

Micro Fiche Scan

Name of device(s) tested:

MSV11-J/L/P

Test description:

MSV11-J MEMORY DIAG

MAINDEC Number or Package Identifier (after SEP 1977):

CVMJAB0

Fiche Document Part Number:

AH-U136B-MC

Fiche preparation date unknown, using copyright year:

1985

Image resolution:

1-bit black&white, compressed for minimal file size

COPYRIGHT (C) 1985 by d|i|g|i|t|a|l

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41

.TITLE CVMJABO MSV11-J MEMORY DIAG.
.MLIST TOC
.REM

IDENTIFICATION

PRODUCT CODE: AC-U135B-MC
PRODUCT NAME: CVMJABO MSV11-J MEMORY DIAGNOSTIC
PRODUCT DATE: OCTOBER 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

43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64

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

66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109

TABLE OF CONTENTS

1.0	GENERAL PROGRAM INFORMATION
1.1	Program Purpose (Abstract)
1.2	System Requirements
1.3	Related Documents And Standards
1.4	Diagnostic Hierarchy Prerequisites
1.5	Assumptions
2.0	OPERATING INSTRUCTIONS
2.1	Loading and Starting Procedures
2.2	Default Test Sequence
2.3	Special Environments
2.4	Program Options
2.5	Execution Times
3.0	ERROR INFORMATION
3.1	Error Reporting
3.2	Error Abbreviations
3.3	Error Halts
4.0	PROGRESS REPORTS
5.0	CSR INFORMATION TABLES
5.1	MSV11-J CSR
5.2	MSV11-L/P CSR
6.0	SUB-TEST SUMMARIES
6.1	Tests
6.2	Patterns
7.0	PROGRAM FEATURES
7.1	Fast Data Access Rates
7.2	Bank Zero Testing
7.3	Memory Configuration Map
7.4	Everything You've Always Wanted To Know About SUPERMAC ...
7.5	Memory Management Mapping

111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152

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

154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197

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)

199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250

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

252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278

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

280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323

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

325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368

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.

370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421

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.

423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479

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 "10!". 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

480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528

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.

530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549

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).

551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603

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 "\$ENVM" 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".

605 SW10 = BELL ON ERROR
606 (2000)
607 This causes a bell (or beep or click) on each error
608 trap
609
610 SW9 = LOOP ON ERROR
611 (1000)
612 This will cause looping from failure point back to the
613 last correctly initialized area of the current test.
614
615 SW8 = HALT PROGRAM
616 (400)
617 This initiates the following sequence:
618
619 1. If program is relocated it moves back to bank zero.
620 2. Flush out all possible DBE's.
621 3. Turns off Memory Management.
622 4. Restore loaders.
623 5. Unmap the Unibus Map (if there is one).
624 6. Halt if under APT or ACT branch sel.
625
626
627
628
629
630
631
632
633
634
635 SW7 = DETAILED ERROR REPORTS
636 (200)
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 This inhibits the printing of a map showing the memory
645 configuration - reference section 7 3
646
647
648
649 SW5 = LIMIT MAX ERRORS PER BANK
650 (40)
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
656

658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710

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	BAFPAW	Banks forward, patterns reverse
(4)	2	BAWPAF	Banks worst first, patterns forward.
(6)	3	BAWPAW	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 Ranks 0 1 are MSV11-J and Banks 2,3,4, 5 are MSV11-L.

712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763

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

```
765 ;RELOCATE TEST PROGRAM SPACE (BANK 0 & 1)
766
767
768 TEST 1. BANK 0
769 TEST 2. BANK 0
770 TEST 3. BANK 0
771 TEST 4. BANK 0
772 TEST 5. BANK 0
773 TEST 26. BANK 0
774 TEST 1. BANK 1
775 TEST 2. BANK 1
776 TEST 3. BANK 1
777 TEST 4. BANK 1
778 TEST 5. BANK 1
779 TEST 26. BANK 1
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
```

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

818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871

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 = "BAFPAF" 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 = "BAFPAR" = 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 = "BAWPAF" = 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 = "BAWPAR" = Banks worst first, patterns reverse

