								
	032442	010437	002044						
9244	032446								
	032446	012737	177777	002612					
9245	032454	104051							
9246	032456								

```

;
; THIS TEST CHECKS TO SEE THAT THE CSR CONTENTS WILL NOT CHANGE
; WITH A SINGLE BIT ERROR WHEN A DOUBLE BIT ERROR ALREADY
; EXISTS.
;
; CACHOFF
; LET R4 := BANK
; TURN OFF CACHE
; GET BANK NUMBER
; MOV BANK,R4
;
; ASH #9,R4
; BIC #1C7740,R4
; BIS #BIT15,R4
; CBREG
; LET CSR := #3144
; SHIFT INTO PLACE
; MASK OUT UNWANTED BITS
; SET UP GOOD DATA
; ENABLE CHECK/SYNDROME BIT REGISTER
; CHECK BITS FOR DOUBLE BIT ERROR
; MOV #3144,CSR
;
; LOADCSR
; LET (R1) := #0
; WRITE DBE CHECK BITS
; CLR (R1)
;
; LET CSR := #104
; WRITE SBE CHECK BITS
; MOV #104,CSR
;
; LOADCSR
; LET (R2) := #0
; WRITE SBE CHECK BITS AT ADDRESS + 4K
; CLR (R2)
;
; CLR1CSR
; TST (R1)
; READCSR
; BIC #BIT13,CSR
; IF CSR NE R4
; CLEAR CSR
; READ DBE LOCATION
; READ FOR CSR DBE INDICATOR
; CLEAR INHIBIT MODE POINTER
;
;
; CMP CSR,R4
; BEQ L331
;
; MOV CSR,BAD
;
; MOV R4,GOOD
;
; MOV #-1,HEADER
;
;
; L331:::
; SET BIT IN GOOD DATA
; READ SBE
; READ CSR FOR NO CHANGE
; CLEAR INHIBIT MODE POINTER
;
;
; CMP CSR,R4
; BEQ L332
;
; MOV CSR,BAD
;
; MOV R4,GOOD
;
; MOV #-1,HEADER
;
;
; LET BAD := CSR
;
; LET GOOD := R4
;
; SET HEADER
;
; ERROR +63
;
; END
;
; BIS #20,R4
; TST (R2)
; READCSR
; BIC #BIT13,CSR
; IF CSR NE R4
;
;
; LET BAD := CSR
;
; LET GOOD := R4
;
; SET HEADER
;
; ERROR +51
;
; END
;

```

9247	032456	104503	CLR1CSR	;CLEAR 1 CSR	L332: : : : :
9248	032460	005011	CLR (R1)	:	
9249	032462	005012	CLR (R2)	:	
9250	032464	104423	CACHON	;TURN ON CACHE	
9251	032466	000207	RETURN		

```

9253                                     .SBTTL  MISC SUBROUTINES
9254
9255 032470  REGCOPY:SUBTST  <<SUBR  COPY R0 TO R4,R1 TO R3, & R2 TO R5>>
;*****
;*SUBTEST      SUBR      COPY R0 TO R4,R1 TO R3, & R2 TO R5
;*****
9256 032470  010004      MOV      R0,R4
9257 032472  010103      MOV      R1,R3
9258 032474  010205      MOV      R2,R5
9259 032476  000207      RETURN
9260
9261 032500  FLIPWARN:SUBTST <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
;*****
;*SUBTEST      FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
;*****
9262 032500  010046      PUSH     R0
9263 032500  005237 002616      INC     FLIPLOC
9264 032506  042737 177774 002616      BIC     #C3,FLIPLOC
9265 032514  022737 000001 002616      CMP     #1,FLIPLOC
9266 032522  001414      BEQ     1$
9267 032524  022737 000002 002616      CMP     #2,FLIPLOC
9268 032532  001413      BEQ     2$
9269 032534  022737 000003 002616      CMP     #3,FLIPLOC
9270 032542  001414      BEQ     3$
9271 032544  005000      CLR     R0
9272 032546  013704 002614      MOV     ONES,R4
9273 032552  000414      BR      4$
9274 032554      1$:  CLEAR    R0,R4
9275 032554  005000      CLR     R0
9276 032556  005004      CLR     R4
9277 032560  000411      BR      4$
9278 032562  012700 000401 2$:  MOV     #401,R0
9279 032566  013704 002614      MOV     ONES,R4
9280 032572  000404      BR      4$
9281 032574  012700 000401 3$:  MOV     #401,R0
9282 032600  012704 000401      MOV     #401,R4
9283 032604  010037 024762 4$:  MOV     R0,WARN2
9284 032610  010037 024776      MOV     R0,WARN3
9285 032614  010037 025022      MOV     R0,WARN4
9286 032620  010037 025036      MOV     R0,WARN5
9287 032624      POP     R0
9288 032624  012600      MOV     (SP)+,R0
9289 032626  000207      RETURN

```

9288 032630

BACKGND:SUBTST <<SUBR WRITE BACKGROUND>>

;*****

;*SUBTEST SUBR WRITE BACKGROUND

;*****

9289

;WRITES DATA FROM R2

9290 032630 104415

SAVREG

9291 032632 012700 060000

MOV #FIRST,R0

9292 032636 012701 040000

MOV #SIZE,R1

9293 032642 012737 000207 024644

MOV #207,MTP000+4

;WARNING PUTTING "RETURN;" AFTER WRITE

9294 032650 012737 024640 002262

MOV #MTP000,SUPDOADD

9295 032656 004737 024446

CALL SUPD03

9296 032662 012737 000240 024644

MOV #240,MTP000+4

;RESTORE 'NOP' AFTER WRITE

9297 032670 104416

RESREG

9298 032672 000207

RETURN

9299

9301 032674

```

GETCSR: SUBTST <<SUBR GET CSR INFORMATION FROM CONFIGURATION TABLE>>
;*****
;*SUBTEST SUBR GET CSR INFORMATION FROM CONFIGURATION TABLE
;*****
;INPUTS : NONE
;
;OUTPUT : CSRNO = CSR NUMBER
;
MOV BANKINDEX,R2 ;GET INDEX INTO CONFIG TABLE
MOV CONFIG(R2),R3 ;MOV IT INTO R3
SWAB R3
ASL R3
BIC #C36,R3 ;CLEAR OFF SOME BITS
MOV R3,CSRNO ;SAVE CSR NUMBER
RETURN

```

```

9302
9303
9304
9305
9306 032674 013702 002104
9307 032700 016203 002664
9308 032704 000303
9309 032706 006303
9310 032710 042703 177741
9311 032714 010337 002152
9312 032720 000207

```

9315 032722

PCONFIG:SUBTST <<SUBR PRINT CONFIGURATION MAP>>

;*****
;SUBTEST SUBR PRINT CONFIGURATION MAP
;*****

PUSH TKVEC,TKVEC+2,R0

MOV TKVEC,-(SP)
MOV TKVEC+2,-(SP)
MOV R0,-(SP)

9316 032722

032722 013746 000060

032726 013746 000062

032732 010046

9317 032734

032734 010637 033334

9318 032740

032740 012737 033302 000060

9319 032746

032746 012737 000340 000062

9320 032754

032754 017700 147664

9321 032760

032760 042737 000200 177776

9322 032766

032766 052777 000100 147646

9323

9324 032774

032774 104401 065123

TYPE MSG001

TYPEIT ,MSG001

.DSABL CRF

9325 033000

033000 104401 065205

TYPE MSG002

TYPEIT ,MSG002

.DSABL CRF

9326 033004

033004 104401 065262

TYPE MSG003

TYPEIT ,MSG003

.DSABL CRF

9327 033010

033010 022737 000060 002556

CMP #60, LASTBANK

9328 033016

033016 002006

BGE NOOJ

9329

9330 033020

033020 032777 000020 147610

;IF FAT PAPER ON TERMINAL GOTO 1\$

IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1

BIT #SW4,@SWR

BEQ L333

JMP PCONF1

L333:::;;

9331 033034

033034 012700 000074

NOOJ:

MOV #60, R0

9332 033040

033040 010004

MOV R0,R4

9333 033042

033042 005001

CLEAR R1,R3

033044 005003

CLR R1

CLP R3

9334 033046

033046 104401 065367

TYPE MSG004

TYPEIT ,MSG004

.DSABL CRF

9335 033052

033052 004737 033336

CALL TCONFIG

;GO TYPE CONFIGURATION (1ST HALF)

9336 033056

033056 022737 000060 002556

CMP #60, LASTBANK

9337 033064

033064 002106

BGE PCONF2

9338 033066

033066 104401 002660

TYPE \$CRLF

TYPEIT , \$CRLF

.DSABL CRF

9339 033072

033072 104401 066076

TYPE MSG017

;PRINT SPACE(S)

TYPEIT ,MSG017

.DSABL CRF

9340 033076

033076 104401 065570

TYPE MSG11A

TYPEIT ,MSG11A

.DSABL CRF

9341 033102

033102 104401 002660

TYPE \$CRLF

TYPEIT , \$CRLF

.DSABL CRF

9342 033106

033106 104401 066076

TYPE MSG017

;PRINT SPACE(S)

TYPEIT ,MSG017

9343	033112			.DSABL	CRF	
	033112	104401	065656	TYPE	MSG011	
				TYPEIT	,MSG011	
9344	033116			.DSABL	CRF	
	033116	104401	002660	TYPE	\$CRLF	
				TYPEIT	, \$CRLF	
9345	033122			.DSABL	CRF	
	033122	104401	066076	TYPE	MSG017	;PRINT SPACE(S)
				TYPEIT	,MSG017	
9346	033126			.DSABL	CRF	
	033126	104401	065744	TYPE	MSG012	
				TYPEIT	,MSG012	
9347	033132	012701	000360	.DSABL	CRF	
9348	033136	010103		MOV	#60.*2*2,R1	
9349	033140	004737	033336	MOV	R1,R3	
9350	033144	022737	000170	CALL	TCONFIG	
9351	033152	002053		CMP	#170, LASTBANK	
9352	033154			BGE	PCONF2	
	033154	104401	002660	TYPE	\$CRLF	
				TYPEIT	, \$CRLF	
9353	033160			.DSABL	CRF	
	033160	104401	066076	TYPE	MSG017	;PRINT SPACE(S)
				TYPEIT	,MSG017	
9354	033164			.DSABL	CRF	
	033164	104401	066041	TYPE	MSG11B	
				TYPEIT	,MSG11B	
9355	033170			.DSABL	CRF	
	033170	104401	002660	TYPE	\$CRLF	
				TYPEIT	, \$CRLF	
9356	033174			.DSABL	CRF	
	033174	104401	066076	TYPE	MSG017	;PRINT SPACE(S)
				TYPEIT	,MSG017	
9357	033200			.DSABL	CRF	
	033200	104401	066043	TYPE	MSG11C	
				TYPEIT	,MSG11C	
9358	033204			.DSABL	CRF	
	033204	104401	002660	TYPE	\$CRLF	
				TYPEIT	, \$CRLF	
9359	033210			.DSABL	CRF	
	033210	104401	066076	TYPE	MSG017	;PRINT SPACE(S)
				TYPEIT	,MSG017	
9360	033214			.DSABL	CRF	
	033214	104401	066045	TYPE	MSG11D	
				TYPEIT	,MSG11D	
9361	033220	012701	000740	.DSABL	CRF	
9362	033224	010103		MOV	#740,R1	
9363	033226	012700	000010	MOV	R1,R3	
9364	033232	010004		MOV	#8.,R0	
9365	033234	004737	033336	MOV	R0,R4	
9366	033240	000420		CALL	TCONFIG	
9367				BR	PCONF2	
9368	033242	012700	000170	PCONF1:	MOV	#120.,R0
9369	033246	010004			MOV	R0,R4
9370	033250				CLEAR	R1,R3
	033250	005001				
	033252	005003				

CLR R1
CLR R3

Address	Offset	Hex	Hex	Label	Instruction	Comment
9371	033254	104401	066060		TYPEIT ,MSG014	;SPACE
	033254				.DSABL CRF	
9372	033260	104401	065656		TYPE MSG011	
	033260				TYPEIT ,MSG011	
					.DSABL CRF	
9373	033264	104401	065367		TYPE MSG004	
	033264				TYPEIT ,MSG004	
					.DSABL CRF	
9374	033270	104401	065744		TYPE MSG012	
	033270				TYPEIT ,MSG012	
					.DSABL CRF	
9375	033274	004737	033336		CALL TCONFIG	
9376	033300	000721			BR NOOJ1	
9377						
9378	033302	013706	033334	PCONF2:	MOV PCONFS,SP	;RESTORE STACK
9379	033306	042777	000100	147326	BIC #BIT6,@TKS	
9380	033314	117700	147324		MOVB @TK8,R0	;READ CHAR TO KILL FLAG
9381	033320				POP R0,TKVEC+2,TKVEC	
	033320	012600				MOV (SP)+,R0
	033322	012637	000062			MOV (SP)+,TKVEC+2
	033326	012637	000060			MOV (SP)+,TKVEC
9382	033332	000207			RETURN	
9383						
9384	033334	000000		PCONFS: 0		;STACK SAVED HERE!

9387 033336

SUBTST <<SUBR TYPE CONFIGURATION>>

```

*****
;SUBTEST      SUBR      TYPE CONFIGURATION
*****
;CALL:  MOV      #N,R0          ;N=NUMBER OF CHARACTERS
;        MOV      R0,R4          ;BACKUP
;        MOV      #K,R1          ;INDEX CONSTANT
;        MOV      R1,R3          ;BACKUP
;        CALL     TCONFIG        ;ACTUAL CALL
;        RETURN          ;ONLY RETURN
*****

```

9388
9389
9390
9391
9392
9393
9394
9395
9396
9397
9398
9399

```

*****
; ** ERROR **
*****
9400 033336 012737 000340 177776 TCONFIG:MOV      #340,PSW          ;DISABLE INTERUPTS
9401 033344          104401 065475      TYPE      MSG005
          033344          TYPEIT    ,MSG005
          .DSABL    CRF
9402 033350 032761 000001 002664 1$:  BIT      #BIT0,CONFIG(R1)      ;ERROR ON THIS BANK?
9403 033356 001403          BEQ      2$          ;NO - SKIP
9404 033360          TYPE      MSG013          ;PRINT "X"
          033360 104401 066056      TYPEIT    ,MSG013
          .DSABL    CRF
9405 033364 000402          BR      3$
9406 033366          TYPE      MSG014          ;PRINT SPACE
          033366 104401 066060      TYPEIT    ,MSG014
          .DSABL    CRF
9407 033372 062701 000004          3$:  ADD      #4,R1          ;BUMP POINTER
9408 033376 077014          SOB      R0,1$          ;LOOP UNTIL DONE
9409 033400 010400          MOV      R4,R0
9410 033402 010301          MOV      R3,R1

```

```

9413 ;*****
9414 ;** INTERLEAVE **
9415 ;*****
9416 ;THIS IS AN ENTRY POINT FROM ERROR REPORTS
9417 033404 012737 000340 177776 TCFIG1: MOV #340,PSW ;DISABLE INTERRUPTS
9418 033412 112737 000040 066062 MOVB #' ,MSG015 ;MOVE A BLANK IN TO BE PRINTED
9419 033420 TYPE MSG015
033420 104401 066062 TYPEIT ,MSG015
.DSABL CRF
IF NOTAB NE #0 THEN $RETURN

9420 033424 005737 002372 TST NOTAB
033424 001401 BEQ L334
033430 000207 RTS PC
033434 L334:;;;;;

9421 033434 010400 MOV R4,R0
9422 033436 010301 MOV R3,R1
9423
9424 ;*****
9425 ;** MEMORY TYPE **
9426 ;*****
9427 .ENABL LSB
9428 033440 TYPE MSG009
033440 104401 065544 TYPEIT ,MSG009
.DSABL CRF
9429 033444 033761 002106 002664 TCFIG2: BIT CPUBIT,CONFIG(R1)
9430 033452 001432 BEQ 17$
9431 033454 016105 002666 MOV CONFIG+2(R1),R5
9432 033460 000305 SWAB R5 ;GET MEMORY TYPE
9433 033462 042705 177770 BIC #C7,R5 ;CLEAR NON INTERESTING BITS
9434 033466 020527 000003 CMP R5,#3 ;IS IT A LEGAL MEMORY TYPE
9435 033472 003022 BGT 17$ ;IF IF SO BRANCH!!!!!!
9436 033474 IF #BIT0 SET.IN R5 ;IS IT AN ECC MEMORY????
033474 032705 000001 BIT #BIT0,R5
033500 001413 BEQ L335
9437 033502 IF #BIT1 SET.IN R5 ;IS IT A MSV11-P OR A MSV11-J???
033502 032705 000002 BIT #BIT1,R5
033506 001404 BEQ L336
9438 033510 112737 000120 066062 MOVB #'P,MSG015 ;IT IS A MSV11-P
9439 033516 000403 ELSE ;
033516 000403 BR L337
033520 L336:;;;;;
9440 033520 112737 000105 066062 MOVB #'E,MSG015 ;IT IS A MSV11-J
9441 033526 END ;
033526 L337:;;;;;
9442 033526 ELSE ;
033526 000403 BR L340
033530 L335:;;;;;
9443 033530 112737 000120 066062 MOVB #'P,MSG015 ;IT IS A MSV11-L/P
9444 033536 END ;
033536 L340:;;;;;
9445 033536 000403 BR 8$
9446 033540 112737 000040 066062 17$: MOVB #' ,MSG015
9447 033546 000403 8$: TYPE MSG015
033546 104401 066062 TYPEIT ,MSG015
.DSABL CRF
9448 033552 IF NOTAB NE #0 THEN $RETURN
033552 005737 002372 TST NOTAB

```

```

033556 001401
033560 000207
033562
9449 033562 062701 000004
9450 033566 077052
9451 033570 010400
9452 033572 010301
9453
9454
9455
9456
9457
9458 033574
033574 104401 066064
9459 033600 112737 000040 066062 TCFIG3:
9460 033606 016105 002664
9461 033612 032705 000002
9462 033616 001414
9463 033620 042705 170377
9464 033624 000305
9465 033626 022705 000011
9466 033632 100002
9467 033634 062705 000007
9468 033640 062705 000060
9469 033644 110537 066062
9470 033650
033650 104401 066062
9471 033654
033654 005737 002372
033660 001401
033662 000207
033664
9472 033664 062701 000004
9473 033670 077035
9474 033672 010400
9475 033674 010301
9476
9477
9478
9479
9480 033676
033676 104401 065556
9481 033702 105761 002664 11$:
9482 033706 100006
9483 033710 112737 000120 066062
9484 033716
033716 104401 066062
9485 033722 000402
9486 033724
033724 104401 066060
9487 033730 062701 000004 15$:
9488 033734 077016

```

```

ADD #4,R1
SOB R0,TCFIG2
MOV R4,R0
MOV R3,R1
.DSABL LSB

;*****
;** CSR **
;*****
TYPE MSG016
TYPEIT ,MSG016
.DSABL CRF
MOVB #' ,MSG015
MOV CONFIG(R1),R5
BIT #BIT1,R5
BEQ 16$
BIC #C7400,R5
SWAB R5
CMP #9.,R5
BPL 10$
ADD #7,R5
ADD #60,R5
MOVB R5,MSG015
TYPE MSG015
TYPEIT ,MSG015
.DSABL CRF
IF NOTAB NE #0 THEN $RETURN

;*****
;** PROTECTED **
;*****
TYPE MSG010
TYPEIT ,MSG010
.DSABL CRF
TSTB CONFIG(R1)
BPL 14$
MOVB #'P,MSG015
TYPE MSG015
TYPEIT ,MSG015
.DSABL CRF
BR 15$
TYPE MSG014
TYPEIT ,MSG014
.DSABL CRF
ADD #4,R1
SOB R0,11$

```

```

;BUMP POINTER
;LOOP UNTIL DONE

L341:;;;;;
BEQ L341
RTS PC

;MAKE ASCII
;PLUG INTO MEMORY

TST NOTAB
BEQ L342
RTS PC

L342:;;;;;
;BUMP POINTER

;BANK PROTECTED?
;NO - SKIP

;PRINT SPACE

;BUMP POINTER
;LOOP UNTIL DONE

```

CVMJAA0 MSV11-J MEMORY DIAG.
SUBR TYPE CONFIGURATION

MACRO V05.01b Friday 15 Feb-85 16:10 Page 296 2

SEQ 0274

9489 033736 010400
9490 033740 010301
9491 033742 000207

MOV R4,R0
MOV R3,R1
RETURN

```

9494          .SBTTL  TRAP  PARITY ERROR HANDLER
9495          ;*****
9496          ;VECTOR TO HERE FROM TRAPS TO 114
9497          ;IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
9498          ;*****
9499          ;
9500          ;      CODE      ACTION
9501          ;      ---
9502          ;      0      PRINT UNEXPECTED PARITY TRAP
9503          ;      1      COUNT ERROR
9504          ;      2      SET "ABORT" / SETUP "BADPC" / RETURN VIA PCBUMP
9505          ;      3      RETURN VIA "PARTHERE"
9506          ;
9507 033744 022737 000001 002076 PARITY: CMP      #1,NOPAR      ;COUNTING PARITY ERRORS?
9508 033752 001003          BNE      1$      ;NO - SKIP
9509 033754 005237 002072          INC      PARCNT      ;PARITY ERROR COUNTER + 1
9510 033760 000002          RTI
9511 033762 022737 000002 002076 1$: CMP      #2,NOPAR      ;ACTION CODE = 2 ?
9512 033770 001013          BNE      2$      ;NO - SKIP
9513 033772          SET      ABORTFLAG      ;YES
9514 034000 012737 177777 002144          MOV      #-1,ABORTFLAG
9515 034004 004737 034152          CALL     BADSTACK      ;FIND BAD SP,PC,PSW OFF STACK
9516 034010 063716 002326          ADD      PCBUMP,(SP)      ;UPDATE RETURN PC
9517 034016 042766 000004 000002          BIC      #BIT2,2(SP)      ;SHOW FAILURE BY .NE.
9518 034020 000002          RTI
9519 034026 022737 000003 002076 2$: CMP      #3,NOPAR      ;ACTION CODE - 3 ?
9520 034030 001003          BNE      3$      ;NO - SKIP
9521 034034 013716 002304          MOV      PARTHERE,(SP)
9522 034036 000002          RTI
9523 034036 004737 034152          3$: CALL     BADSTACK      ;FIND BAD SP,PC,PSW OFF STACK
9523 034042 004737 034152          FATAL     32
9523 034042 005237 002064          INC      FATAL$      ;SET FATAL INDICATOR
9523 034046 104032          ERROR     +32
9523          .DSABL     CRF

```

```

9526 .SBTTL TRAP NON-EXISTANT MEMORY (HOLES) HANDLER
9527 ;*****
9528 ; VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
9529 ; CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
9530 ; 1) IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
9531 ; 2) TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
9532 ;*****
9533
9534 034050 022737 000001 002100 NONEXIST: CMP #1, NONEM ; COUNTING NON EXISTANT MEMORY ERRORS?
9535 034056 001011 BNE 2$ ; NO - SKIP
9536 034060 005237 002070 INC NEMCNT ; BUMP NON-EXISTANT MEMORY COUNTER
9537 034064 022737 000001 002070 CMP #1, NEMCNT ; FIRST ERROR?
9538 034072 001002 BNE 1$ ; NO - SKIP
9539 034074 010037 002034 MOV RO, ADDRESS ; ASSUME RO CONTAINS THE ADDRESS ACCESSED
9540 034100 000002 1$: RTI
9541 034102 005237 002070 2$: INC NEMCNT ; BUMP NON-EXISTANT MEMORY COUNTER
9542 034106 012701 000001 MOV #1, R1 ; DUMMY UP R1 FOR A FORCED SOB EXIT
9543 034112 000002 RTI
9544
9545 ;*****
9546 .SBTTL TRAP TIMEOUT (TRAP TO 4) HANDLER
9547 034114 004737 034152 TIMEOUT: CALL BADSTACK ; FIND BAD SP, PC, PSW OFF STACK
9548 034120 FATAL 6
9549 034120 005237 002064 INC FATAL$ ; SET FATAL INDICATOR
9550 034124 104006 ERROR +6
9551 .DSABL CRF
9552 ;*****
9553 .SBTTL TRAP MEMORY MANAGEMENT (TRAP TO 250) HANDLER
9554 034126 004737 034152 MMTRAP: CALL BADSTACK ; FIND BAD SP, PC, PSW OFF STACK
9555 034132 FATAL 7
9556 034132 005237 002064 INC FATAL$ ; SET FATAL INDICATOR
9557 034136 104007 ERROR +7
9558 .DSABL CRF
9559 .SBTTL TRAP RESERVED INSTRUCTION HANDLER
9560 034140 004737 034152 PDP1105: CALL BADSTACK ; FIND BAD SP, PC, PSW OFF STACK
9561 034144 FATAL 5
9562 034144 005237 002064 INC FATAL$ ; SET FATAL INDICATOR
9563 034150 104005 ERROR +5
9564 .DSABL CRF
9565
9566 BADSTACK: SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
9567 ;*****
9568 ; *SUBTEST FIND BAD SP, PC, & PSW FROM STACK
9569 ;*****
9570 MOV SP, BADSP
9571 ADD #2, BADSP
9572 MOV 2(SP), BADPC
9573 MOV 4(SP), BADPSW
9574 RETURN

```

```

9571          .SBTTL TRAP    KERNEL TRAP HANDLER
9572          ;*****
9573          ;KERNEL IS A TRAP THAT COMES HERE
9574          ;*****
9575
9576 034202 042766 140000 000002 $KERNEL:      BIC      #140000,2(SP)
9577 034210 000002          RTI
9578          ;*****
9579          .SBTTL TRAP    ENERGIZE TRAP HANDLER
9580 034212 052737 000001 177572 $ENERGIZE:BIS  #BIT0,MMR0
9581 034220 000002          RTI
9582          ;*****
9583          .SBTTL TRAP    DEENERGIZE TRAP HANDLER
9584 034222 042737 000001 177572 $DEENERGIZE:BIC #BIT0,MMR0
9585 034230 000002          RTI
9586          ;*****
9587          .SBTTL TRAP    CACHON TRAP HANDLER
9588 034232 005737 002544 $CACHN: TST    CACHKN          ;IS THERE A CACHE
9589 034236 001406          BEQ    1$          ;NO - RETURN
9590 034240 013737 002544 177746          MOV    CACHKN,CONTRL ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
9591 034246 052737 000001 177746          BIS    #BIT0,CONTRL ;DISABLE TRAPS (BUT NOT ABORTS)
9592 034254 000002          1$: RTI
9593          ;*****
9594          .SBTTL TRAP    CACHOFF TRAP HANDLER
9595 034256 005737 002544 $CACHF: TST    CACHKN          ;IS THERE A CACHE?
9596 034262 001403          BEQ    1$          ;NO - RETURN
9597          ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
9598 034264 053737 002550 177746          BIS    CACHKF,CONTRL
9599 034272 000002          1$: RTI

```

```

9602          .SBTTL TRAP LOAD CSR TRAP HANDLER
9603          ;LOAD CORRECT CSR WITH DATA IN CSR
9604          ;PROGRAM CSR'S ASSERT INHIBIT MODE POINTER WHEN LOADED
9605          $LOADC: PUSH      R0,R1          ;SAVE REGISTERS
                                MOV R0,-(SP)
                                MOV R1,-(SP)
                                034274 010046
                                034276 010146
9606          034300 013700 002152          MOV      CSRNO,R0          ;CREATE CSR ADDRESS
9607          034304          IF INHECC IS TRUE THEN GOTO 3$ ;DON'T WANT INH. MODE POINTER ON
                                TST INHECC
                                BNE 3$
                                034304 005737 002536
                                034310 001021
9608          034312 005737 002532          TST      PGMCSR          ;PROGRAM IN INTERLEAVED SPACE?
9609          034316 100007          BPL      1$          ;BRANCH IF NOT
9610          034320 113701 002533          MOVB     PGMCSR+1,R1      ;CHECK SECOND CSR
9611          034324 042701 177740          BIC      #+C37,R1      ;CLEAR UNNECESSARY BITS
9612          034330 020137 002152          CMP      R1,CSRNO      ;IS THIS THE CURRENT CSR?
9613          034334 001404          BEQ      2$          ;BRANCH IF IT IS
9614          034336 123737 002532 002152 1$: CMPB     PGMCSR,CSRNO  ;IS THIS THE CURRENT CSR?
9615          034344 001003          BNE      3$          ;BRANCH IF NOT
9616          034346 052737 020000 002150 2$: BIS      #BIT13,CSR    ;SET THE INHIBIT MODE POINTER TO 1ST 16K
9617          034354 013760 002150 172100 3$: MOV      CSR,CSRADD(R0) ;LOAD THE CSR
9618          034362          POP      R1,R0          ;RESTORE REGISTERS
                                034362 012601          MOV (SP)+,R1
                                034364 012600          MOV (SP)+,R0
9619          034366 000002          RTI
9620
9621          .SBTTL TRAP READ CSR TRAP HANDLER
9622          ;READ THE CORRECT CSR INTO LOCATIONS CSR
9623          $READC: PUSH      R0
                                034370 010046          MOV R0,-(SP)
                                034372 013700 002152
9624          034372 013700 002152          MOV      CSRNO,R0
9625          034376 016037 172100 002150          MOV      CSRADD(R0),CSR      ;READ IT
9626          034404          POP      R0
                                034404 012600          MOV (SP)+,R0
9627          034406 000002          RTI

```


PC	Instruction	Op	Op2	Op3	Op4	Op5	Op6	Op7	Op8	Op9	Op10	Op11	Op12	Op13	Op14	Op15	Op16	Op17	Op18	Op19	Op20	Op21	Op22	Op23	Op24	Op25	Op26	Op27	Op28	Op29	Op30	Op31	Op32	Op33	Op34	Op35	Op36	Op37	Op38	Op39	Op40	Op41	Op42	Op43	Op44	Op45	Op46	Op47	Op48	Op49	Op50	Op51	Op52	Op53	Op54	Op55	Op56	Op57	Op58	Op59	Op60	Op61	Op62	Op63	Op64	Op65	Op66	Op67	Op68	Op69	Op70	Op71	Op72	Op73	Op74	Op75	Op76	Op77	Op78	Op79	Op80	Op81	Op82	Op83	Op84	Op85	Op86	Op87	Op88	Op89	Op90	Op91	Op92	Op93	Op94	Op95	Op96	Op97	Op98	Op99	Op100	Op101	Op102	Op103	Op104	Op105	Op106	Op107	Op108	Op109	Op110	Op111	Op112	Op113	Op114	Op115	Op116	Op117	Op118	Op119	Op120	Op121	Op122	Op123	Op124	Op125	Op126	Op127	Op128	Op129	Op130	Op131	Op132	Op133	Op134	Op135	Op136	Op137	Op138	Op139	Op140	Op141	Op142	Op143	Op144	Op145	Op146	Op147	Op148	Op149	Op150	Op151	Op152	Op153	Op154	Op155	Op156	Op157	Op158	Op159	Op160	Op161	Op162	Op163	Op164	Op165	Op166	Op167	Op168	Op169	Op170	Op171	Op172	Op173	Op174	Op175	Op176	Op177	Op178	Op179	Op180	Op181	Op182	Op183	Op184	Op185	Op186	Op187	Op188	Op189	Op190	Op191	Op192	Op193	Op194	Op195	Op196	Op197	Op198	Op199	Op200	Op201	Op202	Op203	Op204	Op205	Op206	Op207	Op208	Op209	Op210	Op211	Op212	Op213	Op214	Op215	Op216	Op217	Op218	Op219	Op220	Op221	Op222	Op223	Op224	Op225	Op226	Op227	Op228	Op229	Op230	Op231	Op232	Op233	Op234	Op235	Op236	Op237	Op238	Op239	Op240	Op241	Op242	Op243	Op244	Op245	Op246	Op247	Op248	Op249	Op250	Op251	Op252	Op253	Op254	Op255	Op256	Op257	Op258	Op259	Op260	Op261	Op262	Op263	Op264	Op265	Op266	Op267	Op268	Op269	Op270	Op271	Op272	Op273	Op274	Op275	Op276	Op277	Op278	Op279	Op280	Op281	Op282	Op283	Op284	Op285	Op286	Op287	Op288	Op289	Op290	Op291	Op292	Op293	Op294	Op295	Op296	Op297	Op298	Op299	Op300	Op301	Op302	Op303	Op304	Op305	Op306	Op307	Op308	Op309	Op310	Op311	Op312	Op313	Op314	Op315	Op316	Op317	Op318	Op319	Op320	Op321	Op322	Op323	Op324	Op325	Op326	Op327	Op328	Op329	Op330	Op331	Op332	Op333	Op334	Op335	Op336	Op337	Op338	Op339	Op340	Op341	Op342	Op343	Op344	Op345	Op346	Op347	Op348	Op349	Op350	Op351	Op352	Op353	Op354	Op355	Op356	Op357	Op358	Op359	Op360	Op361	Op362	Op363	Op364	Op365	Op366	Op367	Op368	Op369	Op370	Op371	Op372	Op373	Op374	Op375	Op376	Op377	Op378	Op379	Op380	Op381	Op382	Op383	Op384	Op385	Op386	Op387	Op388	Op389	Op390	Op391	Op392	Op393	Op394	Op395	Op396	Op397	Op398	Op399	Op400	Op401	Op402	Op403	Op404	Op405	Op406	Op407	Op408	Op409	Op410	Op411	Op412	Op413	Op414	Op415	Op416	Op417	Op418
----	-------------	----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------

```

9661 .SBTTL TRAP ECC DISABLE ALL CSR'S TRAP HANDLER
9662 034570 012737 000002 002150 $ECCDIS:MOV #BIT1,CSR
9663 034576 004737 035314 CALL CSRROUT
9664 034602 000002 RTI
9665 .SBTTL TRAP ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
9666 034604 012737 000002 002150 $ECC1DIS:MOV #BIT1,CSR
9667 034612 104425 LOADCSR
9668 034614 000002 RTI
9669 .SBTTL TRAP INITIALIZE ALL CSR'S TRAP HANDLER
9670 034616 012737 000001 002150 $ECCINIT:MOV #BIT0,CSR
9671 034624 004737 035314 CALL CSRROUT
9672 034630 000002 RTI
9673 .SBTTL TRAP INITIALIZE 1 SELECTED CSR TRAP HANDLER
9674 034632 012737 000001 002150 $ECC1INIT:MOV #BIT0,CSR
9675 034640 104425 LOADCSR
9676 034642 000002 RTI
9677 .SBTTL TRAP ENABLE SBE PARITY TRAPS ON ALL CSR'S
9678 034644 012737 000003 002150 $ENASBE:MOV #BIT0!BIT1,CSR
9679 034652 004737 035314 CALL CSRROUT
9680 034656 000002 RTI
9681 .SBTTL TRAP ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
9682 034660 012737 000003 002150 $ENA1SBE:MOV #BIT0!BIT1,CSR
9683 034666 104425 LOADCSR
9684 034670 000002 RTI
9685 .SBTTL TRAP WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
9686 034672 013737 002314 002150 $CBCSR:MOV CHECK,CSR ;BITS 11-5
9687 034700 052737 000006 002150 BIS #BIT1!BIT2,CSR ;CHECK MODE
9688 034706 004737 035314 CALL CSRROUT
9689 034712 000002 RTI
9690 .SBTTL TRAP WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
9691 034714 013737 002314 002150 $CB1CSR:MOV CHECK,CSR ;BITS 11-5
9692 034722 052737 000006 002150 BIS #BIT1!BIT2,CSR ;CHECK MODE
9693 034730 104425 LOADCSR
9694 034732 000002 RTI

```

```

9697          .SBTTL TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER
9698 034734 $WASSBE: PUSH R1,R4
          MOV R1,-(SP)
          MOV R4,-(SP)
9699 034734 010146
9700 034736 010446
9701 034740 013701 002224 MOV TOTCSRS,R1 ;GET CSR'S BYTE
9702 034744 005004 CLR R4
9703 034746 BEGIN LWSBE
9704 034746 FOR CSRNO := #0 TO #36 BY #2
9705 034746 005037 002152
9706 034752 006301 ASL R1
9707 034754 103010 ON.ERROR
9708 034756 104426 READCSR
9709 034760 032737 000020 002150 IF #BIT4 SET.IN CSR
9710 034766 001403
9711 034770 SET R4
9712 034770 012704 177777 LEAVE LWSBE
9713 034774 000411 END ;OF IF #BIT4
9714 034776 END ;OF ON.ERROR
9715 034776 IF R1 EQ #0 THEN LEAVE LWSBE
9716 034776 005701
9717 035000 001407
9718 035002 END ;OF FOR CSRNO
9719 035002 062737 000002 002152
9720 035010 023727 002152 000036
9721 035016 003755
9722 035020 END LWSBE
9723 035020 ROR R4 ;SET C BIT FOR ERROR
9724 035022 006004 POP R4,R1
9725 035022 012604
9726 035024 012601
9727 035026 ON.ERROR
9728 035026 103004
9729 035030 052766 000001 000002
9730 035036 000403
9731 035036 000403
9732 035040 042766 000001 000002
9733 035046 035046
9734 035046 000002
9735 035050 104426
9736 035052 042766 000001 000002
9737 035060 032737 000020 002150
9738 035066 001403
9739 035070 052766 000001 000002
          RTI
          .SBTTL TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
          ;ON RETURN IF CARRY IS SET THERE WAS A SBE
          $WAS1SBE: READCSR
          BIC #BIT0,2(SP) ;CLR C BIT ON STACK
          BIT #BIT4,CSR
          BEQ 1$
          BIS #BIT0,2(SP) ;SET C BIT ON STACK

```

B57:::~::~

CLR CSRNO

B60:::~::~

BCC L345

BIT #BIT4,CSR

BEQ L346

MOV #-1,R4

BR E57

L346:::~::~

L345:::~::~

TST R1

BEQ E57

ADD #2,CSRNO

CMP CSRNO,#36

BLE B60

E60:::~::~

E57:::~::~

MOV (SP)+,R4

MOV (SP)+,R1

BCC L347

BR L350

L347:::~::~

L350:::~::~

K6

CVMJAAO MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15-Feb 85 16:10 Page 309 1
TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER

SEQ 0282

9729 035076 000002

14: RTI

L6

```

9732          .SBTTL TRAP WAS THERE A DBE ON ANY CSR TRAP HANDLER
9733 $WASDBE: PUSH R1,R4
          MOV R1, (SP)
          MOV R4, -(SP)
9734 035100 010146 02224 MOV TOTCSRS,R1 ;GET CSR'S BYTE
9735 035110 005004 CLR R4
9736 035112 BEGIN LWDBE
          B61:
9737 035112 FOR CSRNO := #0 TO #36 BY #2
          CLR CSRNO
          B62:
9738 035116 006301 ASL R1
9739 035120 103010 ON.ERROR
          BCC L351
9740 035122 104426 READCSR
9741 035124 032737 100000 002150 IF #BIT15 SET.IN CSR
          BIT #BIT15,CSR
          BEQ L352
9742 035134 012704 177777 SET R4
          MOV #-1,R4
9743 035140 000411 LEAVE LWDBE
          BR E61
9744 035142 END ;OF IF #BIT4
          L352:
9745 035142 END ;OF ON.ERROR
          L351:
9746 035142 IF R1 EQ #0 THEN LEAVE LWDBE
          TST R1
          BEQ E61
          005701
          001407
9747 035146 END ;OF FOR CSRNO
          ADD #2,CSRNO
          CMP CSRNO,#36
          BLE B62
          062737 000002 002152
          023727 002152 000036
          003755
          E62:
9748 035164 END LWDBE
          E61:
9749 035164 006004 ROR R4 ;SET C BIT FOR ERROR
9750 035166 012604 POP R4,R1
          MOV (SP)+,R4
          MOV (SP)+,R1
          035170 012601
          ON.ERROR
          BCC L353
9751 035172 103004
          BIS #BIT0,2(SP)
          052766 000001 000002
          ELSE
          BR L354
          000403
          L353:
9754 035204 042766 000001 000002 BIC #BIT0,2(SP)
9755 035212 END ;OF ON.ERROR
          L354:
9756 035212 000002 RTI
9757 .SBTTL TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
9758 ;ON RETURN IF CARRY IS SET THERE WAS A DBE
9759 $WAS1DBE: READCSR
          TST CSR ;DBE?
          BPL 3$ ;NO - SKIP
          BIS #BIT0,2(SP) ;SET C BIT ON STACK
          RTI
          000001 000002
          000002
9763 035232 000002
  
```

M6

CVMJAAO MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15-Feb-85 16:10 Page 311-1
TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER

SEQ 0284

9764 035234 042766 000001 000002 34: BIC #BIT0,2(SP) ;CLR C BIT ON STACK
9765 035242 000002 RTI

```

9768 .SBTTL TRAP CLEAR ALL ECC CSR'S TRAP HANDLER
9769 035244 $CLRCSR: CLEAR CSR CLR CSR
      035244 005037 002150
9770 035250 004737 035314 CALL CSROUT
9771 035254 000002 RTI
9772 .SBTTL TRAP CLEAR 1 SELECTED CSR TRAP HANDLER
9773 035256 $CLR1CSR: CLEAR CSR CLR CSR
      035256 005037 002150
9774 035262 104425 LOADCSR
9775 035264 000002 RTI
9776 .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
9777 ;CHECKBITS ALREADY IN LOC "CSR"
9778 035266 052737 000006 002150 $CHKDIS: BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
9779 035274 004737 035314 CALL CSROUT
9780 035300 000002 RTI
9781 .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
9782 ;CHECKBITS ALREADY IN LOC "CSR"
9783 035302 052737 000006 002150 $CHK1DIS: BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
9784 035310 104425 LOADCSR
9785 035312 000002 RTI

```

```

9788 035314      CSROUT: SUBTST <<SUBR WRITE IN ALL CSR'S>>
;*****
;*SUBTEST      SUBR   WRITE IN ALL CSR'S
;*****
9789 035314      PUSH    R1
035314 010146      MOV     R1, -(SP)
9790 035316 013701 002224      MOV     TOTCSRS, R1      ;GET CSR'S BYTE
9791 035322      BEGIN LCSROUT
035322
9792 035322      FOR CSRNO := #0 TO #36 BY #2
035322 005037 002152
035326
9793 035326 006301      ASL     R1
9794 035330      ON.ERROR
035330 103001
9795 035332 104425      LOADCSR
9796 035334      END ;OF ON.ERROR
035334
9797 035334      IF R1 EQ #0 THEN LEAVE LCSROUT
035334 005701
035336 001407
9798 035340      END ;OF FOR CSRNO
035340 062737 000002 002152
035346 023727 002152 000036
035354 003764
035356
9799 035356      END LCSROUT
035356
9800 035356      POP     R1
035356 012601
9801 035360 000207      RETURN
9802
9803 035362      $INVALID: SUBTST <<TRAP INVALIDATE BACKGROUND PATTERN>>
;*****
;*SUBTEST      TRAP   INVALIDATE BACKGROUND PATTERN
;*****
9804 035362      PUSH    R0, R1
035362 010046
035364 010146
9805 035366 013701 002102      MOV     BANK, R1
9806 035372 006301      ASL     R1
9807 035374 006301      ASL     R1
9808 035376 042761 020000 002666      BIC     #BIT13, CONFIG+2(R1)
9809 035404      POP     R1, R0
035404 012601
035406 012600
9810 035410 000002      MOV     (SP)+, R1
RTI      MOV     (SP)+, R0

```



```

9812 035412          $ERRGEN:          SUBTST<<TRAP    GENERATE AND TEST ERROR ADDRESS>>
;*****
;SUBTEST          TRAP    GENERATE AND TEST ERROR ADDRESS
;*****
          PUSH      R0,R1,R2,R3
          MOV R0,-(SP)
          MOV R1,-(SP)
          MOV R2,-(SP)
          MOV R3,-(SP)

9813 035412          010046
          035412      010146
          035414      010246
          035416      010346
          035420      010346
9814 035422      013703 002104          MOV      BANKINDEX,R3
9815 035426      005737 002456          TST      NOSUPER
9816 035432      001003          BNE      6$
9817 035434      013700 172246          MOV      SIPAR3,R0          ;GENERATE WHAT ERROR ADDR SHOULD BE
9818 035440      000402          BR      7$
9819 035442      013700 177646          6$: MOV      UIPAR3,R0
9820 035446      072027 177773          7$: ASH      #-5,R0
9821 035452      005737 002132          TST      EQFLAG
9822 035456      001002          BNE      1$
9823 035460      042700 177600          BIC      #C177,R0
9824 035464      000301          1$: SWAB     R1          ;GET CURRENT ADDRESS BITS 11 AND 12
9825 035466      006201          ASR      R1
9826 035470      006201          ASR      R1
9827 035472      006201          ASR      R1
9828 035474      042701 177775          BIC      #C2,R1
9829 035500      060100          ADD      R1,R0          ;ADD THEM TO THE ADJUSTED PAR VALUE
9830          ;GET ERROR ADDRESS FROM CSR UNDER TEST
9831 035502      013701 002150          MOV      CSR,R1
9832 035506      072127 177773          ASH      #-5,R1
9833 035512      042701 177600          BIC      #C177,R1
9834 035516      005737 002454          TST      N022BIT          ;IS THIS AN 11/83,11/23-B OR 11/23 ?
9835 035522      001024          BNE      2$          ;BRANCH IF NOT NECESSARY
9836 035524      005737 002132          TST      EQFLAG          ;IS IT EQB?
9837 035530      001421          BEQ      2$          ;BRANCH IF NOT
9838 035532          010046          PUSH     R0          ;SAVE GENERATED ERROR ADDRESS
          035532      010046          MOV R0,-(SP)
9839 035534      013702 002152          MOV      CSRNO,R2          ;GET CSR NUMBER
9840 035540      052762 040000 172100      BIS      #BIT14,CSRADD(R2)          ;TURN ON EQB BIT CAREFULLY
9841 035546      016200 172100          MOV      CSRADD(R2),R0          ;GET CSR CONTENTS
9842 035552      042762 040000 172100      BIC      #BIT14,CSRADD(R2)          ;TURN OFF EQB BIT CAREFULLY
9843 035560      042700 177037          BIC      #C740,R0          ;CLEAR EVERYTHING BUT ERROR ADDR
9844 035564      006300          ASL      R0
9845 035566      006300          ASL      R0          ;SHIFT ADDR BITS 18-21 INTO POSITION
9846 035570      060001          ADD      R0,R1          ;ADD TO CURRENT ERROR ADDRESS
9847 035572          012600          POP      R0
          035572      012600          MOV (SP)+,R0
9848 035574      020001          2$: CMP      R0,R1          ;COMPARE REAL AND GENERATED ERR. ADDR.
9849 035576      001420          BEQ      5$          ;BRANCH IF THEY ARE THE SAME
9850 035600      005737 002136          TST      INTFLAG          ;INTERLEAVED?
9851 035604      001411          BEQ      3$          ;NO - WE HAVE AN ERROR
9852 035606      062700 000100          ADD      #100,R0
9853 035612      005737 002140          TST      INT64K          ;64K INTERLEAVED MEMORY?
9854 035616      001002          BNE      4$
9855 035620      062700 000100          ADD      #100,R0
9856 035624      020001          4$: CMP      R0,R1
9857 035626      001404          BEQ      5$
9858 035630      005737 002066          3$: TST      SKPERR          ;ARE WE SUPPOSED TO SKIP ERROR P.O.?
9859 035634      001001          BNE      5$          ;YES - SKIP ERROR PRINTOUT

```

```

9860 035636 104462          PERR36          ;ELSE PRINT ERROR ADDRESS ERROR
9861 035640 010137 002460 5$: MOV R1,ERRADD ;SAVE CSR'S ERROR ADDRESS
9862 035644 005037 002066    CLR SKPERR    ;ENABLE THE ERROR PRINTOUT AGAIN
9863 035650          POP R3,R2,R1,R0      ;RESTORE REGISTERS
          035650 012603          MOV (SP),R3
          035652 012602          MOV (SP),R2
          035654 012601          MOV (SP),R1
          035656 012600          MOV (SP),R0
9864 035660 000002          RTI
9865
9866 035662          $CBREG: SUBTST <<TRAP      ENABLE CHECKBIT REGISTER>>
          ;*****
          ;*SUBTEST      TRAP      ENABLE CHECKBIT REGISTER
          ;*****
9867 035662 005037 002150    CLR CSR          ;
9868 035666 052737 000004 002150 BIS #BIT2,CSR ;ENABLE DIAGNOSTIC MODE
9869 035674 104425          LOADCSR          ;LOAD CSR REGISTER
9870 035676 000002          RTI              ;
9871
9872 035700          $SYNREG: SUBTST <<TRAP      ENABLE SYNDROME BIT REGISTER>>
          ;*****
          ;*SUBTEST      TRAP      ENABLE SYNDROME BIT REGISTER
          ;*****
9873 035700 005037 002150    CLR CSR          ;
9874 035704 052737 040004 002150 BIS #BIT14:BIT2,CSR ;ENABLE DIAGNOSTIC MODE
9875 035712 104425          LOADCSR          ;LOAD CSR REGISTER
9876 035714 000002          RTI              ;
9877

```

9880 035716

SUBTST<<SUBR MAPPER>>

;*****
;SUBTEST SUBR MAPPER

;*****

;THIS SUBROUTINE MAPS THE MEMORY BANK (16K WORDS = 1 BANK)
;IN R3 TO THE TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000 - 157777) FOR
;THE 11/83 ; USER VIRTUAL (60000 - 157777) FOR ALL OTHERS.