This mode first tests each known bad bank completely

873 from 0 to 167 except those requiring relocation*, then
874 presumed good banks are tested from 0 to 167 except
875 those requiring relocation*.
876
877 While testing each bank the patterns are run with the
878 most complex ones first, working to the simple ones.
879
880 MODE 4 = "PAFBAF" = Patterns forward, banks forward
881
882 This mode tests each pattern completely with the simple
883 ones first, building to the more complex.
884
885 While testing each pattern the banks are run from 0 to
886 167 except those requiring relocation*.
887
888
889 MODE 5 = "PAFBAW" = Patterns forward, banks worst first
890
891 This mode tests each pattern completely with the simple
892 ones first, building to the more complex.
893
894 While testing each pattern first each known bad bank
895 from 0 to 167 except those requiring relocation* is
896 run, then presumed good banks are run from 0 to 167
897 except those requiring relocation*.
898
899 MODE 6 = "PARBAF" = Patterns Reverse, Banks Forward
900
901 This mode tests each pattern completely with the most
902 complex ones first, working to the simple ones.
903
904 While testing each pattern the banks are run from 0 to
905 167 except those requiring relocation*.
906
907 MODE 7 = "PARBAW" = Patterns Reverse, Banks Worst First
908
909 This mode tests each pattern completely with the most
910 complex ones first, working to the simple ones.
911
912 While testing each pattern first each known bad bank
913 from 0 to 167 except those that require relocation* is
914 run, then presumed good banks are run from 0 to 167
915 except those requiring relocation*.
916
917
918
919
920
921
922
923
924
925

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)!

927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967

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 "PAFBAF".

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.

969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024

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 "CACMK" 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 FCC 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).

1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079

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).

1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125

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" (Quit)

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.

1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176

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.

1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228

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.

1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279

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.

1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333

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.

1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390

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).

1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440

2.5 Execution Times

2.5.1 Typical (System) -

Execution time depends on many variables; however here are some measured times on an 11/238:

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.

1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486

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

1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533

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.

1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583

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

1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608

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 "APHLT".

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.

1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661

4.0 PROGRESS REPORTS

Pass complete typeouts as follows:

END PASS	0	0
END PASS	0	1
END PASS	0QV	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

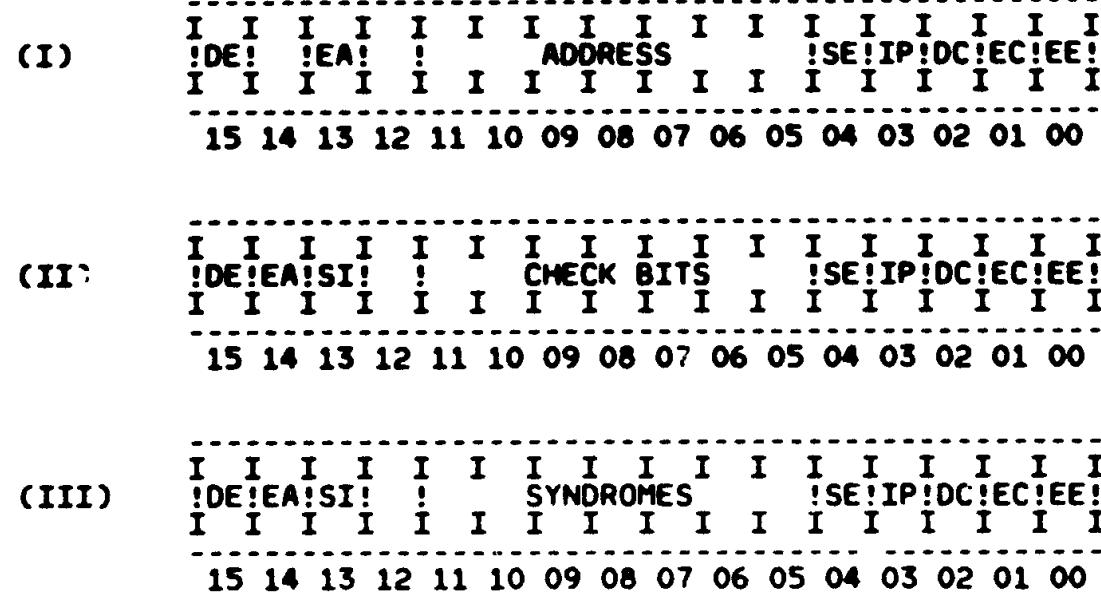
The following is a picture view of the current control status registers which can be tested by this program. It shows bit assignments and definitions to provide a handy reference, and shows the similarities and differences between each one:

NOTE

All unused bits in each CSR are equal to zero.

1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716

5.1 MSV11-J CSR



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.

1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771

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

1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826

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

1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880

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.

1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1930
 1931
 1932
 1933
 1934

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

1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959

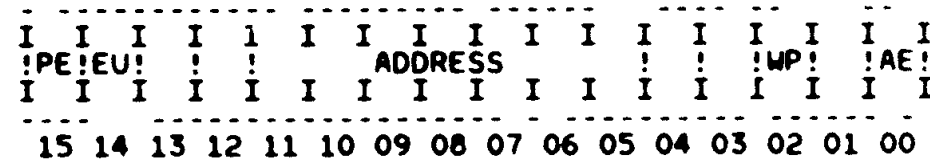
have been generated on the data written. This means that if a single or double error existed in the location accessed, it would be cleared (unless the errors were hard).

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.

1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015

5.2 MSV11-L/P CSR



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 zero.

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

2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062

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

2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109

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:

2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140

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).

2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159

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.

2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174

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.

2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193

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.

2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218

6.2.2.5 Test 4 Rotating Zeros Test

Writes a background pattern of Ones. Rotates a Zero Carry Bit left thru each pair 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.

2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244

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.

2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259

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.

2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285

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.

2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310

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

2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350

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

2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368

6.2.2.11 Test 1/ 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 or correct Test.
3. Do (1-2) again for a complement Test.

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

2370
 2371
 2372
 2373
 2374
 2375
 2376
 2377
 2378
 2379
 2380
 2381
 2382
 2383
 2384
 2385
 2386
 2387
 2388
 2389
 2390
 2391
 2392
 2393
 2394
 2395
 2396
 2397
 2398
 2399
 2400
 2401
 2402
 2403
 2404
 2405
 2406
 2407

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

2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458

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.

2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512

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).

2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534

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).

2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555

6.2.2.16 Test 24 Fast Galloping Pattern Test

This does a classical galloping pattern except that address ng s
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).

2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588

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 "SECDHI" 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.

2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606

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.

2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624

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).

2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678

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).

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
2729

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).

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
2779

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

2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805

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.

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
2835

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.

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
2863

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

2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889

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.

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
2918

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.

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
2954

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

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
2995

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

2997
2998
2999
3000
3001
3002
3003
3004
3005
3006
3007
3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040

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 correct 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

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
3077

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

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
3115

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

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
3150

6.2.2.33 Test 47 NO CSR Update On SBE with existing 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

3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164

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.

3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208
3209
3210
3211
3212
3213
3214
3215
3216
3217
3218
3219

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

3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274

Utilization of the full PDP 11 Instruction Set to speed pattern algorithms (principally the S08).

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

3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321

This map is printed out immediately after sizing the memory unless SW6 is set (reference section 2.4.1). It can also be printed at any later time in Field Service Mode (reference section 2.4.4.8.7)

Example:

```

                                MEMORY CONFIGURATION MAP
                                16K BANKS
                                1       2       3       4       5       6       7
                                0123456701234567012345670123456701234567012345670123
ERRORS                          XX
MEMTYPE EEEEEEP PPPPPPPPPPPPPPPPPPPPPPPPPPEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
CSR      000000001111111122222222222222224444444444444444444444444444444444444444
PROTECT PP      I      I      P      I
                                1       1       1       1       1       1       1
                                0       1       2       3       4       5       6
                                4567012345670123456701234567012345670123456701234567
ERRORS
MEMTYPE EEEE
CSR      4444
PROTECT

```

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:

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
3367

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.

```
3369
3370
3371
3372
3373
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389
3390
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401
3402
3403
3404
3405
3406
3407
3408
3409
3410
3411
3412
3413
3414
3415
3416
3417
3418
3419
3420
3421
3422
```

```
7.4.1 Sample Source File -
.ENABL ABS
.ENABL AMA
.MCALL .SUPER
.SUPER
;LST##=0
BITS=40
A: 0
B: 0
C: 0
D: 0
E: 0
F: 0
G: 0
H: 0
I: 0
J: 0
.PAGE
;LET EXAMPLES
LET RO := A
LET B := C + D
LET E := F + 1
LET G := H + 2
LET J := J + 01
LET A :B= B
;IF EXAMPLES
IF A IS TRUE
MOV 23,D
END ;OF IF A
IF B IS FALSE
MOV 34,E
END ;OF IF B
IF A EQ B THEN LET C := D
IF A LT B
MOV C,D
ELSE
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
```

```
3424          END ;OF IFB A
3425          IF RESULT IS EQ
3426            MOV  A,B
3427          END ;OF IF RESULT
3428          IF BITS SET.IN A
3429            MOV  B,C
3430          END ;OF IF BITS
3431          IF BITS OFF.IN A
3432            MOV  C,D
3433          END ;OF IF BITS
3434 ;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
3435 ;ON.ERROR EXAMPLES
3436          ON.ERROR
3437            MOV  A,B
3438          ELSE
3439            MOV  C,B
3440          END ;OF ON.ERROR
3441          ON.NOERROR
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              MOV  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
3476
```