;

;CALL MOV BANKNO,R3 ;SET UP BANK ARGUMENT
;CALL MAPPER ;ACTUAL CALL
;RETURN ;ONLY RETURN

;SET SUPERVISOR/USER UP FOR 1 TO 1 MAP

MAPPER: PUSH RO,R1,R2,R4,R5

MOV RO,-(SP)

MOV R1,-(SP)

MOV R2,-(SP)

MOV R4,-(SP)

MOV R5,-(SP)

9881 035716 010046
 9882 035716 010146
 9883 035722 010246
 9884 035724 010446
 9885 035726 010546
 9892 035730 012700 172340
 9893 035734 012701 172240
 9894 035740 012704 172200
 9895 035744 005737 002456
 9896 035750 001404
 9897 035752 012701 177640
 9898 035756 012704 177600
 9899 035762 012702 077406
 9900 035766 012705 000010
 9901 035772 012021
 9902 035774 010224
 9903 035776 077503
 9904 036000 012741 177600
 9905 036004 022737 000005 004010 30\$:
 9906 036012 001007
 9907 036014 012704 172206
 9908 036020 012705 000004
 9909 036024 052724 100000 35\$:
 9910 036030 077503
 9911
 9912 036032 022703 000200 40\$:
 9913 036036 001516
 9914 036040 072327 000011
 9915
 9916 036044 012701 172246
 9917 036050 005737 002456
 9918 036054 001402
 9919 036056 012701 177646
 9920 036062 012702 000004 5\$:
 9921 036066 010321 2\$:
 9922 036070 062703 000200
 9923 036074 077204
 9924 036076 005737 002240
 9925 036102 001442
 9926 036104 162701 000010
 9927 036110 010102
 9928 036112 062702 000004

MOV #KIPAR0,R0

MOV #SIPAR0,R1

MOV #SIPDR0,R4

TST NOSUPER

BEQ 4\$

MOV #UIPAR0,R1

MOV #UIPDR0,R4

4\$: MOV #77406,R2

MOV #8.,R5

1\$: MOV (R0)+,(R1)+

MOV R2,(R4)+

SOB R5,1\$

MOV #177600,(R1)

30\$: CMP #5,PROTYP

BNE 40\$

MOV #SIPDR0+6,R4

MOV #4.,R5

35\$: BIS #BIT15,(R4)+

SOB R5,35\$

;SET UP SUPERVISOR/USER FOR TEST AREA

40\$: CMP #200,R3

BEQ 3\$

ASH #9.,R3

MOV #SIPAR3,R1

TST NOSUPER

BEQ 5\$

MOV #UIPAR3,R1

5\$: MOV #4,R2

2\$: MOV R3,(R1)+

ADD #200,R3

SOB R2,2\$

TST SPLTCSR

BEQ 9\$

SUB #10,R1

MOV R1,R2

ADD #4,R2

;FIRST AREA TO MAP TO

;FIRST ADDRESS REGISTER

;FIRST DESCRIPTOR REGISTER

;CAN WE USE SUPERVISOR MODE?

;YES, BRANCH

;FIRST ADDRESS REGISTER

;FIRST DESCRIPTOR REGISTER

;CONSTANT FOR 4K PAGE, UP, R/W

;COUNTER

;PUT IN SUPERVISOR ADDRESS

;PUT IN SUPERVISOR DESCRIPTOR

;LOOP TILL DONE

;CORRECT LAST FIELD FOR PERIPHERALS PAGE

;IS THIS A 11/73?

;BRANCH IF NOT

;POINT TO PDR 3

;COUNTER=4

;SET UNCONDITIONAL CACHE BYPASS IN PDR3-6

;MAP NOTHING (1 TO 1)?

;YES - SKIP

;BANK 1 STARTS AT 100,000 LESS 6 LSB'S

;FOR MEMORY MANAGEMENT = 1000

;SETUP FOR AUTO INCREMENTING

;DO WE HAVE SUPERVISOR MODE?

;YES - BRANCH

;SETUP FOR AUTO INCREMENTING

;COUNTER

;PLUG IN PAR INFO

;BUMP ADDRESS 4K

;LOOP TILL DONE

Address	Offset	Hex	Dec	Label	Operation	Comments
9929	036116	022737	000001	002240	CMP	#1,SPLTCSR
9930	036124	001403			BEQ	10\$
9931	036126	010200			MOV	R2,R0
9932	036130	010102			MOV	R1,R2
9933	036132	010001			MOV	R0,R1
9934	036134	012122		10\$:	MOV	(R1)+,(R2)+
9935	036136	011112			MOV	(R1),(R2)
9936	036140	013700	002104		MOV	BANKINDEX,R0
9937	036144	005737	002140		TST	INT64K
9938	036150	001403			BEQ	11\$
9939	036152	012700	004000		MOV	#4000,R0
9940	036156	000402			BR	12\$
9941	036160	012700	010000	11\$:	MOV	#10000,R0
9942	036164	005737	002456	12\$:	TST	NOSUPER
9943	036170	001403			BEQ	13\$
9944	036172	012701	177652		MOV	#UIPAR5,R1
9945	036176	000402			BR	14\$
9946	036200	012701	172252	13\$:	MOV	#SIPAR5,R1
9947	036204	060021		14\$:	ADD	R0,(R1)+
9948	036206	060011			ADD	R0,(R1)
9949						;IF WE ONLY HAVE AN 124K SYSTEM, WE DON'T WANT TO TEST THE
9950						;LAST 4K, WHERE THE Q BUS DEVICE PAGE IS. INSTEAD, THE
9951						;PROGRAM WILL REMAP THE LAST 4K TO 8-12K. ALSO, IF THERE
9952						;IS A BANK 177 ON AN 11/83, THE PROGRAM WILL REMAP THE LAST
9953						;4K TO 8-12K FOR THE SAME REASON.
9954	036210	022737	000007	002556	9\$:	CMP
9955	036216	001010				#7, LASTBANK
9956	036220	005737	002454			7\$
9957	036224	001423				TST
9958	036226	022737	000007	002102		N022BIT
9959	036234	001017				3\$
9960	036236	000404				BEQ
9961	036240	022737	000177	002102	7\$:	#7,BANK
9962	036246	001012				3\$
9963	036250	005737	002456		8\$:	TST
9964	036254	001404				NOSUPER
9965	036256	013737	177652	177654		6\$
9966	036264	000403				BEQ
9967	036266	013737	172252	172254	6\$:	MOV
9968	036274				3\$:	BR
	036274	012605				3\$
	036276	012604				MOV (SP)+,R5
	036300	012602				MOV (SP)+,R4
	036302	012601				MOV (SP)+,R2
	036304	012600				MOV (SP)+,R1
9969	036306	000207				MOV (SP)+,R0
9970						RETURN
9971	036310				\$KMAP:	.SBTTL
	036310	010046				TRAP
	036312	010146				MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
	036314	010246				PUSH
	036316	010346				R0,R1,R2,R3,R4
	036320	010446				
9972	036322	005000				MOV R0,-(SP)
9973	036324	012701	172340			MOV R1,-(SP)
9974	036330	012702	077406			MOV R2,-(SP)
9975	036334	012703	172300			MOV R3,-(SP)
						MOV R4,-(SP)
						CLR

9976	036340	012704	000010		MOV	#8.,R4	;COUNTER
9977	036344	010021		1\$:	MOV	R0,(R1)+	;PUT IN KERNEL ADDRESS
9978	036346	010223			MOV	R2,(R3)+	;PUT IN KERNEL DISCRIPTOR
9979	036350	062700	000200		ADD	#200,R0	;ADD ADDRESS CONSTANT FOR 4K CHANGE
9980	036354	077405			SOB	R4,1\$;LOOP TILL DONE
9981	036356	012741	177600		MOV	#177600,-(R1)	;THE PERIPHERALS PAGE TO KIPAR7
9982	036362	012741	177400		MOV	#177400,(R1)	;AND NEXT LOWER PAGE TO KIPAR6
9989	036366				POP	R4,R3,R2,R1,R0	
	036366	012604					MOV (SP)+,R4
	036370	012603					MOV (SP)+,R3
	036372	012602					MOV (SP)+,R2
	036374	012601					MOV (SP)+,R1
	036376	012600					MOV (SP)+,R0
9990	036400	000002			RTI		

```

9993 036402      RELOCATE:SUBTST <<RELOCATE PROGRAM>>
;*****
;SUBTEST      RELOCATE PROGRAM
;*****
          IF #SW12 SET.IN @SWR THEN $RETURN ERROR

9994 036402      032777 010000 144226      BIT #SW12,@SWR
          036410      001402      BEQ L356
          036412      000261      SEC
          036414      000207      RTS PC
          036416      L356:::;

9995 036416      IF APTFLAG IS TRUE OR ACTFLAG IS TRUE      TST APTFLAG
          036416      005737 002352      BNE L357
          036422      001003      TST ACTFLAG
          036424      005737 002350      BEQ L360
          036430      001405      L357:::;
          036432      IF $PASS NE #0 THEN $RETURN ERROR      TST $PASS
          036432      005737 056654      BEQ L361
          036436      001402      SEC
          036440      000261      RTS PC
          036442      000207      L361:::;
          036444      END; OF IF APTFLAG      L360:::;
          036444      BEGIN LOADERBANK      B65:::;
          036444      FOR BANK := #1 TO LASTBANK      MOV #1,BANK
          036444      012737 000001 002102      B66:::;
          036452      004737 040052      CALL EXBANK
10000 036452      IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE      TST ACFLAG
10001 036456      005737 002116      BEQ L362
          036462      001431      TST PFLAG
          036464      005737 002122      BNE L362
          036470      001026      TST BMFLAG
          036472      005737 002130      BEQ L362
          036476      001023      BNE L362

10002 036500      013700      MOV BANK,RO
10003 036504      010037      MOV RO,LOADBANK
10004 036510      013701      MOV LOADHOME,R1
10005 036514      004737      CALL BANKMOV
10006 036520      004737      CALL NEWLOAD      ;MAP NEW LOADER BANK IN KERNEL
10007 036524      013701      MOV BANKINDEX,R1
10008 036530      052761 100000 002666      BIS #BIT15,CONFIG+2(R1)      ;MARK LOADER
10009 036536      042761 020000 002666      BIC #BIT13,CONFIG+2(R1)      ;INVALIDATE BACKGROUND PATTERN
10010 036544      LEAVE LOADERBANK
          036544      000416      BR E65
10011 036546      END ;OF IF ACFLAG
          036546      END ;OF FOR BANK      L362:::;
10012 036546      005237 002102      INC BANK
          036552      023737 002102 002556      CMP BANK,LASTBANK
          036560      003734      BLE B66
          036562      E66:::;
10013 036562      IF #SW13 OFF.IN @SWR      BIT #SW13,@SWR
          036562      032777 020000 144046      BNE L363
          036570      001002

```

				TYPE	MSG075		
10014	036572			TYPEIT	MSG075		
	036572	104401	070176	.DSABL	CRF		
				END ;OF IF	#SW13		
10015	036576						
	036576						
10016	036576			\$RETURN	ERROR		
	036576	000261					
	036600	000207					
10017	036602			END	LOADERBANK		
	036602						
10018	036602			BEGIN	FINDBANK		
	036602						
10019	036602	013702	002556	MOV	LASTBANK,R2		
10020	036606	006302		ASL	R2		
10021	036610	006302		ASL	R2		
10022	036612			FOR R1 :=	#2*2 TO R2 BY #4		
	036612	012701	000004				
	036616						
10023	036616			IF #BIT7!BIT0	OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT		
	036616	032761	000201				
	036624	001035	002664				
10024	036626			IF #BIT15	OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK		
	036626	032761	100000				
	036634	001031	002666				
10025	036636			IF CPUBIT	SET.IN CONFIG(R1) ;IF ACCESSABLE		
	036636	033761	002106				
	036644	001425	002664				
10026	036646			IF #BIT8	OFF.IN CONFIG+2(R1) THEN LEAVE FINDBANK ;IF PARITY		
	036646	032761	000400				
	036654	001460	002666				
10027	036656			IF #BIT6	SET.IN CONFIG(R1) AND #BIT7 OFF.IN CONFIG(R1)		
	036656	032761	000100				
	036664	001405	002664				
	036666	032761	000200				
	036674	001001					
10028							
10029	036676						
	036676	000447					
10030	036700			END ;OF IF	#BIT6		
	036700						
10031	036700			IF INHECC	IS FALSE		
	036700	005737	002536				
	036704	001005					
10032	036706			SET	INHECC		
	036706	012737	177777				
10033	036714	010137	002540	MOV	R1,INHBANK		
10034	036720			END ;OF IF	INHECC		
	036720						
10035	036720			END ;OF IF	CPUBIT		
	036720						
10036	036720			END ;OF IF	#BIT15		
	036720						
10037	036720			END ;OF IF	#BIT7		
	036720						
10038	036720			END ;OF	FOR		
	036720	062701	000004				
	036724	020102					

Address	Offset	Label	Operation	Comment
10039	036726	003733		
	036730			
	036730		IF FULLREL IS FALSE	
	036730	005737 002542		
	036734	001012		
10040	036736		IF INHECC IS TRUE	
	036736	005737 002536		
	036742	001407		
10041	036744	013701 002540	MOV INHBANK,R1	
10042	036750	023727 002300 000030	CMP REALPAT,#30	;IS THIS PATTERN 30?
10043	036756	001421	BEQ RELENT1	;YES - SKIP MESSAGE
10044	036760	000420	BR RELENT1	
10045	036762		END; OF IF INHECC	
	036762			
10046	036762		END; OF IF FULLREL	
	036762			
10047	036762	005037 002536	CLR INHECC	
10048	036766		IF #SW13 OFF.IN @SWR	
	036766	032777 020000 143642		
	036774	001006		
10049	036776	023727 002300 000030	CMP REALPAT,#30	;IS THIS PATTERN 30?
10050	037004	001402	BEQ SKUB	;YES - SKIP MESSAGE
10051	037006		TYPE MSG075	;RELOCATION NOT POSSIBLE
	037006	104401 070176	TYPEIT ,MSG075	
			.DSABL CRF	
10052	037012		END ;OF IF #SW13	
	037012			
10053	037012		SKUB: \$RETURN ERROR	
	037012	000261		
	037014	000207		
10054	037016		END FINDBANK	
	037016			
10055	037016		CLEAR INHECC	
	037016	005037 002536		
10056	037022	042761 020000 002666	RELENT1: BIC #BIT13,CONFIG+2(R1)	
10057	037030	005000	CLR R0	
10058	037032	071027 000004	DIV #4,R0	
10059	037036		RELOC1: LET NEWBANK := R0	
	037036	010037 002310		
10060	037042	013737 002532 002534	MOV PGMCSR,PGMCSR+2	
10061	037050	004737 037610	CALL USERMAP	
10062	037054		USER	
	037054	052737 140000 177776	BIS #BIT15!BIT14,PSW	
			.DSABL CRF	
10063	037062		BMOV 0,100000,SIZE	
	037062	004537 041024	JSR R5,BLOCK3	
	037066	040000	SIZE	
	037070	100000	100000	
	037072	000000	0	
			.DSABL CRF	
10064	037074	104417	KERNEL	
10065	037076	042737 000001 177572	JMPRL1: BIC #BIT0,MMR0	
10066	037104	004737 037672	CALL NEWKERNEL	
10067	037110	013700 002310	MOV NEWBANK,R0	
10068	037114	006300	ASL R0	
10069	037116	006300	ASL R0	
10070	037120	016002 002664	MOV CONF1 R0),R2	

10071	037124	000302			SWAB	R2	
10072	037126	042702	177760		BIC	#C17,R2	
10073	037132	006302			ASL	R2	
10074	037134	052737	000001	177572	BIS	#BIT0,MMR0	;ENERGIZE MEMORY MANAGEMENT
10075	037142	010237	002532		MOV	R2,PGMCSR	;PUT NEW PGM. CSR INTO PGMCSR
10076	037146	032760	010000	002666	BIT	#BIT12,CONFIG+2(R0)	;IS THE NEW BANK INTERLEAVED?
10077	037154	001412			BEQ	1\$;BRANCH IF NOT INTERLEAVED
10078	037156	016002	002664		MOV	CONFIG(R0),R2	
10079	037162	042702	007777		BIC	#C170000,R2	
10080	037166	072227	177775		ASH	#-3,R2	
10081	037172	052702	100000		BIS	#BIT15,R2	
10082	037176	050237	002532		BIS	R2,PGMCSR	
10083	037202				SET	RLFLAG	
	037202	012737	177777	002126			MOV #-1,RLFLAG
10084	037210				\$RETURN	NOERROR	
	037210	000241					CLC
	037212	000207					RTS PC

```
10087 037214
10088
10089 037214 010046
10090 037216 013701 002432
10091 037222 013700 002576
10092 037226 004737 037442
10093 037232 004737 037774
10094 037236
10095 037242 013746 002102
10096 037250 013737 002432 002102
10097 037254 004737 040052
10098 037260 013701 002104
10099 037266 042761 100000 002666
10100 037274 013737 002576 002102
10101 037280 004737 040052
10102 037300 013701 002104
10103 037312 042761 020000 002666
10104 037316 012637 002102
10105 037316 005037 002536
10106
10107 037322 042737 020000 002666
10108 037330
10109 037334 005037 002310
10110 037340 004737 037610
10111 037346 052737 140000 177776
10112 037346 004537 041024
10113 037352 040000
10114 037354 100000
10115 037356 000000
10116 037360 104417
10117 037362 042737 000001 177572
10118 037370 004737 037672
10119 037374 013737 002534 002532
10120 037402 052737 000001 177572
10121 037410 005037 002126
10122 037414 012700 002666
10123 037420 042710 020000
10124 037424 062700 000004
10125 037430 020027 003620
10126 037434 003771
10127 037436
10128 037436 012600
10129 037440 000207
10130

UNRELOCATE:SUBTST <<UNRELOCATE PROGRAM>>
;*****
;SUBTEST UNRELOCATE PROGRAM
;*****
;RESTORE LOADERS
PUSH RO
MOV RO, (SP)
MOV LOADBANK,R1
MOV LOADHOME,R0
CALL BANKMOV
CALL NEWLOAD ;MAP NEW LOADER BANK IN KERNEL SPACE
PUSH BANK
MOV BANK,-(SP)
MOV LOADBANK,BANK
CALL EXBANK
MOV BANKINDEX,R1
BIC #BIT15,CONFIG+2(R1) ;CLEAR LOADER FLAG
MOV LOADHOME,BANK
CALL EXBANK
MOV BANKINDEX,R1
BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
POP BANK
MOV (SP)+,BANK
CLEAR INHECC ;MAKE SURE ECC TESTS ARE NOT INHIBITED!
CLR INHECC
;RESTORE BANK 0
BIC #BIT13,CONFIG+2
LET NEWBANK := #0 ;INVALIDATE BACKGROUND PATTERN
CALL USERMAP ;MAP NEWBANK TO USER PAR
USER ;ENTER USER MODE
BIS #BIT15:BIT14,PSW ;GO TO USER MODE
.DSABL CRF
BMOV 0,100000,SIZE ;MOVE PROGRAM
JSR R5,BLOCK3
SIZE
100000
0
.DSABL CRF
KERNEL ;ENTER KERNEL MODE
BIC #BIT0,MMRO ;DEENERGIZE MEMORY MANAGEMENT
CALL NEWKERNEL
MOV PGMCSR+2,PGMCSR ;RESTORE PREVIOUS PGM. CSR
BIS #BIT0,MMRO ;ENERGIZE MEMORY MANAGEMENT
CLR RLFLAG
MOV #CONFIG+2,R0 ;MOVE 2ND WORD OF CONFIG TO R0
BIC #BIT13,(R0) ;CLEAR BACKGROUND VALID BIT
ADD #4,R0 ;INCREMENT TO NEXT BANK
CMP R0,#3620 ;DONE?
BLE 2$ ;NO - BRANCH
POP R0
MOV (SP)+,R0
RETURN
```

10128 037442

BANKMOV:SUBTST <<MOVE BANKS>>

;*****
;SUBTEST MOVE BANKS
;*****

10129
10130
10131
10132
10133 037442 104415
10134 037444 004737 037610
10135 037450 104416
10136 037452 104415
10137 037454 072027 000011
10138 037460 072127 000011
10139 037464 012702 177650
10140 037470 012703 000200
10141
10142 037474 010122
10143 037476 060301
10144 037500 010122
10145 037502 060301
10146
10147 037504 010022
10148 037506 060300
10149 037510 010022
10150 037512 060300
10151
10152 037514
037514 052737 140000 177776

10153 037522
037522 004537 041024
037526 020000
037530 140000
037532 100000

10154 037534 104417
10155
10156 037536 012702 177650
10157
10158 037542 010122
10159 037544 060301
10160 037546 010122
10161 037550 060301
10162
10163 037552 010022
10164 037554 060300
10165 037556 010022
10166 037560 060300
10167
10168 037562
037562 052737 140000 177776

10169 037570
037570 004537 041024
037574 010000
037576 140000

;MOVE 3/4 OF A BANK
;CALLING SEQUENCE
;R0 = DESTINATION BANK
;R1 = SOURCE BANK
SAVREG
CALL USERMAP
RESREG
SAVREG
ASH #9.,R0
ASH #9.,R1
MOV #UIPAR4,R2
MOV #200,R3

MOV R1,(R2)+ ;MAP 1ST HALF BANK
ADD R3,R1 ;BUMP BY 4K
MOV R1,(R2)+
ADD R3,R1

MOV R0,(R2)+
ADD R3,R0
MOV R0,(R2)+
ADD R3,R0

USER
BIS #BIT15!BIT14,PSW ;GO TO USER MODE
.DSABL CRF
BMOV 100000,140000,SIZE/2 ;MOV 1ST HALF BANK
JSR R5,BLOCK3
SIZE/2
140000
100000
.DSABL CRF
KERNEL ;ENTER KERNEL MODE

MOV #UIPAR4,R2

MOV R1,(R2)+ ;MAP 2ND HALF BANK
ADD R3,R1 ;BUMP BY 4K
MOV R1,(R2)+
ADD R3,R1

MOV R0,(R2)+
ADD R3,R0
MOV R0,(R2)+
ADD R3,R0

USER
BIS #BIT15!BIT14,PSW ;GO TO USER MODE
.DSABL CRF
BMOV 100000,140000,SIZE/4 ;MOV 3RD FOURTH OF BANK
JSR R5,BLOCK3
SIZE/4
140000

CVMJAAO MSV11-J MEMORY DIAG.
MOVE BANKS

MACRO V05.01b Friday 15-Feb-85 16:10 Page 324-1

SEQ 0298

037600	100000	100000		
		.DSABL	CRF	
10170	037602	104417	KERNEL	;ENTER KERNEL MODE
10171				
10172	037604	104416	RESREG	
10173	037606	000207	RETURN	

10176 037610

USERMAP:SUBTST <<SUBR MAP USER TO NEW BANK>>

;*****

;SUBTEST SUBR MAP USER TO NEW BANK

;*****

10177 037610 012701 177640

MOV #UIPAR0,R1 ;COPY KERNEL PAR'S & PDR'S (0-3)

10178 037614 012702 172340

MOV #KIPAR0,R2

10179 037620 012703 177600

MOV #UIPDR0,R3

10180 037624 012704 172300

MOV #KIPDR0,R4

10181 037630 012705 000004

MOV #4,R5

10182 037634 012221

1\$: MOV (R2)+,(R1)+

10183 037636 011423

MOV (R4),(R3)+

10184 037640 077503

SOB R5,1\$

10185

10186 037642 013700 002310

MOV NEWBANK,R0

10187 037646 072027 000011

ASH #9.,R0

;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000

10188

10189 037652 012705 000004

MOV #4,R5

10190 037656 010021

2\$: MOV R0,(R1)+

;SETUP UIPAR(4-7)

10191 037660 062700 000200

ADD #200,R0

;BUMP ADDRESS 4K

10192 037664 011423

MOV (R4),(R3)+

;SETUP UIPDR(4-7)

10193 037666 077505

SOB R5,2\$

10194 037670 000207

RETURN

10195

10196 037672

NEWKERNEL:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW BANK>>

;*****

;SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW BANK

;*****

10197 037672

PUSH R0,R1,R5

MOV R0,-(SP)

MOV R1,-(SP)

MOV R5,-(SP)

037672 010046

037674 010146

037676 010546

10198 037700 012700 172340

MOV #KIPAR0,R0

10199 037704 013701 002310

MOV NEWBANK,R1

10200 037710 072127 000011

ASH #9.,R1

;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000

10201

10202 037714 012705 000004

MOV #4,R5

10203 037720 010120

1\$: MOV R1,(R0)+

;SETUP KIPAR(0-3)

10204 037722 062701 000200

ADD #200,R1

10205 037726 077504

SOB R5,1\$

10206 037730

POP R5,R1,R0

MOV (SP)+,R5

MOV (SP)+,R1

MOV (SP)+,R0

037730 012605

037732 012601

037734 012600

10207 037736 000207

RETURN

10208

10209 037740

MAPKERNAL:SUBTST <<SUBR MAP KERNAL PARS 4 AND 5 TO A BANK>>

;*****

;SUBTEST SUBR MAP KERNAL PARS 4 AND 5 TO A BANK

;*****

10210

10211 037740 013705 002102

MOV BANK,R5

;MOV BANK NUMBER TO R5

10212 037744 072527 000011

ASH #9.,R5

;R5 ENTERS 100000 LESS SHIFT TO CREATE MAPPING

10213 037750 013737 172350

MOV KIPAR4,SAVPAR

;SAVE OLD PAR

10214 037756 010537 172350

MOV R5,KIPAR4

;GET NEW PAR'S

10215 037762 062705 000200

ADD #200,R5

10216 037766 010537 172352

MOV R5,KIPAR5

10217 037772 000207

RETURN

002272

10218
10219 037774NEWLOAD: SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
;*****
;*SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK
;*****

10220

10221 037774

037774 010046

037776 010146

10222 040000 012701 172350

10223 040004 072027 000011

10224 040010 010021

10225 040012 062700 000200

10226 040016 010021

10227 040020

040020 012601

040022 012600

10228 040024 000207

10229

10230 040026

;RO CONTAINS THE DESTINATION BANK

PUSH RO,R1

MOV RO,-(SP)

MOV R1,-(SP)

MOV #KIPAR4,R1

ASH #9,R0

;BANK 1 STARTS AT 100000 LESS 6 LSB'S (1000)

MOV RO,(R1)+

;SETUP KIPAR4

ADD #200,R0

MOV RO,(R1)+

;SETUP KIPAR5

POP R1,R0

MOV (SP)+,R1

MOV (SP)+,R0

RETURN

UNMAP: SUBTST <<SUBR UNMAP KERNAL PAR'S 4 AND 5>>

;*****

;*SUBTEST SUBR UNMAP KERNAL PAR'S 4 AND 5

;*****

10231 040026 013737 002272 172350

10232 040034 062737 000200 002272

10233 040042 013737 002272 172352

10234 040050 000207

MOV SAVPAR,KIPAR4

;RESTORE KIPAR4

ADD #200,SAVPAR

;ADD 200 FOR NEXT PAR

MOV SAVPAR,KIPAR5

;RESTORE KIPAR5

RETURN

;

10237 040052

EXBANK: SUBTST <<SUBR EXAMINE BANK>>

;*****
;SUBTEST SUBR EXAMINE BANK
;*****

;DOES THE FOLLOWING:

;(1) SETS UP "BANKINDEX" AND R1 BASED ON VALUE OF "BANK".
 ;(2) SETS THE "MKFLAG" IF THE BANK IS ECC.
 ;(3) SETS THE "KPFLAG" IF THE BANK IS THE PROTECTED REGION OF ECC MEMORY.
 ;(4) SETS THE "ACFLAG" IF THE BANK CAN BE ACCESSED BY THIS CPU.
 ;(5) SETS THE "PFLAG" IF THE BANK IS IN PROGRAM SPACE.
 ;(6) SETS THE "RRFLAG" IF RELOCATION IS REQUIRED TO TEST THIS BANK; HOWEVER,
 ; IT COMPLEMENTS THIS FLAG IF THE RELOCATION FLAG "RLFLAG" IS SET (THIS IS
 ; NECESSARY FOR THE USE OF THE RECURSIVE "MODE" SUBROUTINES). THE "RRFLAG"
 ; IS ALWAYS SET TO DISABLE TESTING IF FIELD SERVICE MODE "SELECTED BANKS"
 ; ARE BEING TESTED AND THIS BANK IS NOT SELECTED.
 ;(7) SETS THE "BMFLAG" IF THE BANK IS A BAD MEMORY; HOWEVER, IT COMPLEMENTS
 ; THIS FLAG IF THE "WORST" FLAG IS NOT SET (THIS IS NECESSARY FOR THE USE
 ; OF THE RECURSIVE "MODE" SUBROUTINES).
 ;(8) SETS THE "INTFLAG" IF THE BANK IS INTERLEAVED.
 ;(9) SETS THE "INT64K" FLAG IF THE BANK IS INTERLEAVED ON 64K WORD BOUNDS.
 ;(10) SETS THE "SKIPMK" FLAG IF THIS BANK IS INTERLEAVED, AND HAS ALREADY
 ; BEEN TESTED.

; PUSH R0,R1,R2

MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)

; CLEAR MKFLAG,KPFLAG

CLR MKFLAG
CLR KPFLAG

; SET ACFLAG

MOV #-1,ACFLAG

; CLEAR PFLAG,RRFLAG,BMFLAG

CLR PFLAG
CLR RRFLAG
CLR BMFLAG

; CLEAR INTFLAG,INT64K,SKIPMK

CLR INTFLAG
CLR INT64K
CLR SKIPMK

; MOV BANK,R1

; ASL R1

; ASL R1

;R1 <- R1 * 4

; MOV R1,BANKINDEX

; BIT #BIT6,CONFIG(R1)

;PROTECTED REGION OF ECC MEMORY?

; BEQ 1\$

;NO - SKIP

; SET KPFLAG

MOV #-1,KPFLAG

; MOV #BIT1,R0

; IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)

BIT R0,CPUBIT
BEQ L374
BIT R0,CONFIG(R1)
BNE L374

; CLR ACFLAG

; END ;OF IF R0

L374::::::

10238

10239

10240

10241

10242

10243

10244

10245

10246

10247

10248

10249

10250

10251

10252

10253

10254

10255

10256

10257

040052

040052

040054

040056

010046

010146

010246

10258

040060

040060

040064

005037

005037

002120

002114

10259

040070

040070

012737

177777

002116

10260

040076

040076

040102

040106

005037

005037

002122

002124

002130

10261

040112

040112

040116

040122

005037

005037

002136

002140

002342

10262

040126

040132

040134

040136

040142

040150

040152

013701

006301

006301

010137

002104

032761

000100

002664

10268

040152

040160

040164

040170

040172

040176

012737

177777

002114

012700

000002

1\$.

030037

002106

001405

030061

002664

001002

10271

040200

040204

040204

005037

002116

```

10273 040204 005737 002116      TST      ACFLAG      ;ACTIVE MEMORY?
10274 040210 001415      BEQ      2$          ;BRANCH IF NOT
10275 040212 016102 002666      MOV      CONFIG+2(R1),R2
10276 040216 000302      SWAB     R2
10277 040220 042702 177770      BIC      #1C7,R2      ;ISOLATE MEM TYPE BITS
10278 040224 020227 000003      CMP      R2,#3       ;IS THIS AN ILLEGAL MEM TYPE?
10279 040230 003405      BLE      2$          ;BRANCH IF NOT
10280 040232      SET      BMFLAG      ;SET BAD BANK FLAG
      040232 012737 177777 002130      MOV      #-1,BMFLAG
10281 040240 000137 040446      JMP      ENEXBK      ;JUMP OVER REST OF FLAG TESTS
10282 040244 032761 000400 002666 2$: BIT      #BIT8,CONFIG+2(R1) ;IS THERE ECC THERE?
10283 040252 001403      BEQ      3$          ;NO - SKIP
10284 040254      SET      MKFLAG      ;YES - SET MKFLAG
      040254 012737 177777 002120      MOV      #-1,MKFLAG
10285 040262 032761 000200 002664 3$: BIT      #BIT7,CONFIG(R1) ;BANK = PROGRAM SPACE?
10286 040270 001406      BEQ      5$          ;NO - SKIP
10287 040272      SET      PFLAG,RRFLAG
      040272 012737 177777 002122      MOV      #-1,PFLAG
      040300 012737 177777 002124      MOV      #-1,RRFLAG
10288 040306 005737 002126      5$: TST      RLFLAG      ;IS PROGRAM RELOCATED?
10289 040312 001402      BEQ      6$          ;NO - SKIP
10290 040314 005137 002124      COM      RRFLAG      ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
10291 040320 032761 000001 002664 6$: BIT      #BIT0,CONFIG(R1) ;ERRORS PRESENT IN THIS BANK?
10292 040326 001403      BEQ      8$          ;NO - SKIP
10293 040330      SET      BMFLAG
      040330 012737 177777 002130      MOV      #-1,BMFLAG
10294 040336 005737 002600      8$: TST      WORST      ;IS THIS A WORST FIRST PASS?
10295 040342 001002      BNE      9$          ;YES - SKIP
10296 040344 005137 002130      COM      BMFLAG      ;NO - COMPLEMENT BAD MEMORY FLAG
10297 040350      9$: IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
      040350 005737 002002      TST      SELONLY
      040354 001407      BEQ      L375
      040356 032761 040000 002666      BIT      #BIT14,CONFIG+2(R1)
      040364 001003      BNE      L375
10298 040366      SET      RRFLAG
      040366 012737 177777 002124      MOV      # 1,RRFLAG
10299 040374      END ;OF IF SELONLY
      040374      L375:::
10300 040374 032761 010000 002666      BIT      #BIT12,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED?
10301 040402 001421      BEQ      ENEXBK      ;BRANCH IF IT IS NOT
10302 040404      SET      INTFLAG
      040404 012737 177777 002136      MOV      #-1,INTFLAG
10303 040412 032761 004000 002666      BIT      #BIT11,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED WITH 64K BOARDS?
10304 040420 001403      BEQ      10$      ;BRANCH IF IT IS NOT
10305 040422      SET      INT64K
      040422 012737 177777 002140      MOV      #-1,INT64K
10306 040430 032761 000040 002664 10$: BIT      #BIT5,CONFIG(R1) ;SHOULD THIS BANK BE TESTED?
10307 040436 001403      BEQ      ENEXBK      ;BRANCH IF IT SHOULD
10308 040440      SET      SKIPMK
      040440 012737 177777 002342      MOV      #-1,SKIPMK
10309 040446      ENEXBK: POP      R2,R1,R0 ;RESTORE REGISTERS
      040446 012602      MOV      (SP)+,R2
      040450 012601      MOV      (SP)+,R1
      040452 012600      MOV      (SP)+,R0
10310 040454 000207      RETURN

```


10313 040456

BANKOK: SUBTST <<SUBR BANK OK?>>

```

;*****
;SUBTEST      SUBR      BANK OK?
;*****

```

10314

10315

10316

10317 040456 013700 002134

10318 040462 005100

10319 040464 013701 002120

10320 040470 074001

10321 040472 000207

10322

10323 040474

10324 040474

```

;TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
;IS OF THE TYPE WE ARE TESTING "TMFLAG".
;RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).

```

MOV TMFLAG,R0

COM R0

MOV MKFLAG,R1

XOR R0,R1

RETURN

;OK = (=OK)

INCRPT:

INCPAT: SUBTST <<SUBR INCREMENT PATTERN TESTING >>

```

;*****
;SUBTEST      SUBR      INCREMENT PATTERN TESTING
;*****

```

;INCREMENT THE PATTERN & SET UP THE CONDITION CODES

;RESULT - Z BIT SET INDICATES OVERFLOW

INC PATTERN

CMP #30,PATTERN

RETURN

;SET UP CONDITION CODES

;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)

10325

10326

10327 040474 005237 002112

10328 040500 022737 000030 002112

10329 040506 000207

10330

10331 040510

10332 040510

SETPAT:

HIPAT: SUBTST <<SUBR SET HIGHEST PATTERN TESTING TYPE>>

```

;*****
;SUBTEST      SUBR      SET HIGHEST PATTERN TESTING TYPE
;*****

```

MOV #27,PATTERN

RETURN

;SET HIGHEST PATTERN

10333 040510 012737 000027 002112

10334 040516 000207

10335

10336 040520

INCBNK: SUBTST <<SUBR INCREMENT BANK & TEST>>

```

;*****
;SUBTEST      SUBR      INCREMENT BANK & TEST
;*****

```

;RESULTS RETURNED IN CONDITION CODES

INC BANK

CMP LASTBANK,BANK

RETURN

;TOO FAR?

10337

10338 040520 005237 002102

10339 040524 023737 002556 002102

10340 040532 000207

10343 040534

QUIT: SUBTST <<QUIT ROUTINE>>

;*****

;*SUBTEST QUIT ROUTINE

;*****

10344

10345

10346

10347

10348

10349

10350 040534 104472

10351 040536 012737 040600 000004

10352 040544

040544 005737 002126

040550 001402

040552 004737 037214

040556

10353 040556 004737 022200

10354 040562 104421

10355 040564 005737 002454

10356 040570 001003

10357 040572 042737 000040 172516

10358 040600 000005

10359 040602 000000

10360

;INITIALIZE ALL CSR'S

;UNRELOCATE IF NECESSARY

;FLUSH OUT ANY DBE'S

;TURN OFF MEMORY MANAGEMENT

;TURN OFF THE Q-BUS MAP

;HALT

ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)

SET4 #QUIT1 ;TRAPS TO 4 GOTO QUIT1

MOV #QUIT1,4

.DSABL CRF

IF RLFLAG IS TRUE THEN \$CALL UNRELOCATE

TST RLFLAG

BEQ L376

JSR PC,UNRELOCATE

L376:::;::;

CALL MT0030

DEENERGIZE

;FLUSH OUT DBE'S

;TURN OFF MEMORY MANAGEMENT

TST N022BIT

;IS THIS AN 11/83,11/23-B OR 11/23?

BNE QUIT1

BIC #BIT5,MMR3

;TURN OFF THE Q-BUS MAP

QUIT1: RESET

HALT

```

10363 040604          EXIT:  SUBTST  <<HALT PROGRAM>>
;*****
;SUBTEST             HALT PROGRAM
;*****
10364 040604 004737 040636          CALL  SHUTUP
10365 040610          EXIT2:  IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
      040610 005737 002352          TST  APTFLAG
      040614 001003          BNE  L377
      040616 005737 002350          TST  ACTFLAG
      040622 001402          BEQ  L400
      040624          L377:::;
10366 040624 000777          BR
10367 040626          ELSE
      040626 000403          BR  L401
      040630          L400:::;
10368 040630 000000          $EXHALT: HALT
10369 040632 000137 003670          JMP  START
10370 040636          END ;OF IF APTFLAG
      040636          L401:::;
10371
10372 040636          SHUTUP: SUBTST  <<SHUTDOWN DIAGNOSTIC>>
;*****
;SUBTEST             SHUTDOWN DIAGNOSTIC
;*****
10373          ;INITIALIZE ALL CSR'S
10374          ;UNRELOCATE
10375          ;FLUSH OUT DBE'S
10376          ;RESTORE LOADERS
10377          ;TURN OFF MEMORY MANAGEMENT
10378          ;UNMAP THE Q-BUS MAP
10382 040636 104472          ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
10383 040640          IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
      040640 005737 002126          TST  RLFLAG
      040644 001402          BEQ  L402
      040646 004737 037214          JSR  PC,UNRELOCATE
      040652          L402:::;
10384 040652          IF QUICK IS FALSE
      040652 005737 002436          TST  QUICK
      040656 001002          BNE  L403
10385 040660 004737 022200          CALL  MT0030 ;FLUSH OUT DBE'S
10386 040664          END ;OF IF QUICK
      040664          L403:::;
10387 040664 012700 000001          MOV   #1,R0 ;DESTINATION BANK
10388 040670 013701 002576          MOV   LOADHOME,R1 ;SOURCE BANK
10389 040674 004737 037442          CALL  BANKMOV
10390 040700 104421          DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
10391 040702 005737 002454          TST   NO22BIT ;DOES THIS PDP-11 HAVE 22-BIT ADDR?
10392 040706 001003          BNE  1$ ;BRANCH IF NOT
10393 040710 042737 000040 172516          BIC   #BIT5,MMR3 ;TURN OFF Q-BUS MAP
10397 040716 000207          1$: RETURN
10398
10399 040720          APTDOWN: SUBTST  <<APT SHUTDOWN SEQUENCE>>
;*****
;SUBTEST             APT SHUTDOWN SEQUENCE
;*****
10400 040720          MAP   #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
      040720 010346          MOV  R3,-(SP)

```

CVMJAA0 MSV11-J MEMORY DIAG.
APT SHUTDOWN SEQUENCE

MACRO V05.01b Friday 15-Feb-85 16:10 Page 334-1

SEQ 0306

	040722	012703	000000		MOV	#0,R3	
	040726	004737	035716		CALL	MAPPER	
					.DSABL	CRF	
	040732	012603					MOV (SP),R3
10401	040734				TESTAREA		;ENTER TEST MODE
	040734	053737	002552	177776	BIS	TESTMODE,PSW	;GO TO SYSTEM TEST MODE
					.DSABL	CRF	
10402	040742	012737	040720	060024	MOV	#APTDOWN,FIRST+24	
10403	040750	012737	000340	060026	MOV	#340,FIRST+26	
10404	040756	012737	000000	120720	MOV	#0,FIRST+APTDOWN	
10405	040764	104417			KERNEL		;ENTER KERNEL MODE
10406	040766	000000			APTHLT: HALT		

10409 040770

SUBTST <<BLOCK MOVE SUBROUTINE>>

```

*****
;SUBTEST      BLOCK MOVE SUBROUTINE
*****

```

10410
10411
10412
10413
10414
10415

```

;BLOCK3 HAS 3 ARGUEMENTS
;BLOCK2 HAS 2 ARGUEMENTS
;BLOCK1 HAS 1 ARGUEMENTS
;
;ALL ARE CALLED BY THE BMOV MACRO
.ENABL  LSB

```

10416 040770

BLOCK1: PUSH R0,R1,R2

```

MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)

```

040770 010046
040772 010146
040774 010246

10417 040776 012702 177640
10418 041002 012701 000020
10419 041006 000413

```

MOV  #FASTCITY,R2
MOV  #16.,R1
BR    3$

```

10420

10421 041010

BLOCK2: PUSH R0,R1,R2

```

MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)

```

041010 010046
041012 010146
041014 010246

10422 041016 012701 000020
10423 041022 000404

```

MOV  #16.,R1
BR    2$

```

10424

10425 041024

BLOCK3: PUSH R0,R1,R2

```

MOV R0,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)

```

041024 010046
041026 010146
041030 010246

10426 041032 012501
10427 041034 012502
10428 041036 012500

```

2$: MOV  (R5)+,R1
    MOV  (R5)+,R2
3$: MOV  (R5)+,R0

```

10429

10430 041040 012022

```

1$: MOV  (R0)+,(R2)+
    SOB  R1,1$
    POP  R2,R1,R0

```

10431 041042 077102

10432 041044

```

MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0

```

041044 012602
041046 012601
041050 012600

10433 041052 000205
10434

```

RTS  R5
.DSABL  LSB

```

```

10436
10437
10438 041054
10439 041054 104415
10440 041056 104401 066116
10441
10442 041062 005737 002126
      041066 001003
      041070 005737 002426
      041074 001404
      041076
10443 041076 104401 067557
      041076
10444 041102 104416
10445 041104 000207
10446 041106
      041106
10447 041106 005737 002544
10448 041112 001402
10449 041114
      041114 013746 177746
10450 041120
      041120 013746 002152
      041124 013746 002006
10451 041130 104424
10452 041132
      041132 012737 177777 002006
10453 041140
      041140 104401 067007
10454 041144 104414
10455 041146
      041146 012600
10456 041150 020027 000022
10457 041154 101403
10458 041156
      041156 104401 066137
10459 041162 000766
10460 041164
      041164 010046
      041166 006316
      041170 004737 041242
10461 041174 041252
10462 041176 041354
10463 041200 041464
10464 041202 041632
10465 041204 042106
10466 041206 042426
10467 041210 043344

      .SBTTL FIELD SERVICE MODE
      FIELDSERVICE:SUBTST <<SUBR FIELD SERVICE COMMAND MODE>>
      ;*****
      ;*SUBTEST SUBR FIELD SERVICE COMMAND MODE
      ;*****
      SAVREG
      TYPE MSG020 ;FIELD SERVICE COMMAND MODE
      TYPEIT ,MSG020
      .DSABL CRF
      IF RLFLAG IS TRUE OR NOFSMODE IS TRUE
      TST RLFLAG
      BNE L404
      TST NOFSMODE
      BEQ L405
      L404:;;;
      TYPE MSG048 ;NOT AVAILABLE NOW - TRY LATER!
      TYPEIT ,MSG048
      .DSABL CRF
      RESREG
      RETURN
      END ;OF IF RLFLAG
      L405:;;;;;
      TST CACHKN
      BEQ 1$
      PUSH CONTRL ;SAVE CACHE STATUS
      MOV CONTRL,-(SP)
      1$: PUSH CSRNO,KAMIKAZE ;SAVE CSR & KAMIKAZE STATUS
      MOV CSRNO,-(SP)
      MOV KAMIKAZE,-(SP)
      CACHOFF
      SET KAMIKAZE ;TURN CACHE OFF
      MOV #-1,KAMIKAZE
      FS1: TYPE MSG026 ;COMMAND:
      TYPEIT ,MSG026
      .DSABL CRF
      RDDEC ;READ A DECIMAL NUMBER
      POP R0 ;COMMAND - > R0
      MOV (SP)+,R0
      CMP R0,#18.
      BLOS 1$
      TYPE MSG021
      TYPEIT ,MSG021
      .DSABL CRF
      BR FS1
      1$: CASE R0
      FSCMD0 ;EXIT FIELD SERVICE COMMANDS
      FSCMD1 ;READ CSR
      FSCMD2 ;LOAD CSR
      FSCMD3 ;EXAMINE MEMORY
      FSCMD4 ;MODIFY MEMORY
      FSCMD5 ;SELECT BANK & PATTERN
      FSCMD6 ;TYPE CONFIGURATION MAP

```

10468	041212	043352	FSCMD7	;SOB-A-LONG TEST
10469	041214	043644	FSCMD8	;ERROR SUMMARY
10470	041216	044072	FSCMD9	;REFRESH TEST
10471	041220	044364	FCMD10	;SET FILL COUNT
10472	041222	044412	FCMD11	;ENTER KAMIKAZE MODE
10473	041224	044434	FCMD12	;EXIT KAMIKAZE MODE
10474	041226	044454	FCMD13	;TURN CACHE OFF
10475	041230	044476	FCMD14	;TURN CACHE ON
10476	041232	044514	FCMD15	;TEST ONLY SELECTED BANKS
10477	041234	044600	FCMD16	;RESUME TESTING ALL BANKS
10478	041236	044642	FCMD17	;ENABLE TRACE
10479	041240	044656	FCMD18	;DISABLE TRACE
10480	041242		END ;OF CASE	

	041242	
	041242	062616
	041244	013646
	041246	004736
10481	041250	000733

BR FS1

L406:::;:::
ADD (SP),@SP
MOV @ (SP),-(SP)
JSR PC,@ (SP).

```

10484 041252
10485 041252 104401 070535
10486 041256 062706 000002
10487 041262 005737 002010
10488 041270 062706 000002
10489 041274 005037 002010
10490 041300 000402
10491 041302 012637 002006
10492 041306
10493 041306 012637 002152
10494 041312 005737 002544
10495 041316 001414
10496 041320 023737 002544 002550
10497 041326 001003
10498 041330 062706 000002
10499 041334 000405
10500 041336 005737 002544
10501 041342 001402
10502 041344 012637 177746
10503 041350 104416
10504 041352 000207
10505
10506 041354
10507 041354 004737 044670
10508 041360 010637 002306
10509 041364 012737 041430 000004
10510 041372 104426
10511 041374 012737 177777 002430
10512 041402 104026
10513 041404 012737 034114 000004
10514 041412 022737 000005 004010
10515 041420 001002

FSCMD0: SUBTST <<COMMAND 0 EXIT>>
;*****
;SUBTEST COMMAND 0 EXIT
;*****
TYPE MSG103
TYPEIT MSG103
DSABL CRF
ADD #2,SP
IF SKIPKAMI IS TRUE
TST SKIPKAMI
BEQ L407
;THROW AWAY OLD KAMIKAZE FLAG
BR L410
L407:;;;;;
;RESTORE OLD KAMIKAZE FLAG
MOV (SP)+,KAMIKAZE
L410:;;;;;
MOV (SP)+,CSRNO
TST CACHKN
BEQ RES0
IF CACHKN EQ CACHKF
;IF CACHE IS OFF
CMP CACHKN,CACHKF
BNE L411
;THROW AWAY CACHE STATUS
BR L412
L411:;;;;;
TST CACHKN
BEQ RES0
POP CONTRL
;RESTORE CACHE STATUS
MOV (SP)+,CONTRL
L412:;;;;;
RES0: RESREG
RETURN

FSCMD1: SUBTST <<FS COMMAND 1 READ CSR>>
;*****
;SUBTEST FS COMMAND 1 READ CSR
;*****
CALL WHICHCSR
MOV SP,FSSTACK
SET4 #RES1
MOV #RES1,4
;TRAPS TO 4 GOTO RES1
DSABL CRF
READCSR
SET NOERROR
MOV #-1,NOERROR
ERROR +26
RES4
MOV #TIMEOUT,4
CMP #5,PROTYP
BNE 101$
;USE ERROR ROUTINE FOR PRINTOUT
;RESET TRAPS TO 4 TO DEFAULT
;IS THIS AN 11/83/84 ?
;BRANCH IF NOT