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
3530

B
C
D
E
F
END ;OF CASE RO
.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			BITS=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

```

3532      36 000174 013737 000010 000006
3533      37 000202
3534      38 000202
3535      39 000222 013737 000012 000014
3536      40 000230
3537      41 000230
3538      42 000250 013737 000012 000014
3539      43 000256
3540      44 000256
3541      45 000276 013737 000016 000022
3542      46 000304
3543      47 000306 013737 000010 000022
3544      48 000314
3545      49 000314
3546      50 000334 013737 000016 000022
3547      51 000342
3548      52 000344 013737 000010 000022
3549      53 000352
3550      54 000352
3551      55 000354 013737 000000 000002
3552      56 000362
3553      57 000362
3554      58 000372 013737 000002 000004
3555      59 000400
3556      60 000400
3557      61 000410 013737 000004 000006
3558      62 000416
3559      63
3560      64
3561      65 000416
3562      66 000420 013737 000000 000002
3563      67 000426
3564      68 000430 013737 000004 000002
3565      69 000436
3566      70 000436
3567      71 000440 013737 000004 000002
3568      72 000446
3569      73 000450 013737 000000 000002
3570      74 000456
3571
3572
3573      .MAIN. MACRO M1111 01-APR-79 16:41 PAGE 3-1
3574
3575
3576      75 000456
3577      76
3578      77 000466
3579      78 000474 005237 000000
3580      79 000500
3581      80 000514
3582      81 000516 005360 000000
3583      82 000522
3584      83 000534
    
```

```

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

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
    
```

```

3586      84 000542 063737 000000 000002
3587      85 000550
3588      86
3589      87 000566
3590      88 000566
3591      89 000570 116037 000000 000002
3592      90 000576
3593      91 000604
3594      92 000614
3595      93 000620
3596      94 000626
3597      95 000636
3598      96
3599      97 000636
3600      98 000640
3601      99 000644
3602     100
3603     101 000650 013700 000000
3604     102 000654
3605     103 000664 000000
3606     104 000666 000002
3607     105 000670 000004
3608     106 000672 000006
3609     107 000674 000010
3610     108 000676 000012
3611     109 000700
3612     110
3613     111      000001

```

```

      ADD  A,B
      END ;OF FOR I
;BEGIN EXAMPLES
      BEGIN ALPHA
      FOR RO := 0 TO 167
      MOVB  A(RO),B
      IF B LT 0 THEN LEAVE ALPHA
      END ;OF FOR RO
      FOR RO := 400 TO 567
      IF B GE 0 THEN LEAVE ALPHA
      END ;OF FOR RO
      END ALPHA
;RETURN EXAMPLES
      $RETURN
      $RETURN ERROR
      $RETURN NOERROR
;CASE EXAMPLES
      MOV  A,RO
      CASE RO
      A
      B
      C
      D
      E
      F
      END ;OF CASE RO
      .END

```

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

```

3618      1 000000
3619      2
3620      3
3621      4 000000
3622      5      000000
3623      6      000040
3624      7 000000 000000
3625      8 000002 000000
3626      9 000004 000000
3627     10 000006 000000
3628     11 000010 000000
3629     12 000012 000000
3630     13 000014 000000
3631     14 000016 000000
3632     15 000020 000000
3633     16 000022 000000

```

```

      .ENABL ABS
      .ENABL AMA
      .MCALL .SUPER
      .SUPER
      LST$$=0
      BITS=40
      A: 0
      B: 0
      C: 0
      D: 0
      E: 0
      F: 0
      G: 0
      H: 0
      I: 0
      J: 0

```


3639	18			
3640	19	000024		
3641		000024	013700	000000
3642	20	000030		
3643		000030	013737	000004
3644		000036	063737	000006
3645	21	000044		
3646		000044	013737	000012
3647		000052	005237	000010
3648	22	000056		
3649		000056	013737	000016
3650		000064	062737	000002
3651	23	000072		
3652		000072	062737	000001
3653	24	000100		
3654		000100	113737	000002
3655	25			
3656	26	000106		
3657		000106	005737	000000
3658		000112	001403	
3659	27	000114	012737	000023
3660	28	000122		
3661		000122		
3662	29	000122		
3663		000122	005737	000002
3664		000126	001003	
3665	30	000130	012737	000034
3666	31	000136		
3667		000136		
3668	32	000136		
3669		000136	023737	000000
3670		000144	001003	
3671		000146	013737	000006
3672		000154		
3673	33	000154		
3674		000154	023737	000000
3675		000162	002004	
3676	34	000164	013737	000004
3677	35	000172		
3678		000172	000403	
3679		000174		
3680	36	000174	013737	000010
3681	37	000202		
3682		000202		
3683	38	000202		
3684		000202	023737	000000
3685		000210	001007	
3686		000212	023737	000004
3687		000220	001403	
3688	39	000222	013737	000012
3689	40	000230		

```

;LET EXAMPLES
LET RO := A
MOV A,RO
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
MOVB 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

```

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
3743

000230
41 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 123737 000000 000002
000256 001010
000264 023727 000004 000001
000274 001004
45 000276 013737 000016 000022
46 000304
000304 000403
000306
47 000306 013737 000010 000022
48 000314
000314
49 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 001003
000352 013737 000000 000002
55 000354 013737 000000 000002
56 000362
000362
57 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

```

3745      61 000410 013737 000004 000006
3746
3747      62 000416
3748          000416
3749      63
3750      64
3751      65 000416
3752          000416 103004
3753      66 000420 013737 000000 000002
3754      67 000426
3755          000426 000403
3756          000430
3757
3758      68 000430 013737 000004 000002
3759
3760      69 000436
3761          000436
3762      70 000436
    
```

```

MOV C,D
END ;OF IF BITS
L16:
;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
;ON.ERROR EXAMPLES
ON.ERROR
BCC L17
MOV A,B
ELSE
BR L20
L17:
MOV C,B
END ;OF ON.ERROR
L20:
ON.NOERROR
    
```

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

```

3766
3767
3768
3769      71 000436 103404
3770      72 000440 013737 000004 000002
3771          000446 000403
3772          000450
3773      73 000450 013737 000000 000002
3774      74 000456
3775          000456
3776      75 000456
3777          000456 103003
3778          000460 113737 000002 000000
3779          000466
3780
3781      76
3782      77 000466
3783          000466 012737 177773 000020
3784          000474
3785      78 000474 005237 000000
3786      79 000500
3787          000500 005237 000020
3788          000504 023727 000020 000023
3789          000512 003770
3790      80 000514
3791          000514 005000
3792          000516
3793      81 000516 05360 000000
3794      82 000522
3795          000522 062700 000004
3796          000526 020027 000140
    
```

```

BCS L21
MOV C,B
ELSE
BR L22
L21:
MOV A,B
END ;OF ON.NOERROR
L22:
ON.ERROR THEN LET A :B= B
BCC L23
MOVB B,A
L23:
;FOR EXAMPLES
FOR I := -5 TO 23
MOV -5,I
B0:
INC A
END ;OF FOR I
INC I
CMP I, 23
BLE B0
E0:
FOR RO := 0 TO 140 BY 4
CLR RO
B1:
DEC A(RO)
END ;OF FOR RO
ADD 4,RO
CMP RO, 140
    
```

3798
3799
3800
3801
3802
3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813
3814
3815
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848
3849
3850

```

000532 003771
000534
83 000534 012737 000133 000020
000542
84 000542 063737 000000 000002
85 000550 162737 000002 000020
000556 023727 000020 000003
000564 002366
000566
86
87 000566
000566
88 000566 005000
000570
89 000570 116037 000000 000002
90 000576 005737 000002
000602 002415
91 000604 005200
000606 020027 000167
000612 003766
000614
92 000614 012700 000400
000614

```

```

BLE B1
E1: FOR I := 133 DOWNT0 3 BY 2
MOV 133,I
B2: ADD A,B
END ;OF FOR 1
SUB 2,I
CMP I, 3
BGE B2
E2: ;BEGIN EXAMPLES
BEGIN ALPHA
B3: FOR RO := 0 TO 167
CLR RO
B4: MOVB A(RO),B
IF B LT 0 THEN LEAVE ALPHA
TST B
BLT E3
END ;OF FOR RO
INC RO
CMP RO, 167
BLE B4
E4: FOR RO := 400 TO 567
MOV 400,RO

```

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

```

000620
93 000620 005737 000002
000624 002004
94 000626 005200
000630 020027 000567
000634 003771
000636
95 000636
000636
96
97 000636 000207
000636
98 000640 000261
000642 000207
99 000644 000241
000646 000207