```


	041422	005037	177766		CLR	CPUERR		;CLEAR OUT THE CPU ERROR REGISTER BITS
	041426			101\$:				;THAT A EXPECTED TRAP COULD HAVE SET
10514	041426	000207			.DSABL	CRF		
10515	041430			RES1:	RETURN			
	041430	104401	066763		TYPE	MSG025		;THIS CSR DOES NOT EXIST
					TYPEIT	MSG025		
10516	041434	013706	002306		.DSABL	CRF		
10517	041440				MOV	FSSTACK,SP		
	041440	012737	034114	000004	RES4			;RESET TRAPS TO 4 TO DEFAULT
	041446	022737	000005	004010	MOV	#TIMEOUT,4		
	041454	001002			CMP	#5,PROTYP		;IS THIS AN 11/83/84 ?
	041456	005037	177766		BNE	101\$;BRANCH IF NOT
	041462			101\$:	CLR	CPUERR		;CLEAR OUT THE CPU ERROR REGISTER BITS
								;THAT A EXPECTED TRAP COULD HAVE SET
10518	041462	000207			.DSABL	CRF		
					RETURN			

```

10521 041464      FSCMD2: SUBTST <<FS  COMMAND 2  LOAD CSR>>
;*****
;SUBTEST      FS  COMMAND 2  LOAD CSR
;*****
10522 041464 004737 044670      CALL  WHICHCSR
10523 041470 010637 002306      MOV   SP,FSSTACK
10524 041474 012737 041576 000004  SET4  #RES2      ;TRAPS TO 4 GOTO RES2
                                MOV   #RES2,4
                                .DSABL CRF
10525 041502 104426      READCSR
10526 041504 104401 067021      TYPE  MSG027
                                TYPEIT ,MSG027
                                .DSABL CRF
10527 041510 012737 177777 002430  SET   NOERROR
                                MOV   #-1,NOERROR
10528 041516 104026      ERROR  +26      ;USE ERROR ROUTINE FOR PRINTOUT
10529 041520 012737 034114 000004  RES4      ;RESET TRAPS TO 4 TO DEFAULT
                                MOV   #TIMEOUT,4
                                CMP   #5,PROTYP      ;IS THIS AN 11/83/84 ?
                                BNE   101$      ;BRANCH IF NOT
                                CLR   CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                101$:
                                ;THAT A EXPECTED TRAP COULD HAVE SET
                                ;FIRST CSR WORD
                                .DSABL CRF
                                TYPE  MSG023
                                TYPEIT ,MSG023
                                .DSABL CRF
10530 041542 104401 066747      RDOCT
                                POP   CSR      ;READ AN OCTAL NUMBER
                                ;PUT IN IN LOC "CSR"
                                MOV (SP),.CSR
10531 041546 104413      LOADCSR
10532 041550 012637 002150      READCSR
                                TYPE  MSG028
                                TYPEIT ,MSG028
                                .DSABL CRF
10533 041554 104425      SET   NOERROR
10534 041556 104426      ERROR  +26      ;USE FOR PRINTOUT - NOT AN ERROR
10535 041560 104401 067036      RETURN
                                TYPE  MSG025
                                TYPEIT ,MSG025
                                .DSABL CRF
10536 041564 012737 177777 002430  RES2:
                                MOV   FSSTACK,SP
                                RES4      ;RESET TRAPS TO 4 TO DEFAULT
                                MOV   #TIMEOUT,4
                                CMP   #5,PROTYP      ;IS THIS AN 11/83/84 ?
                                BNE   101$      ;BRANCH IF NOT
                                CLR   CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                101$:
                                ;THAT A EXPECTED TRAP COULD HAVE SET
                                .DSABL CRF
10537 041572 104026      RETURN
10538 041574 000207
10539 041576 104401 066763
10540 041602 013706 002306
10541 041606 012737 034114 000004  MOV   FSSTACK,SP
                                RES4      ;RESET TRAPS TO 4 TO DEFAULT
                                MOV   #TIMEOUT,4
                                CMP   #5,PROTYP      ;IS THIS AN 11/83/84 ?
                                BNE   101$      ;BRANCH IF NOT
                                CLR   CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                101$:
                                ;THAT A EXPECTED TRAP COULD HAVE SET
                                .DSABL CRF
10542 041630 000207      RETURN

```

```

10545 041632      FSCMD3: SUBTST <<FS  COMMAND 3  EXAMINE MEMORY>>
;*****
;SUBTEST  FS  COMMAND 3  EXAMINE MEMORY
;*****
10546 041632      PUSH  BANK,NOPAR,PARTHERE.4
041632 013746 002102      MOV BANK,-(SP)
041636 013746 002076      MOV NOPAR,-(SP)
041642 013746 002304      MOV PARTHERE,-(SP)
041646 013746 000004      MOV 4,-(SP)
10547 041652 012737 000002 002076      MOV #2,NOPAR ;INDICATE PARITY ACTION
10548 041660      TYPE MSG029 ;EXAMINE MEMORY
041660 104401 067052      TYPEIT ,MSG029
10549 041664      .DSABL CRF
041664 104401 067111      1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-17775776)??
TYPEIT ,MSG031
.DSABL CRF
10550 041670 104413      RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
10551 041672 013737 056146 002102      MOV $HIOCT,BANK ;PUT MSB'S IN BANK
10552 041700      POP RO ;PUT LSB'S IN RO
041700 012600      MOV (SP)+,RO
10553 041702 000241      CLC
10554 041704 006100      ROL RO
10555 041706 006137 002102      ROL BANK
10556 041712 000241      CLC
10557 041714 006000      ROR RO
10558 041716 023737 002102 002556      CMP BANK, LASTBANK ;CHECK FOR BANK TOO HIGH
10559 041724 003357      BGT 1$ ;BRANCH IF TRUE
10560 041726 062700 060000      ADD #FIRST,RO
10561 041732 032700 000001      BIT #BIT0,RO ;CHECK FOR ODD ADDRESS
10562 041736 001352      BNE 1$ ;BRANCH IF ODD ADDRESS
10563 041740 020027 157776      CMP RO,#LAST ;CHECK FOR ADDRESS OVER 16K
10564 041744 101347      BHI 1$ ;BRANCH IF OVER 16K
10565 041746 012737 042020 002304      MOV #3$,PARTHERE ;INCASE OF ABORTS
10566 041754      SET4 #4$ ;TRAPS TO 4 GOTO 4$
041754 012737 042026 000004      MOV #4$,4
.DSABL CRF
10567 041762      MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
041762 010346      MOV R3,-(SP)
041764 013703 002102      MOV BANK,R3
041770 004737 035716      CALL MAPPER
.DSABL CRF
10568 041774 012603      MOV (SP)+,R3
041776      TESTAREA ;ENTER TEST MODE
041776 053737 002552 177776      BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
10569 042004 011001      MOV (R0),R1
10570 042006 104417      KERNEL ;ENTER KERNEL MODE
10571 042010      TYPOCS R1
042010 010146      MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
042012 104403      TYPOS ;GO TYPE--OCTAL ASCII
042014 006 ;TYPE 6 DIGITS
042015 000 ;SUPPRESS LEADING ZEROS
.DSABL CRF
10572 042016 000410      BR EXCMD3
10573
10574 042020      3$: TYPE MSG032 ;PARITY ABORT
042020 104401 067151      TYPEIT ,MSG032

```

```

10575 042024 000405          .DSABL CRF
10576          BR          EXCMD3
10577 042026 062706 000004    4$: ADD #4,SP          ;FIX STACK
10578 042032          TYPE MSG033          ;TIMEOUT TRAP
      042032 104401 067170    TYPEIT ,MSG033
      .DSABL CRF
10579 042036 000400          BR          EXCMD3
10580          EXCMD3: KERNEL          ;ENTER KERNEL MODE
10581 042040 104417          POP 4,PARTHERE,NOPAR,BANK
10582 042042          MOV (SP),.4
      042042 012637 000004          MOV (SP),.PARTHERE
      042046 012637 002304          MOV (SP),.NOPAR
      042052 012637 002076          MOV (SP),.BANK
      042056 012637 002102
10583 042062          RES4          ;RESET TRAPS TO 4 TO DEFAULT
      042062 012737 034114 000004    MOV #TIMEOUT,4
      042070 022737 000005 004010    CMP #5,PROTYP          ;IS THIS AN 11/83/84 ?
      042076 001002          BNE 101$          ;BRANCH IF NOT
      042100 005037 177766          CLR CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
      042104          101$:          ;THAT A EXPECTED TRAP COULD HAVE SET
      .DSABL CRF
10584 042104 000207          RETURN

```

```

10587 042106      FSCMD4: SUBTST <<FS  COMMAND 4  MODIFY MEMORY>>
;*****
;:SUBTEST      FS  COMMAND 4  MODIFY MEMORY
;*****
10588 042106      PUSH      BANK,NOPAR,PARTHERE,4
                                MOV BANK,-(SP)
                                MOV NOPAR,-(SP)
                                MOV PARTHERE,-(SP)
                                MOV 4,-(SP)
10589 042126      013746  002102
10590 042134      013746  002076
                                042112  013746  002304
                                042116  013746  000004
10589 042126      012737  000003  002076      MOV #3,NOPAR ;INDICATE PARITY ACTION
10590 042134      104401  067276      TYPE MSG036 ;MODIFY MEMORY
                                TYPEIT ,MSG036
                                .DSABL CRF
10591 042140      104401  067111      1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-17775776)??
                                TYPEIT ,MSG031
                                .DSABL CRF
10592 042144      104413
10593 042146      013737  056146  002102      RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
10594 042154      012600      MOV $HIOCT,BANK ;PUT MSB'S IN BANK
                                POP RO ;PUT LSB'S IN RO
                                MOV (SP)+,RO
10595 042156      000241      CLC
10596 042160      006100      ROL RO
10597 042162      006137  002102      ROL BANK
10598 042166      000241      CLC
10599 042170      006000      ROR RO
10600 042172      023737  002102  002556      IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
                                CMP BANK, LASTBANK
                                BGT 1$
10601 042202      062700  060000      ADD #FIRST,RO
10602 042206      032700  000001      IF #BIT0 SET.IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
                                BIT #BIT0,RO
                                BNE 1$
10603 042214      020027  157776      IF RO HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER
                                CMP RO,#LAST
                                BHI 1$
10604 042222      012737  042270  002304      MOV #3$,PARTHERE ;INCASE OF ABORTS
10605 042230      012737  042276  000004      SET4 #4$ ;TRAPS TO 4 GOTO 4$
                                MOV #4$,4
                                .DSABL CRF
10606 042236      010346  002102      MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
                                MOV R3,-(SP)
                                042240  013703  002102
                                042244  004737  035716      MOV BANK,R3
                                CALL MAPPER
                                .DSABL CRF
                                MOV (SP)+,R3
10607 042250      012603
10608 042252      104511      INVALIDATE
10608 042254      053737  002552  177776      TESTAREA ;ENTER TEST MODE
                                BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
                                .DSABL CRF
10609 042262      011001      MOV (RO),R1
10610 042264      104417      ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
10611 042266      000410      KERNEL ;ENTER KERNEL MODE
10612 042266      000410      BR 5$
10613 042270      104401  067151      3$: TYPE MSG032 ;PARITY ABORT
                                TYPEIT ,MSG032
                                .DSABL CRF

```

```

10615 042274 000431          BR      EXCMD4          ;EXIT
10616
10617 042276 062706 000004    4$:  ADD      #4,SP          ;FIX STACK
10618 042302 104401 067170    TYPE      MSG033        ;TIMEOUT TRAP
                                TYPEIT     ,MSG033
                                .DSABL      CRF
                                BR          EXCMD4          ;EXIT
10619 042306 000424
10620
10621 042310 104401 067315    5$:  TYPE      MSG037        ;OLD DATA WAS
                                TYPEIT     ,MSG037
                                .DSABL      CRF
                                TYPOCS     R1              ;PRINT IT
                                MOV         R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
                                TYPOS      6               ;;GO TYPE--OCTAL ASCII
                                .BYTE      0               ;;TYPE 6 DIGITS
                                .DSABL      CRF             ;;SUPPRESS LEADING ZEROS
10622 042314 010146
                                042314 104403
                                042316 006
                                042320 000
10623 042322 104401 067352    TYPE      MSG039        ;INPUT NEW DATA
                                TYPEIT     ,MSG039
                                .DSABL      CRF
10624 042326 104413
                                RDOCT
10625 042330 012601          POP      R1              READ ON OCTAL NUMBER ONTO THE STACK
                                ;GET NEW NUMBER
10626 042332 053737 002552 177776 TESTAREA
                                BIS          TESTMODE,PSW    ;ENTER TEST MODE
                                .DSABL      CRF             ;GO TO SYSTEM TEST MODE
10627 042340 010110          MOV      R1,(R0)          ;PUT IT IN MEMORY
10628 042342 011001          MOV      (R0),R1          ;READ IT AGAIN
10629 042344 104417          KERNEL
10630 042346 104401 067334    TYPE      MSG038        ;ENTER KERNEL MODE
                                TYPEIT     ,MSG038          ;DATA IS NOW
                                .DSABL      CRF
10631 042352 010146
                                042352 104403
                                042354 006
                                042356 000
                                TYPOCS     R1              ;PRINT IT
                                MOV         R1,-(SP)        ;;SAVE R1 FOR TYPEOUT
                                TYPOS      6               ;;GO TYPE--OCTAL ASCII
                                .BYTE      0               ;;TYPE 6 DIGITS
                                .DSABL      CRF             ;;SUPPRESS LEADING ZEROS
10632
10633 042360 104417          EXCMD4: KERNEL
10634 042362 012637 000004    POP      4,PARTHERE,NOPAR,BANK ;ENTER KERNEL MODE
                                MOV (SP)+,4
                                MOV (SP)+,PARTHERE
                                MOV (SP)+,NOPAR
                                MOV (SP)+,BANK
10635 042402 012637 002304
                                042372 012637 002076
                                042376 012637 002102
                                RES4
                                MOV      #TIMEOUT,4        ;RESET TRAPS TO 4 TO DEFAULT
                                CMP      #5,PROTYP          ;IS THIS AN 11/83/84 ?
                                BNE      101$              ;BRANCH IF NOT
                                CLR      CPUERR             ;CLEAR OUT THE CPU ERROR REGISTER BITS
10636 042424 000207          101$:
                                .DSABL      CRF             ;THAT A EXPECTED TRAP COULD HAVE SET
                                RETURN

```

```

10639 042426 FSCMD5: SUBTST <<FS COMMAND 5 SELECT BANK & PATTERN>>
;*****
;SUBTEST FS COMMAND 5 SELECT BANK & PATTERN
;*****
10640 042426 013746 002102 PUSH BANK,PATTERN,TESTADD,PCBUMP,TKVEC,TKVEC+2
042426 013746 002102 MOV BANK,-(SP)
042432 013746 002112 MOV PATTERN,(SP)
042436 013746 002412 MOV TESTADD,-(SP)
042442 013746 002326 MOV PCBUMP,-(SP)
042446 013746 000060 MOV TKVEC,(SP)
042452 013746 000062 MOV TKVEC+2,-(SP)
10641 042456 010637 002306 MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
10642 042462 TYPE MSG040 ;SELECT BANK & PATTERN TEST
042462 104401 067374 TYPEIT ,MSG040
.DSABL CRF
10643 042466 104401 067072 1$: TYPE MSG030 ;BANK(0-177)?
042466 TYPEIT ,MSG030
.DSABL CRF
10644 042472 104413 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
10645 042474 012637 002102 POP BANK ;PUT IT IN BANK
10646 042500 023737 002102 002556 IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
042500 MOV (SP)+,BANK
042506 003367 CMP BANK,LASTBANK
BGT 1$
10647 042510 013701 002102 MOV BANK,R1
10648 042514 006301 ASL R1
10649 042516 006301 ASL R1
10650 042520 IF CPUBIT OFF.IN CONFIG(R1)
10651 042520 033761 002106 002664 BIT CPUBIT,CONFIG(R1)
042526 001003 BNE L413
10652 042530 TYPE MSG041 ;BANK NOT ACCESSABLE
042530 104401 067420 TYPEIT ,MSG041
.DSABL CRF
GOTO 1$
10653 042534 000754 BR 1$
042534 000754
10654 042536 END ;OF IF
042536 L413:::;
10655 042536 2$: TYPE MSG042 ;PATTERN(0-45)?
10656 042536 104401 067445 TYPEIT ,MSG042
.DSABL CRF
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP PATTERN ;PUT IT IN PATTERN
10657 042542 104413 MOV (SP)+,PATTERN
10658 042544 012637 002112 IF PATTERN GT #47 THEN GOTO 2$ ;CHECK FOR PATTERN TO HIGH
042544 023727 002112 000047 CMP PATTERN,#47
042556 003367 BGT 2$
10660 042560 IF PATTERN EQ #0
042560 005737 002112 TST PATTERN
042564 001004 BNE L414
10661 042566 TYPE MSG043 ;PATTERN 0 DATA IS?
042566 104401 067463 TYPEIT ,MSG043
.DSABL CRF
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP R2 ;PUT IT IN R2
10662 042572 104413 MOV (SP)+,R2
10663 042574 012602
042574

```

```

10664 042576          END ;OF IF
10665 042576          L414:::
10666
10667 042576          MAP      BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      042576 010346          MOV R3,-(SP)
      042600 013703 002102
      042604 004737 035716          CALL  BANK,R3
      .DSABL CRF          CALL  MAPPER
      .DSABL CRF          .DSABL CRF
      042610 012603          MOV (SP)+,R3
10668 042612 104511
10669 042614 004737 040052          INVALIDATE
      042620 005737 002124          CALL  EXBANK
      042624 001404          IF RRFLAG IS TRUE          ;SET NEW MARGINS
      042626 104401 067617          TST RRFLAG
      042626 104401 067617          BEQ L415
10671 042626          TYPE MSG049          ;BANK REQUIRES RELOCATION
      042626 104401 067617          TYPEIT ,MSG049
      .DSABL CRF          .DSABL CRF
      JMP CMD5C
10672 042632 000137 043246          END ;OF IF RRFLAG
10673 042636
      042636          L415:::
10674 042636          TYPE MSG046          ;TO ESCAPE TYPE ANY KEY!
      042636 104401 067505          TYPEIT ,MSG046
      .DSABL CRF          .DSABL CRF
10675 042642 013737 002152 002154          MOV  CSRNO,SAVCSR          ;SAVE OLD CSR NUMBER
10676 042650 013702 002102          MOV  BANK,R2          ;
10677 042654 072227 000002          ASH  #2,R2          ;GENERATE INDEX INTO CONFIGURATION TABLE
10678 042660 016203 002664          MOV  CONFIG(R2),R3          ;R3 = LOW WORD OF CONFIGURATION TABLE FOR THIS BANK
10679 042664 072327 177770          ASH  #-10,R3          ;POSITION CSR CODE IN BITS 0 3
10680 042670 042703 177760          BIC  #1C17,R3          ;CLEAR ALL BUT THE CSR CODE
10681 042674 006303          ASL  R3          ;ADJUST CSR NUMBER
10682 042676 010337 002152          MOV  R3,CSRNO          ;
10683 042702 012737 043246 000060          MOV  #CMD5C,TKVEC
10684 042710 012737 000340 000062          MOV  #340,TKVEC+2
10685 042716 017700 137722          MOV  @TKB,R0          ;KILL ANY OLD INTERRUPT
10686 042722 042737 000200 177776          BIC  #BIT7,PSW          ;LOWER CPU PRIORITY TO 140
10687 042730 052777 000100 137704          BIS  #BIT6,@TKS          ;ENABLE KEYBOARD INTERRUPTS
10688
10689
10690 042736          SET  HEADER,MUT
      042736 012737 177777 002612          MOV  #-1,HEADER
      042744 012737 177777 002110          MOV  #-1,MUT
10691 042752 013701 002102          CMD5B: MOV  BANK,R1
10692 042756 006301          ASL  R1
10693 042760 006301          ASL  R1
10694 042762 005037 002240          CLR  SPLTCSR
10695 042766 005037 002264          CLR  PASFLG
10696 042772 012737 060000 002412          MOV  #FIRST,TESTADD
10697 043000 012737 060002 002414          MOV  #FIRST+2,TESTADD+2
10698 043006          IF #BIT12 SET.IN CONFIG+2(R1)
      043006 032761 010000 002666          BIT  #BIT12,CONFIG+2(R1)
      043014 001413          BEQ L416
10699 043016 005237 002240          INC  SPLTCSR
10700 043022          MAP  BANK
      043022 010346          MOV R3,-(SP)
      043024 013703 002102          MOV  BANK,R3
      043030 004737 035716          CALL  MAPPER

```



```

      .DSABL CRF
10701 043034 012603      MOV (SP)+,R3
10702 043036 012737 120000 002414      MOV #120000,TESTADD+2
      END; OF IF #BIT12
10703 043044      L416:;;;;;
      043044 032777 000001 137564      IF #SWO SET.IN @SWR
      043052 001402      BIT #SWO,@SWR
10704 043054 104470      BEQ L417
      ECCDIS      ;DISABLE ERROR CORRECTION
10705 043056 000405      ELSE
      043060      BR L420
10706 043060      L417:;;;;;
      043060 013746 002152      PUSH CSRNO
      043064 104502      CLRCR
      043066 012637 002152      POP CSRNO
      043072      MOV (SP)+,CSRNO
10709 043072      END ;OF IF
      043072      L420:;;;;;
10710 043072 012737 000002 002076      MOV #2,NOPAR      ;PARITY ACTION
10711 043100 012737 000002 002326      MOV #2,PCBUMP      ;TRAPS ADD 2 TO PC
10712 043106 013700 002112      MOV PATTERN,R0
10713 043112 006300      ASL R0
10714 043114 004770 043126      CALL @FSPAT(R0)
10715 043120 005037 002076      CLR NOPAR
10716 043124 000712      BR CMD5B      ;LOOP TILL KEYBOARD INTERRUPT
10717
FSPAT: MT0000 ;<1 SEC      DATA PATTERN TEST
10718 043126 017510      MT0001 ;<1 SEC      ADDRESS TEST
10719 043130 017544      MT0002 ;<1 SEC      COMPLEMENT ADDRESS TEST
10720 043132 017640      MT0003 ; 1 SEC      3 XOR 9 WORST CASE NOISE TEST
10721 043134 017754      MT0004 ; 1 SEC      ROTATING ZEROS TEST
10722 043136 020070      MT0005 ; 1 SEC      ROTATING ONES TEST
10723 043140 020140      MT0006 ;<1 SEC      INITIAL DATA TEST
10724 043142 020222      MT0007 ;<1 SEC      ADDRESS BIT TEST
10725 043144 020256      MT0010 ;<1 SEC      BYTE ADDRESSING TEST
10726 043146 020320      MT0999 ;<1 SEC      NULL TEST TO KEEP TABLE ORDER
10727 043150 024220      MT0999 ;<1 SEC      NULL TEST TO KEEP TABLE ORDER
10728 043152 024220      MT0999 ;<1 SEC      NULL TEST TO KEEP TABLE ORDER
10729 043154 024220      MT0014 ; 1 SEC      BASIC DOUBLE BIT ERROR TEST
10730 043156 020354      MT0999 ;<1 SEC      NULL TEST TO KEEP TABLE ORDER
10731 043160 024220      MT0999 ;<1 SEC      NULL TEST TO KEEP TABLE ORDER
10732 043162 024220      MT0017 ;<1 SEC      HOLDING 1'S & 0'S TEST
10733 043164 020434      MT0020 ; 1 SEC      SYNDROMES TO CSR ON SBE TEST
10734 043166 020456      MT0021 ; 1 SEC      MARCHING 0'S & 1'S TEST
10735 043170 020536      MT0022 ;10 SEC      REFRESH & SHIFTING DIAGONAL TEST
10736 043172 021026      MT0023 ;10 SEC      SHIFTING DIAGONAL TEST
10737 043174 021060      MT0024 ;20 SEC      FAST GALLOPING PATTERN TEST
10738 043176 021124      MT0999 ;<1 SEC      NULL TEST TO KEEP TABLE ORDER
10739 043200 024220      MT0026 ;<1 SEC      RANDOM DATA TEST
10740 043202 021332      MT0027 ; 1 SEC      UNIQUE BANK TEST
10741 043204 021610      MT0030 ; 1 SEC      FLUSH OUT DBE'S TEST
10742 043206 022200      MT0031 ; 3 SEC      SOB-A-LONG TEST
10743 043210 022454      MT0032 ;<1 SEC      WRITE RECOVERY TEST
10744 043212 022644      MT0033 ;35 SEC      BRANCH GOBBLE TEST
10745 043214 023160      MT0034 ; 1 SEC      SOFT ERROR TEST
10746 043216 023346      MT0035 ;<1 SEC      WORST CASE NOISE PARITY TEST
10747 043220 023446

```

10748	043222	023560		MT0036	: 1 SEC	CORRECTION CODE TEST
10749	043224	023622		MT0037	: <1 SEC	ECC DISABLE TEST
10750	043226	024220		MT0999	: <1 SEC	NULL TEST TO KEEP TABLE ORDER
10751	043230	023660		MT0041	: 1 SEC	ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
10752	043232	023740		MT0042	: <1 SEC	EXTENDED Q-BUS ADDRESS TEST
10753	043234	023774		MT0043	: 1 SEC	WRITE BYTE CLEARS SBE TEST
10754	043236	024024		MT0044	: 5 SEC	SHIFTING 1/0'S THROUGH THE CHECK BITS
10755	043240	024104		MT0045	: 1 SEC	SYNDROMES TO CSR ON DBE TEST
10756	043242	024134		MT0046	: 1 SEC	CHECK SINGLE BIT ERROR WITH ECC DISABLED TEST
10757	043244	024164		MT0047	: <1 SEC	NO CSR UPDATE WITH SBE ON DBE TEST
10758						
10759	043246	013706	002306	CMD5C: MOV	FSSTACK,SP	;RECOVER OLD STACK POINTER
10760	043252	042777	000100	BIC	#BIT6,@\$TKS	
10761	043260			POP	TKVEC+2,TKVEC	
	043260	012637	000062			MOV (SP)+,TKVEC+2
	043264	012637	000060			MOV (SP)+,TKVEC
10762	043270	117700	137350	MOVB	@\$TKB,R0	;GET CHARACTER TO GET RID OF FLAG
10763	043274			POP	PCBUMP,TESTADD	
	043274	012637	002326			MOV (SP)+,PCBUMP
	043300	012637	002412			MOV (SP)+,TESTADD
10764	043304			POP	PATTERN,BANK	
	043304	012637	002112			MOV (SP)+,PATTERN
	043310	012637	002102			MOV (SP)+,BANK
10765	043314			MAP	BANK	;REMAP OLD BANK
	043314	010346				MOV R3,-(SP)
	043316	013703	002102	MOV	BANK,R3	
	043322	004737	035716	CALL	MAPPER	
				.DSABL	CRF	
	043326	012603				MOV (SP)+,R3
10766	043330	004737	040052	CALL	EXBANK	
10767	043334	013737	002154	MOV	SAVCSR,CSRNO	;RESTORE CSRNO.
10768	043342	000207		RETURN		

10770 043344

FSCMD6: SUBTST <<FS COMMAND 6 TYPE CONFIGURATION MAP>>
;*****
;*SUBTEST FS COMMAND 6 TYPE CONFIGURATION MAP
;*****
CALL PCONFIG
RETURN

10771 043344 004737 032722
10772 043350 000207
10773

```

10776 043352      FSCMD7: SUBTST <<FS      COMMAND 7      SOB-A-LONG TEST>>
;*****
;SUBTEST      FS      COMMAND 7      SOB-A-LONG TEST
;*****
      PUSH      BANK,PATTERN,TKVEC,TKVEC+2,NOPAR

10777 043352      MOV      BANK,-(SP)
      043352      013746      002102      MOV      PATTERN,-(SP)
      043356      013746      002112      MOV      TKVEC,-(SP)
      043362      013746      000060      MOV      TKVEC+2,-(SP)
      043366      013746      000062      MOV      NOPAR,-(SP)
      043372      013746      002076
10778 043376      010637      002306      MOV      SP,FSSTACK      ;SAVE LAST GOOD STACK POINTER
10779 043402      TYPE      MSG055      ;SOB-A-LONG TEST
      043402      104401      067671      TYPEIT      ,MSG055
      .DSABL      CRF

10780
10781 043406      IF      #SW0 SET.IN @SWR
      043406      032777      000001      137222      BIT      #SW0,@SWR
      043414      001402      BEQ      L421
10782 043416      104470      ECCDIS      ;DISABLE ERROR CORRECTION
10783 043420      ELSE
      043420      000401
      043422      BR      L422
10784 043422      104502      CLRCR      ;CLEAR CSRS
      043424      END ;OF IF      L421:;;;;;
10785 043424
      043424      TYPE      MSG056      ;BELL = EACH PASS COMPLETE
      043424      104401      067712      TYPEIT      ,MSG056
      .DSABL      CRF      L422:;;;;;

10787
10788 043430      TYPE      MSG046      ;TO ESCAPE TYPE ANY KEY!
      043430      104401      067505      TYPEIT      ,MSG046
      .DSABL      CRF

10789 043434      012737      043560      000060      MOV      #CMD7C,TKVEC
10790 043442      012737      000340      000062      MOV      #340,TKVEC+2
10791 043450      017700      137170      MOV      @TKB,RO
10792 043454      042737      000200      177776      BIC      #BIT7,PSW
10793 043462      052777      000100      137152      BIS      #BIT6,@TKS
10794
10795
10796 043470      SET      HEADER,MUT
      043470      012737      177777      002612      MOV      #-1,HEADER
      043476      012737      177777      002110      MOV      #-1,MUT

10797
10798 043504      CMD7B: FOR BANK := #0 TO LASTBANK
      043504      005037      002102
      043510      CLR      BANK
10799 043510      004737      040052      CALL      EXBANK
10800 043514      IF      ACFLAG IS TRUE AND RRFLAG IS FALSE
      043514      005737      002116      B71:;;;;;
      043520      001406
      043522      005737      002124      TST      ACFLAG
      043526      001003      BEQ      L423
10801 043530      104511      INVALIDATE
10802 043532      004737      022454      CALL      MT0031
10803 043536      END ;OF IF ACFLAG
      043536      TST      RRFLAG
10804 043536      END ;OF FOR BANK      BNE      L423
      L423:;;;;;

```


10817 043644

FSCMD8: SUBTST <<FS COMMAND 8 ERROR SUMMARY>>

;*****
;*SUBTEST FS COMMAND 8 ERROR SUMMARY
;*****

PUSH R0,R2,R3,BANK

MOV R0,-(SP)
MOV R2,-(SP)
MOV R3,-(SP)
MOV BANK,-(SP)

10818 043644

043644 010046

043646 010246

043650 010346

043652 013746 002102

10819 043656 013737 056654 002434

10820 043664 005337 002434

10821 043670

043670 013746 002434

043674 104405

MOV \$PASS,TEMP

DEC TEMP

TYPDEC TEMP

MOV TEMP,-(SP)

;;SAVE TEMP FOR TYPEOUT

TYPDS

;;GO TYPE--DECIMAL ASCII WITH SIGN

.DSABL CRF

MOV KIPAR4,SAV4

;;;I.L.C.;REV B

MOV #1000,KIPAR4

;;;I.L.C.;REV B

TYPE MSG125

;PASSES COMPLETED

TYPEIT ,MSG125

.DSABL CRF

TYPDEC \$ERTTL

MOV \$ERTTL,-(SP)

;;SAVE \$ERTTL FOR TYPEOUT

TYPDS

;;GO TYPE--DECIMAL ASCII WITH SIGN

.DSABL CRF

TYPE MSG079

;ERROR(S) DETECTED

TYPEIT ,MSG079

.DSABL CRF

IF \$ERTTL NE #0

TST \$ERTTL

BEQ L424

10822 043676

043730 005737 002630

043734 001445

10828 043736 005037 002334

10829 043742

043742 005037 002102

043746

10830 043746 013703 002102

10831 043752 070327 000004

10832 043756

043756 105763 002666

043762 001424

10833 043764

043764 005737 002334

043770 001005

10834 043772

043772 104401 070230

CLR SUCCESS

FOR BANK := #0 TO LASTBANK

CLR BANK

B72:::;;

MOV BANK,R3

MUL #4,R3

IFB CONFIG*2(R3) NE #0

TSTB CONFIG*2(R3)

BEQ L425

IF SUCCESS IS FALSE

TST SUCCESS

BNE L426

10835 043776

043776 012737 177777 002334

10836 044004

044004

10837 044004

044004 013746 002102

044010 104403

044012 006

044013 000

TYPE MSG076

;BANK ERRORS

TYPEIT ,MSG076

.DSABL CRF

SET SUCCESS

MOV #1,SUCCESS

END ;OF IF SUCCESS

L426:::;;

TYPDCS BANK,3

MOV BANK,-(SP)

;;SAVE BANK FOR TYPEOUT

;;3

TYPDS

;;GO TYPE--OCTAL ASCII

.BYTE 6

;;TYPE 6 DIGITS

.BYTE 0

;;SUPPRESS LEADING ZEROS

.DSABL CRF

MOVB

CONFIG*2(R3),R0

10838 044014 116300 002666

10839	044020	042700	177400	BIC	#C377,R0	
10840	044024			TYPDEC	R0	
	044024	010046		MOV	R0,-(SP)	::SAVE R0 FOR TYPEOUT
	044026	104405		TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN
				.DSABL	CRF	
10841	044030			TYPE	\$CRLF	
	044030	104401	002660	TYPEIT	\$CRLF	
				.DSABL	CRF	
				END	;OF IFB CONFIG(R3)	
10842	044034					L425:::~::~
	044034					
10843	044034			END	;OF FOR BANK	
	044034	005237	002102			INC BANK
	044040	023737	002102			CMP BANK, LASTBANK
	044046	003737				BLE B72
	044050					E72:::~::~
10844	044050			END	;OF IF \$ERTTL	
	044050					L424:::~::~
10845	044050	013737	002270	MOV	SAV4,KIPAR4	::I.L.C.:::REV B
10846	044056			POP	BANK,R3,R2,R0	
	044056	012637	002102			MOV (SP)+,BANK
	044062	012603				MOV (SP)+,R3
	044064	012602				MOV (SP)+,R2
	044066	012600				MOV (SP)+,R0
10847	044070	000207		RETURN		

```

10850 044072      FSCMD9: SUBTST <<FS      COMMAND 9      REFRESH TEST>>
;*****
;SUBTEST      FS      COMMAND 9      REFRESH TEST
;*****
      PUSH      BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
10851 044072      MOV      SP,FSSTACK      ;SAVE LAST GOOD STACK POINTER
      044072 013746 002102
      044076 013746 002112      MOV BANK,-(SP)
      044102 013746 000060      MOV PATTERN,-(SP)
      044106 013746 000062      MOV TKVEC,(SP)
      044112 013746 002076      MOV TKVEC+2,-(SP)
10852 044116 010637 002306      MOV      SP,FSSTACK      ;SAVE LAST GOOD STACK POINTER
10853 044122      TYPE      MSG073      ;REFRESH TEST
      044122 104401 070160      TYPEIT   ,MSG073
      .DSABL   CRF
10854
10855 044126      IF #SW0 SET.IN @SWR
      044126 032777 000001 136502      BIT #SW0,@SWR
      044134 001402      BEQ L427
10856 044136 104470      ECCDIS      ;DISABLE ERROR CORRECTION
10857 044140      ELSE
      044140 000401      BR L430
      044142      L427:;;;;;
10858 044142 104502      CLRCR      ;CLEAR CSRS
10859 044144      END ;OF IF      L430:;;;;;
      044144      TYPE      MSG056      ;BELL = EACH PASS COMPLETE
      044144 104401 067712      TYPEIT   ,MSG056
      .DSABL   CRF
10861
10862 044150      TYPE      MSG046      ;TO ESCAPE TYPE ANY KEY!
      044150 104401 067505      TYPEIT   ,MSG046
      .DSABL   CRF
10863 044154 012737 044300 000060      MOV      @CMD9C,TKVEC
10864 044162 012737 000340 000062      MOV      340,TKVEC+2
10865 044170 017700 136450      MOV      $TKB,RO
10866 044174 042737 000200 177776      BIC      $JIT7,PSW
10867 044202 052777 000100 136432      BIS      @BIT6,@TKS
10868
10869 044210      SET      HEADER,MUT
      044210 012737 177777 002612      MOV      #-1,HEADER
      044216 012737 177777 002110      MOV      #-1,MUT
10870
10871 044224      CMD9B: FOR BANK := #0 TO LASTBANK
      044224 005037 002102      CLR BANK
      044230      B73:;;;;;
10872 044230 004737 040052      CALL EXBANK
10873 044234      IF ACFLAG IS TRUE AND RRFLAG IS FALSE
      044234 005737 002116      TST ACFLAG
      044240 001406      BEQ L431
      044242 005737 002124      TST RRFLAG
      044246 001003      BNE L431
10874 044250 104511      INVALIDATE
10875 044252 004737 021026      CALL MT0022
10876 044256      END ;OF IF ACFLAG
      044256      L431:;;;;;
10877 044256      END ;OF FOR BANK
      044256 005237 002102      INC BANK

```



```

044262 023737 002102 002556          CMP BANK, LASTBANK
044270 003757          BLE B73
044272          E73: :::::
10878 044272          TYPE $BELL          ;RING BELL
044272 104401 002653  TYPEIT $BELL
          .DSABL CRF
10879 044276          GOTO CMD98
044276 000752          BR CMD98
10880
10881 044300 013706 002306          CMD9C: MOV FSSTACK, SP          ;RECOVER OLD STACK POINTER
10882 044304 042777 000100 136330 BIC #BIT6, @TKS
10883 044312 117700 136326          MOV @TKB, R0          ;READ CHAR TO KILL FLAG
10884 044316          POP NOPAR, TKVEC+2, TKVEC, PATTERN, BANK
044316 012637 002076          MOV (SP)+, NOPAR
044322 012637 000062          MOV (SP)+, TKVEC+2
044326 012637 000060          MOV (SP)+, TKVEC
044332 012637 002112          MOV (SP)+, PATTERN
044336 012637 002102          MOV (SP)+, BANK
10885 044342          MAP BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
044342 010346          MOV BANK, R3
044344 013703 002102          CALL MAPPER
044350 004737 035716          .DSABL CRF
          MOV R3, -(SP)
044354 012603          MOV (SP)+, R3
10886 044356 004737 040052          CALL EXBANK
10887 044362 000207          RETURN
10888

```

```

10891 044364      FCMD10: SUBTST <<FS      COMMAND 10      SET FILL COUNT>>
;*****
;*SUBTEST      FS      COMMAND 10      SET FILL COUNT
;*****
10892 044364      PUSH      R0
044364 010046
10893 044366      TYPE      MSG085      ;FILL COUNT(OCTAL)?
044366 104401 070311      TYPEIT    ,MSG085
;*****
10894 044372      .DSABL    CRF
10895 044374      RDOCT
044374 012600      POP      R0
10896 044376      BIC      #1C17,R0
10897 044402      MOV      R0,$FILLS
10898 044406      POP      R0
044406 012600
10899 044410      RETURN
000207
10900
10901 044412      FCMD11: SUBTST <<FS      COMMAND 11      ENTER KAMIKAZE MODE>>
;*****
;*SUBTEST      FS      COMMAND 11      ENTER KAMIKAZE MODE
;*****
10902 044412      TYPE      MSG101      ;ENTERING KAMIKAZE MODE
044412 104401 070456      TYPEIT    ,MSG101
;*****
10903 044416      .DSABL    CRF
044416 012737 177777 002006      SET      KAMIKAZE,SKIPKAMI
044424 012737 177777 002010
10904 044432      RETURN
000207
10905
10906 044434      FCMD12: SUBTST <<FS      COMMAND 12      EXIT KAMIKAZE MODE>>
;*****
;*SUBTEST      FS      COMMAND 12      EXIT KAMIKAZE MODE
;*****
10907 044434      TYPE      MSG102      ;LEAVING KAMIKAZE MODE
044434 104401 070506      TYPEIT    ,MSG102
;*****
10908 044440      .DSABL    CRF
10909 044444      CLR      KAMIKAZE
044444 012737 177777 002010      SET      SKIPKAMI
10910 044452      RETURN
000207
10911
10912 044454      FCMD13: SUBTST <<FS      COMMAND 13      TURN CACHE OFF>>
;*****
;*SUBTEST      FS      COMMAND 13      TURN CACHE OFF
;*****
10913 044454      TYPE      MSG106      ;CACHE IS OFF
044454 104401 070634      TYPEIT    ,MSG106
;*****
10914 044460      .DSABL    CRF
10915 044462      CACHOFF
013737 002544 002546      MOV      CACHKN,CACHKN+2      ;TURN CACHE OFF
10916 044470      CLR      CACHKN      ;SAVE OLD CACHE ON STATE
005037 002544      RETURN      ;KEEP CACHE OFF
10917 044474
10918
10919 044476      FCMD14: SUBTST <<FS      COMMAND 14      TURN CACHE ON>>
;*****
;*SUBTEST      FS      COMMAND 14      TURN CACHE ON
;*****

```

```
10920 044476 104401 070652
      044476
10921 044502 013737 002546 002544
10922 044510 104423
10923 044512 000207
10924
```

```
*****
TYPE MSG107
TYPEIT ,MSG107
.DSABL CRF
MOV CACHKN+2,CACHKN
CACHON
RETURN
;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
;RESTORE OLD CACHE ON STATE
;TURN CACHE ON
```

```

10937
10938 044514
FCMD15: SUBTST <<FS COMMAND 15 TEST ONLY SELECTED BANKS>>
;*****
;*SUBTEST FS COMMAND 15 TEST ONLY SELECTED BANKS
;*****
10939 044514 104401 070561
044514 TYPE MSG105 ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
044514 TYPEIT ,MSG105
.DSABL CRF
10940 044520 004737 044610
10941 044524
044524 CALL CMD16A ;ERASE OLD SELECTIONS
044524 BEGIN CMD16LOOP
10942 044524
044524 REPEAT
10943 044524 104401 067072
044524 TYPE MSG030 ;BANK(0-177)?
TYPEIT ,MSG030
.DSABL CRF
10944 044530 104413
10945 044532 012601
044532 RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
044534 IF R1 GT #177 OR R1 LT #0 ;PUT IT IN R1
044534 020127 000177 MOV (SP)+,R1
044540 003002 CMP R1,#177
044542 005701 BGT L432
044544 002001 TST R1
044546 BGE L433
10947 044546
044546 LEAVE CMD16LOOP L432:;;;;;
10948 044550 000406
044550 END ;OF IF R1 BR E74
10949 044550 006301
10950 044552 006301
10951 044554 052761 040000 002666 ASL R1
ASL R1 ;R1 <- R1 * 4
10952 044562 000760 BIS #BIT14,CONFIG+2(R1)
END ;OF REPEAT
10953 044564
044564 END CMD16LOOP
10954 044564 104401 070727
044564 TYPE MSG110 ;ONLY SELECTED BANKS WILL BE TESTED
TYPEIT ,MSG110
.DSABL CRF
10955 044570 012737 177777 002002 SET SEL ONLY
10956 044576 000207 RETURN
10957
10958 044600
FCMD16: SUBTST <<FS COMMAND 16 RESUME TESTING ALL BANKS>>
;*****
;*SUBTEST FS COMMAND 16 RESUME TESTING ALL BANKS
;*****
10959 044600 104401 070773
044600 TYPE MSG111 ;ALL BANKS WILL BE TESTED
TYPEIT ,MSG111
.DSABL CRF
10960 044604 005037 002002 CLR SELONLY
10961
10962
10963 044610 013702 002556 ;ENTRY POINT FROM CMD15
10964 044614 006302 CMD16A: MOV LASTBANK,R2
10965 044616 006302 ASL R2
ASL R2

```

10966 044620 005001 FOR R1 := #0 TO R2 BY #4
044620
044622
10967 044622 042761 040000 002666 BIC #BIT14,CONFIG+2(R1)
10968 044630 062701 000004 END ;OF FOR R1
044630
044634 020102
044636 003771
044640
10969 044640 000207 RETURN

CLR R1
B76:::~::~

ADD #4,R1
CMP R1,R2
BLE B76
E76:::~::~

10972 044642

FCMD17: SUBTST <<FS COMMAND 17 ENABLE TRACE>>

;*****

;*SUBTEST FS COMMAND 17 ENABLE TRACE

;*****

10973 044642

TYPE MSG127

044642 104401 071331

TYPEIT ,MSG127

.DSABL CRF

10974 044646 012737 177777 006216

MOV #-1,TRACE

10975 044654 000207

RETURN

10978 044656

```
FCMD18: SUBTST <<FS COMMAND 18 DISABLE TRACE>>
;*****
;*SUBTEST FS COMMAND 18 DISABLE TRACE
;*****
TYPE MSG128
TYPEIT ,MSG128
.DSABL CRF
CLR TRACE
RETURN
```

10979 044656 104401 071350

10980 044662 005037 006216

10981 044666 000207

10984 044670

WHICHCSR:SUBTST <<SUBR DETERMINE CORRECT CSR>>

;*****
;*SUBTEST SUBR DETERMINE CORRECT CSR
;*****10985 044670 013700 002224
10986 044674 022700 100000
10987 044700 001003
10988 044702 005037 002152
10989 044706 000207MOV TOTCSR,R0 ;GET CSR'S FLAG
CMP #BIT15,R0 ;CSR 0?
BNE 1\$;NO - SKIP
CLR CSRNO ;YES - SET IT UP
RETURN10990
10991 044710
044710 104401 0667251\$: TYPE MSG022 ;WHICH CSR(0-F)
TYPEIT ,MSG022
.DSABL CRF10992 044714 104412
10993 044716RDLIN ;GET CHARACTER
POP R0 ;PUT IN R0044716 012600
10994 044720 011000
10995 044722 020027 000106MOV (R0),R0 ;PUT CHAR IN R0
CMP R0,#106 ;CHECK LIMIT
BHI 1\$;IF BAD LOOP TILL HE TYPES IT RIGHT10996 044726 101370
10997 044730 022700 000101

CMP #'A,R0

10998 044734 103002
10999 044736 162700 000007

BHS 2\$

11000 044742 162700 000060

2\$: SUB #7,R0

11001 044746 006300

ASL R0

11002 044750 010037 002152

MOV R0,CSRNO

11003 044754 000207

RETURN


```

11469 .SBTTL ERROR DATA (SUPERVISOR) SETUP STUFF
11470 044756 $PER25: LET ADDRESS := R1 - #2
      044756 010137 002034
      044762 162737 000002 002034
11471 044770 IF ABORTFLAG IS FALSE
      044770 005737 002144
      044774 001007
11472 044776 TESTAREA ;ENTER TEST MODE
      044776 053737 002552 177776 BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      LET BAD := -2(R1)
11473 045004 MOV -2(R1),BAD
      045004 016137 177776 002052
11474 045012 KERNEL ;ENTER KERNEL MODE
      104417
11475 045014 END ;OF IF ABORTFLAG
      045014
11476 045014 IF 177654 EQ #0 L434:;;;;;
      045014 005737 177654 TST 177654
      045020 001003 BNE L435
11477 045022 LET GOOD := R2
      045022 010237 002044 MOV R2,GOOD
11478 045026 ELSE
      045026 000402 BR L436
      045030 L435:;;;;;
11479 045030 LET GOOD := R3
      045030 010337 002044 MOV R3,GOOD
11480 045034 END ;OF IF
      045034 L436:;;;;;
11481 045034 JMP PERRAW
      000137 050212
11482
11483 045040 PERRA3: SUBTST <<DATA WAS 3 WORDS>>
      ;*****
      ;*SUBTEST DATA WAS 3 WORDS
      ;*****
      IF BADPC EQ #0 THEN $CALL BADSTACK
11484 045040 TST BADPC
      045040 005737 002022 BNE L437
      045044 001002 JSR PC,BADSTACK
      045046 004737 034152 L437:;;;;;
      045052
11485 045052 PUSH R0
      045052 010046 MOV R0,-(SP)
11486 045054 CLR CSR ;MAKE SURE CSR BIT HOLDER IS CLEAR
      005037 002150 ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
11487 045060 CHK1DIS
      104505
11488 045062 TESTAREA
      045062 053737 002552 177776 BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      TST (R1) ;READ LOCATION TO READ CHECKBITS INTO CSR
11489 045070 KERNEL
      005711
11490 045072 READCSR ;GET CSR CONTENTS
      104417 ;SAVE CSR CONTENTS IN R0
11491 045074 MOV CSR,R0 ;RETURN CSR TO NORMAL MODE
      013700 002150 ;MOVE CHECK BITS TO BOTTOM OF WORD
11492 045076 CLR1CSR ;CLEAR OFF EXTRANEIOUS GARBAGE
      104503 ;SAVE VIRTUAL ADDRESS FOR PRINTOUT
11493 045102 ASH #-5,R0
      072027 177773
11494 045104 BIC #+C177,R0
      042700 177600
11495 045110 LET ADDRESS := R1
      045114
11496 045114 MOV R1,ADDRESS
      010137 002034
11497 045120 CLR GOOD ;FIRST TEST WORD WRITTEN SHOULD ALWAYS BE ZERO
      005037 002044
11498 045124 TESTAREA
      045124 053737 002552 177776 BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE

```

11499	045132	011137	002052	.DSABL	CRF	
11500	045136	011437	002054	MOV	(R1),BAD	;GET BAD DATA FROM MUT - FIRST WORD
11501	045142	104417		MOV	(R4),BAD2	;AND SECOND WORD
11502	045144	110037	002056	KERNEL		;ENTER KERNEL MODE
11503	045150	105037	002057	MOVB	R0,BAD3	;MOVE BAD CHECKBITS FOR PRINTOUT
11504	045154	004737	050446	CLRB	BAD3+1	;CLEAR OFF THE OTHER UNUSED BITS
11505	045160	104033		CALL	PERBNK	;MARK BANK AS BAD IN CONFIG TABLE
11506	045162			ERROR	+33	
	045162	012600		POP	R0	;RESTORE R0
11507	045164					MOV (SP),R0
	045164	032777	000001 135444	IF #SW0 SET.IN @SWR		BIT #SW0,@SWR
	045172	001402				BEQ L440
11508	045174	104506		ENASBE		;TRAP ON SINGLE BIT ERRORS
11509	045176			ELSE		
	045176	000401				BR L441
	045200					L440:::;;
11510	045200	104472		ECCINIT		;TRAP ON UNCORRECTABLE ERRORS
11511	045202			END; OF IF #SW0		
	045202					L441:::;;
11512	045202	000002		RTI		

```

11515 045204          $PER30: LET GOOD := R1
      045204 010137 002044
11516 045210          LET ADDRESS := (SP) - 16
      045210 011637 002034
      045214 163737 000016 002034
11517 045222          IF ABORTFLAG IS FALSE
      045222 005737 002144
      045226 001007
11518 045230          TESTAREA          ;ENTER TEST MODE
      045230 053737 002552 177776      BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      LET BAD := @ADDRESS
11519 045236          MOV @ADDRESS,BAD
      045236 017737 134572 002052
11520 045244          KERNEL          ;ENTER KERNEL MODE
      104417
11521 045246          END ;OF IF ABORTFLAG
      045246
11522 045246          JMP PERRAW          L442:;;;;;;
      000137 050212
11523
11524 045252          GETDATA:SUBTST <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
      ;*****
      ;*SUBTEST          GET DATA FROM ABORTED AREA IF POSSIBLE
      ;*****
      PUSH          R0,4,114
11525 045252          MOV R0,-(SP)
      045252 010046          MOV 4,-(SP)
      045254 013746 000004          MOV 114,(SP)
      045260 013746 000114
11526 045264          MOV SP,GETDA1
      010637 045350
11527 045270          MOV #14,4
      012737 045330 000004
11528 045276          MOV #14,114
      012737 045330 000114
11529 045304          MOV ADDRESS,R0
      013700 002034
11530 045310          TESTAREA
      045310 053737 002552 177776      BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      MOV (R0),BAD
11531 045316          KERNEL
      011037 002052
11532 045322          CLR ABORTFLAG
      104417
11533 045324          MOV GETDA1,SP          ;RESTORE KNOWN GOOD STACK POINTER
      005037 002144
11534 045330          POP 114,4,R0
      013706 045350
11535 045334          MOV (SP),114
      045334 012637 000114
      045340 012637 000004
      045344 012600
11536 045346          RETURN
      000207
11537 045350          GETDA1: 0
      000000

```

```

11540          .SBTTL POWER FAIL AUTO RESTART
11541          .SBTTL ROUTINE POWER DOWN AND UP
11542          ;*****
11543          ;POWER DOWN ROUTINE
11544 045352      $PWRDN:
11552          ;SAVE CACHE STATUS
11553 045352 005737 002544      TST CACHKN
11554 045356 001403          BEQ 5$
11555 045360          PUSH CONTRL
11556 045360 013746 177746          MOV CONTRL,-(SP)
11557 045364 104423          CACHON          ;TURN CACHE ON
11558 045374 012737 046300 000024 5$:      MOV #ILLUP,PWRVEC ;SET FOR FAST UP
11559 045402 012737 000340 000026          MOV #340,PWRVEC+2 ;:PRIO:7
11559 045402          PUSH R0,R1,R2,R3,R4,R5,CSRNO
11559 045402 010046          MOV R0,(SP)
11559 045404 010146          MOV R1,-(SP)
11559 045406 010246          MOV R2,-(SP)
11559 045410 010346          MOV R3,-(SP)
11559 045412 010446          MOV R4,-(SP)
11559 045414 010546          MOV R5,-(SP)
11559 045416 013746 002152          MOV CSRNO,-(SP)
11560          ;SAVE USER PAR'S & PDR7
11561 045422 012700 177700      MOV #177700,R0
11562 045426 012701 000021      MOV #17.,R1
11563 045432          1$:      PUSH -(R0)
11564 045432 014046          MOV -(R0),-(SP)
11564 045434 077102          SOB R1,1$
11565          ;SAVE SUPERVISOR PAR'S
11566 045436 005737 002456      TST NOSUPER
11567 045442 001013          BNE PD1
11568 045444 012700 172300      MOV #172300,R0
11569 045450 012701 000020      MOV #16.,R1
11570 045454          2$:      PUSH -(R0)
11570 045454 014046          MOV -(R0),-(SP)
11571 045456 077102          SOB R1,2$
11572 045460          IF RLFLAG IS TRUE THEN $CALL WOOPS
11572 045460 005737 002126          TST RLFLAG
11572 045464 001402          BEQ L443
11572 045466 004737 046306          JSR PC,WOOPS
11572 045472          L443:::
11573          ;COPY KERNEL MAP TO USER & SUPERVISOR
11574 045472 012700 172300      PD1:      MOV #KIPDR0,R0
11575 045476 012701 177600      MOV #UIPDR0,R1
11576 045502 012702 172200      MOV #SIPDR0,R2
11577 045506 012703 000040      MOV #32.,R3
11578 045512 011021          3$:      MOV (R0),(R1)+
11579 045514 012022          MOV (R0)+,(R2)+
11580 045516 077303          SOB R3,3$

```

```

11582 ;SAVE USER & SUPERVISOR STACK POINTERS
11583 045520 USER
      045520 052737 140000 177776 BIS #BIT15:BIT14,PSW ;GO TO USER MODE
      .DSABL CRF
      MOV USP,R0
      KERNEL ;ENTER KERNEL MODE
      PUSH R0
      MOV R0,(SP)
11584 045526 010600
11585 045530 104417
11586 045532 010046
      045532 005737 002456 TST NOSUPER
11587 045534 001006 BNE 7$
11588 045540 001006 SUPERVISOR ;ENTER SUPERVISOR MODE
11589 045542 052737 040000 177776 BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
      .DSABL CRF
      MOV SSP,R0
      KERNEL ;ENTER KERNEL MODE
      PUSH R0
      MOV R0,--(SP)
11593 ;SAVE ECC REGISTERS
11594 045556 013701 002224 7$: MOV TOTCSRS,R1 ;GET CSR'S
11595 045562 BEGIN LCSRSAVE
      045562
      045562 005037 002152 FOR CSRNO := #0 TO #36 BY #2
      045566
11596 045562 006301
      045562 ASL R1
11597 045566 103003 ON.ERROR
11598 045570 104426 READCSR
      045570 103003 PUSH CSR
11599 045572 104426
11600 045574 013746 002150 END ;OF ON.ERROR
      045574
11601 045600 IF R1 EQ #0 THEN LEAVE LCSRSAVE
      045600
11602 045600 005701
      045600 001407
11603 045604 062737 000002 002152
      045604 023727 002152 000036
      045612
      045620 003762
      045622
11604 045622 END LCSRSAVE
      045622
11605 ;SAVE MMRO,1,2,3
11606 045622 PUSH MMRO,MMR1,MMR2
      045622 013746 177572
      045626 013746 177574
      045632 013746 177576
11607 045636 005737 002456 TST NOSUPER
11608 045642 001002 BNE 8$
11609 045644 013746 172516 PUSH MMR3
      045644
11610 ;SAVE KERNEL PAR'S
11611 045650 012700 172400 8$: MOV #172400,R0
11612 045654 012701 000020 MOV #16.,R1
11613 045660 014046 4$: PUSH -(R0)
      045660
11614 045662 077102 SOB R1,4$
      MOV (R0),(SP)

```

```

11615                                     ;SAVE Q-BUS MAP REGISTERS
11616 045664 022737 000005 004010      CMP      #5,PROTYP      ;IS THIS AN 11/83 ?
11617 045672 001004                      BNE      9$           ;BRANCH IF NOT
11618 045674                      PUSH     MAPHO,MAPLO
                                045674 013746 170202          MOV MAPHO,-(SP)
                                045700 013746 170200          MOV MAPLO,-(SP)
11619                                     ;SAVE POSSIBLE SOFTWARE SWITCH REGISTER
11620 045704                      9$:    PUSH     @SWR
                                045704 017746 134726          MOV @SWR,-(SP)
11621                                     ;SAVE STACK POINTER
11622 045710 010637 046304      MOV      SP,$SAVR6      ;;SAVE SP
11623                                     ;NOW SET UP REAL VECTOR
11624 045714 012737 045726 000024      MOV      #$PWRUP,PWRVEC ;;SET UP VECTOR
11625 045722 000000                      $DOWN:  HALT
11626 045724 000776                      BR       $DOWN      ;;HANG UP

```

```

11629 ;*****
11630 ;POWER UP ROUTINE
11631 045726 $PWRUP:
11635 045726 012737 046300 000024 MOV    $$ILLUP,PWRVEC ;;SET FOR FAST DOWN
11636 ;RESTORE STACK POINTER
11637 045734 013706 046304 MOV    $SAVR6,SP      ;;GET SP
11638 045740 005037 046304 CLR     $SAVR6        ;;WAIT LOOP FOR THE TTY
11639 045744 005237 046304 1$: INC    $SAVR6        ;;WAIT FOR THE INC
11640 045750 001375 BNE     1$           ;;OF A WORD
11641 ;RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
11642 045752 POP     @SWR
11643 045752 012677 134660 MOV    (SP)+,@SWR
11644 045756 012700 172340 10$: MOV    #172340,R0
11645 045762 012702 172300 MOV    #KIPDR0,R2
11646 045772 012701 000020 6$: MOV    #16.,R1
11647 045772 012620 POP    (R0)+
11648 045774 012722 077406 MOV    #77406,(R2)+
11649 046000 077104 SOB     R1,6$
11650 046002 005737 002456 ;RESTORE MMR3,2,1,0
11651 046006 001002 TST     NOSUPER
11652 046010 012637 172516 BNE     11$
11653 046014 012637 177576 POP     MMR3
11654 046014 012637 177574 MOV    (SP)+,MMR3
11655 046014 012637 177574 MOV    (SP)+,MMR2
11656 046014 012637 177574 MOV    (SP)+,MMR1
11657 046014 012637 177574 MOV    (SP)+,MMR0
11658 ;RESTORE ECC REGISTERS
11659 046030 013701 002224 MOV    TOTCSRS,R1    ;GET CSR'S
11660 046034 042701 177400 BIC     #177400,R1
11661 046040 BEGIN LCSRRESTORE
11662 046040 FOR CSRNO := #36 DOWNT0 #0 BY #2
11663 046040 012737 000036 002152 ASR     R1
11664 046046 006201 ON.ERROR
11665 046050 103003 POP     CSR
11666 046052 012637 002150 LOADCSR
11667 046056 104425 END ;OF ON.ERROR
11668 046060 IF R1 EQ #0 THEN LEAVE LCSRRESTORE
11669 046060 005701 TST    R1
11670 046062 001407 BEQ    E101
11671 046064 162737 000002 002152 END ;OF FOR CSRNO
11672 046072 023727 002152 000000 SUB    #2,CSRNO
11673 046100 002362 CMP    CSRNO,#0
11674 046102 BGE    B102
11675 046102 E102:
11676 046102 E101:
11677 046102 END LCSRRESTORE
11678 ;COPY KERNEL MAP TO USER & SUPERVISOR
11679 046102 012700 172300 MOV    #KIPDR0,R0
11680 046106 012701 177600 MOV    #UIPDR0,R1

```

CVMJAAO MSV11-J MEMORY DIAG.
ROUTINE POWER DOWN AND UP

MACRO V05.01b Friday 15-Feb-85 16:10 Page 384 1

SEQ 0342

11670	046112	012702	172200		MOV	#SIPDR0,R2
11671	046116	012703	000040		MOV	#32.,R3
11672	046122	011021		3\$:	MOV	(R0),(R1)+
11673	046124	012022			MOV	(R0)+,(R2)+
11674	046126	077303			SOB	R3,3\$


```

11676 ;RESTORE SUPERVISOR & USER STACK POINTERS
11677 046130 005737 002456 TST NOSUPER
11678 046134 001006 BNE 13$
11679 046136 POP RO
                                MOV (SP)+,RO
11680 046136 012600
                                SUPERVISOR ;ENTER SUPERVISOR MODE
                                BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
                                .DSABL CRF
                                MOV RO,SSP
                                KERNEL ;ENTER KERNEL MODE
                                POP RO
                                MOV (SP)+,RO
11681 046146 010006
11682 046150 104417
11683 046152 012600 13$:
11684 046154 052737 140000 177776 USER
                                BIS #BIT15!BIT14,PSW ;GO TO USER MODE
                                .DSABL CRF
                                MOV RO,USP
                                KERNEL ;ENTER KERNEL MODE
                                ;RESTORE SUPERVISOR PAR'S
                                MOV #172240,RO
                                MOV #16.,R1
                                POP (RO)+
                                MOV (SP)+,(RO)+
11691 046200 077102 SOB R1,7$
11692 ;RESTORE USER PAR'S & PDR7
11693 046202 012700 177636 MOV #177636,RO
11694 046206 012701 000021 MOV #17.,R1
11695 046212 012620 8$:
                                POP (RO)+
                                MOV (SP)+,(RO)+
11696 046214 077102 SOB R1,8$
11697 ;RESTORE POSSIBLE SOFTWARE DISPLAY REGISTER
11698 046216 013777 002012 134414 MOV $PATMAR,@DISPLAY
11699 046224 013737 002012 000174 MOV $PATMAR,DISPREG
11700 046232 012637 002152 POP CSRNO,R5,R4,R3,R2,R1,RO
                                MOV (SP)+,CSRNO
                                MOV (SP)+,R5
                                MOV (SP)+,R4
                                MOV (SP)+,R3
                                MOV (SP)+,R2
                                MOV (SP)+,R1
                                MOV (SP)+,RO
11701 046252 012737 045352 000024 MOV $$PWRDN,PWRVEC ;;SET UP THE POWER DOWN VECTOR
11702 046260 104401 067652 TYPE MSG051 ;REPORT THE POWER FAILURE
                                TYPEIT ,MSG051
                                .DSABL CRF
                                ;RESTORE CACHE STATUS
                                TST CACHKN
                                BEQ 9$
                                POP CONTRL
                                MOV (SP)+,CONTRL
11703
11704 046264 005737 002544
11705 046270 001402
11706 046272 012637 177746
11707 046276 000002 9$:
11708 046300 000000 $ILLUP: HALT
11709 046302 000776 BR $ILLUP
11710 046304 000000 $SAVR6: 0
11711 .EVEN
                                ;;THE POWER UP SEQUENCE WAS STARTED
                                ;; BEFORE THE POWER DOWN WAS COMPLETE
                                ;;PUT THE SP HERE

```

11723 046306

WOOPS: SUBTST <<POWER FAIL WHILE RELOCATED>>

;*****

;SUBTEST POWER FAIL WHILE RELOCATED

;*****

11724 046306

11725 046312 013746 002102

11726 046316 005037 002102

046316 010346

046320 013703 002102

046324 004737 035716

046330 012603

11727 046332 052737 040000 177776

11728 046340 013737 060024 046704

11729 046346 013737 060026 046706

11730 046354 004537 041024

046354 000105

046360 046710

046362 126472

11731 046366 012737 046472 060024

11732 046374 012737 000340 060026

11733 046402 004537 041024

046402 000071

046406 126472

046410 046472

11734 046414 012700 172340

11735 046420 012701 126654

11736 046424 012702 000010

11737 046430 012021

11738 046432 077202

11739 046434 005737 002456

11740 046440 001002

11741 046442 013721 172516

11742 046446 013721 177576

11743 046452 013721 177574

11744 046456 013721 177572

11745 046462 104417

11746 046464 012637 002102

11747 046470 000207

PUSH BANK

MOV BANK,-(SP)

CLR BANK

MAP BANK

;MAP SUPERVISOR SPACE (TEST AREA) TO BANK

MOV R3,-(SP)

MOV BANK,R3

CALL MAPPER

.DSABL CRF

MOV (SP)+,R3

SUPERVISOR

;ENTER SUPERVISOR MODE

BIS #BIT14,PSW

;GO TO SUPERVISOR MODE

.DSABL CRF

MOV FIRST+PWRVEC,WOOPSAV

MOV FIRST+PWRVEC+2,WOOPSAV+2

BMOV FIRST+WOOPUP,WOOPSAV+4,WOOPEND-WOOPUP/2+12.

JSR R5,BLOCK3

WOOPEND-WOOPUP/2+12.

WOOPSAV+4

FIRST+WOOPUP

.DSABL CRF

MOV #WOOPUP,FIRST+PWRVEC

MOV #340,FIRST+PWRVEC+2

BMOV WOOPUP,FIRST+WOOPUP,WOOPEND-WOOPUP/2

JSR R5,BLOCK3

WOOPEND-WOOPUP/2

FIRST+WOOPUP

WOOPUP

.DSABL CRF

MOV #KIPAR0,P0

MOV #FIRST+WOOPEND,R1

MOV #8,R2

MOV (R0)+,(R1)+

SOB R2,1\$

TST NOSUPER

BNE 2\$

MOV MMR3,(R1)+

MOV MMR2,(R1)+

MOV MMR1,(R1)+

MOV MMRO,(R1)+

KERNEL

;ENTER KERNEL MODE

POP BANK

MOV (SP)+,BANK

RETURN

11750 046472

11751 046472 012700 046654
11752 046476 012701 172340
11753 046502 012703 172300
11754 046506 012702 000010
11755 046512 012021
11756 046514 012723 077406
11757 046520 077204
11758 046522 005737 002456
11759 046526 001002
11760 046530 012037 172516
11761 046534 012037 177576
11762 046540 012037 177574
11763 046544 012037 177572
11764 046550 013706 046304
11765 046554
046554 013746 002102
11766 046560 005037 002102
11767 046564
046564 010346
046566 013703 002102
046572 004737 035716

046576 012603
11768 046600
046600 052737 040000 177776

11769 046606 013737 046704 060024
11770 046614 013737 046706 060026
11771
11772
11773 046622 012700 046710
11774 046626 012701 000105
11775 046632 012702 126472
11776 046636 012022
11777 046640 077102
11778
11779 046642 104417
11780 046644
046644 012637 002102
11781 046650 000137 045726
11782 046654 000014
11785 046704 000107

WOOPUP: SUBTST <<POWER UP FROM BANK 0 TO RELOCATION>>

;*****
;*SUBTEST POWER UP FROM BANK 0 TO RELOCATION
;*****

MOV #WOOPEND,R0
MOV #KIPAR0,R1
MOV #KIPDR0,R3
MOV #8.,R2
1\$: MOV (R0)+,(R1)+
MOV #77406,(R3)+
SOB R2,1\$
TST NOSUPER
BNE 3\$
MOV (R0)+,MMR3
3\$: MOV (R0)+,MMR2
MOV (R0)+,MMR1
MOV (R0)+,MMR0
MOV \$SAVR6,SP
PUSH BANK

MOV BANK,-(SP)
CLR BANK
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
MOV R3,(SP)

MOV (SP)+,R3
SUPERVISOR ;ENTER SUPERVISOR MODE
BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
.DSABL CRF
MOV WOOPSAV,FIRST+PWRVEC
MOV WOOPSAV+2,FIRST+PWRVEC+2
;SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
;BMOV WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
MOV #WOOPSAV+4,R0
MOV #WOOPEND-WOOPUP/2+12.,R1
MOV #FIRST+WOOPUP,R2
2\$: MOV (R0)+,(R2)+
SOB R1,2\$

KERNEL ;ENTER KERNEL MODE
POP BANK

MOV (SP)+,BANK
JMP \$PWRUP
WOOPEND: .REPT 12.
WOOPSAV: .REPT WOOPEND WOOPUP/2+12.+2

.SBTTL IO SUBROUTINES

.SBTTL ROUTINE TYPE

```
11790
11791
11792
11793
11794
11795
11796
11797
11798
11799
11800
11801
11802
11803
11804
11805
11806
11807
11808
11809 047122 105737 002360
11810 047126 100407
11811 047130 010046
11812 047132 017600 000002
11813 047136 112046
11814 047140 001005
11815 047142 005726
11816 047144 012600
11817 047146 062716 000002
11818 047152 000002
11819 047154 122716 000011
11820 047160 001002
11821 047162 112716 000040
11822 047166 122716 000200
11823 047172 001006
11824 047174 005726
11825 047176
      047176 104401
11826 047200 002660
11827 047202 105037 047434
11828 047206 000753
11829 047210 004737 047250
11830 047214 123726 002652
11831 047220 001346
11832 047222 013746 002356
11833
11834 047226 105366 000001
11835 047232 002770
11836 047234 004737 047250
11837 047240 105337 047434
11838 047244 000770
11839 047246 000000
11840 047250
      047250 010146
11841 047252 116601 000004
11842 047256 005737 002544
11843 047262 001402

;*****
;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
;*
;*CALL:
;*1) USING A TRAP INSTRUCTION
;*      TYPE      MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;*      TYPE
;*      MESADR
;*
$TYPE:  TSTB      $TPFLG      ;;IS THERE A TERMINAL?
        BMI       6$          ;;BR IF NO
1$:      MOV       R0,-(SP)    ;;SAVE R0
        MOV       @2(SP),R0   ;;GET ADDRESS OF ASCIZ STRING
4$:      MOVB      (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE       7$          ;;BR IF IT ISN'T THE TERMINATOR
        TST       (SP)+       ;;IF TERMINATOR POP IT OFF THE STACK
5$:      MOV       (SP)+,R0    ;;RESTORE R0
6$:      ADD       #2,(SP)     ;;ADJUST RETURN PC
        RTI                    ;;RETURN
7$:      CMPB      #HT,(SP)    ;;BRANCH IF NOT <HT>
        BNE       11$
        MOVB      #'',(SP)    ;;REPLACE TAB WITH SPACE
11$:     CMPB      #CRLF,(SP)  ;;BRANCH IF NOT <CRLF>
        BNE       8$
        TST       (SP)+       ;;POP <CR><LF> EQUIV
                                ;;TYPE A CR AND LF
        TYPEIT
        .DSABL    CRF
        $CRLF
        CLRB      $CHARCNT    ;;CLEAR CHARACTER COUNT
        BR        4$          ;;GET NEXT CHARACTER
8$:      CALL      $TYPEC      ;;GO TYPE THIS CHARACTER
9$:      CMPB      $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
        BNE       4$          ;;IF NO GO GET NEXT CHAR.
        MOV       $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
                                ;;AND THE NULL CHAR.
10$:     DECB      1(SP)       ;;DOES A NULL NEED TO BE TYPED?
        BLT       9$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
        CALL      $TYPEC      ;;GO TYPE A NULL
        DECB      $CHARCNT    ;;DO NOT COUNT AS A COUNT
        BR        10$        ;;LOOP
XOCHAR: .WORD 0
$TYPEC: PUSH      R1
                                MOV R1,-(SP)
        MOVB      4(SP),R1
        TST       CACHKN
        BEQ       2$
```

```

11844 047264          PUSH    CONTRL
11845 047264 013746 177746      2$: PUSH    R0
11846 047270 010046          MOV    CONTRL, -(SP)
11846 047272 104424          MOV    R0, -(SP)
11871 047274 105777 133346      3$: CACHOFF          ;TURN CACHE OFF
11872 047300 100375          TSTB   @TPS          ;WAIT UNTIL PRINTER IS READY
11873 047302 005037 047246      BPL    3$
11874 047306 105777 133330      CLR    XCHAR          ;CHECK FOR XOFF
11875 047312 100032          TSTB   @TKS          ;SKIP IF NO CHARACTER
11876 047314 117737 133324 047246  MOVB   @TKB,XCHAR ;SAVE THE CHARACTER
11877 047322 042737 177600 047246  BIC    #C177,XCHAR ;STRIP OFF ASCII
11878 047330 023727 047246 000023  CMP    XCHAR,#023 ;WAS IT A CONTROL S?
11879 047336 001020          BNE     NC          ;BRANCH IF NOT
11880 047340 105777 133276      CONTS3: TSTB   @TKS          ;WAIT FOR CHARACTER
11881 047344 100375          BPL     CONTS3
11882 047346 117737 133272 047246  MOVB   @TKB,XCHAR ;GET CHARACTER
11883 047354 042737 177600 047246  BIC    #C177,XCHAR ;STRIP OFF ASCII
11884 047362          IF XCHAR EQ #21 ; IF IT IS A +Q
11885 047362 023727 047246 000021  CMP    XCHAR,#21
11885 047370 001002          BNE     L446
11886 047372 000402          BR      NC
11886 047374          ELSE
11887 047376 000401          BR      L447
11887 047376 000760          L446:;;;;;
11888 047400          BR      CONTS3
11888 047400          END ;OF IF XCHAR
11889 047400 110177 133244      NC:   MOVB   R1,@TPB          ;LOAD CHAR TO BE TYPED INTO DATA REG.
11893 047404 122766 000015 000002  CMPB   #CR,2(SP) ;IS CHARACTER A CARRIAGE RETURN?
11894 047412 001003          BNE     1$          ;BRANCH IF NO
11895 047414 105037 047434      CLRB   $CHARCNT ;YES--CLEAR CHARACTER COUNT
11896 047420 000406          BR      $TYPEX ;EXIT
11897 047422 122766 000012 000002  1$: CMPB   #LF,2(SP) ;IS CHARACTER A LINE FEED?
11898 047430 001402          BEQ     $TYPEX ;BRANCH IF YES
11899 047432 105227          INCB   (PC)+ ;COUNT THE CHARACTER
11900 047434 000000          $CHARCNT: .WORD 0 ;CHARACTER COUNT STORAGE
11901 047436          $TYPEX: POP    R0
11902 047436 012600          TST     CACHKN          ;IS THERE A CACHE?
11903 047440 005737 002544      BEQ     2$          ;BRANCH IF NOT
11904 047444 001402          POP     CONTRL ;POP CACHE STATUS
11905 047446 012637 177746      MOV    (SP)+,R0
11905 047452          2$: POP     R1
11905 047452 012601          MOV    (SP)+,R1
11906 047454 000207          RETURN
11907 047456          SUPLIMIT:;!!!!!!!!!!!!!!!!THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE

```

.SBTTL ERROR DATA SETUP

USE THIS IF THIS CONDITION DESCRIBES THE ERROR

PERR01 TRAP
BAD DATA IN R0 UNLESS ABORTED
THEN BAD DATA IS POINTED TO BY -(R4)
GOOD DATA IN R5PERR02 TRAP
BAD DATA IN R1 UNLESS ABORTED
THEN BAD DATA IS POINTED TO BY -(R4)
GOOD DATA IN R2PERR03 TRAP
BAD DATA IS POINTED TO BY -(R1)
GOOD DATA IN R4PERR04 TRAP
BAD DATA IN R4 UNLESS ABORTED
THEN BAD DATA IS POINTED TO BY 2(R0)
GOOD DATA IN R2PERR05 JSR PC
BAD DATA IS POINTED TO BY -(R0)
GOOD DATA IN R2
RETURN AFTER SETTING UP GOOD,BAD,ADDRESSPERR06 JSR PC
BAD DATA IS POINTED TO BY -(R0)
GOOD DATA IS ZERO
RETURN AFTER SETTING UP GOOD,BAD,ADDRESSPERR07 TRAP
BAD DATA IN R2 UNLESS ABORTED
THEN BAD DATA IS POINTED TO BY (R1)
GOOD DATA IN DATBUFPERR10 TRAP
BAD DATA IN R2 UNLESS ABORTED
THEN BAD DATA IS POINTED TO BY 2(R1)
GOOD DATA IN DATBUF+2PERR11 TRAP
BYTE TEST
BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
THEN BAD DATA IS POINTED TO BY (R1)
GOOD DATA IS A ZERO BYTEPERR12 TRAP
BYTE TEST
BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
THEN BAD DATA IS POINTED TO BY (R1)
GOOD DATA IS A BYTE OF ONESPERR13 TRAP
BAD DATA IN R0 UNLESS ABORTED12585
12586
12587
12588
12589
12590
12591
12592
12593
12594
12595
12596
12597
12598
12599
12600
12601
12602
12603
12604
12605
12606
12607
12608
12609
12610
12611
12612
12613
12614
12615
12616
12617
12618
12619
12620
12621
12622
12623
12624
12625
12626
12627
12628
12629
12630
12631
12632
12633
12634
12635
12636
12637
12638
12639
12640
12641

12642	:		THEN BAD DATA IS POINTED TO BY (R1)
12643	:		GOOD DATA IS ZERO
12644	:		
12645	:	PERR14	TRAP
12646	:		BAD DATA IN R0 UNLESS ABORTED
12647	:		THEN BAD DATA IS POINTED TO BY (R1)
12648	:		GOOD DATA IS ONES
12649	:		
12650	:	PERR15	TRAP
12651	:		BAD DATA IN R0 UNLESS ABORTED
12652	:		THEN BAD DATA IS POINTED TO BY (R1)
12653	:		GOOD DATA IN TSTDAT
12654	:		
12655	:	PERR16	TRAP
12656	:		BAD DATA IN R0 UNLESS ABORTED
12657	:		THEN BAD DATA IS POINTED TO BY (R1)
12658	:		GOOD DATA IN TSTDAT+2
12659	:		
12660	:	PERR17	TRAP
12661	:		BAD DATA IN R0 UNLESS ABORTED
12662	:		THEN BAD DATA IS POINTED TO BY (R1)
12663	:		GOOD DATA IN R2
12664	:		
12665	:	PERR20	TRAP
12666	:		BAD DATA IN R0 UNLESS ABORTED
12667	:		THEN BAD DATA IS POINTED TO BY (R1)
12668	:		GOOD DATA IN R3
12669	:		
12670	:	PERR21	TRAP
12671	:		7 BIT BYTE TEST
12672	:		BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
12673	:		THEN BAD DATA IS POINTED TO BY (R1)
12674	:		GOOD DATA IS A 7 BIT BYTE ON ONES
12675	:		
12676	:	PERR22	TRAP
12677	:		BAD DATA IN R2 UNLESS ABORTED
12678	:		THEN BAD DATA IS POINTED TO BY (R1)
12679	:		GOOD DATA IN R0
12680	:		
12681	:	PERR23	TRAP
12682	:		BAD DATA IN R0 UNLESS ABORTED
12683	:		THEN BAD DATA IS POINTED TO BY (R1)
12684	:		GOOD DATA IN R4
12685	:		
12686	:	PERR24	TRAP
12687	:		BAD DATA IN R0 UNLESS ABORTED
12688	:		THEN BAD DATA IS POINTED TO BY (R2)
12689	:		GOOD DATA IN R3
12690	:		
12691	:	PERR25	TRAP
12692	:		BAD DATA POINTED TO BY -(R1)
12693	:		GOOD DATA IN R2 UNLESS LOC V177654 IS SET
12694	:		THEN GOOD DATA IS IN R3
12695	:		
12696	:	PERR26	TRAP
12697	:		BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
12698	:		GOOD DATA IS 000000,,100000,,100

```

12699      ;
12700      ; PERR27      TRAP
12701      ;           BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
12702      ;           GOOD DATA IS 000000,,000000,,077
12703      ;
12704      ; PERR30      TRAP
12705      ;           BAD DATA IS POINTED TO BY -16(SP)
12706      ;           GOOD DATA IS IN R1
12707      ;
12708      ; PERR31      TRAP
12709      ;           SPECIAL ECC FAILURE HANDLER
12710      ;
12711      ; PERR32      TRAP
12712      ;           SPECIAL ECC FAILURE HANDLER
12713      ;
12714      ; PERR33      TRAP
12715      ;           SPECIAL ECC FAILURE HANDLER
12716      ;
12717      ; PERR34      TRAP
12718      ;           SPECIAL ECC FAILURE HANDLER
12719      ;
12720      ; PERR35      TRAP
12721      ;           SPECIAL BRANCH GOBBLE FAILURE HANDLER
12722      ;
12723      ;           CALLING SEQUENCE FOR TRAP TYPES
12724      ; BEQ      2$           ;NO - ERROR,BRANCH FOR CARD
12725      ; PERRXX           ;TRAP TO ERROR ROUTINE
12726      ;2$: NEXT INSTRUCTION ;CONTINUE TESTING

```


12729	047454	010437	002034		\$PER01: MOV R4, ADDRESS	
12730	047462	162737	000002	002034	SUB #2, ADDRESS	
12731	047470	010037	002052		MOV R0, BAD	
12732	047474	010537	002044		MOV R5, GOOD	
12733	047500	000137	050212		JMP PERRAW	
12734						
12735	047504	010437	002034		\$PER02: MOV R4, ADDRESS	
12736	047510	162737	000002	002034	SUB #2, ADDRESS	
12737	047516	010137	002052		MOV R1, BAD	
12738	047522	010237	002044		MOV R2, GOOD	
12739	047526	000137	050212		JMP PERRAW	
12740						
12741	047532	010137	002034		\$PER03: MOV R1, ADDRESS	
12742	047536	162737	000002	002034	SUB #2, ADDRESS	
12743	047544	010437	002044		MOV R4, GOOD	
12744	047550	016137	177776	002052	MOV -2(R1), BAD	
12745	047556	000137	050212		JMP PERRAW	
12746						
12747	047562	010037	002034		\$PER04: MOV R0, ADDRESS	
12748	047566	162737	000002	002034	SUB #2, ADDRESS	
12749	047574	010437	002052		MOV R4, BAD	
12750	047600	010237	002044		MOV R2, GOOD	
12751	047604	000137	050212		JMP PERRAW	
12752						
12753	047610	010237	002044		PERR05: MOV R2, GOOD	
12754	047614	014037	002052		PERR05: MOV -(R0), BAD	
12755	047620	010037	002034		MOV R0, ADDRESS	
12756	047624	062700	000002		ADD #2, R0	;RESTORE R0
12757	047630	004737	034152		CALL BADSTACK	
12758	047634	000207			RETURN	
12759						
12760	047636	005037	002044		PERR06: CLR GOOD	
12761	047642	000764			BR PERR05	
12762						
12763	047644	010137	002034		\$PER07: MOV R1, ADDRESS	
12764	047650	010237	002052		MOV R2, BAD	
12765	047654	013737	002242	002044	MOV DATBUF, GOOD	
12766	047662	000137	050212		JMP PERRAW	
12767						
12768	047666				\$PER10: LET ADDRESS := R1 + #2	
	047666	010137	002034			MOV R1, ADDRESS
	047672	062737	000002	002034		ADD #2, ADDRESS
12769	047700				LET BAD := R2	
	047700	010237	002052			MOV R2, BAD
12770	047704				LET GOOD := DATBUF+2	
	047704	013737	002244	002044		MOV DATBUF+2, GOOD
12771	047712	000137	050212		JMP PERRAW	
12772						
12773	047716				\$PER11: LET ADDRESS := R1	
	047716	010137	002034			MOV R1, ADDRESS
12774	047722				LET BAD := R0	
	047722	010037	002052			MOV R0, BAD
12775	047726				LET GOOD := #0	
	047726	005037	002044			CLR GOOD
12776	047732	000137	050264		JMP PERRAB	
12777						
12778	047736				\$PER12: LET ADDRESS := R1	

12779 047736 010137 002034
047742
12780 047742 010037 002052
047746
12781 047746 012737 000377 002044
047754 000137 050264

LET BAD := R0
LET GOOD := #377
JMP PERRAB

MOV R1, ADDRESS
MOV R0, BAD
MOV #377, GOOD

12784	047760			\$PER13: LET ADDRESS := R1	
	047760	010137	002034		MOV R1,ADDRESS
12785	047764			LET BAD := R0	
	047764	010037	002052		MOV R0,BAD
12786	047770			LET GOOD := #0	
	047770	005037	002044		CLR GOOD
12787	047774	000137	050212	JMP PERRAW	
12788					
12789	050000			\$PER14: LET ADDRESS := R1	
	050000	010137	002034		MOV R1,ADDRESS
12790	050004			LET BAD := R0	
	050004	010037	002052		MOV R0,BAD
12791	050010			LET GOOD := ONES	
	050010	013737	002614 002044		MOV ONES,GOOD
12792	050016	000137	050212	JMP PERRAW	
12793					
12794	050022			\$PER15: LET ADDRESS := R1	
	050022	010137	002034		MOV R1,ADDRESS
12795	050026			LET BAD := R0	
	050026	010037	002052		MOV R0,BAD
12796	050032			LET GOOD := TSTDAT	
	050032	013737	002246 002044		MOV TSTDAT,GOOD
12797	050040	000137	050212	JMP PERRAW	
12798					
12799	050044			\$PER16: LET ADDRESS := R1	
	050044	010137	002034		MOV R1,ADDRESS
12800	050050			LET BAD := R0	
	050050	010037	002052		MOV R0,BAD
12801	050054			LET GOOD := TSTDAT+2	
	050054	013737	002250 002044		MOV TSTDAT+2,GOOD
12802	050062	000453		BR PERRAW	
12803					
12804	050064			\$PER17: LET ADDRESS := R1	
	050064	010137	002034		MOV R1,ADDRESS
12805	050070			LET BAD := R0	
	050070	010037	002052		MOV R0,BAD
12806	050074			LET GOOD := R2	
	050074	010237	002044		MOV R2,GOOD
12807	050100	000444		BR PERRAW	
12808					
12809	050102			\$PER20: LET ADDRESS := R1	
	050102	010137	002034		MOV R1,ADDRESS
12810	050106			LET BAD := R0	
	050106	010037	002052		MOV R0,BAD
12811	050112			LET GOOD := R3	
	050112	010337	002044		MOV R3,GOOD
12812	050116	000435		BR PERRAW	
12813					
12814	050120			\$PER21: LET ADDRESS := R1	
	050120	010137	002034		MOV R1,ADDRESS
12815	050124			LET BAD := R0	
	050124	010037	002052		MOV R0,BAD
12816	050130			LET GOOD := #177	
	050130	012737	000177 002044		MOV #177,GOOD
12817	050136	000477		BR PERRAW	
12818					
12819	050140			\$PER22: LET ADDRESS := R1	

12820	050140	010137	002034		LET BAD := R2	MOV R1,ADDRESS
	050144					MOV R2,BAD
12821	050144	010237	002052		LET GOOD := R0	MOV R0,GOOD
	050150					
12822	050150	010037	002044		BR PERRAW	
12823	050154	000416				
12824	050156			\$PER23:	LET ADDRESS := R1	MOV R1,ADDRESS
	050156	010137	002034		LET BAD := R0	MOV R0,BAD
12825	050162				LET GOOD := R4	MOV R4,GOOD
	050162	010037	002052			
12826	050166				BR PERRAW	
	050166	010437	002044			
12827	050172	000407				
12828						
12829	050174			\$PER24:	LET ADDRESS := R2	MOV R2,ADDRESS
	050174	010237	002034		LET BAD := R0	MOV R0,BAD
12830	050200				LET GOOD := R3	MOV R3,GOOD
	050200	010037	002052			
12831	050204				BR PERRAW	
	050204	010337	002044			
12832	050210	000400				

12834 050212

PERRAW: SUBTST <<DATA WAS A WORD>>

```

*****
;SUBTEST      DATA WAS A WORD
*****

```

12835 050212 004737 050446

12836 050216 005737 002144

050216 005737 002144

050222 001402

050224 004737 045252

050230

12837 050230 005737 002022

050230 005737 002022

050234 001002

050236 004737 034152

050242

12838 050242 004737 050422

12839 050246 005737 002144

050246 005737 002144

050252 001002

12840 050254 104011

12841 050256 000401

050256 000401

050260

12842 050260 104012

12843 050262

050262

12844 050262 000002

12845

12846 050264

```

CALL PERBNK
IF ABORTFLAG IS TRUE THEN $CALL GETDATA

```

IF BADPC EQ #0 THEN \$CALL BADSTACK

```

CALL PERXOR
IF ABORTFLAG IS FALSE

```

```

ERROR +11
ELSE

```

```

ERROR +12
END ;OF IF ABORTFLAG

```

RTI

PERRAB: SUBTST <<DATA WAS A BYTE>>

```

*****
;SUBTEST      DATA WAS A BYTE
*****

```

12847 050264 004737 050446

12848 050270 005737 002144

050270 005737 002144

050274 001402

050276 004737 045252

050302

12849 050302 005737 002022

050302 005737 002022

050306 001002

050310 004737 034152

050314

12850 050314 004737 050422

12851 050320 005737 002144

050320 005737 002144

050324 001002

12852 050326 104014

12853 050330 000401

050330 000401

050332

12854 050332 104015

12855 050334

050334

12856 050334 000002

```

CALL PERBNK
IF ABORTFLAG IS TRUE THEN $CALL GETDATA

```

IF BADPC EQ #0 THEN \$CALL BADSTACK

```

CALL PERXOR
IF ABORTFLAG IS FALSE

```

```

ERROR +14
ELSE

```

```

ERROR +15
END ;OF IF ABORTFLAG

```

RTI

```

TST ABORTFLAG
BEQ L450
JSR PC,GETDATA
L450:;:;:;:

```

```

TST BADPC
BNE L451
JSR PC,BADSTACK
L451:;:;:;:

```

```

TST ABORTFLAG
BNE L452

```

```

BR L453
L452:;:;:;:

```

L453:;:;:;:

```

TST ABORTFLAG
BEQ L454
JSR PC,GETDATA
L454:;:;:;:

```

```

TST BADPC
BNE L455
JSR PC,BADSTACK
L455:;:;:;:

```

```

TST ABORTFLAG
BNE L456

```

```

BR L457
L456:;:;:;:

```

L457:;:;:;:

12859 050336

PERRA7: SUBTST <<DATA WAS A 7 BIT BYTE>>

;*****
;*SUBTEST DATA WAS A 7 BIT BYTE;*****
IF BADPC EQ #0 THEN \$CALL BADSTACK

12860 050336

050336 005737 002022

050342 001002

050344 004737 034152

050350

12861 050350 004737 050422

12862 050354 004737 050446

12863 050360 104022

12864 050362 000002

12865

12866 050364

050364 012737 100000 002046

12867 050372

050372 012737 000100 002050

12868 050400 000137 045040

12869

12870 050404 005037 002046

12871 050410

050410 012737 000077 002050

12872 050416 000137 045040

12873

12874 050422

\$PER26: LET GOOD2 := #100000

LET GOOD3 := #100

JMP PERRA3

\$PER27: CLR GOOD2
LET GOOD3 := #077

JMP PERRA3

PERXOR: SUBTST <<DETERMINE XOR OF GOOD & BAD>>

;*****
;*SUBTEST DETERMINE XOR OF GOOD & BAD;*****
PUSH R0

MOV R0, (SP)

MOV GOOD,R0
MOV BAD,BAD XOR
XOR R0,BAD XOR
POP R0

MOV (SP)+,R0

RETURN

12875 050422

050422 010046

12876 050424 013700 002044

12877 050430 013737 002052 002060

12878 050436 074037 002060

12879 050442

050442 012600

12880 050444 000207

12883 050446

PERBNK: SUBTST<<LOG ERROR ON BAD BANK>>

;*****
;SUBTEST LOG ERROR ON BAD BANK
;*****

;WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE

PUSH RO,R1

MOV RO,-(SP)

MOV R1,-(SP)

12884

12885 050446

050446 010046

050450 010146

12886 050452 013701 002102

MOV BANK,R1

12887 050456 006301

ASL R1

12888 050460 006301

ASL R1

12889 050462 052761 000001 002664

BIS #BIT0,CONFIG(R1)

12890 050470 105261 002666

INCB CONFIG+2(R1)

;BUMP BANK COUNTER

12891 050474 001002

BNE 12\$

;NO OVERFLOW - SKIP

12892 050476 105361 002666

DECB CONFIG+2(R1)

;SET BACK TO 255.

12893 050502 126137 002666 002554 12\$:

CMPB CONFIG+2(R1),ERRMAX

;IS IT PAST MAX?