```

```

B5: IF B GE 0 THEN LEAVE ALPHA
TST B
BGE E3
END ;OF FOR RO
INC RO
CMP RO, 567
BLE B5
E5: END ALPHA
E3: ;RETURN EXAMPLES
;RETURN
RTS PC
;RETURN ERROR
SEC
RTS PC
;RETURN NOERROR
CLC
RTS PC

```

3852
3853
3854
3855
3856
3857
3858
3859
3860
3861
3862
3863
3864
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901

```

100
101 000650 013700 000000
102 000654
    000654 010046
    000656 006316
    000660 004737 000700
103 000664 000000
104 000666 000002
105 000670 000004
106 000672 000006
107 000674 000010
108 000676 000012
109 000700
    000700
    000700 062616
    000702 013646
    000704 004736
110
111          000001
    
```

```

;CASE EXAMPLES
MOV     A,RO
CASE RO
MOV RO,-(SP)
ASL @SP
JSR PC,L24
A
B
C
D
E
F
END ;OF CASE RO
L24:
ADD (SP)+,@SP
MOV @ (SP)+,-(SP)
JSR PC,@(SP)+
.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 11/238)

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

3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915

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



```
3919 .LIST TOC
3920 000000 .ENABL ABS
3921 .ENABL AMA
3922 .DSABL GBL
3923 ;NOTE: CVMJAO.SML IS THE SUPER.MAC SOURCE AND IS RELEASED WITH
3924 ;THIS PROGRAM. ALL THESE .MCALL STATEMENTS REFERENCE THAT FILE.
3925 .MCALL SMACIT,. .PUSH,. .POP,. .TAG,. .BRAN,. .EMIT,. .EMITN,. .EMITL,. .EMITP
3926 .MCALL .IFOPR,. .IS,. .GENBR,. .OPADD,. .OPSUB,. .CLEAR,. .SET,. .CLEARB,. .SETB
3927 .MCALL RNE,REQ,RLT,RGE,RGT,RLE,RPL,RMI,RHI,RLOS,RHIS,RLO,RCS,RCC
3928 .MCALL IF,.OR,.IFARI,.LEAVE,.GOTO,OR,AND,THEN,ELSE,WHILE,CASE
3929 .MCALL FOR,TO,DOWNTO,REPEAT,UNTIL,THRU,END,BEGIN
3930 .MCALL ##END,LEAVE,JUMPTO,GOTO,PUSH,POP,LET
3931 .MCALL .SIMPLE,.ARITH,ORB,ANDB,IFB,UNTILB,WHILEB,ON.ERROR,ON.NOERROR
3932 .MCALL $CALL,$RETURN
3933
3934 .NLIST TTM ;I WANT FAT PAPER!
3935 .LIST MC.SYM ;LIST MACRO CALLS, SYMBOL TABLE
3936 .NLIST MD.CND,ME ;DON'T LIST MACRO DEFS & CONDITIONALS & EXPANSIONS
3937 000000 LST##= 0 ;DEFINED TO LIST SUPERMAC EXPANSIONS
3938 163000 $SWR= 163000 ;USE THESE SYSMAC SWITCHES
3939 000001 $TN= 1 ;FIRST TEST NUMBER TO ONE(1)
3940 000000 SMACIT
```