12894 050510 101403

BLOS 11\$

;NO SKIP

12895 050512

SET TOOMANY

;YES

050512 012737 177777 002406

MOV #-1,TOOMANY

12896 050520

11\$: POP R1,R0

050520 012601

MOV (SP)+,R1

050522 012600

MOV (SP)+,R0

12897 050524 000207

RETURN

12898

12899 050526 010037 002052

PERECC: MOV RO,BAD

12900 050532 023737 002034 002412

IF ADDRESS EQ TESTADD

CMP ADDRESS,TESTADD

050540 001004

BNE L461

12901 050542 013737 002246 002044

MOV TSTDAT,GOOD

12902 050550

ELSE

BR L462

050550 000403

L461:;;;;;

12903 050552 013737 002250 002044

MOV TSTDAT+2,GOOD

12904 050560

END ;OF IF (R1)

L462:;;;;;

050560 004737 050422

CALL PERXOR

12905 050564

SET HEADER

12906 050564

SET HEADER

12907 050572

SET HEADER

MOV #-1,HEADER

12908 050572

SET HEADER

12909 050574

SET HEADER

12910 050574

SET HEADER

12911 050574

SET HEADER

12912 050574

SET HEADER

12913 050574

SET HEADER

12914 050574

SET HEADER

12915 050574

SET HEADER

12916 050574

SET HEADER

12917 050574

SET HEADER

12918 050574

SET HEADER

12919 050574

SET HEADER

12920 050574

SET HEADER

12921 050574

SET HEADER

12922 050574

SET HEADER

12923 050574

SET HEADER

12924 050574

SET HEADER

12925 050574

SET HEADER

12926 050574

SET HEADER

12927 050574

SET HEADER

12928 050574

SET HEADER

12929 050574

SET HEADER

12930 050574

SET HEADER

12931 050574

SET HEADER

12932 050574

SET HEADER

12933 050574

SET HEADER

12934 050574

SET HEADER

12935 050574

SET HEADER

12936 050574

SET HEADER

12937 050574

SET HEADER

12938 050574

SET HEADER

12939 050574

SET HEADER

12940 050574

SET HEADER

CVMJAAO MSV11-J MEMORY DIAG.
LOG ERROR ON BAD BANK

MACRO V05.01b Friday 15-Feb 85 16:10 Page 414-1

SEQ 0358

12917 050636
050636 023727 002300 000015
050644 001001
12918 050646 104043
12919 050650
050650
12920 050650
050650 023727 002300 000016
050656 001001
12921 050660 104044
12922 050662
050662
12923 050662
050662 012737 177777 002612
12924 050670 000002

IF REALPAT EQ #15

ERROR +43
END ;OF IF REALPAT

IF REALPAT EQ #16

ERROR +44
END ;OF IF REALPAT

SET HEADER

RTI

L465:::~::~

CMP REALPAT,#15
BNE L466

L466:::~::~

CMP REALPAT,#16
BNE L467

L467:::~::~

MOV #-1,HEADER

PC	Address	Op Code	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10	Op 11	Op 12	Op 13	Op 14	Op 15	Op 16	Op 17	Op 18	Op 19	Op 20	Op 21	Op 22	Op 23	Op 24	Op 25	Op 26	Op 27	Op 28	Op 29	Op 30	Op 31	Op 32	Op 33	Op 34	Op 35	Op 36	Op 37	Op 38	Op 39	Op 40	Op 41	Op 42	Op 43	Op 44	Op 45	Op 46	Op 47	Op 48	Op 49	Op 50	Op 51	Op 52	Op 53	Op 54	Op 55	Op 56	Op 57	Op 58	Op 59	Op 60	Op 61	Op 62	Op 63	Op 64	Op 65	Op 66	Op 67	Op 68	Op 69	Op 70	Op 71	Op 72	Op 73	Op 74	Op 75	Op 76	Op 77	Op 78	Op 79	Op 80	Op 81	Op 82	Op 83	Op 84	Op 85	Op 86	Op 87	Op 88	Op 89	Op 90	Op 91	Op 92	Op 93	Op 94	Op 95	Op 96	Op 97	Op 98	Op 99	Op 100	Op 101	Op 102	Op 103	Op 104	Op 105	Op 106	Op 107	Op 108	Op 109	Op 110	Op 111	Op 112	Op 113	Op 114	Op 115	Op 116	Op 117	Op 118	Op 119	Op 120	Op 121	Op 122	Op 123	Op 124	Op 125	Op 126	Op 127	Op 128	Op 129	Op 130	Op 131	Op 132	Op 133	Op 134	Op 135	Op 136	Op 137	Op 138	Op 139	Op 140	Op 141	Op 142	Op 143	Op 144	Op 145	Op 146	Op 147	Op 148	Op 149	Op 150	Op 151	Op 152	Op 153	Op 154	Op 155	Op 156	Op 157	Op 158	Op 159	Op 160	Op 161	Op 162	Op 163	Op 164	Op 165	Op 166	Op 167	Op 168	Op 169	Op 170	Op 171	Op 172	Op 173	Op 174	Op 175	Op 176	Op 177	Op 178	Op 179	Op 180	Op 181	Op 182	Op 183	Op 184	Op 185	Op 186	Op 187	Op 188	Op 189	Op 190	Op 191	Op 192	Op 193	Op 194	Op 195	Op 196	Op 197	Op 198	Op 199	Op 200	Op 201	Op 202	Op 203	Op 204	Op 205	Op 206	Op 207	Op 208	Op 209	Op 210	Op 211	Op 212	Op 213	Op 214	Op 215	Op 216	Op 217	Op 218	Op 219	Op 220	Op 221	Op 222	Op 223	Op 224	Op 225	Op 226	Op 227	Op 228	Op 229	Op 230	Op 231	Op 232	Op 233	Op 234	Op 235	Op 236	Op 237	Op 238	Op 239	Op 240	Op 241	Op 242	Op 243	Op 244	Op 245	Op 246	Op 247	Op 248	Op 249	Op 250	Op 251	Op 252	Op 253	Op 254	Op 255	Op 256	Op 257	Op 258	Op 259	Op 260	Op 261	Op 262	Op 263	Op 264	Op 265	Op 266	Op 267	Op 268	Op 269	Op 270	Op 271	Op 272	Op 273	Op 274	Op 275	Op 276	Op 277	Op 278	Op 279	Op 280	Op 281	Op 282	Op 283	Op 284	Op 285	Op 286	Op 287	Op 288	Op 289	Op 290	Op 291	Op 292	Op 293	Op 294	Op 295	Op 296	Op 297	Op 298	Op 299	Op 300	Op 301	Op 302	Op 303	Op 304	Op 305	Op 306	Op 307	Op 308	Op 309	Op 310	Op 311	Op 312	Op 313	Op 314	Op 315	Op 316	Op 317	Op 318	Op 319	Op 320	Op 321	Op 322	Op 323	Op 324	Op 325	Op 326	Op 327	Op 328	Op 329	Op 330	Op 331	Op 332	Op 333	Op 334	Op 335	Op 336	Op 337	Op 338	Op 339	Op 340	Op 341	Op 342	Op 343	Op 344	Op 345	Op 346	Op 347	Op 348	Op 349	Op 350	Op 351	Op 352	Op 353	Op 354	Op 355	Op 356	Op 357	Op 358	Op 359	Op 360	Op 361	Op 362	Op 363	Op 364	Op 365	Op 366	Op 367	Op 368	Op 369	Op 370	Op 371	Op 372	Op 373	Op 374	Op 375	Op 376	Op 377	Op 378	Op 379	Op 380
----	---------	---------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------

12963							
12964	051106	010037	002044	\$PER36:	MOV	R0,GOOD	
12965	051112	010137	002052		MOV	R1,BAD	
12966	051116				SET	HEADER	
	051116	012737	177777 002612				MOV # -1,HEADER
12967	051124	104023			ERROR	+23	
12968	051126				SET	HEADER	
	051126	012737	177777 002612				MOV # -1,HEADER
12969	051134	000002			RTI		
12970							
12971	051136	104053		\$PER37:	ERROR	+53	;ILC;;REV B
12972	051140	000002			RTI		;ILC;;REV B
12973							
12974	051142	104054		\$PER40:	ERROR	+54	;ILC;;REV B
12975	051144	000002			RTI		;ILC;;REV B

```

12978          .SBTTL ROUTINE SCOPE HANDLER
12979          ;*****
12980          ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
12981          ;*AND LOAD THE DISPLAY DATA INTO THE DISPLAY REGISTER
12982          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
12983          ;*SW14=1      LOOP ON TEST
12984          ;*SW9=1      LOOP ON ERROR
12985          ;*CALL
12986          ;*
12987 051146 005237 056656          $SCOPE: INC $DEVCT      ;;SCOPE=IOT
12988 051152          IF RESULT IS LT          ;TELL AP WE ARE ALIVE
12989 051152 002004          CLR $DEVCT          BGE L475
12990 051160 005037 056656          INCB $UNIT
12991 051164 105237 056660          END ;OF IF RESULT
12992 051164 104410          CKSWR          L475:;;;;;
12993 051166 005737 006216          TST TRACE          ;;TEST FOR CHANGE IN SOFT-SWR
12994 051172 001402          BEQ NOTRCE
12995 051174 004737 055152          CALL CONTT          ;TRACE
12996 051200          NOTRCE:
12997 051200 005737 052314          TST CPERRF          ;IS THERE A CPU ERROR REGISTER?      ;R-C
12998 051204 001410          BEQ SKJ          ;BRANCH IF NOT      ;R-C
12999 051206 013737 177766 052312          MOV @#177766,CPSAVE ;GET CONTENTS OF ERROR REGISTER      ;R-C
13000 051214 032737 000001 052312          BIT #BIT0,CPSAVE ;IS THE POWER FAIL MONITOR BIT SET?      ;R-C
13001 051222 001401          BEQ SKJ          ;BRANCH IF NOT      ;R-C
13002 051224 104177          ERROR +177          ;REPORT IF SO      ;R-C
13011 051226          SKJ: IF STOPOK IS TRUE AND #SW8 SET.IN @SWR      ;R-C
13012 051226 005737 002420          TST STOPOK
13013 051232 001410          BEQ L476
13014 051234 032777 000400 131374          BIT #SW8,@SWR
13015 051242 001404          BEQ L476
13016 051244 005037 002420          CLR STOPOK
13017 051250 000137 040604          JMP EXIT
13018 051254          END ;OF IF STOPOK
13019 051254          IF NOSCOPE IS TRUE          L476:;;;;;
13020 051254 005737 002440          TST NOSCOPE
13021 051260 001401          BEQ L477
13022 051262 000002          RTI
13023 051264          END ;OF IF NOSCOPE
13024 051264          L477:;;;;;
13025 051264 032777 040000 131344          1$: IF #SW14 SET.IN @SWR THEN GOTO $OVER
13026 051272 001051          BIT #SW14,@SWR
13027 051274 000425          BNE $OVER
13028 051276 013746 000004          ;####START OF CODE FOR THE XOR TESTER####
13029 051302 012737 051322 000004          $XTSTR: BR 2$          ;;IF RUNNING ON THE "XOR" TESTER CHANGE
13030 051310 005737 177060          MOV ERRVEC,-(SP)          ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
13031 051314 012637 000004          MOV #1$,ERRVEC          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
13032 051320 000430          TST 177060          ;;SET FOR TIMEOUT
13033 051322 062706 000004          MOV (SP)+,ERRVEC          ;;RESTORE THE ERROR VECTOR
13034 051334 001002          BR $SVLAD          ;;GO TO THE NEXT TEST
13035 051336 005037 177766          1$: ADD #4,SP          ;FIX STACK FROM TRAP
13036          CMP #5,PROTYP          ;IS THIS AN 11/83 ?
13037          BNE 6$          ;BRANCH IF NOT
13038          CLR CPUERR          ;RESET CPU ERROR REGISTER

```

13031	051342	012637	000004	6\$:	MOV	(SP)+,ERRVEC	::RESTORE THE ERROR VECTOR
13032	051346	000407			BR	4\$::LOOP ON THE PRESENT TEST
13033	051350			2\$::	####END OF CODE FOR THE XOR TESTER####		
13034	051350	105737	002014	3\$:	TSTB	\$ERFLG	::HAS AN ERROR OCCURRED?
13035	051354	001412			BEQ	\$SVLAD	::BR IF NO
13036	051356	032777	001000	131252	BIT	\$SW9,\$SWR	::LOOP ON ERROR?
13037	051364	001404			BEQ	5\$::BR IF NO
13038	051366	013737	002624	002622	4\$:	MOV	\$LPERR,\$LPADR
13039	051374	000410			BR	\$OVER	::SET LOOP ADDRESS TO LAST SCOPE
13040	051376	105037	002014	5\$:	CLRB	\$ERFLG	::ZERO THE ERROR FLAG
13041	051402	011637	002622	\$SVLAD:	MOV	(SP),\$LPADR	::SAVE SCOPE LOOP ADDRESS
13042	051406	011637	002624		MOV	(SP),\$LPERR	::SAVE ERROR LOOP ADDRESS
13043	051412	005037	002362		CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS
13044	051416	004737	051430	\$OVER:	CALL	GETDIS	
13045	051422	013716	002622		MOV	\$LPADR,(SP)	::FUDGE RETURN ADDRESS
13046	051426	000002			RTI		::FIXES PS

13048 051430

GETDIS: SUBTST <<SUBR DISPLAY>>

;*****
;SUBTEST SUBR DISPLAY
;*****

13049 051430 113737 002102 002013

MOVB BANK,\$BANK

13050 051436 113737 002300 002012

MOVB REALPAT,\$PATMAR

13051 051444

PUSH RO

MOV RO,-(SP)

051444 010046

13052 051446 005737 002126

TST RLFLAG

;ARE WE RELOCATED?

13053 051452 001403

BEQ 1\$

;NO - SKIP

13054 051454 052737 100000 002012

BIS #BIT15,\$PATMAR

;YES - SET MSB

13055 051462

1\$:

13059 051462 013777 002012 131150

MOV \$PATMAR,@DISPLAY

13060 051470 013737 002012 000174

MOV \$PATMAR,DISPREG

;SOFTWARE DISPLAY REGISTER

13061 051476

POP RO

MOV (SP)+,RO

051476 012600

13062 051500 000207

RETURN

```

13065          SBTTL ROUTINE ERROR HANDLER
13066          ;*****
13067          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
13068          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
13069          ;*AND GO TO $ERRTYP ON ERROR
13070          ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
13071          ;*SW15=1      HALT ON ERROR
13072          ;*SW13=1      INHIBIT ERROR TYPEOUTS
13073          ;*SW10=1      BELL ON ERROR
13074          ;*SW9=1      LOOP ON ERROR
13075          ;*CALL
13076          ;*      ERROR      N      ;;ERROR-EMT AND N=ERROR ITEM NUMBER
13077
13078          .ENABL  LSB
13079 051502 105737 056666 $ERROR: TSTB $ENV      ;ARE WE RUNNING IN APT MODE ?
13080 051506 001402      BEQ      NOAPT      ;YES - DO UFD ABORT
13081 051510 004737 000000      JSR      PC,ABORT      ;GOTO UFD ABORT
13082 051514 105037 052310 NOAPT: CLRB      IBSAVE      ;R-C
13083 051520      IF NOERROR IS FALSE
13084 051524 005737 002430      TST NOERROR
13085 051526 001027      BNE L500
13086 051530      CKSWR      ;;TEST FOR CHANGE IN SOFT SWR
13087 051534 105237 002014 BACK: 1$: INCB $ERFLG      ;;SET THE ERROR FLAG
13088 051536 001775      BEQ 1$      ;;DON'T LET THE FLAG GO TO ZERO
13089 051542 004737 051430      CALL GETDIS      ;SETUP DISPLAY STUFF
13090 051544 013737 002012 056652      MOV $PATMAR,$TESTN ;FOR APT
13091 051550 032777 002000 131060      BIT $SW10,$SWR      ;BELL ON ERROR?
13092 051556 001404      BEQ 2$      ;NO SKIP
13093 051560      TYPE $BELL      ;RING BELL
13094 051564 104401 002653      TYPEIT , $BELL
13095 051564 104401 066060      .DSABL CRF
13096 051570 005237 002630      TYPE MSG014      ;CONTROL Z
13097 051574 100003      TYPEIT ,MSG014
13098 051576 012737 077777 002630      .DSABL CRF
13099 051604      INC $ERTTL      ;COUNT THE NUMBER OF ERRORS
13100 051604      IF RESULT IS MI
13101 051604      MOV #77777,$ERTTL
13102 051604      END ;OF IF RESULT
13103 051604      L501:;;;;;
13104 051604      END ;OF IF NOERROR
13105 051604 011637 002020      MOV (SP),ERRPC      ;GET ADDRESS OF ERROR INSTRUCTION
13106 051610 162737 000002 002020      SUB #2,ERRPC
13107 051616 010637 002024      MOV SP,ERRSP
13108 051622 016637 000002 002030      MOV 2(SP),ERRPSW
13109 051630 117737 130164 002015      MOVB @ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
13110 051636 122737 000177 002015      CMPB #177,$ITEMB ;IS THIS THE POWER FAIL CALL? ;R-C
13111 051644 001431      BEQ 1001$ ;BRANCH IF SO ;R-C
13112 051646 105737 052310      TSTB IBSAVE ;2ND ERROR CALL? ;R-C
13113 051652 001024      BNE 1000$ ;BRANCH IF SO ;R-C
13114 051654 005737 052314      TST CPERRF ;IS THERE A CPU ERROR REGISTER? ;R-C
13115 051660 001423      BEQ 1001$ ;BRANCH IF NOT ;R-C
13116 051662 013737 177766 052312      MOV 177766,CPSAVE ;SAVE CONTENTS ;R-C
13117 051670 032737 000001 052312      BIT #BIT0,CPSAVE ;POWER MONITOR BIT SET? ;R-C

```

```

13113 051676 001414      BEQ      1001$      ;BRANCH IF NOT      ;R-C
13114 051700 042737 000001 177766      BIC      #BIT0,177766 ;CLEAR THE BIT      ;R-C
13115 051706 112737 002015 052310      MOVB     #$ITEMB,IBSAVE ;MAKE IBSAVE NON-ZERO FOR DUAL CALL ;R-C
13116 051714 112737 000177 002015      MOVB     #177,$ITEMB ;SET $ITEMB TO POWER FAIL POINTER ;R-C
13117 051722 000402      BR      1001$      ;R-C
13118 051724 105037 052310      CLRB     IBSAVE      ;R-C
13119 051730      1001$:      ;R-C
13120 051730      IF NOERROR IS FALSE
      051730 005737 002430      TST NOERROR
      051734 001043      BNE L502
13121 051736      IF BADPC NE #0
      051736 005737 002022      TST BADPC
      051742 001416      BEQ L503
13122 051744 013737 002022 002020      MOV BADPC,ERRPC
13123 051752 162737 000002 002020      SUB #2,ERRPC
13124 051760 013737 002026 002024      MOV BADSP,ERRSP
13125 051766 013737 002032 002030      MOV BADPSW,ERRPSW
13126 051774 005037 002022      CLR BADPC
13127 052000      END ;IF
      052000      L503:;
13128 052000 013737 002020 056650      MOV ERRPC,$FATAL ;FOR APT
13129 052006 004737 050446      CALL PERBNK ;LOG ERROR ON BANK
13130 052012      IF #SW13 SET.IN @SWR
      052012 032777 020000 130616      BIT #SW13,@SWR
      052020 001401      BEQ L504
13131 052022 000420      BR 3$
13132 052024      END ;OF IF #SW13
      052024      L504:;
13133 052024      IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE
      052024 032777 000040 130604      BIT #SW5,@SWR
      052032 001404      BEQ L505
      052034 005737 002406      TST TOOMANY
      052040 001401      BEQ L505
13134 052042      GOTO 3$
      052042 000410      BR 3$
13135 052044      END ;OF IF #SW5
      052044      L505:;
13136 052044      END ;OF IF NOERROR
      052044      L502:;
13137 052044 004737 052316      CALL $ERRTYP ;GO TO USER ERROR ROUTINE
13138 052050      IF MONFLG IS TRUE ;SHOULD WE RETURN TO XXDP MONITOR???
      052050 005737 002276      TST MONFLG
      052054 001403      BEQ L506
13139 052056 013706 002274      MOV SAVMON,SP ;GET MONITOR ADDRESS
13140 052062 000207      RTS PC ;GO TO MONITOR
13141 052064      END ;
      052064      L506:;

```

```

13143 052064      3$: IF NOERROR IS FALSE
      052064 005737 002430
      052070 001072
13144 052072 005777 130540
13145 052076 100002
13146 052100 000000
13147 052102 104410
13148 052104      7$: IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR
      052104 005737 002440
      052110 001006
      052112 032777 001000 130516
      052120 001402
13149 052122 013716 002624
13150 052126
      052126
13151 052126 005737 002362
13152 052132 001402
13153 052134 013716 002362
13154 052140      9$: IF DETFLAG IS FALSE
      052140 005737 002220
      052144 001043
13155 052146 022737 000005 004010
13156 052154 001002
13157 052156 005037 177766
13158 052162
      052162 005737 002350
      052166 001006
      052170 005737 002352
      052174 001003
      052176 005737 002064
      052202 001405
      052204
13159 052204 012737 000001 056646
13160 052212 000137 040604
13161 052216
      052216
13162 052216
      052216 005737 002354
      052222 001414
      052224 023727 002630 000020
      052232 101410
13163 052234
      052234 104401 070051
13164 052240 013700 000042
13165 052244 005037 000042
13166 052250 000137 014022
13167 052254
      052254
13168 052254
      052254
13169 052254
      052254 000403
      052256
13170 052256
      052256 012737 177777 002612
13171 052264

```

```

      TST NOERROR
      BNE L507

      TST @SWR          ;;HALT ON ERROR
      BPL 7$           ;;SKIP IF CONTINUE
      $HALT:          ;;HALT ON ERROR!
      CKSWR           ;;TEST FOR CHANGE IN SOFT SWR
      7$: IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR

      TST NOSCOPE
      BNE L510
      BIT #SW9,@SWR
      BEQ L510

      MOV $LPERR,(SP)   ;;FUDGE RETURN FOR LOOPING
      END ;OF IF NOSCOPE

      TST $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
      BEQ 9$           ;;BR IF NONE
      MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
      9$: IF DETFLAG IS FALSE

      TST DETFLAG
      BNE L511

      CMP #5,PROTYP    ;IS THIS AN 11/83 ?
      BNE 11$
      CLR CPUERR
      11$: IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL$ IS TRUE

      TST ACTFLAG
      BNE L512
      TST APTFLAG
      BNE L512
      TST FATAL$
      BEQ L513
      L512:;;;;;

      MOV #1,$MSGTY    ;FOR APT
      JMP EXIT
      END ;OF IF ACTFLAG

      L513:;;;;;

      IF XXDPCHAIN IS TRUE AND $ERTTL HI #20

      TST XXDPCHAIN
      BEQ L514
      CMP $ERTTL,#20
      BLOS L514
      ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN

      TYPE MSG066
      TYPEIT ,MSG066
      .DSABL CRF
      MOV 42,R0
      CLR 42
      JMP $ZAP42
      END ;OF IF XXDPCHAIN

      L514:;;;;;

      END ;OF IF DETFLAG
      L511:;;;;;

      ELSE
      BR L515
      L507:;;;;;

      SET HEADER
      MOV #-1,HEADER
      END ;OF IF NOERROR

```



```
13172 052264 005037 002406 10$: CLEAR TOOMANY,NOERROR LS15:;;;;;
13173 052270 005037 002430 CLR TOOMANY
13174 052274 105737 052310 CLR NOERROR
13175 052302 000137 051530 TSTB IBSAVE ;POWER FAIL ERROR CALL? ;R C
13176 052306 000002 BEQ 213$ ;R-C
13177 052310 000000 JMP BACK ;JUMP IF SO ;R-C
13178 052312 000000 213$: RTI ;;RETURN ;R-C
13179 052314 000000 IBSAVE: .WORD 0 ;R-C
13180 CPSAVE: .WORD 0 ;R C
        CPERRF: .WORD 0 ;R-C
        .DSABL LSB
```

.SBTTL ROUTINE ERROR MESSAGE TYPEOUT

```

13183
13184
13185
13186
13187
13188
13189
13190 052316 104415
13191 052320 104401 002660
13192 052324 005000
13193 052326 153700 002015
13194 052332 001004
13195
13196 052334 013746 002020
13197 052342 000511
13198 052344 122700 000177
13199 052350 001003
13200 052352 012700 052626
13201 052356 000406
13202 052360 005300
13203 052362 006300
13204 052364 006300
13205 052366 006300
13206 052370 062700 057274
13207 052374 012037 052432
13208 052400 001417
13209 052402 005737 002430
13210 052406 001003
13211 052410 005737 002612
13212 052414 100011
13213 052416 005737 002064
13214 052422 001402
13215 052424 104401 070120
13216 052430 104401
13217 052432 000000
13218 052434 104401 002660
13219 052440 012037 052464
13220 052444 001412
13221 052446 005737 002430
13222 052452 001003
13223 052454 005737 002612
13224 052460 100004
13225 052462 104401

*****
;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
$ERRTYP: SAVREG
TYPE $CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
TYPEIT , $CRLF
.DSABL CRF
CLR RO ;; PICKUP THE ITEM INDEX
BISB $ITEMB, RO
BNE 1$ ;; IF ITEM NUMBER IS ZERO, JUST
;; TYPE THE PC OF THE ERROR
TYPOCT ERRPC, <ERROR ADDRESS>
MOV ERRPC, -(SP) ;; SAVE ERRPC FOR TYPEOUT
;; ERROR ADDRESS
;; GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
BR 11$ ;; GET OUT
1$: CMPB #177, RO ;; POWER MONITOR CALL? ;R-C
BNE 100$ ;; BRANCH IF NOT ;R-C
MOV #PFECWS, RO ;; MOV ADDRESS OF PFE BIT ERROR TO RO ;R-C
BR 110$ ;; ;R-C
100$: DEC RO ;; ADJUST THE INDEX SO THAT IT WILL ;R-C
ASL RO ;; WORK FOR THE ERROR TABLE
ASL RO
ASL RO
ADD # $ERRTB, RO ;; FORM TABLE POINTER
110$: MOV (RO)+, 3$ ;; PICKUP "ERROR MESSAGE" POINTER ;R-C
BEQ 4$ ;; SKIP TYPEOUT IF NO POINTER
TST NOERROR ;; IS THIS REALLY AN ERROR?
BNE 12$ ;; YES - SKIP
TST HEADER ;; TYPE HEADER?
BPL 4$ ;; NO - SKIP
12$: TST FATAL$ ;; WAS IT A FATAL ERROR?
BEQ 2$ ;; NO - SKIP
TYPE MSG067 ;; FATAL
TYPEIT , MSG067
.DSABL CRF
2$: TYPE ;; TYPE THE "ERROR MESSAGE"
TYPEIT
.DSABL CRF
3$: .WORD 0 ;; "ERROR MESSAGE" POINTER GOES HERE
TYPE $CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
TYPEIT , $CRLF
.DSABL CRF
4$: MOV (RO)+, 5$ ;; PICKUP "DATA HEADER" POINTER
BEQ 6$ ;; SKIP TYPEOUT IF 0
TST NOERROR ;; IS THIS REALLY AN ERROR?
BNE 13$ ;; YES - SKIP
TST HEADER ;; TYPE HEADER?
BPL 6$ ;; NO - SKIP
13$: TYPE ;; TYPE THE "DATA HEADER"
TYPEIT
.DSABL CRF

```

13226 052464 000000
13227 052466
052466 104401 002,60

13228 052472 012001
13229 052474 001427
13230 052476 012002

5\$: .WORD 0 ;; "DATA HEADER" POINTER GOES HERE
TYPE \$CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
TYPEIT , \$CRLF
.DSABL CRF
6\$: MOV (R0)+, R1 ;; PICKUP "DATA TABLE" POINTER
BEQ 10\$;; BR IF NO DATA TO BE TYPED
MOV (R0)+, R2 ;; PICKUP "DATA FORMAT" POINTER

```

13233 052500 112203      7$:  MOVB  (R2)+,R3
13234 052502 006303      ASL   R3           ;MAKE IT A WORD ADDRESS
13235 052504 004773 052512 CALL  @8$(R3)
13236 052510 000412      BR    9$
13237 052512 052736      8$:  TAG70$
13238 052514 052746      TAG71$
13239 052516 052756      TAG72$
13240 052520 053026      TAG73$
13241 052522 053066      TAG74$
13242 052524 053100      TAG75$
13243 052526 053112      TAG76$
13244 052530 053156      TAG77$
13245 052532 053164      TAG78$
13246 052534 053244      TAG79$
13251 052536 062701 000002 9$:  ADD   #2,R1           ;UPDATE DATA TABLE POINTER
13252 052542 005711      TST   (R1)           ;;IS THERE ANOTHER NUMBER?
13253 052544 001403      BEQ   10$           ;;BR IF NO
13254 052546 104401 066107 TYPE  MSG018       ;TYPE 2 SPACES
                                TYPEIT .MSG018
                                .DSABL CRF
                                BR    7$           ;;LOOP
13255 052552 000752
13256
13257 052554 005737 002110 10$:  TST   MUT           ;IS THERE A MEMORY UNDER TEST
13258 052560 001402      BEQ   11$           ;NO - SKIP
13259 052562 005237 002612 INC    HEADER       ;YES  BUMP HEADER FLAG
13260 052566 104416
13261 052570      11$:  RESREG
                                IF #SW7 SET.IN @SWR AND DETFLAG IS FALSE AND NOERROR IS FALSE
                                BIT #SW7,@SWR
                                BEQ L516
                                TST DETFLAG
                                BNE L516
                                TST NOERROR
                                BNE L516
                                052570 032777 000200 130040
                                052576 001410
                                052600 005737 002220
                                052604 001005
                                052606 005737 002430
                                052612 001002
                                13262 052614 004737 053266
                                CALL DETAIL
                                END ;OF IF #SW7
                                13263 052620
                                052620
                                13264 052620
                                052620 104401 070557
                                TYPE  MSG104           ;CONTROL Z
                                TYPEIT .MSG104
                                .DSABL CRF
                                RETURN
                                .EVEN
                                13265 052624 000207
                                13266
                                13267 052626 052636 052672 052722 PFECWS: .WORD  PFECM,PFECDH,PFECDT,PFECDF ;R-C
                                052634 052732 ;R C
                                13268 052636 120 117 127 PFECM: .ASCIZ  "POWER MONITOR BIT FOUND SET" ;R-C
                                052641 105 122 040
                                052644 115 117 116
                                052647 111 124 117
                                052652 122 040 102
                                052655 111 124 040
                                052660 106 117 125
                                052663 116 104 040
                                052666 123 105 124
                                052671 000
                                13269 052672 124 105 123 PFECDH: .ASCIZ  "TESTNO ERR PC CPUERR" ;R C
                                052675 124 116 117
                                052700 040 040 105
                                052703 122 122 040

```


13276
13277
13278
13279 052736 017146 000000
052736 104402
052742
13280 052744 000207
13281
13282
13283
13284
13285 052746 017146 000000
052746 104405
052752
13286 052754 000207
13287
13288
13289
13290
13291 052756 010146
052756 010546
052760 013701 002102
13292 052762 070127 000004
13293 052766 012737 177777 002372
13294 052772 053000 104401 066060
053000
13295 053004 004737 033404
13296 053010 005037 002372
13297 053014 012605
13298 053016 012601
053020
13299 053020 104401 066060
053020
13300 053024 000207
13301
13302
13303
13304
13305 053026 010146
053026 010546
053030 013701 002102
13306 053032 070127 000004
13307 053036 012737 177777 002372
13308 053042 053050 004737 033600
053054 005037 002372
13309 053060 012605
13310 053062 012601
13311 053064 000207
13312

```
*****
*** OCTAL ***
*****
TAG70$: TYP OCT @ (R1) ; TYPE AN OCTAL NUMBER
        MOV @ (R1), -(SP) ; SAVE @ (R1) FOR TYPEOUT
        TYP OCT ; GO TYPE--OCTAL ASCII (ALL DIGITS)
        .DSABL CRF
        RETURN

*****
*** DECIMAL ***
*****
TAG71$: TYP DEC @ (R1) ; TYPE A DECIMAL NUMBER
        MOV @ (R1), -(SP) ; SAVE @ (R1) FOR TYPEOUT
        TYP DS ; GO TYPE--DECIMAL ASCII WITH SIGN
        .DSABL CRF
        RETURN

*****
*** INTERLEAVE ***
*****
TAG72$: PUSH R1, R5
                                MOV R1, -(SP)
                                MOV R5, -(SP)
                                MOV BANK, R1
                                MUL #4, R1
                                SET NOTAB ; INDICATE NO TABLE TO BE PRINTED - NOW
                                MOV #1, NOTAB
                                TYPE MSG014
                                TYPEIT ,MSG014
                                .DSABL CRF
                                CALL TCFIG1
                                CLR NOTAB
                                POP R5, R1
                                MOV (SP)+, R5
                                MOV (SP)+, R1
                                TYPE MSG014 ; 1 SPACE
                                TYPEIT ,MSG014
                                .DSABL CRF
                                RETURN

*****
*** CSR ***
*****
TAG73$: PUSH R1, R5
                                MOV R1, -(SP)
                                MOV R5, -(SP)
                                MOV BANK, R1
                                MUL #4, R1
                                SET NOTAB
                                MOV #-1, NOTAB
                                CALL TCFIG3
                                CLR NOTAB
                                POP R5, R1
                                MOV (SP)+, R5
                                MOV (SP)+, R1
                                RETURN
```

13313
 13314
 13315
 13316
 13317 053066 013746 002300
 053066 013746 002300
 053072 104403
 053074 002
 053075 001
 13318 053076 000207
 13319
 13320
 13321
 13322
 13323 053100 013746 002102
 053100 013746 002102
 053104 104403
 053106 003
 053107 000
 13324 053110 000207

```

*****
*** PATTERN ***
*****
TAG74$: TYPOCS REALPAT,<TYPE (0-77)>,2,Z
        MOV     REALPAT,-(SP)      ;;SAVE REALPAT FOR TYPEOUT
                                   ;;TYPE (0-77)
        TYPOS                                ;;GO TYPE--OCTAL ASCII
        .BYTE  2                          ;;TYPE 2 DIGIT(S)
        .BYTE  1                          ;;TYPE LEADING ZEROS
        .DSABL CRF
        RETURN

*****
*** BANK ***
*****
TAG75$: TYPOCS BANK,<TYPE (0-176)>,3
        MOV     BANK,-(SP)        ;;SAVE BANK FOR TYPEOUT
                                   ;;TYPE (0-176)
        TYPOS                                ;;GO TYPE--OCTAL ASCII
        .BYTE  3                          ;;TYPE 3 DIGIT(S)
        .BYTE  0                          ;;SUPPRESS LEADING ZEROS
        .DSABL CRF
        RETURN
  
```

```
13326 ;*****
13327 ;*** MTYPE ***
13328 ;*****
13329 053112 TAG76$: PUSH R1,R5
      053112 010146 MOV R1,-(SP)
      053114 010546 MOV R5,-(SP)
13330 053116 013701 002102 MOV BANK,R1
13331 053122 070127 000004 MUL #4,R1
13332 053126 012737 177777 002372 SET NOTAB
      053126 012737 177777 002372 MOV #1,NOTAB
13333 053134 TYPE MSG019
      053134 104401 066112 TYPEIT ,MSG019
      .DSABL CRF
      CALL TCFIG2
      CLR NOTAB
      POP R5,R1
      MOV (SP)+,R5
      MOV (SP)+,R1
13337 RETURN
13338
13339 ;*****
13340 ;*** UNKNOWN DATA ***
13341 ;*****
13342 053156 TAG77$: TYPE MSG061
      053156 104401 067767 TYPEIT ,MSG061
      .DSABL CRF
      RETURN
13343 053162 000207
13344
13345 ;*****
13346 ;*** PHYSICAL ADDRESS ***
13347 ;*****
13348 053164 013737 002034 002040 TAG78$: MOV ADDRESS,PHYADD
13349 053172 162737 060000 002040 SUB #FIRST,PHYADD
13350 053200 013737 002102 002042 MOV BANK,PHYADD+2
13351 053206 006237 002042 ASR PHYADD+2
13352 053212 103003 BCC 1$
13353 053214 052737 100000 002040 BIS #BIT15,PHYADD
13354 053222 012746 002040 1$: MOV #PHYADD,-(SP) ;POINTER TO DOUBLE WORD ON STACK
13355 053226 004737 056526 CALL $DB20 ;CALL DOUBLE PRECISION CONVERSION ROUTINE
13356 053232 062706 000002 ADD #2,SP ;FIX STACK
13357 053236 TYPE $OCT8
      053236 104401 056634 TYPEIT , $OCT8
      .DSABL CRF
      RETURN
13358 053242 000207
13359
13360 ;*****
13361 ;*** OCTAL BYTE ***
13362 ;*****
13363 053244 TAG79$: TYPE MSG018 ;2 SPACES
      053244 104401 066107 TYPEIT ,MSG018
      .DSABL CRF
      TYPOCS @ (R1),<TYPE BYTE>,3,Z
      MOV @ (R1),-(SP) ;;SAVE @ (R1) FOR TYPEOUT
      TYPOS ;;TYPE BYTE
      .BYTE 3 ;;GO TYPE--OCTAL ASCII
      .BYTE 1 ;;TYPE 3 DIGIT(S)
      ;;TYPE LEADING ZEROS
13364 053250
      053250 017146 000000
      053254 104403
      053256 003
      053257 001
```


CVMJAAO MSV11-J MEMORY DIAG.
ROUTINE ERROR MESSAGE TYPEOUT

MACRO V05.01b Friday 15 Feb-85 16:10 Page 429 1

SEQ 0375

13365 053260
053260 104401 066060
13366 053264 000207

.DSABL CRF
TYPE MSG014 ;SPACE
TYPEIT ,MSG014
.DSABL CRF
RETURN

13410 053266

DETAIL: SUBTST <<SUBR DETAILED ERROR REPORT>>

```

*****
;SUBTEST      SUBR      DETAILED ERROR REPORT
*****

```

13411	053266	005237	002220		INC	DETFLAG	
13412	053272	022737	000003	002220	CMP	#3,DETFLAG	
13413	053300	101473			BLOS	4\$	
13414	053302	022737	000002	002220	CMP	#2,DETFLAG	
13415	053310	001435			BEQ	2\$	
13416	053312				PUSH	HEADER,MUT	
	053312	013746	002612				MOV HEADER,-(SP)
	053316	013746	002110				MOV MUT,-(SP)
13417	053322				SET	HEADER	
	053322	012737	177777	002612			MOV #-1,HEADER
13418	053330	005037	002110		CLR	MUT	
13419	053334	010037	002200		MOV	R0,DETRO	
13420	053340	012700	002202		MOV	#DETR1,R0	
13421	053344	010120			MOV	R1,(R0)+	
13422	053346	010220			MOV	R2,(R0)+	
13423	053350	010320			MOV	R3,(R0)+	
13424	053352	010420			MOV	R4,(R0)+	
13425	053354	010520			MOV	R5,(R0)+	
13426	053356	013720	002024		MOV	ERRSP,(R0)+	
13427	053362	013720	002030		MOV	ERRPSW,(R0)+	
13428	053366	013700	002200		MOV	DETRO,R0	
13429	053372				SET	NOERROR	
	053372	012737	177777	002430			MOV #-1,NOERROR
13430	053400	104013			ERROR	+13	
13431	053402	000423			BR	1\$	
13432	053404				PUSH	HEADER,MUT	
	053404	013746	002612				MOV HEADER,-(SP)
	053410	013746	002110				MOV MUT,-(SP)
13433	053414				SET	HEADER	
	053414	012737	177777	002612			MOV #-1,HEADER
13434	053422	005037	002110		CLR	MUT	
13435	053426				SET	NOERROR	
	053426	012737	177777	002430			MOV #-1,NOERROR
13436	053434	104031			ERROR	+31	
13437	053436	022737	000005	004010	CMP	#5,PROTYP	
13438	053444	001002			BNE	1\$	
13439	053446	005037	177766		CLR	CPUERR	
13440	053452				POP	MUT,HEADER	
	053452	012637	002110				MOV (SP)+,MUT
	053456	012637	002612				MOV (SP)+,HEADER
13441					;WARNING RECURSIVE		
13442	053462	004737	053266		CALL DETAIL		
13443	053466	000207			RETURN		

;IS THIS AN 11/83 ?


```

13472 053622          FOR      RO :- SP TO R1 BY #2
      053622 010600
      053624
13473 053624          TYPE      $CRLF
      053624 104401 002660    TYPEIT  , $CRLF
                                .DSABL  CRF
                                TYPOCT  RO
      13474 053630          MOV      RO, -(SP)      ;;SAVE RO FOR TYPEOUT
      053630 010046          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      053632 104402          .DSABL  CRF
      13475 053634          TYPE      MSG018          ;2 SPACES
      053634 104401 066107    TYPEIT  ,MSG018
                                .DSABL  CRF
      13476 053640          TYPOCT  (RO)
      053640 011046          MOV      (RO), -(SP)    ;;SAVE (RO) FOR TYPEOUT
      053642 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                .DSABL  CRF
      13477 053644          END ;OF FOR RO
      053644 062700 000002
      053650 020001
      053652 003764
      053654
                                ADD #2,RO
                                CMP RO,R1
                                BLE B105
      13478          ;SET PREVIOUS MODE TO SUPERVISOR
      13479 053654 005737 002456    TST      NOSUPER
      13480 053660 001036          BNE      DET1
      13481 053662 042737 030000 177776    BIC      #BIT13!BIT12,PSW
      13482 053670 052737 010000 177776    BIS      #BIT12,PSW
      13483 053676 006506          MFPI      SSP
      13484 053700          POP      R1,RO
                                MOV (SP)+,R1
                                MOV (SP)+,RO
      13485 053704          TYPE      MSG089          ;SUPERVISOR STACK
      053704 104401 070354    TYPEIT  ,MSG089
                                .DSABL  CRF
      13486 053710          IF RO LT #SUPSTK
      053710 020027 000740
      053714 002016
                                CMP RO,#SUPSTK
                                BGE L520
      13487 053716          FOR RO := RO TO #SUPSTK-2 BY #2
      053716
                                B106:;;;;;
      13488 053716          TYPE      $CRLF
      053716 104401 002660    TYPEIT  , $CRLF
                                .DSABL  CRF
                                TYPOCT  RO
      13489 053722          MOV      RO, -(SP)      ;;SAVE RO FOR TYPEOUT
      053722 010046          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      053724 104402          .DSABL  CRF
      13490 053726          TYPE      MSG018          ;2 SPACES
      053726 104401 066107    TYPEIT  ,MSG018
                                .DSABL  CRF
      13491 053732          TYPOCT  (RO)
      053732 011046          MOV      (RO), -(SP)    ;;SAVE (RO) FOR TYPEOUT
      053734 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                .DSABL  CRF
      13492 053736          END ;OF FOR RO
      053736 062700 000002
      053742 020027 000736
      053746 003763
                                ADD #2,RO
                                CMP RO,#SUPSTK-2
                                BLE B106

```

```

13493 053750 000402 ELSE
13494 053752 104401 070412 TYPE MSG091 ;IS EMPTY
                                TYPEIT ,MSG091
                                .DSABL CRF
                                END ;OF IF R0
                                E106:::
                                BR L521
                                L520:::
13495 053756 052737 030000 177776 DET1: ;SET PREVIOUS MODE TO USER
                                BIS #BIT13!BIT12,PSW
                                MFPI USP
                                POP R0
                                L521:::
13500 053770 104401 070376 TYPE MSG090 ;USER STACK
                                TYPEIT ,MSG090
                                .DSABL CRF
                                IF R0 LT #USESTK
                                MOV (SP)+,R0
                                CMP R0,#USESTK
                                BGE L522
13501 053774 020027 000700
                                054000 002016
                                FOR R0 := R0 TO #USESTK-2 BY #2
                                B107:::
13503 054002 104401 002660 TYPE $CRLF
                                TYPEIT , $CRLF
                                .DSABL CRF
                                TYPOCT R0
                                MOV R0,-(SP) ;;SAVE R0 FOR TYPEOUT
                                TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
                                .DSABL CRF
                                TYPE MSG018 ;2 SPACES
                                TYPEIT ,MSG018
                                .DSABL CRF
                                TYPOCT (R0)
                                MOV (R0),-(SP) ;;SAVE (R0) FOR TYPEOUT
                                TYPOC ;;GO TYPE- OCTAL ASCII(ALL DIGITS)
                                .DSABL CRF
                                END ;OF FOR R0
                                ADD #2,R0
                                CMP R0,#USESTK-2
                                BLE B107
                                E107:::
13508 054034 000402 ELSE
                                BR L523
                                L522:::
13509 054036 104401 070412 TYPE MSG091 ;IS EMPTY
                                TYPEIT ,MSG091
                                .DSABL CRF
                                END ;OF IF R0
                                L523:::
13510 054042 104401 002660 TYPE $CRLF
                                TYPEIT , $CRLF
                                .DSABL CRF
                                CLR DETFLAG
                                POP R0
                                MOV (SP)+,R0
13512 054046 005037 002220
13513 054052 012600
13514 054054 000207 RETURN

```

.SBTTL ROUTINE BINARY TO OCTAL (ASCII) AND TYPE

```

13552
13553
13554
13555
13556
13557
13558
13559
13560
13561
13562
13563
13564
13565
13566
13567
13568
13569
13570
13571
13572
13573
13574
13575
13576
13577 054056 017646 000000
13578 054062 116637 000001 054301
13579 054070 112637 054303
13580 054074 062716 000002
13581 054100 000406
13582 054102 112737 000001 054301
13583 054110 112737 000006 054303
13584 054116 112737 000005 054300
13585 054124 010346
13586 054126 010446
13587 054130 010546
13588 054132 113704 054303
13589 054136 005404
13590 054140 062704 000006
13591 054144 110437 054302
13592 054150 113704 054301
13593 054154 016605 000012
13594 054160 005003
13595 054162 006105
13596 054164 000404
13597 054166 006105
13598 054170 006105
13599 054172 006105
13600 054174 010503
13601 054176 006103
13602 054200 105337 054302
13603 054204 100016
13604 054206 042703 177770
13605 054212 001002
13606 054214 005704
13607 054216 001403
13608 054220 005204

;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;OCTAL (ASCII) NUMBER AND TYPE IT.
;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;CALL:
;*      MOV      NUM, (SP)      ;;NUMBER TO BE TYPED
;*      TYPOS      ;;CALL FOR TYPEOUT
;*      .BYTE    N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;*      .BYTE    M      ;;M=1 OR 0
;*                               ;;1=TYPE LEADING ZEROS
;*                               ;;0=SUPPRESS LEADING ZEROS
;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;*$TYPOS OR $TYPOC
;CALL:
;*      MOV      NUM, -(SP)      ;;NUMBER TO BE TYPED
;*      TYPON      ;;CALL FOR TYPEOUT
;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;CALL:
;*      MOV      NUM, -(SP)      ;;NUMBER TO BE TYPED
;*      TYPOC      ;;CALL FOR TYPEOUT
;*$TYPOS: MOV      @ (SP), -(SP)      ;;PICKUP THE MODE
;MOV      1(SP), $OFILL      ;;LOAD ZERO FILL SWITCH
;MOV      (SP)+, $OMODE+1      ;;NUMBER OF DIGITS TO TYPE
;ADD      #2, (SP)      ;;ADJUST RETURN ADDRESS
;BR      $TYPON
;*$TYPOC: MOV      #1, $OFILL      ;;SET THE ZERO FILL SWITCH
;MOV      #6, $OMODE+1      ;;SET FOR SIX(6) DIGITS
;*$TYPON: MOV      #5, $OCNT      ;;SET THE ITERATION COUNT
;MOV      R3, -(SP)      ;;SAVE R3
;MOV      R4, -(SP)      ;;SAVE R4
;MOV      R5, -(SP)      ;;SAVE R5
;MOV      $OMODE+1, R4      ;;GET THE NUMBER OF DIGITS TO TYPE
;NEG      R4
;ADD      #6, R4      ;;SUBTRACT IT FOR MAX. ALLOWED
;MOV      R4, $OMODE      ;;SAVE IT FOR USE
;MOV      $OFILL, R4      ;;GET THE ZERO FILL SWITCH
;MOV      12(SP), R5      ;;PICKUP THE INPUT NUMBER
;CLR      R3      ;;CLEAR THE OUTPUT WORD
;1$: ROL      R5      ;;ROTATE MSB INTO "C"
;BR      3$      ;;GO DO MSB
;2$: ROL      R5      ;;FORM THIS DIGIT
;ROL      R5
;ROL      R5
;MOV      R5, R3
;3$: ROL      R3      ;;GET LSB OF THIS DIGIT
;DECB     $OMODE      ;;TYPE THIS DIGIT?
;BPL      6$      ;;BR IF NO
;BIC      #177770, R3      ;;GET RID OF JUNK
;BNE      4$      ;;TEST FOR 0
;TST      R4      ;;SUPPRESS THIS 0?
;BEQ      5$      ;;BR IF YES
;4$: INC      R4      ;;DON'T SUPPRESS ANYMORE 0'S

```

13609	054222	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
13610	054226	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
13611	054232	110337	054276		MOVB	R3,8\$::SAVE FOR TYPING
13612	054236				TYPE	8\$::GO TYPE THIS DIGIT
	054236	104401	054276		TYPEIT	,8\$	
					.DSABL	CRF	
13613	054242	105337	054300	6\$:	DECB	\$OCNT	::COUNT BY 1
13614	054246	003347			BGT	2\$::BR IF MORE TO DO
13615	054250	002402			BLT	7\$::BR IF DONE
13616	054252	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
13617	054254	000744			BR	2\$::GO DO THE LAST DIGIT
13618	054256	012605		7\$:	MOV	(SP)+,R5	::RESTORE R5
13619	054260	012604			MOV	(SP)+,R4	::RESTORE R4
13620	054262	012603			MOV	(SP)+,R3	::RESTORE R3
13621	054264	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
13622	054272	012616			MOV	(SP)+,(SP)	
13623	054274	000002			RTI		::RETURN
13624	054276	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
13625	054277	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
13626	054300	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
13627	054301	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
13628	054302	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

```

13630 .SBTTL ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
13631 ;*****
13632 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
13633 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
13634 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
13635 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
13636 ;*REPLACED WITH SPACES.
13637 ;*CALL:
13638 ;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
13639 ;*      TYPDS    ;;GO TO THE ROUTINE
13640 $TYPDS: PUSH      R0,R1,R2,R3,R5

13641 054304 010046      MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
13642 054306 010146      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
13643 054310 010246      BPL      1$      ;;BR IF INPUT IS POS.
13644 054312 010346      NEG      R5      ;;MAKE THE BINARY NUMBER POS.
13645 054314 010546      MOVB     #' -,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
13646 054316 012746 020200 1$: CLR      R0      ;;ZERO THE CONSTANTS INDEX
13647 054322 016605 000020  MOV      #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
13648 054326 100004      MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
13649 054330 005405      CLR      R2      ;;CLEAR THE BCD NUMBER
13650 054332 112766 000055 000001 2$: MOV      $DTBL(R0),R1      ;;GET THE CONSTANT
13651 054334 005000      SUB      R1,R5      ;;FORM THIS BCD DIGIT
13652 054336 012703 054520 3$: BLT      4$      ;;BR IF DONE
13653 054338 002402      INC      R2      ;;INCREASE THE BCD DIGIT BY 1
13654 054340 005202      BR      3$
13655 054342 000774      ADD      R1,R5      ;;ADD BACK THE CONSTANT
13656 054344 060105 4$: TST      R2      ;;CHECK IF BCD DIGIT=0
13657 054346 005702      BNE      5$      ;;FALL THROUGH IF 0
13658 054348 001002      TSTB     (SP)      ;;STILL DOING LEADING 0'S?
13659 054350 105716      BMI      7$      ;;BR IF YES
13660 054352 100407      ASLB     (SP)      ;;MSD?
13661 054354 106316 5$: BCC      6$      ;;BR IF NO
13662 054356 103003      MOVB     1(SP),-1(R3)      ;;YES--SET THE SIGN
13663 054358 116663 000001 177777 6$: BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
13664 054360 052702 000060 7$: BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
13665 054362 052702 000040      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
13666 054364 110223      TST      (R0)+      ;;JUST INCREMENTING
13667 054366 005720      CMP      R0,#10      ;;CHECK THE TABLE INDEX
13668 054368 020027 000010      BLT      2$      ;;GO DO THE NEXT DIGIT
13669 054370 002746      BGT      8$      ;;GO TO EXIT
13670 054372 003002      MOV      R5,R2      ;;GET THE LSD
13671 054374 010502      BR      6$      ;;GO CHANGE TO ASCII
13672 054376 000764      TSTB     (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO?
13673 054378 105726 8$: BPL      9$      ;;BR IF NO
13674 054380 100003      MOVB     -1(SP),-2(R3)      ;;YES- SET THE SIGN FOR TYPING
13675 054382 116663 177777 177776 9$: CLRB     (R3)      ;;SET THE TERMINATOR
13676 054384 105013      POP      R5,R3,R2,R1,R0
13677 054460 012605      MOV      (SP)+,R5
13678 054462 012603      MOV      (SP)+,R3
13679 054464 012602      MOV      (SP)+,R2
13680 054466 012601      MOV      (SP)+,R1
13681 054470 012600      MOV      (SP)+,R0

```


				TYPE	\$DBLK	
13677	054472			TYPEIT	,\$DBLK	::NOW TYPE THE NUMBER
	054472	104401	054520	.DSABL	CRF	
13678	054476	016666	000002	MOV	2(SP),4(SP)	::ADJUST THE STACK
13679	054504	012616		MOV	(SP)+,(SP)	
13680	054506	000002		RTI		::RETURN TO USER
13681	054510	023420		\$DTBL:	10000.	
13682	054512	001750			1000.	
13683	054514	000144			100.	
13684	054516	000012			10.	
13685	054520	000000	000000	\$DBLK:	.WORD	0,0,0,0
	054526	000000				

```
13687 .SBTTL ROUTINE TTY INPUT
13688 ;*****
13689 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
13690 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
13691 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
13692 ;*WHEN OPERATING IN TTY FLAG MODE.
13693 .ENABLE LSB
13694 $CKSWR:
13695 054530 005737 047246 TST XOCHAR ;;SOMETHING THERE?
13696 054534 001406 BEQ NOCH ;; GO ON IF NOT
13697 054536 013746 047246 MOV XOCHAR, -(SP) ;; USE IT
13698 054542 005037 047246 CLR XOCHAR
13699 054546 000137 054570 JMP CONTS1
13700 054552 105777 126064 NOCH: TSTB @TKS ;;CHAR THERE?
13701 054556 100136 BPL SWREND ;;IF NO, DON'T WAIT AROUND
13702 054560 117746 126060 MOVB @TKB, -(SP) ;;SAVE THE CHAR
13703 054564 042716 177600 BIC #C177, (SP) ;;STRIP-OFF THE ASCII
13704 054570 022716 000006 CONTS1: CMP #6, (SP) ;;IS IT CONTROL F?
13705 054574 001002 BNE 1$ ;;NO SKIP
13706 054576 004737 041054 CALL FIELDSERVICE
13707 054602 022716 000024 1$: CMP #24, (SP) ;;IS IT CONTROL T?
13708 054606 001002 BNE 16$ ;;NO - SKIP
13709 054610 004737 055152 CALL CONTT ;;YES - CALL CONTROL T ROUTINE
13710 054614 022716 000003 16$: CMP #3, (SP) ;;IS IT CONTROL C?
13711 054620 001454 BEQ 5$ ;;YES EXIT *****NOTE***** STACK IS SCREWF UP!
13712 054622 022716 000023 2$: CMP #23, (SP) ;;IS IT CONTROL S?
13713 054626 001002 BNE 17$ ;;NO - SKIP
13714 054630 004737 055226 CALL CONTS ;;YES - CALL CONTROL S ROUTINE
13715 054634 022716 000013 17$: CMP #13, (SP) ;;IS IT CONTROL K?
13716 054640 001005 BNE 6$ ;;NO - SKIP
13717 054642 TYPE $CNTLK ;;TYPE A +K
13718 054642 104401 055144 TYPEIT , $CNTLK
13718 054646 013706 002146 .DSABL CRF
13719 054652 000207 MOV CTLKVEC, SP ;;RESET KSP TO AFTER PATTERN EXEC ROUTINE
13720 054654 022737 000176 002636 6$: RETURN ;;RETURN TO PATTERN EXEC ROUTINE
13721 054662 001075 CMP #SWREG, SWR ;;IS THE SOFT-SWR SELECTED?
13722 054664 022716 000007 BNE CKEND ;;BRANCH IF NO
13723 054670 001072 CMP #7, (SP) ;;IS IT A CONTROL G?
13724 054672 005737 002062 BNE CKEND ;;NO, RETURN TO USER
13725 054676 001067 TST $AUTO ;;ARE WE RUNNING IN AUTO-MODE?
13726 054700 TYPE $CNTLG ;;BRANCH IF YES
13727 054700 104401 055750 TYPEIT , $CNTLG ;;ECHO THE CONTROL-G (+G)
13727 054704 $GTSWR: TYPE $MSWR ;;TYPE CURRENT CONTENTS
13728 054710 TYPEIT , $MSWR
13728 054710 017746 125722 .DSABL CRF
13729 054716 TYPOCT @SWR ;;OF THE SWR
13730 054722 104402 055766 MOV @SWR, (SP) ;;SAVE @SWR FOR TYPEOUT
13731 054724 TYPOC @SWR, (SP) ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
13732 054726 105777 125710 .DSABL CRF
13732 054726 105777 125710 TYPE $MNEW ;;PROMPT FOR NEW SWR
13732 054726 105777 125710 TYPEIT , $MNEW
13732 054726 105777 125710 .DSABL CRF
13732 054726 105777 125710 3$: CLR -(SP) ;;CLEAR COUNTER
13732 054726 105777 125710 4$: CLR -(SP) ;;THE NEW SWR
13732 054726 105777 125710 5$: TSTB @TKS ;;CHAR THERE?
```

```

13733 054732 100375      BPL      4$      ;;IF NOT TRY AGAIN
13734 054734 117746 125704  MOVB    @TKB,-(SP)  ;;PICK UP CHAR
13735 054740 042716 177600  BIC     #177,(SP)  ;;MAKE IT 7-BIT ASCII
13736 054744 021627 000003  CMP     (SP),#3    ;;IS IT A CONTROL-C?
13737 054750 001006      BNE     7$      ;;BRANCH IF NOT
13738 054752      5$:  TYPE    $CNTLC  ;;YES, ECHO CONTROL-C (↑C)
      054752 104401 055736  TYPEIT  , $CNTLC
      .DSABL CRF
13739 054756 062706 000006  ADD     #6,SP      ;;CLEAN UP STACK
13740 054762 000137 040534  JMP     QUIT      ;;CONTROL-C HALT
13741 054766 021627 000025  7$:  CMP     (SP),#25  ;;IS IT A CONTROL-U?
13742 054772 001005      BNE     9$      ;;BRANCH IF NOT
13743 054774      TYPE    $CNTLU   ;;YES, ECHO CONTROL-U (↑U)
      054774 104401 055743  TYPEIT  , $CNTLU
      .DSABL CRF
13744 055000 062706 000006  8$:  ADD     #6,SP      ;;IGNORE PREVIOUS INPUT
13745 055004 000746      BR      3$      ;;LET'S TRY IT AGAIN
13746 055006 021627 000015  9$:  CMP     (SP),#15  ;;IS IT A <CR>?
13747 055012 001024      BNE     13$     ;;BRANCH IF NO
13748 055014 005766 000004  TST     4(SP)      ;;YES, IS IT THE FIRST CHAR?
13749 055020 001403      BEQ     10$     ;;BRANCH IF YES
13750 055022 016677 000002 125606  MOV     2(SP),@SWR  ;;SAVE NEW SWR
13751 055030      10$: IF SWRFLG IS TRUE
      055030 005737 002566      TST SWRFLG
      055034 001403      BEQ L524
13752 055036 062706 000006      ADD     #6,SP      ;;CLEAR UP STACK
13753 055042      ELSE
      055042 000402
      055044      BR L525
13754 055044 062706 000010      ADD     #10,SP     ;;CLEAR UP STACK
13755 055050      END
      055050      L524:;;;;;;
13756 055050      TYPE    $CRLF   ;;ECHO <CR> AND <LF>
      055050 104401 002660  TYPEIT  , $CRLF
      .DSABL CRF
      L525:;;;;;;
13757 055054 000002  SWREND: RTI      ;;RETURN

```

13758	055056	062706	000002	CKEND:	ADD	#2,SP	;FIX STACK
13759	055062	000002			RTI		;RETURN
13760	055064	004737	047250	13\$:	CALL	\$TYPEC	::ECHO CHAR
13761	055070	021627	000060		CMP	(SP),#60	::CHAR < 0?
13762	055074	002420			BLT	15\$::BRANCH IF YES
13763	055076	021627	000067		CMP	(SP),#67	::CHAR > 7?
13764	055102	003015			BGT	15\$::BRANCH IF YES
13765	055104	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
13766	055110	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
13767	055114	001403			BEQ	14\$::BRANCH IF YES
13768	055116	006316			ASL	(SP)	::NO, SHIFT PRESENT
13769	055120	006316			ASL	(SP)	::CHAR OVER TO MAKE
13770	055122	006316			ASL	(SP)	::ROOM FOR NEW ONE.
13771	055124	005266	000002	14\$:	INC	2(SP)	::KEEP COUNT OF CHAR
13772	055130	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
13773	055134	000674			BR	4\$::GET THE NEXT ONE
13774	055136			15\$:	TYPE	\$QUES	::TYPE ?<CR><LF>
	055136	104401	002657		TYPEIT	, \$QUES	
					.DSABL	CRF	
13775	055142	000716			BR	8\$::SIMULATE CONTROL-U
13776	055144	136	113	015	\$CNTLK:	.ASCIZ	/↑K/<15><12>
	055147	012	000				;CONTROL K ASCII STRING
13777					.EVEN		
13778					.DSABL	LSB	

```

13781 055152          CONTT: SUBTST <<CONTROL T>>
;*****
;SUBTEST          CONTROL T
;*****
13782 055152          PUSH      R0
055152 010046          MOV R0,-(SP)
13783 055154          TYPE      $CRLF
055154 104401 002660  TYPEIT    $CRLF
                                .DSABL CRF
                                IF RLFLAG IS TRUE
13793 055160          TST RLFLAG
055160 005737 002126  BEQ L526
055164 001402
13794 055166          TYPE      MSG092          ;RELOCATED
055166 104401 070424  TYPEIT    MSG092
                                .DSABL CRF
                                END ;OF IF RLFLAG
13795 055172
055172
13796 055172          TYPE      MSG093          ;BANK=          L526:;;;;;
055172 104401 070440  TYPEIT    MSG093
                                .DSABL CRF
                                TYPOCS          BANK,,3          ;TYPE 3 DIGITS
13797 055176          MOV      BANK,-(SP)          ;;SAVE BANK FOR TYPEOUT
055176 013746 002102  TYPOS          ;;GO TYPE--OCTAL ASCII
055202 104403          .BYTE      3          ;;TYPE 3 DIGIT(S)
055204          003          ;;SUPPRESS LEADING ZEROS
055205          000
13798 055206          TYPE      MSG095          ;PAT=
055206 104401 070446  TYPEIT    MSG095
                                .DSABL CRF
                                TYPOCS          REALPAT,,2          ;TYPE 2 DIGITS
13799 055212          MOV      REALPAT,-(SP)          ;;SAVE REALPAT FOR TYPEOUT
055212 013746 002300  TYPOS          ;;GO TYPE--OCTAL ASCII
055216 104403          .BYTE      2          ;;TYPE 2 DIGIT(S)
055220          002          ;;SUPPRESS LEADING ZEROS
055221          000
13803 055222          POP      R0
055222 012600          MOV (SP)+,R0
13804 055224 000207
13805

```

13806 055226

CONTS: SUBTST <<CONTROL S & CONTROL Q>>

;*****

;SUBTEST CONTROL S & CONTROL Q

;*****

POF RO

;GET RID OF RETURN ADDRESS FROM STACK

MOV (SP),RO

13807 055226

055226 012600

13808 055230 105777 125406

13809 055234 100375

13810 055236 117716 125402

13811 055242 042716 177600

13812 055246

055246 021627 000021

055252 001003

13813 055254 000137 054570

13814 055260

055260 000401

055262

13815 055262 000762

13816 055264

055264

CONTS2: TSTB

@TKS

;WAIT FOR CHARACTER

BPL CONTS2

MOVB @TKB,(SP)

;REPLACE OVER OLD CHARACTER ON STACK

BIC #C177,(SP)

;STRIP ALL BUT ASCII

IF (SP) EQ #21

;IF IT IS A CONTROL Q

CMP (SP),#21

BNE L527

JMP CONTS1

ELSE

BR L530

L527:::::

BR CONTS2

END ;OF IF (SP)

L530:::::

```

13818 ;*****
13819 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
13820 ;*CALL:
13821 ;*      RDCHR      ;;INPUT A SINGLE CHARACTER FROM THE TTY
13822 ;*      RETURN HERE ;;CHARACTER IS ON THE STACK
13823 ;*                ;;WITH PARITY BIT STRIPPED OFF
13824 ;
13825
13826 055264 011646 $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC
13827 055266 016666 000004 000002 MOV      4(SP),2(SP)      ;;SAVE THE PS
13828 055274 105777 125342 1$:      TSTB      @TKS      ;;WAIT FOR
13829 055300 100375      BPL      1$      ;;A CHARACTER
13830 055302 117766 125336 000004 MOVB      @TKB,4(SP)      ;;READ THE TTY
13831 055310 042766 177600 000004 BIC      @C<177>,4(SP)      ;;GET RID OF JUNK IF ANY
13832 055316 026627 000004 000023 CMP      4(SP),#23      ;;IS IT A CONTROL-S?
13833 055324 001013      BNE      3$      ;;BRANCH IF NO
13834 055326 105777 125310 2$:      TSTB      @TKS      ;;WAIT FOR A CHARACTER
13835 055332 100375      BPL      2$      ;;LOOP UNTIL ITS THERE
13836 055334 117746 125304      MOVB      @TKB,-(SP)      ;;GET CHARACTER
13837 055340 042716 177600      BIC      @C177,(SP)      ;;MAKE IT 7-BIT ASCII
13838 055344 022627 000021      CMP      (SP)+,#21      ;;IS IT A CONTROL-Q?
13839 055350 001366      BNE      2$      ;;IF NOT DISCARD IT
13840 055352 000750      BR      1$      ;;YES, RESUME
13841 055354 026627 000004 000021 3$:      CMP      4(SP),#21      ;;IS IT A RANDOM CONTROL-Q?      ;R-C
13842 055362 001744      BEQ      1$      ;;BRANCH BACK IF SO      ;R-C
13843 055364 026627 000004 000140      CMP      4(SP),#140      ;;IS IT UPPER CASE?
13844 055372 002407      BLT      4$      ;;BRANCH IF YES
13845 055374 026627 000004 000175      CMP      4(SP),#175      ;;IS IT A SPECIAL CHAR?
13846 055402 003003      BGT      4$      ;;BRANCH IF YES
13847 055404 042766 000040 000004      BIC      #40,4(SP)      ;;MAKE IT UPPER CASE
13848 055412 000002 4$:      RTI      ;;GO BACK TO USER
13849 ;*****
13850 ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
13851 ;*CALL:
13852 ;*      RDLIN      ;;INPUT A STRING FROM THE TTY
13853 ;*      RETURN HERE ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
13854 ;*                ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
13855 055414 010346 $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
13856 055416 005046      CLR      -(SP)      ;;CLEAR THE RUBOUT KEY
13857 055420 012703 055712 1$:      MOV      @TTYIN,R3      ;;GET ADDRESS
13858 055424 022703 055736 2$:      CMP      @TTYIN+20.,R3      ;;BUFFER FULL?
13859 055430 101477      BLOS      8$      ;;BR IF YES
13860 055432 104411      RDCHR      ;;GO READ ONE CHARACTER FROM THE TTY
13861 055434 112613      MOVB      (SP)+,(R3)      ;;GET CHARACTER
13862 055436 122713 000003      CMPB      #3,(R3)      ;;IS IT A CONTROL-C?
13863 055442 001016      BNE      3$      ;;BRANCH IF NO
13864 055444      TYPE      $CNTLC      ;;TYPE A CONTROL-C (+C)
13865 055450 005726      TYPEIT , $CNTLC
13866 055452 012603      .DSABL CRF
13867 055454 032777 000400 125154 TST      (SP)+      ;;CLEAN RUBOUT KEY OFF OF THE STACK
13868 055462 001404      MOV      (SP)+,R3      ;;RESTORE R3
13869 055464 005037 002420      BIT      @BIT8,@SWR      ;;IS THERE A HALT FLAG SET IN THE SWR?
13870 055470 000137 040604      BEQ      11$      ;;BRANCH IF NOT TO HALT ROUTINE
13871 055474 000137 040534      CLR      STOPOK      ;;GET READY TO HALT PROGRAM
13872 055500 122713 000177 11$:      JMP      EXIT      ;;GO HALT PROGRAM
3$:      JMP      QUIT      ;;GOTO CONTROL-C HALT
          CMPB      #177,(R3)      ;;IS IT A RUBOUT

```

13873	055504	001022		BNE	5\$::BR IF NO
13874	055506	005716		TST	(SP)	::IS THIS THE FIRST RUBOUT?
13875	055510	001007		BNE	4\$::BR IF NO
13876	055512	112737	000134 055710	MOVB	#'\,10\$::TYPE A BACK SLASH
13877	055520			TYPE	10\$	
	055520	104401	055710	TYPEIT	,10\$	
				.DSABL	CRF	
13878	055524	012716	177777	MOV	#-1.(SP)	::SET THE RUBOUT KEY
13879	055530	005303		DEC	R3	::BACKUP BY ONE
13880	055532	020327	055712	CMP	R3,#\$TTYIN	::STACK EMPTY?
13881	055536	103434		BLO	8\$::BR IF YES
13882	055540	111337	055710	MOVB	(R3),10\$::SETUP TO TYPEOUT THE DELETED CHAR.
13883	055544			TYPE	10\$::GO TYPE
	055544	104401	055710	TYPEIT	,10\$	
				.DSABL	CRF	
13884	055550	000725		BR	2\$::GO READ ANOTHER CHAR.
13885	055552	005716		TST	(SP)	::RUBOUT KEY SET?
13886	055554	001406		BEQ	6\$::BR IF NO
13887	055556	112737	000134 055710	MOVB	#'\,10\$::TYPE A BACK SLASH
13888	055564			TYPE	10\$	
	055564	104401	055710	TYPEIT	,10\$	
				.DSABL	CRF	
13889	055570	005016		CLR	(SP)	::CLEAR THE RUBOUT KEY
13890	055572	122713	000025	CMPB	#25,(R3)	::IS CHARACTER A CTRL U?
13891	055576	001003		BNE	7\$::BR IF NO
13892	055600			TYPE	\$CNTLU	::TYPE A CONTROL "U"
	055600	104401	055743	TYPEIT	, \$CNTLU	
				.DSABL	CRF	
13893	055604	000705		BR	1\$::GO START OVER
13894	055606	122713	000022	CMPB	#22,(R3)	::IS CHARACTER A "+R"?
13895	055612	001011		BNE	9\$::BRANCH IF NO
13896	055614	105013		CLRB	(R3)	::CLEAR THE CHARACTER
13897	055616			TYPE	\$CRLF	::TYPE A "CR" & "LF"
	055616	104401	002660	TYPEIT	, \$CRLF	
				.DSABL	CRF	
13898	055622			TYPE	\$TTYIN	::TYPE THE INPUT STRING
	055622	104401	055712	TYPEIT	, \$TTYIN	
				.DSABL	CRF	
13899	055626	000676		BR	2\$::GO PICKUP ANOTHER CHACTER
13900	055630			TYPE	\$QUES	::TYPE A '?'
	055630	104401	002657	TYPEIT	, \$QUES	
				.DSABL	CRF	
13901	055634	000671		BR	1\$::CLEAR THE BUFFER AND LOOP
13902	055636	111337	055710	MOVB	(R3),10\$::ECHO THE CHARACTER
13903	055642			TYPE	10\$	
	055642	104401	055710	TYPEIT	,10\$	
				.DSABL	CRF	
13904	055646	122723	000015	CMPB	#15,(R3)+	::CHECK FOR RETURN
13905	055652	001264		BNE	2\$::LOOP IF NOT RETURN
13906	055654	105063	177777	CLRB	-1(R3)	::CLEAR RETURN (THE 15)
13907	055660			TYPE	\$LF	::TYPE A LINE FEED
	055660	104401	002661	TYPEIT	, \$LF	
				.DSABL	CRF	
13908	055664	005726		TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
13909	055666	012603		MOV	(SP)+,R3	::RESTORE R3
13910	055670	011646		MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
13911	055672	016666	000004 000002	MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT


```

13912 055700 012766 055712 000004      MOV      #TTYIN,4(SP)
13913 055706 000002                      RTI                      ;;RETURN
13914 055710      000                      10$: .BYTE 0           ;;STORAGE FOR ASCII CHAR. TO TYPE
13915 055711      000                      .BYTE 0           ;;TERMINATOR
13916 055712 000024      $TTYIN: .REPT 20.           ;;RESERVE SIZE BYTES FOR TTY INPUT
13919 055736      136      103      015 $CNTLC: .ASCIZ /+C/<15><12> ;;CONTROL "C"
      055741      012      000
13920 055743      136      125      015 $CNTLU: .ASCIZ /+U/<15><12> ;;CONTROL "U"
      055746      012      000
13921 055750      136      107      015 $CNTLG: .ASCIZ /+G/<15><12> ;;CONTROL "G"
      055753      012      000
13922 055755      015      012      123 $MSWR: .ASCIZ <15><12>/SWR = /
      055760      127      122      040
      055763      075      040      000
13923 055766      040      040      116 $MNEW: .ASCIZ / NEW = /
      055771      105      127      040
      055774      075      040      000
13924                      .EVEN

```

```

13926 .SBTTL ROUTINE READ AN OCTAL NUMBER FROM THE TTY
13927 ;*****
13928 ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
13929 ;*CHANGE IT TO BINARY.
13930 ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
13931 ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
13932 ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
13933 ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
13934 ;*CALL:
13935 ;*      RDOCT          ;;READ AN OCTAL NUMBER
13936 ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
13937 ;*                  ;;HIGH ORDER BITS ARE IN $HIOCT
13938 056000 011646 $RDOCT: MOV (SP),-(SP) ;;PROVIDE SPACE FOR THE
13939 056002 016666 000004 000002 MOV 4(SP),2(SP) ;;INPUT NUMBER
13940 056010 010046 PUSH R0,R1,R2
13941 056012 010146 MOV R0,-(SP)
13942 056014 010246 MOV R1,-(SP)
13943 056016 104412 MOV R2,-(SP)
13944 056020 012600 1$: RDLIN ;;READ AN ASCII LINE
13945 056022 010037 056126 MOV (SP)+,R0 ;;GET ADDRESS OF 1ST CHARACTER
13946 056026 005001 MOV R0,5$ ;;AND SAVE IT
13947 056030 005002 CLR R1 ;;CLEAR DATA WORD
13948 056032 112046 CLR R2
13949 056034 001420 2$: MOVB (R0)+,-(SP) ;;PICKUP THIS CHARACTER
13950 056036 122716 000060 BEQ 3$ ;;IF ZERO GET OUT
13951 056042 003026 CMPB #'0,(SP) ;;MAKE SURE THIS CHARACTER
13952 056044 122716 000067 CMPB #'7,(SP) ;;IS AN OCTAL DIGIT
13953 056050 002423 BLT 4$
13954 056052 006301 ASL R1 ;;*2
13955 056054 006102 ROL R2
13956 056056 006301 ASL R1 ;;*4
13957 056060 006102 ROL R2
13958 056062 006301 ASL R1 ;;*8
13959 056064 006102 ROL R2
13960 056066 042716 177770 BIC #1C7,(SP) ;;STRIP THE ASCII JUNK
13961 056072 062601 ADD (SP)+,R1 ;;ADD IN THIS DIGIT
13962 056074 000756 BR 2$ ;;LOOP
13963 056076 005726 3$: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
13964 056100 010166 000012 MOV R1,12(SP) ;;SAVE THE RESULT
13965 056104 010237 056146 MOV R2,$HIOCT
13966 056110 012602 POP R2,R1,R0
13967 056112 012601 MOV (SP)+,R2
13968 056114 012600 MOV (SP)+,R1
13969 056116 000002 MOV (SP)+,R0
13970 056120 005726 4$: RTI ;;RETURN
13971 056122 105010 TST (SP)+ ;;CLEAN PARTIAL FROM STACK
13972 056124 104401 CLR R0 ;;SET A TERMINATOR
13973 056126 000000 TYPEIT ;;TYPE UP THRU THE BAD CHAR.
13974 056130 104401 067776 .DSABL CRF
13975 056134 070016 .WORD 0
13976 056136 070016 TYPE MSG062 ;INPUT MUST BE A
13977 056138 070016 TYPEIT MSG062
13978 056140 070016 .DSABL CRF
13979 056142 070016 TYPE MSG063 ;N OCTAL
13980 056144 070016 TYPEIT ,MSG063

```

13972	056140				.DSABL	CRF		
	056140	104401	070027		TYPE	MSG064	:	NUMBER
					TYPEIT	MSG064		
					.DSABL	CRF		
13973	056144	000724			BR	1\$:	TRY AGAIN
13974	056146	000000			\$HIOCT:	.WORD	0	;;HIGH ORDER BITS GO HERE
13975					.SBTTL	ROUTINE READ A DECIMAL NUMBER FROM THE TTY		
13976								
13977								
13978					;*****			
13979					;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND			
13980					;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS			
13981					;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.			
13982					;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE			
13983					;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS			
13984					;*POSITIVE 32767 TO NEGATIVE 32768.			
13985					;*CALL:			
13986					RDDEC		:	READ A DECIMAL NUMBER
13987					RETURN HERE		:	NUMBER IS ON TOP OF THE STACK
13988								
13989	056150	011646			\$RDDEC:	MOV	(SP),-(SP)	;;PROVIDE SPACE FOR
13990	056152	016666	000004	000002		MOV	4(SP),2(SP)	;;THE INPUT NUMBER
13991	056160					PUSH	R0,R1,R2	
	056160	010046						MOV R0,-(SP)
	056162	010146						MOV R1,-(SP)
	056164	010246						MOV R2,-(SP)
13992	056166	104412			1\$:	RDLIN		;;READ AN ASCII LINE
13993	056170	012600				MOV	(SP)+,R0	;;ADDRESS OF 1ST CHAR.
13994	056172	010037	056316			MOV	R0,6\$;;SAVE IN CASE OF BAD INPUT
13995	056176	005046				CLR	-(SP)	;;CLEAR DATA WORD
13996	056200	005002				CLR	R2	;;SIGN SET POSITIVE
13997	056202	122710	000055			CMPB	#'-,(R0)	;;SEE IF A MINUS SIGN WAS TYPED
13998	056206	001001				BNE	2\$;;BR IF NO MINUS SIGN
13999	056210	112002				MOVB	(R0)+,R2	;;SAVE FOR LATER USE
14000	056212	112001			2\$:	MOVB	(R0)+,R1	;;PICKUP THIS CHARACTER
14001	056214	001424				BEQ	3\$;;GET OUT IF ZERO
14002	056216	122701	000060			CMPB	#'0,R1	;;MAKE SURE THIS CHARACTER
14003	056222	003032				BGT	5\$;;IS A DIGIT BETWEEN 0 & 9
14004	056224	122701	000071			CMPB	#'9,R1	
14005	056230	002427				BLT	5\$	
14006	056232	032716	170000			BIT	#C7777,(SP)	;;DON'T LET NUMBER GET TO BIG
14007	056236	001024				BNE	5\$;;BR IF NUMBER WOULD OVERFLOW
14008	056240	006316				ASL	(SP)	;;*2
14009	056242	011646				MOV	(SP),-(SP)	;;SAVE FOR LATER
14010	056244	006316				ASL	(SP)	;;*4
14011	056246	006316				ASL	(SP)	;;*8
14012	056250	062616				ADD	(SP)+,(SP)	;;*10
14013	056252	102416				BVS	5\$;;OVERFLOW ISN'T ALLOWED
14014	056254	162701	000060			SUB	#'0,R1	;;STRIP AWAY THE ASCII JUNK
14015	056260	060116				ADD	R1,(SP)	;;ADD IN THIS DIGIT
14016	056262	102412				BVS	5\$;;OVERFLOW ISN'T ALLOWED
14017	056264	000752				BR	2\$;;LOOP
14018	056266	005702			3\$:	TST	R2	;;CHECK IF NUMBER IS NEG
14019	056270	001401				BEQ	4\$;;BR IF NO
14020	056272	005416				NEG	(SP)	;;YES--NEGATE THE NUMBER
14021	056274	012666	000012		4\$:	MOV	(SP)+,12(SP)	;;SAVE THE RESULT
14022	056300					POP	R2,R1,R0	

NO	ADDRESS	OPERATION	OPERAND	COMMENT
	056300	012602		
	056302	012601		
	056304	012600		
14023	056306	000002	RTI	::RETURN
14024				
14025	056310	005726	5\$: TST (SP)+	::CLEAN PARTIAL NUMBER FROM STACK
14026	056312	105010	CLRB (RO)	::SET A TERMINATOR
14027	056314		TYPE	::TYPE THE INPUT UP TO BAD CHAR.
	056314	104401	TYPEIT	
			.DSABL CRF	
14028	056316	000000	6\$: .WORD 0	::POINTER GOES HERE
14029	056320		TYPE MSG062	;INPUT MUSST BE A
	056320	104401 067776	TYPEIT ,MSG062	
			.DSABL CRF	
14030	056324		TYPE MSG065	;DECIMAL
	056324	104401 070037	TYPEIT ,MSG065	
			.DSABL CRF	
14031	056330		TYPE MSG064	;NUMBER
	056330	104401 070027	TYPEIT ,MSG064	
			.DSABL CRF	
14032	056334	000714	BR 1\$::TRY AGAIN

```

14034 .SBTTL ROUTINE SAVE AND RESTORE R0-R5
14035
14036 ;*****
14037 ;*SAVE R0-R5
14038 ;*CALL:
14039 ;* SAVREG
14040 ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
14041 ;*
14042 ;*TOP---(+16)
14043 ;* +2---(+18)
14044 ;* +4---R5
14045 ;* +6---R4
14046 ;* +8---R3
14047 ;*+10---R2
14048 ;*+12---R1
14049 ;*+14---R0
14050
14051 $SAVREG:
14052 PUSH R0,R1,R2,R3,R4,R5
14053
14054 MOV R0,-(SP)
14055 MOV R1,-(SP)
14056 MOV R2,-(SP)
14057 MOV R3,-(SP)
14058 MOV R4,-(SP)
14059 MOV R5,-(SP)
14060
14061 MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
14062 MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
14063 MOV 22(SP),-(SP) ;;SAVE PS OF CALL
14064 MOV 22(SP),-(SP) ;;SAVE PC OF CALL
14065 RTI
14066
14067 ;*RESTORE R0-R5
14068 ;*CALL:
14069 ;* RESREG
14070 $RESREG:
14071 MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
14072 MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
14073 MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
14074 MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW
14075 POP R5,R4,R3,R2,R1,R0
14076
14077 MOV (SP)+,R5
14078 MOV (SP)+,R4
14079 MOV (SP)+,R3
14080 MOV (SP)+,R2
14081 MOV (SP)+,R1
14082 MOV (SP)+,R0
14083
14084 RTI

```

```

14070 .SBTTL ROUTINE RANDOM NUMBER GENERATOR
14071
14072 ;*****
14073 ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
14074 ;*WITH A RANGE OF 0 TO 2**(+33)-1.
14075 ;*CALL:
14076 ;*      CALL    $RAND          ;;CALL THE ROUTINE
14077 ;*      RETURN          ;;RETURN HERE THE RANDOM
14078 ;*                      ;;NUMBER WILL BE IN
14079 ;*                      ;;$HINUM,$LONUM
14080
14081 $RAND: PUSH    R0,R1,R2
14082          056432 010046          MOV R0,-(SP)
14083          056434 010146          MOV R1,-(SP)
14084          056436 010246          MOV R2,-(SP)
14085 14082 056440 013700 002604      MOV SEEDLO,R0      ;SET R0 WITH LOW
14086 14083 056444 013701 002602      MOV SEEDHI,R1      ;SET R1 WITH HIGH
14087 14084 056450 012702 000007      MOV #7,R2          ;SET SHIFT COUNT
14088 14085 056454 006300          1$: ASL R0              ;;SHIFT R0 LEFT AND
14089 14086 056456 006101          ROL R1                  ;;ROTATE CARRY INTO R1 AND
14090 14087 056460 077203          SOB R2,1$
14091 14088 056462 063700 002604      ADD SEEDLO,R0      ;ADD NUMBER TO MAKE X 129
14092 14089 056466 005501          ADC R1                  ;;PROPOGATE CARRY
14093 14090 056470 063701 002602      ADD SEEDHI,R1      ;ADD NUMBER TO MAKE X 129
14094 14091 056474 062700 001057      ADD #1057,R0        ;;ADD LOW CONSTANT
14095 14092 056500 005501          ADC R1                  ;;PROPOGATE CARRY
14096 14093 056502 062701 047401      ADD #47401,R1       ;;ADD HIGH CONSTANT
14097 14094 056506 010037 002604      MOV R0,SEEDLO      ;SAVE R0
14098 14095 056512 010137 002602      MOV R1,SEEDHI      ;SAVE R1
14099 14096 056516          POP R2,R1,R0
14100          056516 012602          MOV (SP)+,R2
14101          056520 012601          MOV (SP)+,R1
14102          056522 012600          MOV (SP)+,R0
14103 14097 056524 000207          RETURN

```

SEQ 0397

14100				.SBTTL ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
14101				*****
14102				;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
14103				;*UNSIGNED OCTAL ASCIIZ NUMBER.
14104				;*CALL
14105				;* MOV #PNTR,-(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
14106				;* CALL \$DB20 ;; CALL THE ROUTINE
14107				;* RETURN ;; THE ADDRESS OF THE FIRST ASCIIZ CHAR. IS ON THE STACK
14108				
14109				
14110	056526	104415		\$DB20: SAVREG ;; SAVE ALL REGISTERS
14111	056530	016601	000002	MOV 2(SP),R1 ;; PICKUP THE POINTER TO LOW WORD
14112	056534	012705	056645	MOV #\$OCTVL+13.,R5 ;; POINTER TO DATA TABLE
14113	056540	012704	000014	MOV #12.,R4 ;; DO ELEVEN CHARACTERS
14114	056544	012703	177770	MOV #+C7,R3 ;; MASK
14115	056550	012100		MOV (R1)+,R0 ;; LOWER WORD
14116	056552	012101		MOV (R1)+,R1 ;; HIGH WORD
14117	056554	005002		CLR R2 ;; TERMINATOR
14118	056556	110245		1\$: MOV R2,-(R5) ;; PUT CHARACTER IN DATA TABLE
14119	056560	010002		MOV R0,R2 ;; GET THIS DIGIT
14120	056562	005304		DEC R4 ;; COUNT THIS CHARACTER
14121	056564	003007		BGT 3\$;; BR IF NOT THE LAST DIGIT
14122	056566	001405		BEQ 2\$;; BR IF IT IS THE LAST DIGIT
14123	056570	005205		INC R5 ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
14124	056572	010566	000002	MOV R5,2(SP) ;; ASCIIZ CHAR. & PUT IT ON THE STACK
14125	056576	104416		RESREG ;; RESTORE ALL REGISTERS
14126	056600	000207		RETURN ;; RETURN TO USER
14127	056602	006203		2\$: ASR R3 ;; POSITION THE MASK FOR THE LAST DIGIT
14128	056604	006001		3\$: ROR R1 ;; POSITION THE BINARY NUMBER FOR
14129	056606	006000		ROR R0 ;; THE NEXT OCTAL DIGIT
14130	056610	006001		ROR R1
14131	056612	006000		ROR R0
14132	056614	006001		ROR R1
14133	056616	006000		ROR R0
14134	056620	040302		BIC R3,R2 ;; MASK OUT ALL JUNK
14135	056622	062702	000060	ADD #0,R2 ;; MAKE THIS CHAR. ASCII
14136	056626	000753		BR 1\$;; GO PUT IT IN THE DATA TABLE
14137	056630	000016		\$OCTVL: .REPT 14. ;; RESERVE DATA TABLE
14140	056634			\$OCT8=\$OCTVL+4 ;; POINTER TO 11 DIGIT NUMBER

14142			.SBTTL TABLES	
14143				
14144			.SBTTL APT MAILBOX ETABLE	
14145	056646		\$MAIL:	
14146	056646	000000	\$MSGTY: .WORD	0 ;:MESSAGE TYPE CODE
14147	056650	000000	\$FATAL: .WORD	0 ;:FATAL ERROR NUMBER (ERROR PC)
14148	056652	000000	\$TESTN: .WORD	0 ;:TEST PATTERN NUMBER
14149	056654	000000	\$PASS: .WORD	0 ;:PASS COUNT
14150	056656	000000	\$DEVCT: .WORD	0 ;:DEVICE COUNT
14151	056660	000000	\$UNIT: .WORD	0 ;:I/O UNIT NUMBER
14152	056662	000000	\$MSGAD: .WORD	0 ;:MESSAGE ADDRESS
14153	056664	000000	\$MSGLG: .WORD	0 ;:MESSAGE LENGTH
14154	056666		\$ETABLE:	0 ;:APT ENVIRONMENT TABLE
14155	056666	000	\$ENV: .BYTE	0 ;:ENVIRONMENT BYTE ;SET TO A 1 FOR APT AUTO MODE
14156			;NOTE: IF BIT #7 IS SET IN \$ENVM THE TABLE BELOW (BEGINNING AT \$MAMS1 AND	
14157			ENDING AT \$MADR4) MUST BE FILLED IN TO INDICATE THE PROPER AMOUNT OF	
14158			EACH TYPE OF MEMORY.	
14159	056667	000	\$ENVM: .BYTE	0 ;:ENVIRONMEN/ MODE
14160			;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE	
14161	056670	000101	\$SWREG: .WORD	101 ;:APT SWITCH REGISTER
14162	056672	000000	\$USWR: .WORD	0 ;:USED TO LIMIT THE NUMBER OF PASSES
14163	056674	000000	\$CPUOP: .WORD	0 ;:CPU TYPE,OPTIONS
14164			BITS 15-11=CPU TYPE	
14165			11/04=01,11/05=02,11/20=03,11/40=04,11/45=05	
14166			11/70=06,PDQ=07,Q=10	
14167			BIT 10=REAL TIME CLOCK	
14168			BIT 9=FLOATING POINT PROCESSOR	
14169			BIT 8=MEMORY MANAGEMENT	
14170	056676	001	\$MAMS1: .BYTE	1 ;:HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
14171	056677	004	\$MTYP1: .BYTE	4 ;:MEM. TYPE,BLK#1
14172			MEM.TYPE BYTE -- (HIGH BYTE)	
14173			900 NSEC CORE=001	
14174			300 NSEC BIPOLAR=002	
14175			PARITY MOS=003	
14176			ERROR CORRECTING MOS=004	
14177	056700	177776	\$MADR1: .WORD	177776 ;:HIGH ADDRESS,BLK#1
14178			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE	
14179	056702	000	\$MAMS2: .BYTE	0 ;:HIGH ADDRESS,M.S. BYTE
14180	056703	000	\$MTYP2: .BYTE	0 ;:MEM.TYPE,BLK#2
14181	056704	000000	\$MADR2: .WORD	0 ;:MEM.LAST ADDRESS,BLK#2
14182	056706	000	\$MAMS3: .BYTE	0 ;:HIGH ADDRESS,M.S.BYTE
14183	056707	000	\$MTYP3: .BYTE	0 ;:MEM.TYPE,BLK#3
14184	056710	000000	\$MADR3: .WORD	0 ;:MEM.LAST ADDRESS,BLK#3
14185	056712	000	\$MAMS4: .BYTE	0 ;:HIGH ADDRESS,M.S.BYTE
14186	056713	000	\$MTYP4: .BYTE	0 ;:MEM.TYPE,BLK#4
14187	056714	000000	\$MADR4: .WORD	0 ;:MEM.LAST ADDRESS,BLK#4
14188	056716	000000	\$VECT1: .WORD	0 ;:INTERRUPT VECTOR#1,BUS PRIORITY#1
14189	056720	000000	\$VECT2: .WORD	0 ;:INTERRUPT VECTOR#2BUS PRIORITY#2
14190	056722	000000	\$BASE: .WORD	0 ;:BASE ADDRESS OF EQUIPMENT UNDER TEST
14191	056724	000000	\$DEVM: .WORD	0 ;:DEVICE MAP
14192				
14193	056726	000000	\$CDW1: .WORD	0
14194	056730	000000	\$CDW2: .WORD	0
14195	056732	000000	\$DDW7: .WORD	0 ;:UFD MODE FLAG 1-UFD MODE


```

14197 ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
14198 ;ARE TO BE RUN FOR PARTICULAR MEMORIES
14199 ;
14200 ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
14201 ;BITO SET WILL RUN THE FIRST ENTRY IN THE TABLE, BITO SET
14202 ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
14203 ;
14204 ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
14205 ;
14206 056734 177777 ;FIELD SERVICE VALUE
14207 056736 177777 $DDW0: .WORD 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
14208 056740 177777 $DDW1: .WORD 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
14209 056742 177777 $DDW2: .WORD 177777 ;ECC PATTERNS 103777 TABLE = MKPAT:
14210 056744 177777 $DDW3: .WORD 177777 ;ECC PATTERNS 177777 TABLE = MKPAT:
14211 056746 177777 $DDW4: .WORD 177777 ;PARITY PATTERNS 003777 TABLE = MJPAT:
14212 056750 177777 $DDW5: .WORD 177777 ;PARITY PATTERNS 177774 TABLE = MJPAT:
14213 ;ETEND:
14214 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
14215 ;INTERFACE SPEC.
14216
14217
14218
14219 056750 $APTHD:
14220 056750 000000 $HIBTS: .WORD 0 ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
14221 056752 056646 $MBADR: .WORD $MAIL ;ADDRESS OF APT MAILBOX (BITS 0-15)
14222 056754 000043 $TSTM: .WORD 35. ;RUN TIM OF LONGEST TEST
14223 056756 001274 $PASTM: .WORD 700. ;RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
14224 056760 000000 $UNITM: .WORD 0. ;EXTRA RUN TIME OF A PASS FOR EACH ADDJTIONAL 128K (QV)
14225 056762 000041 .WORD $ETEND $MAIL/2 ;LENGTH MAILBOX ETABLE(WORDS)

```

.SBTTL ROUTINE TRAP DECODER

14227
14228
14229
14230
14231
14232
14233
14234

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

14235 056764 010046
14236 056766 016600 000002
14237 056772 005740
14238 056774 111000
14239 056776 006300
14240 057000 016000 057026
14241 057004 000200

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

14242
14243
14244
14245

;;THIS IS USE TO HANDLE THE "CETPRI" MACRO

14246 057006 011646
14247 057010 016666 000004 000002
14248 057016 000002
14249

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

14250 057020
057020 104401 065507

\$NOTRAP: TYPE MSG006 ;UNDEFINED TRAP INSTRUCTION
TYPEIT ,MSG006
.DSABL CRF

14251 057024 000000

\$HALT2: HALT

.SBTTL TRAP TABLE

;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;*BY THE "TRAP" INSTRUCTION.