```

3943 .SBTTL DEFINE TRAPS
3944 ;ALL ENTRIES HERE MUST HAVE A CORRESPONDING ENTRY IN THE
3945 ;TRAP TABLE "$TRPAD" (NEAR END OF PROGRAM).
3946 ;*TRAP DEFINITIONS
3947 ;
3948 ;HERE IS HOW TRAPS WORK IN THIS PROGRAM
3949 ;
3950 ;ALL TRAPS EXECUTE A "TRAP" INSTRUCTION WHICH TAKES THE PROGRAM
3951 ;TO SYMBOLIC LOCATION "$TRAP"
3952 ;
3953 ;AT $TRAP THE PROGRAM PICKS UP THE RIGHT BYTE OF THE TRAP INSTRUCTION
3954 ;AND INDEXES INTO A TABLE AT LOCATION "$TRPAD" WHICH SENDS THE PROGRAM TO
3955 ;THE SPECIFIC ROUTINE TO HANDLE THAT SPECIFIC TRAPS TASK.
3956 ;
3957 ;THE ULTIMATE DESTINATION OF A TRAP INSTRUCTION CAN BE GUESSED AT AS FOLLOWS
3958 ;
3959 ;EXAMPLE:      NOP
3960 ;              NOP
3961 ;              NOP
3962 ;              KERNEL          ;ENTER KERNEL MODE
3963 ;              NOP
3964 ;
3965 ;              ADD A DOLLAR SIGN TO THE SYMBOLIC NAME AND CHECK THE CRF FOR SOMETHING CLOSE
3966 ;              IN THIS CASE THE CRF HAS $KERNE LISTED AS 032546
3967 ;              AT LOCATION 32546 YOU FIND THE ROUTINE $KERNEL
3968 ;
3969 ;              NOTE THAT CRF SYMBOLS ARE TRUCNATED TO 6 CHARACTERS
3970 ;              SYMBOLIC NAMES GREATER THAT 6 CHARACTERS ARE USED SO I CAN
3971 ;              REMEMBER WHAT THEY MEAN!
3972 ;
3973 ;              ALL GOOD TRAP ROUTINES RETURN VIA AN "RTI" INSTRUCTION
3974 ;              TYPEIT= 104401      ;;TTY TYPEOUT ROUTINE
3975 ;              TYPDC= 104402      ;;TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3976 ;              TYPOS= 104403      ;;TYPE OCTAL NUMBER (NO LEADING ZEROS)
3977 ;              TYPON= 104404      ;;TYPE OCTAL NUMBER (AS PER LAST CALL)
3978 ;              TYPDS= 104405      ;;TYPE DECIMAL NUMBER (WITH SIGN)
3979 ;              TYPBN= 104406      ;;TYPE BINARY (ASCII) NUMBER
3980 ;
3981 ;              GTSWR= 104407      ;;GET SOFT-SWR SETTING
3982 ;              CKSWR= 104410      ;;TEST FOR CHANGE IN SOFT-SWR
3983 ;
3984 ;              RDCHR= 104411      ;;TTY TYPEIN CHARACTER ROUTINE
3985 ;              RDLIN= 104412      ;;TTY TYPEIN STRING ROUTINE
3986 ;              RDOCT= 104413      ;;READ AN OCTAL NUMBER FROM TTY
3987 ;              RDDEC= 104414      ;;READ A DECIMAL NUMBER FROM TTY
3988 ;
3989 ;              SAVREG= 104415      ;;SAVE R0-R5 ROUTINE
3990 ;              RESREG= 104416      ;;RESTORE R0-R5 ROUTINE
3991 ;
3992 ;              KERNEL= 104417      ;ENTER KERNEL MODE
3993 ;
3994 ;              ENERGIZE=104420      ;TURN ON MEMORY MANAGEMENT & TRAPS
3995 ;              DEENERGIZE=104421    ;TURN OFF MEMORY MANAGEMENT & TRAPS
3996 ;              KMAP= 104422        ;MAP KERNEL 1 TO 1
3997 ;
3998 ;              CACHON= 104423      ;TURN ON CACHE
3999 ;              CACHOFF=104424      ;TURN OFF CACHE

```


4000			
4001	104425	LOADCSR=104425	;LOAD CORRECT CSR
4002	104426	READCSR=104426	;READ CORRECT CSR
4003			
4004	104427	PERR01= 104427	;PROGRAM DETECTED ERROR
4005	104430	PERR02= 104430	;PROGRAM DETECTED ERROR
4006	104431	PERR03= 104431	;PROGRAM DETECTED ERROR
4007	104432	PERR04= 104432	;PROGRAM DETECTED ERROR
4008	104433	PERR07= 104433	;PROGRAM DETECTED ERROR
4009	104434	PERR10= 104434	;PROGRAM DETECTED ERROR
4010	104435	PERR11= 104435	;PROGRAM DETECTED ERROR
4011	104436	PERR12= 104436	;PROGRAM DETECTED ERROR
4012	104437	PERR13= 104437	;PROGRAM DETECTED ERROR
4013	104440	PERR14= 104440	;PROGRAM DETECTED ERROR
4014	104441	PERR15= 104441	;PROGRAM DETECTED ERROR