	ROUTINE
14261 057026 057006	\$TRPAD: .WORD \$TRAP2
14262 057030 047122	\$TYPE ;CALL=TYPEIT TRAP+1(104401) TTY TYPEOUT ROUTINE
14263 057032 054102	\$TYPOC ;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
14264 057034 054056	\$TYPOS ;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
14265 057036 057020	\$NOTRAP;\$TYPON ;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
14266 057040 054304	\$TYPDS ;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
14267 057042 057020	\$NOTRAP;\$TYPBN ;CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER
14268	
14269 057044 054704	\$GTSWR ;CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING
14270 057046 054530	\$CKSWR ;CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR
14271	
14272 057050 055264	\$RDCHR ;CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
14273 057052 055414	\$RDLIN ;CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
14274 057054 056000	\$RDOCT ;CALL=RDOCT TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
14275 057056 056150	\$RDDEC ;CALL=RDDEC TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY
14276	
14277 057060 056336	\$SAVREG ;CALL=SAVREG TRAP+15(104415) SAVE R0-R5 ROUTINE
14278 057062 056374	\$RESREG ;CALL=RESREG TRAP+16(104406) RESTORE R0-R5 ROUTINE
14279	
14280 057064 034202	\$KERNEL ;CALL=KERNEL TRAP+17(104417) ENTER KERNEL MODE
14281 057066 034212	\$ENERGIZE;CALL=ENERGIZETRAP+20(104420) TURN ON MEMORY MANAGEMENT & TRAPS
14282 057070 034222	\$DEENERGI;CALL=DEENERGITRAP+21(104421) TURN OFF MEMORY MANAGEMENT & TRAPS
14283	
14284 057072 036310	\$KMAP ;CALL=KMAP TRAP+22(104422) MAP KERNEL 1 TO 1
14285	
14286 057074 034232	\$CACHN ;CALL=CACHON TRAP+23(104423) TURN CACHE ON
14287 057076 034256	\$CACHF ;CALL=CACHOFF TRAP+24(104424) TURN CACHE OFF
14288	
14289 057100 034274	\$LOADC ;CALL=LOADCSR TRAP+25(104425) LOAD CORRECT CSR
14290 057102 034370	\$READC ;CALL=READCSR TRAP+26(104426) READ CORRECT CSR
14291	
14292 057104 047456	\$PER01 ;CALL=PERR01 TRAP+27(104427) PROGRAM DETECTED ERROR
14293 057106 047504	\$PER02 ;CALL=PERR02 TRAP+30(104430) PROGRAM DETECTED ERROR
14294 057110 047532	\$PER03 ;CALL=PERR03 TRAP+31(104431) PROGRAM DETECTED ERROR
14295 057112 047562	\$PER04 ;CALL=PERR04 TRAP+32(104432) PROGRAM DETECTED ERROR
14296 057114 047644	\$PER07 ;CALL=PERR07 TRAP+33(104433) PROGRAM DETECTED ERROR
14297 057116 047666	\$PER10 ;CALL=PERR10 TRAP+34(104434) PROGRAM DETECTED ERROR
14298 057120 047716	\$PER11 ;CALL=PERR11 TRAP+35(104435) PROGRAM DETECTED ERROR
14299 057122 047736	\$PER12 ;CALL=PERR12 TRAP+36(104436) PROGRAM DETECTED ERROR
14300 057124 047760	\$PER13 ;CALL=PERR13 TRAP+37(104437) PROGRAM DETECTED ERROR
14301 057126 050000	\$PER14 ;CALL=PERR14 TRAP+40(104440) PROGRAM DETECTED ERROR
14302 057130 050022	\$PER15 ;CALL=PERR15 TRAP+41(104441) PROGRAM DETECTED ERROR
14303 057132 050044	\$PER16 ;CALL=PERR16 TRAP+42(104442) PROGRAM DETECTED ERROR
14304 057134 050064	\$PER17 ;CALL=PERR17 TRAP+43(104443) PROGRAM DETECTED ERROR
14305 057136 050102	\$PER20 ;CALL=PERR20 TRAP+44(104444) PROGRAM DETECTED ERROR
14306 057140 050120	\$PER21 ;CALL=PERR21 TRAP+45(104445) PROGRAM DETECTED ERROR
14307 057142 050140	\$PER22 ;CALL=PERR22 TRAP+46(104446) PROGRAM DETECTED ERROR
14308 057144 050156	\$PER23 ;CALL=PERR23 TRAP+47(104447) PROGRAM DETECTED ERROR
14309 057146 050174	\$PER24 ;CALL=PERR24 TRAP+50(104450) PROGRAM DETECTED ERROR
14310 057150 044756	\$PER25 ;CALL=PERR25 TRAP+51(104451) PROGRAM DETECTED ERROR

14311 057152 050364	\$PER26 ;CALL=PERR26	TRAP.52(104452) PROGRAM DETECTED ERROR	
14312 057154 050404	\$PER27 ;CALL=PERR27	TRAP.53(104453) PROGRAM DETECTED ERROR	
14313 057156 045204	\$PER30 ;CALL=PERR30	TRAP.54(104454) PROGRAM DETECTED ERROR	
14314 057160 050574	\$PER31 ;CALL=PERR31	TRAP.55(104455) PROGRAM DETECTED ERROR	
14315 057162 050672	\$PER32 ;CALL=PERR32	TRAP.56(104456) PROGRAM DETECTED ERROR	
14316 057164 050740	\$PER33 ;CALL=PERR33	TRAP.57(104457) PROGRAM DETECTED ERROR	
14317 057166 051020	\$PER34 ;CALL=PERR34	TRAP.60(104460) PROGRAM DETECTED ERROR	
14318 057170 051052	\$PER35 ;CALL=PERR35	TRAP.61(104461) PROGRAM DETECTED ERROR	
14319 057172 051106	\$PER36 ;CALL=PERR36	TRAP.62(104462) PROGRAM DETECTED ERROR	
14320 057174 051136	\$PER37 ;CALL=PERR37	TRAP.63(104463) PROGRAM DETECTED ERROR	;;ILC;REV B
14321 057176 051142	\$PER40 ;CALL=PERR40	TRAP.64(104464) PROGRAM DETECTED ERROR	;;ILC;REV B
14322 057200 057020	\$NOTRAP ;CALL=PERR41	TRAP.65(104465) PROGRAM DETECTED ERROR	
14323 057202 057020	\$NOTRAP ;CALL=PERR42	TRAP.66(104466) PROGRAM DETECTED ERROR	
14324 057204 057020	\$NOTRAP ;CALL=PERR43	TRAP.67(104467) PROGRAM DETECTED ERROR	
14325			
14326 057206 034570	\$ECCDIS ;CALL=ECCDIS	TRAP.70(104470) DISABLE ECC ON ALL CSR'S	
14327 057210 034604	\$ECC1DIS;CALL=ECC1DIS	TRAP.71(104471) DISABLE ECC ON 1 SELECTED CSR	
14328 057212 034616	\$ECCINIT;CALL=ECCINIT	TRAP.72(104472) INITIALIZE ALL MK11 CSR'S	
14329 057214 034632	\$ECC1INIT;CALL=ECC1INIT	TRAP.73(104473) INITIALIZE 1 SELECTED MK11 CSR	
14330 057216 034672	\$CBCSR ;CALL=CBCSR	TRAP.74(104474) WRITE GENERATED CHECKBITS IN ALL CSR'S	
14331 057220 034714	\$CB1CSR ;CALL=CB1CSR	TRAP.75(104475) WRITE GENERATED CHECKBITS IN 1 SELECTED CSR	
14332 057222 034734	\$WASSBE ;CALL=WASSBE	TRAP.76(104476) WAS THERE A SBE ON ANY CSR?	
14333 057224 035050	\$WAS1SBE;CALL=WAS1SBE	TRAP.77(104477) WAS THERE A SBE ON 1 SELECTED CSR?	
14334 057226 035100	\$WASDBE ;CALL=WASDBE	TRAP.100(104500) WAS THERE A DBE ON ANY CSR?	
14335 057230 035214	\$WAS1DBE;CALL=WAS1DBE	TRAP.101(104501) WAS THERE A DBE ON 1 SELECTED CSR?	
14336 057232 035244	\$CLRCR ;CALL=CLRCR	TRAP.102(104502) CLEAR ALL CSR'S	
14337 057234 035256	\$CLR1CSR;CALL=CLR1CSR	TRAP.103(104503) CLEAR 1 SELECTED CSR	
14338 057236 035266	\$CHKDIS ;CALL=CHKDIS	TRAP.104(104504) DISABLE ECC & WRITE CKBITS FROM ALL CSR S	
14339 057240 035302	\$CHK1DIS;CALL=CHK1DIS	TRAP.105(104505) DISABLE ECC & WRITE CKBITS FROM 1 CSR	
14340 057242 034644	\$ENASBE ;CALL=ENASBE	TRAP.106(104506) ENABLE TRAPS ON SBE'S FROM ALL CSR'S	
14341 057244 034660	\$ENA1SBE;CALL=ENA1SBE	TRAP.107(104507) ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR	
14342 057246 034410	\$TSTRD ;CALL=TSTREAD	TRAP.110(104510) TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES	
14343 057250 035362	\$INVALID;CALL=INVALID	TRAP.111(104511) INVALIDATE BACKGROUND PATTERN ON BANK	
14344 057252 035412	\$ERRGEN ;CALL=ERRGEN	TRAP.112(104512) TEST ERROR ADDRESS	
14345 057254 035662	\$CBREG ;CALL=CBREG	TRAP.113(104513) ENABLE CHECK BIT REGISTER	
14346 057256 035700	\$SYNREG ;CALL=SYNREG	TRAP.114(104514) ENABLE SYNDROME BIT REGISTER	
14347 057260 057020	\$NOTRAP		
14348 057262 057020	\$NOTRAP		
14349 057264 057020	\$NOTRAP		
14350 057266 057020	\$NOTRAP		
14351 057270 057020	\$NOTRAP		
14352 057272 057020	\$NOTRAP		
14355	177776	ST = 177776	;STATUS REGISTER

14358
14359
14360
14361
14362
14363
14364
14365
14366
14367
14368
14369
14370
14371
14372 057274
14373 057274 061762
14374 057276 064172
14375 057300 060320
14376 057302 060701
14377
14378 057304 060747
14379 057306 063501
14380 057310 060144
14381 057312 060557
14382
14383 057314 061005
14384 057316 063561
14385 057320 060162
14386 057322 060674
14387
14388 057324 061037
14389 057326 063561
14390 057330 060172
14391 057332 060674
14392
14393 057334 061105
14394 057336 063615
14395 057340 060202
14396 057342 060557
14397
14398 057344 061162
14399 057346 063615
14400 057350 060202
14401 057352 060557
14402
14403 057354 061207
14404 057356 063615
14405 057360 060202
14406 057362 060557
14407
14408 057364 063173
14409 057366 064714
14410 057370 060476
14411 057372 060557

.SBTTL TABLE ERROR POINTER

;*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: ;ERROR 1
EM24
DH13
DT13
DF11
;ERROR 2
EM2
DH1
DT1
DF2
;ERROR 3
EM3
DH3
DT3
DF9
;ERROR 4
EM4
DH3
DT4
DF9
;ERROR 5
EM5
DH5
DT5
DF2
;ERROR 6
EM6
DH5
DT5
DF2
;ERROR 7
EM7
DH5
DT5
DF2
;ERROR 10
EM53
DH25
DT25
DF2

14414			;ERROR	11
14415	057374	061247	EM11	
14416	057376	063741	DH7	
14417	057400	060234	DT7	
14418	057402	060603	DF3	
14419			;ERROR	12
14420	057404	061247	EM11	
14421	057406	063741	DH7	
14422	057410	060234	DT7	
14423	057412	060616	DF4	
14424			;ERROR	13
14425	057414	061271	EM12	
14426	057416	064051	DH10	
14427	057420	060264	DT10	
14428	057422	060557	DF2	
14429			;ERROR	14
14430	057424	061247	EM11	
14431	057426	063741	DH7	
14432	057430	060234	DT7	
14433	057432	060631	DF5	
14434			;ERROR	15
14435	057434	061247	EM11	
14436	057436	063741	DH7	
14437	057440	060234	DT7	
14438	057442	060644	DF6	
14439			;ERROR	16
14440	057444	061315	EM13	
14441	057446	064172	DH13	
14442	057450	060320	DT13	
14443	057452	060701	DF11	
14444			;ERROR	17
14445	057454	061347	EM14	
14446	057456	064172	DH13	
14447	057460	060320	DT13	
14448	057462	060701	DF11	
14449			;ERROR	20
14450	057464	061413	EM15	
14451	057466	064172	DH13	
14452	057470	060320	DT13	
14453	057472	060701	DF11	
14454			;ERROR	21
14455	057474	063222	EM55	
14456	057476	064752	DH26	
14457	057500	060510	DT26	
14458	057502	060557	DF2	
14459			;ERROR	22
14460	057504	061461	EM17	
14461	057506	063741	DH7	
14462	057510	060234	DT7	
14463	057512	060631	DF5	
14464			;ERROR	23
14465	057514	063042	EM50	
14466	057516	064566	DH23	
14467	057520	060434	DT23	
14468	057522	060712	DF13	

14471			;ERROR 24	
14472	057524	061521	EM19	
14473	057526	064172	DH13	
14474	057530	060320	DT13	
14475	057532	060701	DF11	
14476			;ERROR 25	
14477	057534	061573	EM20	
14478	057536	064172	DH13	
14479	057540	060320	DT13	
14480	057542	060701	DF11	
14481			;ERROR 26	
14482	057544	000000	0	;NO MESSAGE
14483	057546	064165	DH12	
14484	057550	060314	DT12	
14485	057552	060557	DF2	
14486			;ERROR 27	
14487	057554	061652	EM21	
14488	057556	064147	DH11	
14489	057560	060306	DT11	
14490	057562	060557	DF2	
14491			;ERROR 30	
14492	057564	061706	EM22	
14493	057566	064172	DH13	
14494	057570	060320	DT13	
14495	057572	060701	DF11	
14496			;ERROR 31	
14497	057574	000000	0	;NO MESSAGE
14498	057576	064267	DH14	
14499	057600	060342	DT14	
14500	057602	060557	DF2	
14501			;ERROR 32	
14502	057604	061733	EM23	
14503	057606	063615	DH5	
14504	057610	060202	DT5	
14505	057612	060557	DF2	
14506			;ERROR 33	
14507	057614	062041	EM25	
14508	057616	064346	DH15	
14509	057620	060360	DT16	
14510	057622	060657	DF7	
14511			;ERROR 34	
14512	057624	062066	EM26	
14513	057626	064465	DH16	
14514	057630	060410	DT17	
14515	057632	060603	DF3	

14518			;ERROR 35
14519	057634	063146	EM52
14520	057636	064714	DH25
14521	057640	060476	DT25
14522	057642	060557	DF2
14523			;ERROR 36
14524	057644	062137	EM27
14525	057646	064465	DH16
14526	057650	060410	DT17
14527	057652	060672	DF8
14528			;ERROR 37
14529	057654	062634	EM35
14530	057656	063741	DH7
14531	057660	060234	DT7
14532	057662	060603	DF3
14533			;ERROR 40
14534	057664	062227	EM29
14535	057666	063741	DH7
14536	057670	060234	DT7
14537	057672	060603	DF3
14538			;ERROR 41
14539	057674	062311	EM30
14540	057676	063741	DH7
14541	057700	060234	DT7
14542	057702	060631	DF5
14543			;ERROR 42
14544	057704	063343	EM60
14545	057706	064507	DH20
14546	057710	060434	DT23
14547	057712	060712	DF13
14548			;ERROR 43
14549	057714	062421	EM32
14550	057716	063741	DH7
14551	057720	060234	DT7
14552	057722	060603	DF3
14553			;ERROR 44
14554	057724	062526	EM33
14555	057726	063741	DH7
14556	057730	060234	DT7
14557	057732	060603	DF3
14558			;ERROR 45
14559	057734	063076	EM51
14560	057736	064645	DH24
14561	057740	060456	DT24
14562	057742	060722	DF14
14563			;ERROR 46
14564	057744	062721	EM36
14565	057746	063674	DH6
14566	057750	060220	DT6
14567	057752	060557	DF2

14570			;ERROR 47
14571	057754	062770	EM40
14572	057756	063536	DH2
14573	057760	060416	DT20
14574	057762	060557	DF2
14575			;ERROR 50
14576	057764	063243	EM56
14577	057766	064770	DH27
14578	057770	060516	DT27
14579	057772	060556	DF1
14580			;ERROR 51
14581	057774	063405	EM61
14582	057776	064645	DH24
14583	060000	060456	DT24
14584	060002	060722	DF14
14585			;ERROR 52
14586	060004	062227	EM29
14587	060006	064172	DH13
14588	060010	060320	DT13
14589	060012	060701	DF11
14590			;ERROR 53
14591	060014	062041	EM25
14592	060016	065044	DH30
14593	060020	060536	DT30
14594	060022	060740	DF16
14595			;ERROR 54
14596	060024	063275	EM57
14597	060026	065044	DH30
14598	060030	060536	DT30
14599	060032	060740	DF16
14600			;ERROR 55
14601	060034	061573	EM20
14602	060036	064645	DH24
14603	060040	060456	DT24
14604	060042	060722	DF14
14605			;ERROR 56
14606	060044	061573	EM20
14607	060046	065044	DH30
14608	060050	060536	DT30
14609	060052	060740	DF16
14610			;ERROR 57
14611	060054	061521	EM19
14612	060056	064645	DH24
14613	060060	060456	DT24
14614	060062	060722	DF14
14615			;ERROR 60
14616	060064	061315	EM13
14617	060066	064507	DH20
14618	060070	060434	DT23
14619	060072	060712	DF13
14620			;ERROR 61
14621	060074	062311	EM30
14622	060076	064507	DH20
14623	060100	060434	DT23
14624	060102	060712	DF13
14625			;ERROR 62
14626	060104	061573	EM20

14627	060106	064645	DH24
14628	060110	060456	DT24
14629	060112	060722	DF14
14630			;ERROR 63
14631	060114	061706	EM22
14632	060116	064645	DH24
14633	060120	060456	DT24
14634	060122	060722	DF14
14635			;ERROR 64
14636	060124	063426	EM62
14637	060126	063615	DH5
14638	060130	060202	DT5
14639	060132	060557	DF2
14640			;ERROR 65
14641	060134	061706	EM22
14642	060136	064507	DH20
14643	060140	060434	DT23
14644	060142	060712	DF13

14646						.SBTTL	ERROR DATA TAGS (DT)
14647	060144	002020	002034	002044	DT1:	.WORD	ERRPC, ADDRESS, GOOD, BAD, 0
	060152	002052	000000				
14648	060156	002020	000000		DT2:	.WORD	ERRPC, 0
14649	060162	002020	002036	002072	DT3:	.WORD	ERRPC, PADDRESS, PARCNT, 0
	060170	000000					
14650	060172	002020	002034	002070	DT4:	.WORD	ERRPC, ADDRESS, NEMCNT, 0
	060200	000000					
14651	060202	002020	177572	177574	DT5:	.WORD	ERRPC, MMRO, MMR1, MMR2, MMR3, CPUERR, 0
	060210	177576	172516	177766			
	060216	000000					
14652	060220	002020	002422	002400	DT6:	.WORD	ERRPC, APTPAR, LSIZE, APTECC, MSIZE, 0
	060226	002424	002402	000000			
14653	060234	002020	002176	002034	DT7:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, GOOD, BAD, BADXOR
	060242	002176	002044	002052			
	060250	002060					
14654	060252	002176	002176	002176		.WORD	DUMMY, DUMMY, DUMMY, DUMMY, 0
	060260	002176	000000				
14655	060264	002200	002202	002204	DT10:	.WORD	DETRO, DETR1, DETR2, DETR3, DETR4, DETR5, DETSP, DETPSW, 0
	060272	002206	002210	002212			
	060300	002214	002216	000000			
14656	060306	002020	002150	000000	DT11:	.WORD	ERRPC, CSR, 0
14657	060314	002150	000000		DT12:	.WORD	CSR, 0
14658	060320	002020	002176	002034	DT13:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, TSTDAT, TSTDAT+2, CHECK, CSR, 0
	060326	002176	002246	002250			
	060334	002314	002150	000000			
14659	060342	177746	177572	177574	DT14:	.WORD	CONTRL, MMRO, MMR1, MMR2, MMR3, CPUERR, 0
	060350	177576	172516	177766			
	060356	000000					
14660	060360	002020	002176	002176	DT16:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, GOOD2, GOOD3
	060366	002044	002046	002050			
14661	060374	002052	002054	002056		.WORD	BAD, BAD2, BAD3, DUMMY, DUMMY, 0
	060402	002176	002176	000000			
14662	060410	002020	002176	000000	DT17:	.WORD	ERRPC, DUMMY, 0
14663	060416	002020	002044	002052	DT20:	.WORD	ERRPC, GOOD, BAD, 0
	060424	000000					
14664	060426	002020	002176	000000	DT22:	.WORD	ERRPC, DUMMY, 0
14665	060434	002020	002176	002044	DT23:	.WORD	ERRPC, DUMMY, GOOD, BAD, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060442	002052	002176	002176			
	060450	002176	002176	000000			
14666	060456	002020	002176	002150	DT24:	.WORD	ERRPC, DUMMY, CSR, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060464	002176	002176	002176			
	060472	002176	000000				
14667	060476	002020	002044	002150	DT25:	.WORD	ERRPC, GOOD, CSR, CSRNO, 0
	060504	002152	000000				
14668	060510	002020	002052	000000	DT26:	.WORD	ERRPC, BAD, 0
14669	060516	002020	002176	002034	DT27:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060524	002176	002176	002176			
	060532	002176	000000				
14670	060536	002020	002176	002176	DT30:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, BAD, CSR, DUMMY, 0
	060544	002044	002052	002150			
	060552	002176	000000				

					.SBTTL	ERROR DATA FORMATS (DF)
14673						
14674	060556	000			DF1: .BYTE	0
14675	060557	000	000	000	DF2: .BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	060562	000	000	000		
	060565	000	000	000		
	060570	000	000	000		
	060573	000	000	000		
	060576	000	000	000		
	060601	000	000			
14676	060603	000	005	000	DF3: .BYTE	0,5,0,8.,0,0,0,3,6,2,4
	060606	010	000	000		
	060611	000	003	006		
	060614	002	004			
14677	060616	000	005	000	DF4: .BYTE	0,5,0,8.,0,8.,8.,3,6,2,4
	060621	010	000	010		
	060624	010	003	006		
	060627	002	004			
14678	060631	000	005	000	DF5: .BYTE	0,5,0,8.,9.,9.,9.,3,6,2,4
	060634	010	011	011		
	060637	011	003	006		
	060642	002	004			
14679	060644	000	005	000	DF6: .BYTE	0,5,0,8.,9.,8.,8.,3,6,2,4
	060647	010	011	010		
	060652	010	003	006		
	060655	002	004			
14680	060657	000	005	010	DF7: .BYTE	0,5,8.,0,0,9.,0,0,9.,2,4
	060662	000	000	011		
	060665	000	000	011		
	060670	002	004			
14681	060672	000	005		DF8: .BYTE	0,5
14682	060674	000	001	001	DF9: .BYTE	0,1,1,1,1
	060677	001	001			
14683	060701	000	005	000	DF11: .BYTE	0,5,0,8.,0,0,0,0,0
	060704	010	000	000		
	060707	000	000	000		
14684	060712	000	005	000	DF13: .BYTE	0,5,0,0,3,6,2,4
	060715	000	003	006		
	060720	002	004			
14685	060722	000	005	000	DF14: .BYTE	0,5,0,3,6,2,4
	060725	003	006	002		
	060730	004				
14686	060731	000	005	000	DF15: .BYTE	0,5,0,8.,3,6,4
	060734	010	003	006		
	060737	004				
14687	060740	000	005	010	DF16: .BYTE	0,5,8.,0,0,3,4
	060743	000	000	003		
	060746	004				

14690					
14696	060747	103	101	116	EM2: .SBTTL ERROR MESSAGES (EM)
	060752	047	124	040	.ASCIZ /CAN'T SET 22 BIT MODE IN MMR3/
	060755	123	105	124	
	060760	040	062	062	
	060763	040	102	111	
	060766	124	040	115	
	060771	117	104	105	
	060774	040	111	116	
	060777	040	115	115	
	061002	122	063	000	
14697	061005	120	101	122	EM3: .ASCIZ /PARITY ERROR(S) IN BANK 0/
	061010	111	124	131	
	061013	040	105	122	
	061016	122	117	122	
	061021	050	123	051	
	061024	040	111	116	
	061027	040	102	101	
	061032	116	113	040	
	061035	060	000		
14698	061037	116	117	116	EM4: .ASCIZ /NON-EXISTANT MEMORY (HOLES) IN BANK 0/
	061042	055	105	130	
	061045	111	123	124	
	061050	101	116	124	
	061053	040	115	105	
	061056	115	117	122	
	061061	131	040	050	
	061064	110	117	114	
	061067	105	123	051	
	061072	040	111	116	
	061075	040	102	101	
	061100	116	113	040	
	061103	060	000		
14699	061105	111	114	114	EM5: .ASCIZ /ILLEGAL OR RESERVED INSTRUCTION (TRAP TO 10)/
	061110	105	107	101	
	061113	114	040	117	
	061116	122	040	122	
	061121	105	123	105	
	061124	122	126	105	
	061127	104	040	111	
	061132	116	123	124	
	061135	122	125	103	
	061140	124	111	117	
	061143	116	040	050	
	061146	124	122	101	
	061151	120	040	124	
	061154	117	040	061	
	061157	060	051	000	
14700	061162	125	116	105	EM6: .ASCIZ /UNEXPECTED TRAP TO 4/
	061165	130	120	105	
	061170	103	124	105	
	061173	104	040	124	
	061176	122	101	120	
	061201	040	124	117	
	061204	040	064	000	
14701	061207	115	105	115	EM7: .ASCIZ /MEMORY MANAGEMENT (TRAP TO 250)/
	061212	117	122	131	

	061215	040	115	101	
	061220	116	101	107	
	061223	105	115	105	
	061226	116	124	040	
	061231	050	124	122	
	061234	101	120	040	
	061237	124	117	040	
	061242	062	065	060	
	061245	051	000		
14702	061247	115	105	115	EM11: .ASCIZ /MEMORY DATA ERROR/
	061252	117	122	131	
	061255	040	104	101	
	061260	124	101	040	
	061263	105	122	122	
	061266	117	122	000	
14703	061271	104	105	124	EM12: .ASCIZ /DETAILED ERROR DUMP/
	061274	101	111	114	
	061277	105	104	040	
	061302	105	122	122	
	061305	117	122	040	
	061310	104	125	115	
	061313	120	000		
14704	061315	115	111	123	EM13: .ASCIZ /MISSING EXPECTED SBE FLAG/
	061320	123	111	116	
	061323	107	040	105	
	061326	130	120	105	
	061331	103	124	105	
	061334	104	040	123	
	061337	102	105	040	
	061342	106	114	101	
	061345	107	000		
14705	061347	127	122	111	EM14: .ASCIZ /WRITE BYTE FAILED TO CLEAR SBE FLAG/
	061352	124	105	040	
	061355	102	131	124	
	061360	105	040	106	
	061363	101	111	114	
	061366	105	104	040	
	061371	124	117	040	
	061374	103	114	105	
	061377	101	122	040	
	061402	123	102	105	
	061405	040	106	114	
	061410	101	107	000	
14706	061413	106	101	111	EM15: .ASCIZ /FAILED TO GET INTERRUPT WITH DBE FLAG/
	061416	114	105	104	
	061421	040	124	117	
	061424	040	107	105	
	061427	124	040	111	
	061432	116	124	105	
	061435	122	122	125	
	061440	120	124	040	
	061443	127	111	124	
	061446	110	040	104	
	061451	102	105	040	
	061454	106	114	101	
	061457	107	000		
14707	061461	115	105	115	EM17: .ASCIZ /MEMORY DATA ERROR IN CHECK BITS/

	061464	117	122	131	
	061467	040	104	101	
	061472	124	101	040	
	061475	105	122	122	
	061500	117	122	040	
	061503	111	116	040	
	061506	103	110	105	
	061511	103	113	040	
	061514	102	111	124	
	061517	123	000		
14708	061521	123	102	105	EM19: .ASCIZ /SBE-DBE CAUSED PARITY TRAP WHEN INHIBITED/
	061524	055	104	102	
	061527	105	040	103	
	061532	101	125	123	
	061535	105	104	040	
	061540	120	101	122	
	061543	111	124	131	
	061546	040	124	122	
	061551	101	120	040	
	061554	127	110	105	
	061557	116	040	111	
	061562	116	110	111	
	061565	102	111	124	
14709	061570	105	104	000	
	061573	123	102	105	EM20: .ASCIZ /SBE-DBE DID NOT CAUSE PARITY TRAP WHEN ENABLED/
	061576	055	104	102	
	061601	105	040	104	
	061604	111	104	040	
	061607	116	117	124	
	061612	040	103	101	
	061615	125	123	105	
	061620	040	120	101	
	061623	122	111	124	
	061626	131	040	124	
	061631	122	101	120	
	061634	040	127	110	
	061637	105	116	040	
	061642	105	116	101	
	061645	102	114	105	
14710	061650	104	000		
	061652	123	102	105	EM21: .ASCIZ /SBE-DBE ON MASTER TEST WORD/
	061655	055	104	102	
	061660	105	040	117	
	061663	116	040	115	
	061666	101	123	124	
	061671	105	122	040	
	061674	124	105	123	
	061677	124	040	127	
	061702	117	122	104	
	061705	000			
14711	061706	115	111	123	EM22: .ASCIZ /MISSING EXPECTED DBE/
	061711	123	111	116	
	061714	107	040	105	
	061717	130	120	105	
	061722	103	124	105	
	061725	104	040	104	
	061730	102	105	000	

14712	061733	125	116	105	EM23:	.ASCIZ /UNEXPECTED PARITY TRAP/
	061736	130	120	105		
	061741	103	124	105		
	061744	104	040	120		
	061747	101	122	111		
	061752	124	131	040		
	061755	124	122	101		
	061760	120	000			
14713	061762	122	105	103	EM24:	.ASCIZ /RECEIVED DBE FLAG WHEN EXPECTING ONLY SBE FLAG/
	061765	105	111	126		
	061770	105	104	040		
	061773	104	102	105		
	061776	040	106	114		
	062001	101	107	040		
	062004	127	110	105		
	062007	116	040	105		
	062012	130	120	105		
	062015	103	124	111		
	062020	116	107	040		
	062023	117	116	114		
	062026	131	040	123		
	062031	102	105	040		
	062034	106	114	101		
	062037	107	000			
14714	062041	103	110	105	EM25:	.ASCIZ /CHECK BIT DATA ERROR/
	062044	103	113	040		
	062047	102	111	124		
	062052	040	104	101		
	062055	124	101	040		
	062060	105	122	122		
	062063	117	122	000		
14715	062066	101	104	104	EM26:	.ASCIZ /ADDRESS PARITY ERROR DID NOT CAUSE ABORT/
	062071	122	105	123		
	062074	123	040	120		
	062077	101	122	111		
	062102	124	131	040		
	062105	105	122	122		
	062110	117	122	040		
	062113	104	111	104		
	062116	040	116	117		
	062121	124	040	103		
	062124	101	125	123		
	062127	105	040	101		
	062132	102	117	122		
	062135	124	000			
14716	062137	105	103	103	EM27:	.ASCIZ /ECC INHIBIT MODE POINTER FAILURE - DID NOT PROTECT BANK/
	062142	040	111	116		
	062145	110	111	102		
	062150	111	124	040		
	062153	115	117	104		
	062156	105	040	120		
	062161	117	111	116		
	062164	124	105	122		
	062167	040	106	101		
	062172	111	114	125		
	062175	122	105	040		
	062200	055	040	104		

	062203	111	104	040	
	062206	116	117	124	
	062211	040	120	122	
	062214	117	124	105	
	062217	103	124	040	
	062222	102	101	116	
	062225	113	000		
14717	062227	103	117	122	EM29: .ASCIZ /CORRECTION FAILURE WITH ECC ENABLED ON FORCED SBE/
	062232	122	105	103	
	062235	124	111	117	
	062240	116	040	106	
	062243	101	111	114	
	062246	125	122	105	
	062251	040	127	111	
	062254	124	110	040	
	062257	105	103	103	
	062262	040	105	116	
	062265	101	102	114	
	062270	105	104	040	
	062273	117	116	040	
	062276	106	117	122	
	062301	103	105	104	
	062304	040	123	102	
	062307	105	000		
14718	062311	127	122	111	EM30: .ASCII /WRITE BYTE WITH ECC ENABLED FAILED TO CLEAR DATA AT/<CRLF>
	062314	124	105	040	
	062317	102	131	124	
	062322	105	040	127	
	062325	111	124	110	
	062330	040	105	103	
	062333	103	040	105	
	062336	116	101	102	
	062341	114	105	104	
	062344	040	106	101	
	062347	111	114	105	
	062352	104	040	124	
	062355	117	040	103	
	062360	114	105	101	
	062363	122	040	104	
	062366	101	124	101	
	062371	040	101	124	
	062374	200			
14719	062375	106	117	122	.ASCIZ /FORCED SBE LOCATION/
	062400	103	105	104	
	062403	040	123	102	
	062406	105	040	114	
	062411	117	103	101	
	062414	124	111	117	
	062417	116	000		
14720	062421	115	117	126	EM32: .ASCIZ /MOVB #360,(R2)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	062424	102	040	043	
	062427	063	066	060	
	062432	054	050	122	
	062435	062	051	053	
	062440	040	127	111	
	062443	124	110	040	
	062446	105	103	103	

	062451	040	105	116	
	062454	101	102	114	
	062457	105	104	040	
	062462	103	110	101	
	062465	116	107	105	
	062470	104	040	104	
	062473	101	124	101	
	062476	040	101	124	
	062501	040	106	117	
	062504	122	103	105	
	062507	104	040	104	
	062512	102	105	040	
	062515	114	117	103	
	062520	101	124	111	
	062523	117	116	000	
14721	062526	115	117	126	EM33: .ASCIZ /MOV #177400,(R1) WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	062531	040	043	061	
	062534	067	067	064	
	062537	060	060	054	
	062542	050	122	061	
	062545	051	040	127	
	062550	111	124	110	
	062553	040	105	103	
	062556	103	040	105	
	062561	116	101	102	
	062564	114	105	104	
	062567	040	103	110	
	062572	101	116	107	
	062575	105	104	040	
	062600	104	101	124	
	062603	101	040	101	
	062606	124	040	106	
	062611	117	122	103	
	062614	105	104	040	
	062617	104	102	105	
	062622	040	114	117	
	062625	103	101	124	
	062630	111	117	116	
	062633	000			
14722	062634	125	116	105	EM35: .ASCIZ /UNEXPECTED CORRECTION WITH ECC DISABLE ON FORCED SBE/
	062637	130	120	105	
	062642	103	124	105	
	062645	104	040	103	
	062650	117	122	122	
	062653	105	103	124	
	062656	111	117	116	
	062661	040	127	111	
	062664	124	110	040	
	062667	105	103	103	
	062672	040	104	111	
	062675	123	101	102	
	062700	114	105	040	
	062703	117	116	040	
	062706	106	117	122	
	062711	103	105	104	
	062714	040	123	102	
	062717	105	000		

14723	062721	101	120	124	EM36:	.ASCIZ /APT SIZE DISAGREES WITH PROGRAM SIZING/
	062724	040	123	111		
	062727	132	105	040		
	062732	104	111	123		
	062735	101	107	122		
	062740	105	105	123		
	062743	040	127	111		
	062746	124	110	040		
	062751	120	122	117		
	062754	107	122	101		
	062757	115	040	123		
	062762	111	132	111		
	062765	116	107	000		
14724	062770	102	122	101	EM40:	.ASCIZ /BRANCH GOBBLE FAILED CONDITION CODES TEST/
	062773	116	103	110		
	062776	040	107	117		
	063001	102	102	114		
	063004	105	040	106		
	063007	101	111	114		
	063012	105	104	040		
	063015	103	117	116		
	063020	104	111	124		
	063023	111	117	116		
	063026	040	103	117		
	063031	104	105	123		
	063034	040	124	105		
	063037	123	124	000		
14725	063042	102	101	104	EM50:	.ASCIZ /BAD ERROR ADDRESS GENERATED/
	063045	040	105	122		
	063050	122	117	122		
	063053	040	101	104		
	063056	104	122	105		
	063061	123	123	040		
	063064	107	105	116		
	063067	105	122	101		
	063072	124	105	104		
	063075	000				
14726	063076	106	114	101	EM51:	.ASCIZ /FLAGS NOT SET ON FORCED UNCORRECTED SBE/
	063101	107	123	040		
	063104	116	117	124		
	063107	040	123	105		
	063112	124	040	117		
	063115	116	040	106		
	063120	117	122	103		
	063123	105	104	040		
	063126	125	116	103		
	063131	117	122	122		
	063134	105	103	124		
	063137	105	104	040		
	063142	123	102	105		
	063145	000				
14727	063146	102	111	124	EM52:	.ASCIZ /BIT SET ERROR IN CSR/
	063151	040	123	105		
	063154	124	040	105		
	063157	122	122	117		
	063162	122	040	111		
	063165	116	040	103		

	063170	123	122	000	
14728	063173	102	111	124	EM53: .ASCIZ /BIT CLEAR ERROR IN CSR/
	063176	040	103	114	
	063201	105	101	122	
	063204	040	105	122	
	063207	122	117	122	
	063212	040	111	116	
	063215	040	103	123	
	063220	122	000		
14729	063222	111	114	114	EM55: .ASCIZ /ILLEGAL CSR TYPE/
	063225	105	107	101	
	063230	114	040	103	
	063233	123	122	040	
	063236	124	131	120	
	063241	105	000		
14730	063243	102	101	104	EM56: .ASCIZ /BAD PARITY TRAP GENERATED/
	063246	040	120	101	
	063251	122	111	124	
	063254	131	040	124	
	063257	122	101	120	
	063262	040	107	105	
	063265	116	105	122	
	063270	101	124	105	
	063273	104	000		
14731	063275	127	122	117	EM57: .ASCIZ /WRONG CHECK BIT READ BACK FROM MEMORY/
	063300	116	107	040	
	063303	103	110	105	
	063306	103	113	040	
	063311	102	111	124	
	063314	040	122	105	
	063317	101	104	040	
	063322	102	101	103	
	063325	113	040	106	
	063330	122	117	115	
	063333	040	115	105	
	063336	115	117	122	
	063341	131	000		
14732	063343	127	122	117	EM60: .ASCIZ /WRONG SYNDROME BITS READ INTO CSR/
	063346	116	107	040	
	063351	123	131	116	
	063354	104	122	117	
	063357	115	105	040	
	063362	102	111	124	
	063365	123	040	122	
	063370	105	101	104	
	063373	040	111	116	
	063376	124	117	040	
	063401	103	123	122	
	063404	000			
14733	063405	103	123	122	EM61: .ASCIZ /CSR UPDATE ERROR/
	063410	040	125	120	
	063413	104	101	124	
	063416	105	040	105	
	063421	122	122	117	
	063424	122	000		
14734	063426	120	122	117	EM62: .ASCIZ /PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC/
	063431	103	105	123	

F1

CVMJAAO MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15-Feb-85 16:10 Page 469 8
ERROR MESSAGES (EM)

SEQ 0419

063434	123	117	122
063437	040	116	117
063442	124	040	123
063445	125	120	120
063450	117	122	124
063453	105	104	040
063456	102	131	040
063461	124	110	111
063464	123	040	104
063467	111	101	107
063472	116	117	123
063475	124	111	103
063500	000		

14737	.SBTTL ERROR DATA HEADERS (DH)				
14738	063501	040	040	120	DH1: .ASCIZ / PC DEV ADD GOOD BAD/
	063504	103	040	040	
	063507	040	104	105	
	063512	126	040	101	
	063515	104	104	040	
	063520	040	040	107	
	063523	117	117	104	
	063526	040	040	040	
	063531	040	102	101	
	063534	104	000		
14739	063536	040	040	120	DH2: .ASCIZ / PC GD CC BD-CC/
	063541	103	040	040	
	063544	040	107	104	
	063547	055	103	103	
	063552	040	102	104	
	063555	055	103	103	
	063560	000			
14740	063561	040	040	120	DH3: .ASCIZ / PC 1ST ADD # OF ERRORS/
	063564	103	040	040	
	063567	040	061	123	
	063572	124	040	101	
	063575	104	104	040	
	063600	040	043	040	
	063603	117	106	040	
	063606	105	122	122	
	063611	117	122	123	
	063614	000			
14741	063615	040	040	120	DH5: .ASCIZ / PC MMRO MMR1 MMR2 MMR3 CPUERR/
	063620	103	040	040	
	063623	040	040	040	
	063626	115	115	122	
	063631	060	040	040	
	063634	040	040	115	
	063637	115	122	061	
	063642	040	040	040	
	063645	040	115	115	
	063650	122	062	040	
	063653	040	040	040	
	063656	115	115	122	
	063661	063	040	040	
	063664	040	103	120	
	063667	125	105	122	
	063672	122	000		
14742	063674	040	040	120	DH6: .ASCIZ / PC APTPAR LSIZE APTECC MSIZE/
	063677	103	040	040	
	063702	040	101	120	
	063705	124	120	101	
	063710	122	040	040	
	063713	040	114	123	
	063716	111	132	105	
	063721	040	040	101	
	063724	120	124	105	
	063727	103	103	040	
	063732	040	115	123	
	063735	111	132	105	
	063740	000			

14743	063741	040	040	120	DH7:	.ASCII	/	PC	BANK	VADD	PADD	GOOD/
	063744	103	040	040								
	063747	040	040	102								
	063752	101	116	113								
	063755	040	040	126								
	063760	101	104	104								
	063763	040	040	040								
	063766	040	040	120								
	063771	101	104	104								
	063774	040	040	040								
	063777	040	040	107								
	064002	117	117	104								
14744	064005	040	040	040		.ASCIZ	/		BAD	XOR	CSR	MTYP INT PAT/
	064010	040	040	102								
	064013	101	104	040								
	064016	040	040	040								
	064021	040	130	117								
	064024	122	040	040								
	064027	103	123	122								
	064032	040	040	115								
	064035	124	131	120								
	064040	040	111	116								
	064043	124	040	120								
	064046	101	124	000								
14745	064051	040	040	122	DH10:	.ASCIZ	/	R0	R1	R2	R3	R4 R5 SP PSW/
	064054	060	040	040								
	064057	040	040	040								
	064062	040	122	061								
	064065	040	040	040								
	064070	040	040	040								
	064073	122	062	040								
	064076	040	040	040								
	064101	040	040	122								
	064104	063	040	040								
	064107	040	040	040								
	064112	040	122	064								
	064115	040	040	040								
	064120	040	040	040								
	064123	122	065	040								
	064126	040	040	040								
	054131	040	040	123								
	064134	120	040	040								
	064137	040	040	040								
	064142	040	120	123								
	064145	127	000									
14746	064147	040	040	120	DH11:	.ASCIZ	/	PC	CSR/			
	064152	103	040	040								
	064155	040	040	040								
	064160	040	103	123								
	064163	122	000									
14747	064165	040	103	123	DH12:	.ASCIZ	/	CSR/				
	064170	122	000									
14748	064172	040	040	120	DH13:	.ASCII	/	PC	BANK	VADD	PADD	WROTE

[illegible]

	064461	120	101	124						
	064464	000								
14753	064465	040	040	120	DH16:	.ASCIZ	/	PC	BANK/	
	064470	103	040	040						
	064473	040	040	102						
	064476	101	116	113						
	064501	000								
14754	064502	040	040	120	DH19:	.ASCIZ	/	PC/		
	064505	103	000							
14755	064507	040	040	120	DH20:	.ASCIZ	/	PC	BANK GD-CSR (CSR) CSR MTYP INT PAT/	
	064512	103	040	040						
	064515	040	040	102						
	064520	101	116	113						
	064523	040	107	104						
	064526	055	103	123						
	064531	122	040	040						
	064534	050	103	123						
	064537	122	051	040						
	064542	040	103	123						
	064545	122	040	040						
	064550	115	124	131						
	064553	120	040	111						
	064556	116	124	040						
	064561	040	120	101						
	064564	124	000							
14756	064566	040	040	120	DH23:	.ASCIZ	/	PC	BANK GD-ERR BAD-ERR CSR MTYP INT PAT/	
	064571	103	040	040						
	064574	040	040	102						
	064577	101	116	113						
	064602	040	107	104						
	064605	055	105	122						
	064610	122	040	102						
	064613	101	104	055						
	064616	105	122	122						
	064621	040	103	123						
	064624	122	040	040						
	064627	115	124	131						
	064632	120	040	111						
	064635	116	124	040						
	064640	040	120	101						
	064643	124	000							
14757	064645	040	040	120	DH24:	.ASCIZ	/	PC	BANK (CSR) CSR MTYP INT PAT/	
	064650	103	040	040						
	064653	040	040	102						
	064656	101	116	113						
	064661	040	040	050						
	064664	103	123	122						
	064667	051	040	103						
	064672	123	122	040						
	064675	040	115	124						
	064700	131	120	040						
	064703	111	116	124						
	064706	040	040	120						
	064711	101	124	000						
14758	064714	040	040	120	DH25:	.ASCIZ	/	PC	GD-DAT (CSR) CSRNO/	
	064717	103	040	040						
	064722	040	040	107						

SEQ 0425

[illegible]

14769	065367	200	040	040	MSG004: .ASCII <CRLF>/	012345670123456701234567/
	065372	040	040	040		
	065375	040	040	040		
	065400	060	061	062		
	065403	063	064	065		
	065406	066	067	060		
	065411	061	062	063		
	065414	064	065	066		
	065417	067	060	061		
	065422	062	063	064		
	065425	065	066	067		
14770	065430	060	061	062	.ASCIZ /012345670123456701234567012345670123/	
	065433	063	064	065		
	065436	066	067	060		
	065441	061	062	063		
	065444	064	065	066		
	065447	067	060	061		
	065452	062	063	064		
	065455	065	066	067		
	065460	060	061	062		
	065463	063	064	065		
	065466	066	067	060		
	065471	061	062	063		
	065474	000				
14771	065475	200	105	122	MSG005: .ASCIZ <CRLF>/ERRORS /	
	065500	122	117	122		
	065503	123	040	040		
	065506	000				
14772	065507	200	125	116	MSG006: .ASCIZ <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>	
	065512	104	105	106		
	065515	111	116	105		
	065520	104	040	124		
	065523	122	101	120		
	065526	040	111	116		
	065531	123	124	122		
	065534	125	103	124		
	065537	111	117	116		
	065542	032	000			
14773	065544	200	115	105	MSG009: .ASCIZ <CRLF>/MENTYPE /	;MEMORY TYPE
	065547	115	124	131		
	065552	120	105	040		
	065555	000				
14774	065556	200	120	122	MSG010: .ASCIZ <CRLF>/PROTECT /	;MEMORY PROTECTED
	065561	117	124	105		
	065564	103	124	040		
	065567	000				
14775	065570	040	040	040	MSG11A: .ASCIZ /	1 1 1 1 1 1 1/
	065573	040	061	040		
	065576	040	040	040		
	065601	040	040	040		
	065604	061	040	040		
	065607	040	040	040		
	065612	040	040	061		
	065615	040	040	040		
	065620	040	040	040		
	065623	040	061	040		
	065626	040	040	040		

	065631	040	040	040	
	065634	061	040	040	
	065637	040	040	040	
	065642	040	040	061	
	065645	040	040	040	
	065650	040	040	040	
	065653	040	061	000	
14776	065656	040	040	040	MSG011: .ASCIZ / 0 1 2 3 4 5 6/
	065661	040	060	040	
	065664	040	040	040	
	065667	040	040	040	
	065672	061	040	040	
	065675	040	040	040	
	065700	040	040	062	
	065703	040	040	040	
	065706	040	040	040	
	065711	040	063	040	
	065714	040	040	040	
	065717	040	040	040	
	065722	064	040	040	
	065725	040	040	040	
	065730	040	040	065	
	065733	040	040	040	
	065736	040	040	040	
	065741	040	066	000	
14777	065744	064	065	066	MSG012: .ASCIZ /456701234567012345670123456701234567012345670123456701234567/
	065747	067	060	061	
	065752	062	063	064	
	065755	065	066	067	
	065760	060	061	062	
	065763	063	064	065	
	065766	066	067	060	
	065771	061	062	063	
	065774	064	065	066	
	065777	067	060	061	
	066002	062	063	064	
	066005	065	066	067	
	066010	060	061	062	
	066013	063	064	065	
	066016	066	067	060	
	066021	061	062	063	
	066024	064	065	066	
	066027	067	060	061	
	066032	062	063	064	
	066035	065	066	067	
	066040	000			
14778	066041	061	000		MSG11B: .ASCIZ /1/
14779	066043	067	000		MSG11C: .ASCIZ /7/
14780	066045	060	061	062	MSG11D: .ASCIZ /01234567/
	066050	063	064	065	
	066053	066	067	000	
14781	066056	130	000		MSG013: .ASCIZ /X/
14782	066060	040	000		MSG014: .ASCIZ / / ;SPACE
14783	066062	000	000		MSG015: .BYTE 0,0 ;FOR SINGLE ASCII CHARACTERS & TERMINATOR
14784	066064	200	103	123	MSG016: .ASCIZ <CRLF>/CSR /
	066067	122	040	040	
	066072	040	040	040	

14785	066075	000				
	066076	040	040	040	MSG017: .ASCIZ / /	;8 SPACES
	066101	040	040	040		
	066104	040	040	000		
14786	066107	040	040	000	MSG018: .ASCIZ / /	;2 SPACES
14787	066112	040	040	040	MSG019: .ASCIZ / /	;3 SPACES
	066115	000				
14788	066116	200	106	123	MSG020: .ASCIZ <CRLF>/FS COMMAND MODE/	
	066121	040	103	117		
	066124	115	115	101		
	066127	116	104	040		
	066132	115	117	104		
	066135	105	000			
14789	066137	200	103	117	MSG021: .ASCII <CRLF>/COMMANDS AVAILABLE:/	
	066142	115	115	101		
	066145	116	104	123		
	066150	040	101	126		
	066153	101	111	114		
	066156	101	102	114		
	066161	105	072			
14790	066163	200	060	040	.ASCII <CRLF>/0 = EXIT/	
	066166	075	040	105		
	066171	130	111	124		
14791	066174	200	061	040	.ASCII <CRLF>/1 = READ CSR/	
	066177	075	040	122		
	066202	105	101	104		
	066205	040	103	123		
	066210	122				
14792	066211	200	062	040	.ASCII <CRLF>/2 = LOAD CSR/	
	066214	075	040	114		
	066217	117	101	104		
	066222	040	103	123		
	066225	122				
14793	066226	200	063	040	.ASCII <CRLF>/3 = EXAMINE MEMORY/	
	066231	075	040	105		
	066234	130	101	115		
	066237	111	116	105		
	066242	040	115	105		
	066245	115	117	122		
	066250	131				
14794	066251	200	064	040	.ASCII <CRLF>/4 = MODIFY MEMORY/	
	066254	075	040	115		
	066257	117	104	111		
	066262	106	131	040		
	066265	115	105	115		
	066270	117	122	131		
14795	066273	200	065	040	.ASCII <CRLF>/5 = SELECT BANK & TEST/	
	066276	075	040	123		
	066301	105	114	105		
	066304	103	124	040		
	066307	102	101	116		
	066312	113	040	046		
	066315	040	124	105		
	066320	123	124			
14796	066322	200	066	040	.ASCII <CRLF>/6 = TYPE CONFIG MAP/	
	066325	075	040	124		
	066330	131	120	105		

	066333	040	103	117	
	066336	116	106	111	
	066341	107	040	115	
	066344	101	120		
14797	066346	200	067	040	ASCII <CRLF>/7 = SOB A-LONG TEST/
	066351	075	040	123	
	066354	117	102	055	
	066357	101	055	114	
	066362	117	116	107	
	066365	040	124	105	
	066370	123	124		
14798	066372	200	070	040	.ASCII <CRLF>/8 = ERROR SUMMARY/
	066375	075	040	105	
	066400	122	122	117	
	066403	122	040	123	
	066406	125	115	115	
	066411	101	122	131	
14799	066414	200	071	075	.ASCII <CRLF>/9= REFRESH TEST/
	066417	040	040	122	
	066422	105	106	122	
	066425	105	123	110	
	066430	040	124	105	
	066433	123	124		
14800	066435	200	061	060	.ASCII <CRLF>/10= SET FILL COUNT/
	066440	075	040	123	
	066443	105	124	040	
	066446	106	111	114	
	066451	114	040	103	
	066454	117	125	116	
	066457	124			
14801	066460	200	061	061	.ASCII <CRLF>/11= ENTER KAMIKAZE MODE/
	066463	075	040	105	
	066466	116	124	105	
	066471	122	040	113	
	066474	101	115	111	
	066477	113	101	132	
	066502	105	040	115	
	066505	117	104	105	
14802	066510	200	061	062	.ASCII <CRLF>/12= EXIT KAMIKAZE MODE/
	066513	075	040	105	
	066516	130	111	124	
	066521	040	113	101	
	066524	115	111	113	
	066527	101	132	105	
	066532	040	115	117	
	066535	104	105		
14803	066537	200	061	063	.ASCII <CRLF>/13= TURN CACHE OFF/
	066542	075	040	124	
	066545	125	122	116	
	066550	040	103	101	
	066553	103	110	105	
	066556	040	117	106	
	066561	106			
14804	066562	200	061	064	.ASCII <CRLF>/14= TURN CACHE ON/
	066565	075	040	124	
	066570	125	122	116	
	066573	040	103	101	

	066576	103	110	105	
	066601	040	117	116	
14805	066604	200	061	065	.ASCII <CRLF>/15= TEST SELECTED BANKS/
	066607	075	040	124	
	066612	105	123	124	
	066615	040	123	105	
	066620	114	105	103	
	066623	124	105	104	
	066626	040	102	101	
	066631	116	113	123	
14806	066634	200	061	066	.ASCII <CRLF>/16= TEST ALL BANKS/
	066637	075	040	124	
	066642	105	123	124	
	066645	040	101	114	
	066650	114	040	102	
	066653	101	116	113	
	066656	123			
14807	066657	200	061	067	.ASCII <CRLF>/17= ENABLE TRACE/
	066662	075	040	105	
	066665	116	101	102	
	066670	114	105	040	
	066673	124	122	101	
	066676	103	105		
14808	066700	200	061	070	.ASCII <CRLF>/18= DISABLE TRACE/
	066703	075	040	104	
	066706	111	123	101	
	066711	102	114	105	
	066714	040	124	122	
	066717	101	103	105	
14809	066722	015	012	000	.BYTE 15,12,0
14810	066725	200	127	110	MSG022: .ASCIIZ <CRLF>/WHICH CSR(0-F)? /
	066730	111	103	110	
	066733	040	103	123	
	066736	122	050	060	
	066741	055	106	051	
	066744	077	040	000	
14811	066747	200	103	123	MSG023: .ASCIIZ <CRLF>/CSR WORD? /
	066752	122	040	127	
	066755	117	122	104	
	066760	077	040	000	
14812	066763	200	103	123	MSG025: .ASCIIZ <CRLF>/CSR DOES NOT EXIST/
	066766	122	040	104	
	066771	117	105	123	
	066774	040	116	117	
	066777	124	040	105	
	067002	130	111	123	
	067005	124	000		
14813	067007	200	103	117	MSG026: .ASCIIZ <CRLF>/COMMAND:/
	067012	115	115	101	
	067015	116	104	072	
	067020	000			
14814	067021	200	117	114	MSG027: .ASCIIZ <CRLF>/OLD CSR WAS/
	067024	104	040	103	
	067027	123	122	040	
	067032	127	101	123	
	067035	000			
14815	067036	200	103	123	MSG028: .ASCIIZ <CRLF>/CSR IS NOW/

	067041	122	040	111	
	067044	123	040	116	
	067047	117	127	000	
14816	067052	200	105	130	MSG029: .ASCIZ <CRLF>/EXAMINE MEMORY/
	067055	101	115	111	
	067060	116	105	040	
	067063	115	105	115	
	067066	117	122	131	
	067071	000			
14817	067072	200	102	101	MSG030: .ASCIZ <CRLF>/BANK(0 177)? /
	067075	116	113	050	
	067100	060	055	061	
	067103	067	067	051	
	067106	077	040	000	
14818	067111	200	120	110	MSG031: .ASCIZ <CRLF>/PHYSICAL ADDRESS(0-17757776)? /
	067114	131	123	111	
	067117	103	101	114	
	067122	040	101	104	
	067125	104	122	105	
	067130	123	123	050	
	067133	060	055	061	
	067136	067	067	065	
	067141	067	067	067	
	067144	066	051	077	
	067147	040	000		
14819	067151	200	120	101	MSG032: .ASCIZ <CRLF>/PARITY ABORT/<32>
	067154	122	111	124	
	067157	131	040	101	
	067162	102	117	122	
	067165	124	032	000	
14820	067170	200	124	111	MSG033: .ASCIZ <CRLF>/TIMEOUT TRAP/<32>
	067173	115	105	117	
	067176	125	124	040	
	067201	124	122	101	
	067204	120	032	000	
14821	067207	200	102	131	MSG034: .ASCIZ <CRLF>/BYPASSING ECC TESTS ON BANK /
	067212	120	101	123	
	067215	123	111	116	
	067220	107	040	105	
	067223	103	103	040	
	067226	124	105	123	
	067231	124	123	040	
	067234	117	116	040	
	067237	102	101	116	
	067242	113	040	000	
14822	067245	040	104	125	MSG034: .ASCIZ / DUE TO SBE LOCATIONS/
	067250	105	040	124	
	067253	117	040	123	
	067256	102	105	040	
	067261	114	117	103	
	067264	101	124	111	
	067267	117	116	123	
	067272	000			
14823	067273	121	126	000	MSG035: .ASCIZ /QV/
14824	067276	200	115	117	MSG036: .ASCIZ <CRLF>/MODIFY MEMORY/
	067301	104	111	106	
	067304	131	040	115	

	067307	105	115	117	
	067312	122	131	000	
14825	067315	200	117	114	MSG037: .ASCIZ <CRLF>/OLD DATA WAS /
	067320	104	040	104	
	067323	101	124	101	
	067326	040	127	101	
	067331	123	040	000	
14826	067334	200	104	101	MSG038: .ASCIZ <CRLF>/DATA IS NOW /
	067337	124	101	040	
	067342	111	123	040	
	067345	116	117	127	
	067350	040	000		
14827	067352	200	111	116	MSG039: .ASCIZ <CRLF>/INPUT NEW DATA? /
	067355	120	125	124	
	067360	040	116	105	
	067363	127	040	104	
	067366	101	124	101	
	067371	077	040	000	
14828	067374	200	123	105	MSG040: .ASCIZ <CRLF>/SELECT BANK & TEST/
	067377	114	105	103	
	067402	124	040	102	
	067405	101	116	113	
	067410	040	046	040	
	067413	124	105	123	
	067416	124	000		
14829	067420	200	102	101	MSG041: .ASCIZ <CRLF>/BANK NOT ACCESSABLE/
	067423	116	113	040	
	067426	116	117	124	
	067431	040	101	103	
	067434	103	105	123	
	067437	123	101	102	
	067442	114	105	000	
14830	067445	200	124	105	MSG042: .ASCIZ <CRLF>/TEST(0-47)? /
	067450	123	124	050	
	067453	060	055	064	
	067456	067	051	077	
	067461	040	000		
14831	067463	200	124	105	MSG043: .ASCIZ <CRLF>/TEST 0 DATA IS? /
	067466	123	124	040	
	067471	060	040	104	
	067474	101	124	101	
	067477	040	111	123	
	067502	077	040	000	
14832	067505	200	124	117	MSG046: .ASCIZ <CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
	067510	040	105	123	
	067513	103	101	120	
	067516	105	040	124	
	067521	131	120	105	
	067524	040	101	116	
	067527	131	040	113	
	067532	105	131	200	
	067535	012	012	000	
14833	067540	200	124	105	MSG047: .ASCIZ <CRLF>/TEST COMPLETE/
	067543	123	124	040	
	067546	103	117	115	
	067551	120	114	105	
	067554	124	105	000	

14834	067557	040	116	117	MSG048: .ASCIZ / NOT AVAILABLE NOW - TRY LATER!/ 101
	067562	124	040	101	
	067565	126	101	111	
	067570	114	101	102	
	067573	114	105	040	
	067576	116	117	127	
	067601	040	055	040	
	067604	124	122	131	
	067607	040	114	101	
	067612	124	105	122	
	067615	041	000		
14835	067617	200	102	101	MSG049: .ASCIZ <CRLF>/BANK REQUIRES RELOCATION/ 040
	067622	116	113	121	
	067625	122	105	122	
	067630	125	111	040	
	067633	105	123	114	
	067636	122	105	101	
	067641	117	103	117	
	067644	124	111		
	067647	116	000		
14836					.EVEN
14837	067652	120	117	127	MSG051: .ASCIZ /POWER RECOVERY/ 040
	067655	105	122	103	
	067660	122	105	105	
	067663	117	126	000	
	067666	122	131		
14838	067671	200	123	117	MSG055: .ASCIZ <CRLF>/SOB-A-LONG TEST/ 101
	067674	102	055	117	
	067677	055	114	040	
	067702	116	107	123	
	067705	124	105		
	067710	124	000		
14839	067712	200	102	105	MSG056: .ASCIZ <CRLF>/BELL = EACH PASS COMPLETE/ 040
	067715	114	114	105	
	067720	075	040	110	
	067723	101	103	101	
	067726	040	120	040	
	067731	123	123	115	
	067734	103	117	105	
	067737	120	114	000	
	067742	124	105		
14840	067745	200	040	040	MSG058: .ASCIZ <CRLF>/ CSR CSR .../ 122
	067750	103	123	040	
	067753	040	040	123	
	067756	040	103	056	
	067761	122	040	000	
	067764	056	056		
14841	067767	077	077	077	MSG061: .ASCIZ /??????/ 077
	067772	077	077		
	067775	000			
14842	067776	111	116	120	MSG062: .ASCIZ /INPUT MUST BE A/ 040
	070001	125	124	123	
	070004	115	125	102	
	070007	124	040	101	
	070012	105	040		
	070015	000			
14843	070016	116	040	117	MSG063: .ASCIZ /N OCTAL /

	070021	103	124	101	
	070024	114	040	000	
14844	070027	116	125	115	MSG064: .ASCIZ /NUMBER/<CRLF>
	070032	102	105	122	
	070035	200	000		
14845	070037	040	104	105	MSG065: .ASCIZ / DECIMAL /
	070042	103	111	115	
	070045	101	114	040	
	070050	000			
14846	070051	200	105	122	MSG066: .ASCIZ <CRLF>/ERRORS > 20 - ABORTING FOR XXDP CHAIN/
	070054	122	117	122	
	070057	123	040	076	
	070062	040	062	060	
	070065	040	055	040	
	070070	101	102	117	
	070073	122	124	111	
	070076	116	107	040	
	070101	106	117	122	
	070104	040	130	130	
	070107	104	120	040	
	070112	103	110	101	
	070115	111	116	000	
14847	070120	106	101	124	MSG067: .ASCIZ /FATAL /
	070123	101	114	040	
	070126	000			
14848	070127	113	040	127	MSG070: .ASCIZ /K WORDS OF MEMORY TOTAL/<CRLF>
	070132	117	122	104	
	070135	123	040	117	
	070140	106	040	115	
	070143	105	115	117	
	070146	122	131	040	
	070151	124	117	124	
	070154	101	114	200	
	070157	000			
14849	070160	200	122	105	MSG073: .ASCIZ <CRLF>/REFRESH TEST/
	070163	106	122	105	
	070166	123	110	040	
	070171	124	105	123	
	070174	124	000		
14850	070176	200	122	105	MSG075: .ASCIZ <CRLF>/RELOCATION NOT POSSIBLE/<32>
	070201	114	117	103	
	070204	101	124	111	
	070207	117	116	040	
	070212	116	117	124	
	070215	040	120	117	
	070220	123	123	111	
	070223	102	114	105	
	070226	032	000		
14851	070230	200	040	040	MSG076: .ASCIZ <CRLF>/ BANK ERRORS/<CRLF>
	070233	102	101	116	
	070236	113	040	040	
	070241	105	122	122	
	070244	117	122	123	
	070247	200	000		
14852	070251	200	105	116	MSG077: .ASCIZ <CRLF>/END PASS #/
	070254	104	040	120	
	070257	101	123	123	

	070262	040	043	000	
14853	070265	040	105	122	MSG079: .ASCIZ / ERROR(S) DETECTED/<CRLF>
	070270	122	117	122	
	070273	050	123	051	
	070276	040	104	105	
	070301	124	105	103	
	070304	124	105	104	
	070307	200	000		
14854	070311	200	106	111	MSG085: .ASCIZ <CRLF>/FILL COUNT(OCTAL)? /
	070314	114	114	040	
	070317	103	117	125	
	070322	116	124	050	
	070325	117	103	124	
	070330	101	114	051	
	070333	077	040	000	
14855	070336	200	113	105	MSG088: .ASCIZ <CRLF>/KERNEL STACK/
	070341	122	116	105	
	070344	114	040	123	
	070347	124	101	103	
	070352	113	000		
14856	070354	200	123	125	MSG089: .ASCIZ <CRLF>/SUPERVISOR STACK/
	070357	120	105	122	
	070362	126	111	123	
	070365	117	122	040	
	070370	123	124	101	
	070373	103	113	000	
14857	070376	200	125	123	MSG090: .ASCIZ <CRLF>/USER STACK/
	070401	105	122	040	
	070404	123	124	101	
	070407	103	113	000	
14858	070412	040	111	123	MSG091: .ASCIZ / IS EMPTY/
	070415	040	105	115	
	070420	120	124	131	
	070423	000			
14859	070424	122	105	114	MSG092: .ASCIZ /RELOCATED /
	070427	117	103	101	
	070432	124	105	104	
	070435	040	040	000	
14860	070440	102	101	116	MSG093: .ASCIZ /BANK= /
	070443	113	075	000	
14861	070446	040	040	124	MSG095: .ASCIZ / TEST= /
	070451	105	123	124	
	070454	075	000		
14862	070456	200	105	116	MSG101: .ASCIZ <CRLF>/ENTERING KAMIKAZE MODE/
	070461	124	105	122	
	070464	111	116	107	
	070467	040	113	101	
	070472	115	111	113	
	070475	101	132	105	
	070500	040	115	117	
	070503	104	105	000	
14863	070506	200	114	105	MSG102: .ASCIZ <CRLF>/LEAVING KAMIKAZE MODE/
	070511	101	126	111	
	070514	116	107	040	
	070517	113	101	115	
	070522	111	113	101	
	070525	132	105	040	

	070530	115	117	104	
	070533	105	000		
14864	070535	200	114	105	MSG103: .ASCIZ <CRLF>/LEAVING FS MODE/<CRLF>
	070540	101	126	111	
	070543	116	107	040	
	070546	106	123	040	
	070551	115	117	104	
	070554	105	200	000	
14865	070557	032	000		MSG104: .BYTE 32,0 ;CONTROL Z
14866	070561	200	105	116	MSG105: .ASCIZ <CRLF>/ENTER BANKS - USE NUMBER 200 TO TERMINATE/
	070564	124	105	122	
	070567	040	102	101	
	070572	116	113	123	
	070575	040	055	040	
	070600	125	123	105	
	070603	040	116	125	
	070606	115	102	105	
	070611	122	040	062	
	070614	060	060	040	
	070617	124	117	040	
	070622	124	105	122	
	070625	115	111	116	
	070630	101	124	105	
	070633	000			
14867	070634	200	103	101	MSG106: .ASCIZ <CRLF>/CACHE IS OFF/
	070637	103	110	105	
	070642	040	111	123	
	070645	040	117	106	
	070650	106	000		
14868	070652	200	103	101	MSG107: .ASCIZ <CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
	070655	103	110	105	
	070660	040	111	123	
	070663	040	117	116	
	070666	040	050	105	
	070671	130	103	105	
	070674	120	124	040	
	070677	104	125	122	
	070702	111	116	107	
	070705	040	101	103	
	070710	124	125	101	
	070713	114	040	120	
	070716	101	124	124	
	070721	105	122	116	
	070724	123	051	000	
14869	070727	200	117	116	MSG110: .ASCIZ <CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
	070732	114	131	040	
	070735	123	105	114	
	070740	105	103	124	
	070743	105	104	040	
	070746	102	101	116	
	070751	113	123	040	
	070754	127	111	114	
	070757	114	040	102	
	070762	105	040	124	
	070765	105	123	124	
	070770	105	104	000	
14870	070773	200	101	114	MSG111: .ASCIZ <CRLF>/ALL BANKS WILL BE TESTED/

	070776	114	040	102	
	071001	101	116	113	
	071004	123	040	127	
	071007	111	114	114	
	071012	040	102	105	
	071015	040	124	105	
	071020	123	124	105	
	071023	104	000		
14871	071025	113	040	117	MSG112: .ASCIZ /K OF Q-BUS PARITY MEMORY/<CRLF>
	071030	106	040	121	
	071033	055	102	125	
	071036	123	040	120	
	071041	101	122	111	
	071044	124	131	040	
	071047	115	105	115	
	071052	117	122	131	
	071055	200	000		
14872	071057	113	040	117	MSG113: .ASCIZ /K OF ECC MEMORY/<CRLF>
	071062	106	040	105	
	071065	103	103	040	
	071070	115	105	115	
	071073	117	122	131	
	071076	200	000		
14873	071100	200	040	040	M1184: .ASCIZ <CRLF>" 11/84"
	071103	040	061	061	
	071106	057	070	064	
	071111	000			
14874	071112	200	040	040	MSG117: .ASCIZ <CRLF>" 11/83"
	071115	040	061	061	
	071120	057	070	063	
	071123	000			
14875	071124	200	040	040	MSG119: .ASCIZ <CRLF>/ NO/
	071127	040	116	117	
	071132	000			
14876	071133	040	103	101	MSG120: .ASCIZ / CACHE AVAILABLE/
	071136	103	110	105	
	071141	040	101	126	
	071144	101	111	114	
	071147	101	102	114	
	071152	105	000		
14877	071154	040	103	101	MSG121: .ASCIZ / CACHE BYPASSED/
	071157	103	110	105	
	071162	040	102	131	
	071165	120	101	123	
	071170	123	105	104	
	071173	000			
14878	071174	200	103	123	MSG122: .ASCII <CRLF>/CSR NUMBER /
	071177	122	040	116	
	071202	125	115	102	
	071205	105	122	040	
14879	071210	000			MSG122: .BYTE 0
14880	071211	040	103	117	.ASCIZ / CONTROLS TOO MANY BANKS/
	071214	116	124	122	
	071217	117	114	123	
	071222	040	124	117	
	071225	117	040	115	
	071230	101	116	131	

	071233	040	102	101	
	071236	116	113	123	
	071241	000			
14881	071242	040	120	101	MSG125: .ASCIZ / PASSES COMPLETED/
	071245	123	123	105	
	071250	123	040	103	
	071253	117	115	120	
	071256	114	105	124	
	071261	105	104	000	
14882	071264	200	120	122	MSG126: .ASCIZ <CRLF>/PROGRAM CSR COULD NOT BE DETERMINED/
	071267	117	107	122	
	071272	101	115	040	
	071275	103	123	122	
	071300	040	103	117	
	071303	125	114	104	
	071306	040	116	117	
	071311	124	040	102	
	071314	105	040	104	
	071317	105	124	105	
	071322	122	115	111	
	071325	116	105	104	
	071330	000			
14883	071331	200	124	122	MSG127: .ASCIZ <CRLF>/TRACE ENABLED/
	071334	101	103	105	
	071337	040	105	116	
	071342	101	102	114	
	071345	105	104	000	
14884	071350	200	124	122	MSG128: .ASCIZ <CRLF>/TRACE DISABLED/
	071353	101	103	105	
	071356	040	104	111	
	071361	123	101	102	
	071364	114	105	104	
	071367	000			
14885	071370	200	200	040	MSG008: .ASCIZ <CRLF><CRLF>/ CSR MAP/<CRLF>
	071373	040	040	040	
	071376	040	040	040	
	071401	040	040	040	
	071404	040	040	040	
	071407	040	040	103	
	071412	123	122	040	
	071415	115	101	120	
	071420	200	000		
14886	071422	200	040	103	MSG000: .ASCIZ <CRLF>" CVMJAA0 ECC/PARITY MEMORY DIAGNOSTIC"
	071425	126	115	112	
	071430	101	060	040	
	071433	040	105	103	
	071436	103	057	120	
	071441	101	122	111	
	071444	124	131	040	
	071447	115	105	115	
	071452	117	122	131	
	071455	040	104	111	
	071460	101	107	116	
	071463	117	123	124	
	071466	111	103	000	
14887	071471	200	200	000	MSG129: .ASCIZ <CRLF><CRLF>
14888	071474	120	122	117	MSG130: .ASCIZ /PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC/

	071477	103	105	123	
	071502	123	117	122	
	071505	040	116	117	
	071510	124	040	123	
	071513	125	120	120	
	071516	117	122	124	
	071521	105	104	040	
	071524	102	131	040	
	071527	124	110	111	
	071532	123	040	104	
	071535	111	101	107	
	071540	116	117	123	
	071543	124	111	103	
	071546	000			
14889	071547	200	125	116	NOUBMT: .ASCIZ <CRLF>/UNIBUS MEMORY WILL NOT BE TESTED/<CRLF>
	071552	111	102	125	
	071555	123	040	115	
	071560	105	115	117	
	071563	122	131	040	
	071566	127	111	114	
	071571	114	040	116	
	071574	117	124	040	
	071577	102	105	040	
	071602	124	105	123	
	071605	124	105	104	
	071610	200	000		
14890					.EVEN
14896	071612				\$\$END
14897	071612				END:
14898	071612	010322			.PRINT 60000-SUPLIMIT ;SUPERVISOR ADDRESSES LEFT
14902		000200			.END START3

ABORT	000000	B5IZE	002374	B61	035112	CPUERR=	177766	DH13	064172
ABORTF	002144	B0	004652	B62	035116	CR	= 000015	DH14	064267
ABORT1	000032	B1	005656	B63	035322	CRLF	= 000200	DH15	064346
ABORT2	000046	B10	012262	B64	035326	CSR	002150	DH16	064465
ACFLAG	002116	B100	045566	B65	036444	CSRADD=	172100	DH19	064502
ACTFLA	002350	B101	046040	B66	036452	CSRBMF	006024	DH2	063536
ADDRES	002034	B102	046046	B67	036602	CSRCS	017004	DH20	064507
ANA2	007372	B103	053522	B7	012220	CSRFBA	002232	DH23	064566
APTDOW	040720	B104	053526	B70	036616	CSRFIR	002226	DH24	064645
APTECC	002424	B105	053624	B71	043510	CSRHOL	002526	DH25	064714
APTFLA	002352	B106	053716	B72	043746	CSRINC	002330	DH26	064752
APTHAN	014126	B107	054002	B73	044230	CSRINF	002462	DH27	064770
APTHLT	040766	B11	013010	B74	044524	CSRINT	002236	DH3	063561
APTPAR	002422	B12	013060	B75	044524	CSRLAS	002230	DH30	065044
APTSIZ	002444	B13	013122	B76	044622	CSRLBA	002234	DH5	063615
BACK	051530	B14	013400	B77	045562	CSRL00	002332	DH6	063674
BACKGN	032630	B15	014172	CACHKF	002550	CSRMAP	006034	DH7	063741
BAD	002052	B16	015610	CACHKN	002544	CSRNO	002152	DIAGFL	002004
BADPC	002022	B17	016060	CACHOF=	104424	CSROUT	035314	DISPLA	002640
BADPSW	002032	B2	006050	CACHON=	104423	CSRSTU=	000021	DISPRE=	000174
BADSP	002026	B20	016066	CACHVE=	000114	CSRIS	002322	DISPTB	013600
BADSTA	034152	B21	016104	CBCSR	= 104474	CTEST	006714	DOBACK	014162
BADXOR	002060	B22	016146	CBITS	002316	CTLKVE	002146	DONE	006700
BAD2	002054	B23	016220	CBREG	= 104513	DATARG=	177754	DSWR	= 177570
BAD3	002056	B24	021642	CB1CSR=	104475	DATBUF	002242	DT1	060144
BAFPAF	014262	B25	021646	CHECK	002314	DBEMSK	002256	DT10	060264
BAFPAR	014370	B26	022026	CHKDIS=	104504	DDISP	= 177570	DT11	060306
BAKPAT	002632	B27	022034	CHKTRP	032220	DEENER=	104421	DT12	060314
BANK	002102	B3	006120	CHK1DI=	104505	DETAIL	053266	DT13	060320
BANKIN	002104	B30	022264	CKEND	055056	DETFLA	002220	DT14	060342
BANKMO	037442	B31	025750	CKSWR	= 104410	DETPSW	002216	DT16	060360
BANKOK	040456	B32	026140	CLRCR=	104502	DETRO	002200	DT17	060410
BAWPAF	014476	B33	026176	CLREX	007342	DETR1	002202	DT2	060156
BAWPAR	014626	B34	026620	CLRMEM	007232	DETR2	002204	DT20	060416
BGTEST	027610	B35	026656	CLR1CS=	104503	DETR3	002206	DT22	060426
BITNO	002324	B36	026672	CMD16A	044610	DETR4	002210	DT23	060434
BIT0	= 000001	B37	027012	CMD16L=	000074	DETR5	002212	DT24	060456
BIT1	= 000002	B4	006400	CMD5B	042752	DETSP	002214	DT25	060476
BIT10	= 002000	B40	027172	CMD5C	043246	DET1	053756	DT26	060510
BIT11	= 004000	B41	027214	CMD7B	043504	DF1	060556	DT27	060516
BIT12	= 010000	B42	027742	CMD7C	043560	DF11	060701	DT3	060162
BIT13	= 020000	B43	030102	CMD9B	044224	DF13	060712	DT30	060536
BIT14	= 040000	B44	030140	CMD9C	044300	DF14	060722	DT4	060172
BIT15	= 100000	B45	030432	CONFGE	002450	DF15	060731	DT5	060202
BIT2	= 000004	B46	030646	CONFIE	003670	DF16	060740	DT6	060220
BIT3	= 000010	B47	031050	CONFIG	002664	DF2	060557	DT7	060234
BIT4	= 000020	B5	006434	CONTFL	002222	DF3	060603	DUMMY	002176
BIT5	= 000040	B50	031224	CONTRL=	177746	DF4	060616	DUMPCS=	000103
BIT6	= 000100	B51	031244	CONTS	055226	DF5	060631	ECCDIS=	104470
BIT7	= 000200	B52	031254	CONTS1	054570	DF6	060644	ECCINI=	104472
BIT8	= 000400	B53	031572	CONTS2	055230	DF7	060657	ECC1DI=	104471
BIT9	= 001000	B54	031742	CONTS3	047340	DF8	060672	ECC1IN=	104473
BLOCK1	040770	B55	031756	CONTT	055152	DF9	060674	EMTSV	003670
BLOCK2	041010	B56	032022	COUNT	002370	DH1	063501	EMTVEC=	000030
BLOCK3	041024	B57	034746	CPERRF	052314	DH10	064051	EM11	061247
BMFLAG	002130	B6	011544	CPSAVE	052312	DH11	064147	EM12	061271
BRG08B	027612	B60	034752	CPUBIT	002106	DH12	064165	EM13	061315

EM14	061347	E0	004700	E62	035164	GTSWR =	104407	LOADBA	002432
EM15	061413	E1	005776	E63	035356	HEADER	002612	LOADCS =	104425
EM17	061461	E10	012346	E64	035356	HIPAT	040510	LOADER =	006065
EM19	061521	E100	045622	E65	036602	HOLDLO =	000016	LOADHO	002576
EM2	060747	E101	046102	E66	036562	HT =	000011	LOOP	013552
EM20	061573	E102	046102	E67	037016	I	002452	LSIZE	002400
EM21	061652	E103	053570	E7	012362	IBSAVE	052310	LST# =	000000
EM22	061706	E104	053570	E70	036730	IIII =	177777	LWDBE =	000061
EM23	061733	E105	053654	E71	043552	ILLCSR	012724	LWSBE =	000057
EM24	061762	E106	053750	E72	044050	IMPTES	011750	LO	004662
EM25	062041	E107	054034	E73	044272	INCBNK	040520	L1	004724
EM26	062066	E11	013052	E74	044564	INCPAT	040474	L10	005222
EM27	062137	E12	013120	E75	044564	INCRPT	040474	L100	012334
EM29	062227	E13	013150	E76	044640	INHBAN	002540	L101	012340
EM3	061005	E14	013520	E77	045622	INHECC	002536	L102	013042
EM30	062311	E15	014260	FASTCI =	177640	INTFLA	002136	L103	013036
EM32	062421	E16	015630	FATAL#	002064	INT64K	002140	L104	013042
EM33	062526	E17	016414	FCMD10	044364	INVALID =	104511	L105	013136
EM35	062634	E2	006114	FCMD11	044412	IOTVEC =	000020	L106	013206
EM36	062721	E20	016374	FCMD12	044434	JMPRL1	037076	L107	013352
EM4	061037	E21	016324	FCMD13	044454	KAMIKA	002006	L11	005230
EM40	062770	E22	016324	FCMD14	044476	KAMITE	024234	L110	013552
EM5	061105	E23	016274	FCMD15	044514	KDIAG =	000010	L111	013510
EM50	063042	E24	022004	FCMD16	044600	KDPAR0 =	172360	L112	013426
EM51	063076	E25	021770	FCMD17	044642	KDPAR6 =	172374	L113	013430
EM52	063146	E26	022172	FCMD18	044656	KDPAR7 =	172376	L114	013470
EM53	063173	E27	022156	FIELDS	041054	KERNEL =	104417	L115	013504
EM55	063222	E3	006210	FINDBA =	000067	KERSTK =	002000	L116	013550
EM56	063243	E30	022342	FINT#	007214	KFLAG	002530	L117	013552
EM57	063275	E31	026034	FIRST =	060000	KIPAR0 =	172340	L12	005300
EM6	061162	E32	026414	FLIPLO	002616	KIPAR4 =	172350	L120	013666
EM60	063343	E33	026402	FLIPWA	032500	KIPAR5 =	172352	L121	013776
EM61	063405	E34	027164	FLUSH	013672	KIPAR6 =	172354	L122	014006
EM62	063426	E35	027150	FSCMD0	041252	KIPDR0 =	172300	L123	014156
EM7	061207	E36	026764	FSCMD1	041354	KMAP =	104422	L124	014156
ENASBE =	104506	E37	027134	FSCMD2	041464	KPFLAG	002114	L125	014244
ENA1SB =	104507	E4	006622	FSCMD3	041632	KSIZE	002376	L126	014356
END	071612	E40	027210	FSCMD4	042106	KSTACK	002574	L127	014464
ENDADD	002562	E41	027234	FSCMD5	042426	LAST =	157776	L13	005274
ENDFLG	002564	E42	030064	FSCMD6	043344	LASTBA	002556	L130	014614
ENERGI =	104420	E43	030270	FSCMD7	043352	LASTBL	002560	L131	014744
ENEXBK	040446	E44	030256	FSCMD8	043644	LASTER	002016	L132	015074
EQFLAG	002132	E45	030536	FSCMD9	044072	LBLS0 =	000530	L133	015246
ERRADD	002460	E46	031022	FSINFL	002442	LBLS1 =	000107	L134	015376
ERRGEN =	104512	E47	031220	FSPAT	043126	LBLS2 =	000520	L135	015550
ERRMAX	002554	E5	006612	FSSTAC	002306	LBLS3 =	000513	L136	015630
ERROR =	104000	E50	031542	FS1	041140	LBLS4 =	000366	L137	015666
ERRPC	002020	E51	031524	FS7FLA	002446	LBLS5 =	000370	L14	005300
ERRPSW	002030	E52	031510	FULLRE	002542	LBLS6 =	000023	L140	015676
ERRSP	002024	E53	031726	GBLENG =	000076	LCSROU =	000063	L141	016324
ERRVEC =	000004	E54	032212	GETCSR	032674	LCSRRE =	000101	L142	016306
EVEN	002364	E55	032202	GETDAT	045252	LCSRSA =	000077	L143	016356
EXBANK	040052	E56	032170	GETDA1	045350	LEGALC =	000010	L144	016626
EXCMD3	042040	E57	035020	GETDIS	051430	LF =	000012	L145	016736
EXCMD4	042360	E6	011676	GOOD	002044	LINK1	002522	L146	017074
EXIT	040604	E60	035020	GOOD2	002046	LINK2	002524	L147	017122
EXIT2	040610	E61	035164	GOOD3	002050	LKS =	177546	L15	005610

L150	017126	L237	026764	L334	033434	L416	043044	L501	051604
L151	017130	L24	006156	L335	033530	L417	0430	L502	052044
L152	017174	L240	026714	L336	033520	L42	00655	L503	052000
L153	017230	L241	026726	L337	033526	L420	04307	L504	052024
L154	017234	L242	026744	L34	006374	L421	043422	L505	052044
L155	017236	L243	026752	L340	033536	L422	043424	L506	052064
L156	017404	L244	026776	L341	033562	L423	043536	L507	052256
L157	017764	L245	027134	L342	033664	L424	044050	L51	006700
L16	005610	L246	027034	L343	034532	L425	044034	L510	052126
L160	020370	L247	027046	L344	034540	L426	044004	L511	052254
L161	020400	L25	006146	L345	034776	L427	044142	L512	052204
L162	020400	L250	027100	L346	034776	L43	006554	L513	052216
L163	020472	L251	027064	L347	035040	L430	044144	L514	052254
L164	020502	L252	027076	L35	006430	L431	044256	L515	052264
L165	020502	L253	027122	L350	035046	L432	044546	L516	052620
L166	021036	L254	027110	L351	035142	L433	044550	L517	053546
L167	021070	L255	027122	L352	035142	L434	045014	L52	010636
L17	005724	L256	027234	L353	035204	L435	045030	L520	053752
L170	021134	L257	030134	L354	035212	L436	045034	L521	053756
L171	021754	L26	006154	L355	035334	L437	045052	L522	054036
L172	021732	L260	030140	L356	036416	L44	006576	L523	054042
L173	021754	L261	030224	L357	036432	L440	045200	L524	055044
L174	022020	L262	030242	L36	006434	L441	045202	L525	055050
L175	022142	L263	030246	L360	036444	L442	045246	L526	055172
L176	022120	L266	030364	L361	036444	L443	045472	L527	055262
L177	022142	L267	030522	L362	036546	L444	045600	L53	010642
L2	005032	L27	006164	L363	036576	L445	046060	L530	055264
L20	005752	L271	030570	L364	036720	L446	047376	L54	010644
L200	022326	L272	030576	L365	036720	L447	047400	L55	011374
L201	022326	L273	030722	L366	036720	L45	006602	L56	011410
L202	022434	L274	030772	L367	036700	L450	050230	L57	011512
L203	022406	L276	031132	L37	006464	L451	050242	L6	005230
L204	022464	L277	031202	L370	036720	L452	050260	L60	011516
L205	022654	L3	005054	L371	036762	L453	050262	L61	011662
L206	023170	L301	031354	L372	036762	L454	050302	L62	011662
L207	023424	L302	031430	L373	037012	L455	050314	L63	011662
L21	005770	L303	031446	L374	040204	L456	050332	L64	011662
L210	023444	L304	031474	L375	040374	L457	050334	L65	011656
L211	023544	L31	006262	L376	040556	L460	050350	L66	011734
L212	023724	L310	031636	L377	040624	L461	050552	L67	011740
L213	023732	L311	031700	L4	005072	L462	050560	L7	005230
L214	024070	L313	032016	L40	006510	L463	050606	L70	011742
L215	024076	L314	032022	L400	040630	L464	050620	L71	012124
L216	024256	L315	032074	L401	040636	L465	050636	L72	012124
L217	024264	L316	032076	L402	040652	L466	050650	L73	012124
L220	024270	L317	032134	L403	040664	L467	050662	L74	012130
L221	025704	L32	006266	L404	041076	L470	050704	L75	012346
L222	025742	L320	032154	L405	041106	L471	050752	L76	012346
L223	026022	L321	032160	L406	041242	L472	051032	L77	012336
L225	026172	L325	032252	L407	041302	L473	051046	MAINT	= 177750
L226	026176	L326	032250	L41	006562	L474	051050	MAPHO	= 170202
L227	026272	L327	032272	L410	041306	L475	051164	MAPKER	037740
L230	026350	L33	006332	L411	041336	L476	051254	MAPLO	= 170200
L231	026366	L330	032272	L412	041350	L477	051264	MAPL1	= 170204
L232	026372	L331	032410	L413	042536	L5	005150	MAPPER	035716
L235	026642	L332	032456	L414	042576	L50	006636	MASK	002320
L236	026652	L333	033034	L415	042636	L500	051604	MBERR	012602

MEMDON	013620	MSG037	067315	MSG126	071264	MT0005	020140	OLDCSR	002156
MFPT	= 000007	MSG038	067334	MSG127	071331	MT0006	020222	ONES	002614
MJPAT	017430	MSG039	067352	MSG128	071350	MT0007	020256	PADDRE	002036
MJTEST	017324	MSG040	067374	MSG129	071471	MT0010	020320	PAFBAF	014756
MKCNT	016476	MSG041	067420	MSG130	071474	MT0014	020354	PAFBAW	015106
MKCONT	015656	MSG042	067445	MSIZE	002402	MT0017	020434	PARBAF	015260
MKCSRT	017014	MSG043	067463	MTA030	022212	MT0020	020456	PARBAW	015410
MKFLAG	002120	MSG046	067505	MTEST	015562	MT0021	020536	PARCNT	002072
MKLOOP	016040	MSG047	067540	MTPA03	024750	MT0022	021026	PARITY	033744
MKPAT	017244	MSG048	067557	MTPA04	025106	MT0023	021060	PARTHE	002304
MKTEST	017104	MSG049	067617	MTPA21	026442	MT0024	021124	PARVEC	= 000114
MMR0	= 177572	MSG051	067652	MTPA24	027266	MT0026	021332	PASCNT	002570
MMR1	= 177574	MSG055	067671	MTPA26	027360	MT0027	021610	PASFLG	002264
MMR2	= 177576	MSG056	067712	MTPB03	025010	MT0030	022200	PASSNO	002266
MMR3	= 172516	MSG058	067745	MTPB04	025142	MT0031	022454	PATERR	002074
MMTRAP	034126	MSG061	067767	MTPB21	026472	MT0032	022644	PATPLU	004644
MMVEC	= 000250	MSG062	067776	MTPB24	027326	MT0033	023160	PATTER	002112
MONFLG	002276	MSG063	070016	MTPB26	027374	MT0034	023346	PCBUMP	002326
MSEEDH	002606	MSG064	070027	MTPC03	025050	MT0035	023446	PCONFI	032722
MSEEDL	002610	MSG065	070037	MTPC21	026526	MT0036	023560	PCONFS	033334
MSG012	071210	MSG066	070051	MTPC24	027342	MT0037	023622	PCONF1	033242
MSG034	067207	MSG067	070120	MTPC26	027430	MT0041	023660	PCONF2	033302
MSG034	067245	MSG070	070127	MTPD03	025066	MT0042	023740	PDP110	034140
MSG000	071422	MSG073	070160	MTPD21	026562	MT0043	023774	PD1	045472
MSG001	065123	MSG075	070176	MTPD26	027450	MT0044	024024	PORA05	047614
MSG002	065205	MSG076	070230	MTP000	024640	MT0045	024104	PERBNK	050446
MSG003	065262	MSG077	070251	MTP001	024664	MT0046	024134	PERECC	050526
MSG004	065367	MSG079	070265	MTP002	024716	MT0047	024164	PERRAB	050264
MSG005	065475	MSG085	070311	MTP005	025162	MT0999	024220	PERRAW	050212
MSG006	065507	MSG088	070336	MTP006	025216	MT1	015636	PERRA3	045040
MSG008	071370	MSG089	070354	MTP007	025416	MT2	015642	PERRA7	050336
MSG009	065544	MSG090	070376	MTP010	025516	MUT	002110	PERR01=	104427
MSG010	065556	MSG091	070412	MTP014	025624	M1184	071100	PERR02=	104430
MSG011	065656	MSG092	070424	MTP017	026050	NC	047400	PERR03=	104431
MSG012	065744	MSG093	070440	MTP020	026126	NEMCNT	002070	PERR04=	104432
MSG013	066056	MSG095	070446	MTP022	026612	NEWBAN	002310	PERR05	047610
MSG014	066060	MSG101	070456	MTP030	027466	NEWKER	037672	PERR06	047636
MSG015	066062	MSG102	070506	MTP031	027476	NEWLOA	037774	PERR07=	104433
MSG016	066064	MSG103	070535	MTP032	027554	NOAPT	051514	PERR10=	104434
MSG017	066076	MSG104	070557	MTP033	027606	NOCH	054552	PERR11=	104435
MSG018	066107	MSG105	070561	MTP034	027704	NOERRO	002430	PERR12=	104436
MSG019	066112	MSG106	070634	MTP035	027730	NOFSMO	002426	PERR13=	104437
MSG020	066116	MSG107	070652	MTP036	030072	NONEM	002100	PERR14=	104440
MSG021	066137	MSG11A	065570	MTP037	030316	NONEXI	034050	PERR15=	104441
MSG022	066725	MSG11B	066041	MTP041	030370	NOOJ	033034	PERR16=	104442
MSG023	066747	MSG11C	066043	MTP042	030542	NOOJ1	033144	PERR17=	104443
MSG025	066763	MSG11D	066045	MTP043	031030	NOPAR	002076	PERR20=	104444
MSG026	067007	MSG110	070727	MTP044	031224	NORES	003722	PERR21=	104445
MSG027	067021	MSG111	070773	MTP045	031546	NOSCOF	002440	PERR22=	104446
MSG028	067036	MSG112	071025	MTP046	031734	NOSUPE	002456	PERR23=	104447
MSG029	067052	MSG113	071057	MTP047	032274	NOTAB	002372	PERR24=	104450
MSG030	067072	MSG117	071112	MTST3	011414	NOTRCE	051200	PERR25=	104451
MSG031	067111	MSG119	071124	MT0000	017510	NOUBMT	071547	PERR26=	104452
MSG032	067151	MSG120	071133	MT0001	017544	NO22BI	002454	PERR27=	104453
MSG033	067170	MSG121	071154	MT0002	017640	NULLFL	002344	PERR30=	104454
MSG035	067273	MSG122	071174	MT0003	017754	NXTCSR	005752	PERR31=	104455
MSG036	067276	MSG125	071242	MT0004	020070	OLDCAC	002302	PERR32=	104456

PERR33=	104457	SAV30	003706	SUPDR0	002160	TMFLAG	002134	WOOPSA	046704
PERR34=	104460	SAV32	003710	SUPDR1	002162	TOOMAN	002406	WOOPUP	046472
PERR35=	104461	SAV4	002270	SUPDR2	002164	TOTCSR	002224	WORST	002600
PERR36=	104462	SBEMSK	002252	SUPDR3	002166	TRACE	006216	XOCHAR	047246
PERR37=	104463	SBENT	016756	SUPDR4	002170	TRAPVE=	000034	XXDPCH	002354
PERR40=	104464	SBESYN	026422	SUPDR5	002172	TSTBAN	011302	ZEROS	002336
PERR41=	104465	SBETES	016500	SUPDR6	002174	TSTDAT	002246	\$APTHD	056750
PERR42=	104466	SCOPE =	000004	SUPLIM	047456	TSTRD1	034542	\$AUTO	002062
PERR43=	104467	SDPAR0=	172260	SUPSTK=	000740	TSTREA=	104510	\$BANK	002013
PERXOR	050422	SDPAR5=	172272	SWAPAT	002634	TST1	005630	\$BASE	056722
PFECDF	052732	SDPAR6=	172274	SWR	002636	TST2	010460	\$BELL	002653
PFECDH	052672	SDPAR7=	172276	SWREG =	000176	TST3	010644	\$CACHF	034256
PFECDT	052722	SEEDHI	002602	SWREND	055054	TST4	011526	\$CACHN	034232
PFECEN	052636	SEEDLO	002604	SWRFLG	002566	TST5	013552	\$CBCSR	034672
PFECWS	052626	SELONL	002002	SWO	= 000001	TST6	013624	\$CBREG	035662
PFLAG	002122	SETPAT	040510	SW1	= 000002	TYPDS =	104405	\$CB1CS	034714
PGMCSR	002532	SHADL1	011444	SW10	= 002000	TYPEIT=	104401	\$CDW1	056726
PHEBE	012604	SHUTUP	040636	SW11	= 004000	TYPOC =	104402	\$CDW2	056730
PHYADD	002040	SIPAR0=	172240	SW12	= 010000	TYP05 =	104403	\$CHARC	047434
PHEMFL	002142	SIPAR3=	172246	SW13	= 020000	TYP50 =	000000	\$CHKDI	035266
PROTYP	004010	SIPAR5=	172252	SW14	= 040000	TYP51 =	000002	\$CHK1D	035302
PSIZE	002404	SIPAR6=	172254	SW15	= 100000	TYP52 =	000000	\$CKSWR	054530
PSW	= 177776	SIPDR0=	172200	SW2	= 0C0004	TYP53 =	000000	\$CLRCS	035244
PTABLE	030276	SIZE =	040000	SW3	= 000010	TYP54 =	000000	\$CLR1C	035256
PWRVEC=	000024	SKIPKA	002010	SW4	= 000020	TYP55 =	000000	\$CMTAG	002000
QUICK	002436	SKIPMK	002342	SW5	= 000040	TYP56 =	000002	\$CMTGE	002544
QUIT	040534	SKJ	051226	SW6	= 000100	UDPAR0=	177660	\$CNTLC	055736
QUIT1	040600	SKPERR	002066	SW7	= 000200	UDPAR7=	177676	\$CNTLG	055750
QVFLAG	002346	SKUB	037012	SW8	= 000400	UFDSET	002000	\$CNTLK	055144
RANODD	027410	SKUJ	012606	SW9	= 001000	UIPAR0=	177640	\$CNTLU	055743
RDCHR =	104411	SOBK	002572	SYNREG=	104514	UIPAR1=	177642	\$CPUOP	056674
RDDEC =	104414	SOBLEN=	000056	SYSSIZ	004012	UIPAR2=	177644	\$CRLF	002660
RDLIN =	104412	SOFTPA	002620	TAG2\$	011046	UIPAR3=	177646	\$DBLK	054520
RDOCT =	104413	SOURCE	002312	TAG3\$	011102	UIPAR4=	177650	\$DB20	056526
READCS=	104426	SPLTCS	002240	TAG4\$	024364	UIPAR5=	177652	\$DDW0	056734
READON	002410	SSP =*	000006	TAG70\$	052736	UIPAR6=	177654	\$DDW1	056736
REALPA	002300	ST =	177776	TAG71\$	052746	UIPDRO=	177607	\$DDW2	056740
REFRES	027166	STACK =	002000	TAG72\$	052756	UNITOP	002416	\$DDW3	056742
REFSUB	027236	START	003670	TAG73\$	053026	UNMAP	040026	\$DDW4	056744
REGCOP	032470	START1	000300	TAG74\$	053066	UNRELO	037214	\$DDW5	056746
RELENT	037022	START2	000310	TAG75\$	053100	UPPFLG	002265	\$DDW7	056732
RELOCA	036402	START3	000200	TAG76\$	053112	USERMA	037610	\$DEENE	034222
RELOC1	037036	STAR27	021634	TAG77\$	053156	USESTK=	000700	\$DEVCT	056656
RESREG=	104416	STOPOK	002420	TAG78\$	053164	USP =*	000006	\$DEVM	056724
RESTAR	002626	STRIPE	002366	TAG79\$	053244	VMKOR	003712	\$DIDDO=	000000
RESVEC=	000010	SUBAAA	004702	TAG9\$	010710	WARN1	010776	\$DOAGA	014156
RES0	041350	SUBAAB	005032	TBG4\$	024542	WARN2	024762	\$DOAGN	014052
RES1	041430	SUBAAI	011440	TCFIG1	033404	WARN3	024776	\$DOWN	045722
RES2	041576	SUBAAP	012766	TCFIG2	033444	WARN4	025022	\$DTBL	054510
RLFLAG	002126	SUBAAR	012172	TCFIG3	033600	WARN5	025036	\$ECCDI	034570
RRFLAG	002124	SUBAAS	010454	TCONFI	033336	WASDBE=	104500	\$ECCIN	034616
RTNVAL=*	000000	SUCCESS	002334	TEMP	002434	WASSBE=	104476	\$ECC1D	034604
RWCSR	006220	SUPDOA	002262	TESTAD	002412	WAS1DB=	104501	\$ECC1I	034632
SAVCSR	002154	SUPD01	024270	TESTMO	002552	WAS1SB=	104477	\$ENASB	034644
SAVMON	002274	SUPD02	024304	TIME	002340	WHICHC	044670	\$ENA1S	034660
SAVPAR	002272	SUPD03	024446	TIMEOU	034114	WOOPEN	046654	\$ENDAD	014042
SAVREG=	104415	SUPD04	024462	TKVEC =	000060	WOOPS	046306	\$ENERG	034212

Symbol table

\$ENV	056666	\$LL	= 000106	\$O\$	= 000000	\$PER37	051136	\$TRAP2	057006
\$ENVM	056667	\$LOADC	034274	\$PASS	056654	\$PER40	051142	\$TRPAD	057026
\$EOP	013676	\$LPADR	002622	\$PASTM	056756	\$PWRDN	045352	\$TSTM	056754
\$ERFLG	002014	\$LPERR	002624	\$PATMA	002012	\$PWRUP	045726	\$TSTRD	034410
\$ERRGE	035412	\$L\$	= 000000	\$PER01	047456	\$QUES	002657	\$TTYIN	055712
\$ERROR	051502	\$MADR1	056700	\$PER02	047504	\$R	= 177777	\$TYPDS	054304
\$ERRTB	057274	\$MADR2	056704	\$PER03	047532	\$RAND	056432	\$TYPE	047122
\$ERRTY	052316	\$MADR3	056710	\$PER04	047562	\$RDCHR	055264	\$TYPEC	047250
\$ERTTL	002630	\$MADR4	056714	\$PER07	047644	\$RDDEC	056150	\$TYPEX	047436
\$ESCAP	002362	\$MAIL	056646	\$PER10	047666	\$RDLIN	055414	\$TYPOC	054102
\$ETABL	056666	\$MAMS1	056676	\$PER11	047716	\$RDOCT	056000	\$TYPON	054116
\$ETEND	056750	\$MAMS2	056702	\$PER12	047736	\$READC	034370	\$TYPOS	054056
\$EXHAL	040630	\$MAMS3	056706	\$PER13	047760	\$RESRE	056374	\$T1	= 000000
\$E\$	= 000001	\$MAMS4	056712	\$PER14	050000	\$SAVRE	056336	\$T2	= 000530
\$FATAL	056650	\$MBADR	056752	\$PER15	050022	\$SAVR6	046304	\$UNIT	056660
\$FILLC	002652	\$MNEW	055766	\$PER16	050044	\$SCOPE	051146	\$UNITM	056760
\$FILLS	002357	\$MSGAD	056662	\$PER17	050064	\$STN	= 000001	\$USWR	056672
\$F\$	= 000000	\$MSGLG	056664	\$PER20	050102	\$SVLAD	051402	\$VECT1	056716
\$GTSWR	054704	\$MSGTY	056646	\$PER21	050120	\$SV\$	= 000000	\$VECT2	056720
\$HALT	052100	\$MSWR	055755	\$PER22	050140	\$SWR	= 163000	\$WASDB	035100
\$HALT2	057024	\$MTYP1	056677	\$PER23	050156	\$SWREG	056670	\$WASSB	034734
\$HIBTS	056750	\$MTYP2	056703	\$PER24	050174	\$SYNRE	035700	\$WAS1D	035214
\$HIOCT	056146	\$MTYP3	056707	\$PER25	044756	\$T	= 000531	\$WAS1S	035050
\$ILLUP	046300	\$MTYP4	056713	\$PER26	050364	\$TESTN	056652	\$XTSTR	051274
\$INVAL	035362	\$NOTRA	057020	\$PER27	050404	\$TKB	002644	\$Y\$	= 000000
\$ITEMB	002015	\$NULL	002356	\$PER30	045204	\$TKS	002642	\$ZAP42	014022
\$I\$	= 000001	\$NWTST	= 000001	\$PER31	050574	\$TN	= 000007	\$Z\$	= 000000
\$KERNE	034202	\$OCNT	054300	\$PER32	050672	\$TPB	002650	\$S\$	= 000000
\$KMAP	036310	\$OCTVL	056630	\$PER33	050740	\$TPFLG	002360	\$T	= 000512
\$K\$	= 000103	\$OCT8	= 056634	\$PER34	051020	\$TPS	002646	\$TT	= 000520
\$L	= 000110	\$OMODE	054302	\$PER35	051052	\$TRAP	056764	\$OFILL	054301
\$LF	002661	\$OVER	051416	\$PER36	051106				

. ABS. 071612 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 1286
 Work file writes: 949
 Size of work file: 25487 Words (100 Pages)
 Size of core pool: 17688 Words (68 Pages)
 Operating system: RSX-11M/PLUS

Elapsed time: 00:32:57.57
 CVMJAO.BIN,CVMJAO/NL:TOC=CVMJAO/ML,CVMJAO.MAC