4015	104442	PERR16= 104442	;PROGRAM DETECTED ERROR
4016	104443	PERR17= 104443	;PROGRAM DETECTED ERROR
4017	104444	PERR20= 104444	;PROGRAM DETECTED ERROR
4018	104445	PERR21= 104445	;PROGRAM DETECTED ERROR
4019	104446	PERR22= 104446	;PROGRAM DETECTED ERROR
4020	104447	PERR23= 104447	;PROGRAM DETECTED ERROR
4021	104450	PERR24= 104450	;PROGRAM DETECTED ERROR
4022	104451	PERR25= 104451	;PROGRAM DETECTED ERROR
4023	104452	PERR26= 104452	;PROGRAM DETECTED ERROR
4024	104453	PERR27= 104453	;PROGRAM DETECTED ERROR
4025	104454	PERR30= 104454	;PROGRAM DETECTED ERROR
4026	104455	PERR31= 104455	;PROGRAM DETECTED ERROR
4027	104456	PERR32= 104456	;PROGRAM DETECTED ERROR
4028	104457	PERR33= 104457	;PROGRAM DETECTED ERROR
4029	104460	PERR34= 104460	;PROGRAM DETECTED ERROR
4030	104461	PERR35= 104461	;PROGRAM DETECTED ERROR
4031	104462	PERR36= 104462	;PROGRAM DETECTED ERROR
4032	104463	PERR37= 104463	;PROGRAM DETECTED ERROR
4033	104464	PERR40= 104464	;PROGRAM DETECTED ERROR
4034	104465	PERR41= 104465	;PROGRAM DETECTED ERROR
4035	104466	PERR42= 104466	;PROGRAM DETECTED ERROR
4036	104467	PERR43= 104467	;PROGRAM DETECTED ERROR

4037			
4038	104470	ECCDIS= 104470	:DISABLE ECC ON ALL CSR'S
4039	104471	ECC1DIS=104471	:DISABLE ECC ON 1 SELECTED CSR
4040	104472	ECCINIT=104472	:INITIALIZE ALL ECC CSR'S
4041	104473	ECC1INIT=104473	:INITIALIZE 1 SELECTED ECC CSR
4042	104474	CBCSR= 104474	:WRITE GENERATED CHECKBITS IN ALL CSR'S
4043	104475	CB1CSR= 104475	:WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
4044	104476	WASSBE= 104476	:WAS THERE A SBE ON ANY CSR?
4045	104477	WAS1SBE=104477	:WAS THERE A SBE ON 1 SELECTED CSR?
4046	104500	WASDBE= 104500	:WAS THERE A DBE ON ANY CSR?
4047	104501	WAS1DBE=104501	:WAS THERE A DBE ON 1 SELECTED CSR?
4048	104502	CLRCSR= 104502	:CLEAR ALL CSR'S
4049	104503	CLR1CSR=104503	:CLEAR 1 SELECTED CSR
4050	104504	CHKDIS= 104504	:DISABLE ECC & WRITE CHECKBITS FROM ALL CSR'S
4051	104505	CHK1DIS=104505	:DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
4052	104506	ENASBE= 104506	:ENABLE TRAPS ON SBE'S FROM ALL CSR'S
4053	104507	ENA1SBE=104507	:ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
4054	104510	TSTREAD=104510	:TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
4055	104511	INVALID=104511	:INVALIDATE BACKGROUND PATTERN ON "BANK"
4056	104512	ERRGEN =104512	:CHECK ERROR ADDRESS
4057	104513	CBREG =104513	:ENABLES CHECKBIT REGISTER
4058	104514	SYNREG =104514	:ENABLES SYNDROME BIT REGISTER

```

4061          .SBTTL DEFINE BASIC PDP11 STUFF
4062
4063          ;*INITIAL ADDRESS OF THE STACK POINTER
4064          002000  STACK= 2000          ;;FIRST ADDRESS OF THE STACK
4065          002000  KERSTK= STACK       ;;KERNEL STACK
4066          000740  SUPSTK= 740        ;;SUPERVISOR STACK
4067          000700  USESTK= 700        ;;USER STACK
4068          104000  ERROR=EMT          ;;BASIC DEFINITION OF ERROR CALL
4069          000004  SCOPE=IOT          ;;BASIC DEFINITION OF SCOPE CALL
4070          177776  PSW= 177776       ;;PROCESSOR STATUS WORD
4071          ;STKLM=177774              ;;STACK LIMIT REGISTER
4072          ;PIRQ= 177772              ;;PROGRAM INTERRUPT REQUEST REGISTER
4073          177570  DSWR= 177570       ;;HARDWARE SWITCH REGISTER
4074          177570  DDISP= 177570     ;;HARDWARE DISPLAY REGISTER
4075          177546  LKS= 177546       ;;LINE CLOCK (KW11-L) STATUS REGISTER
4076
4077          ;*MISCELLANEOUS DEFINITIONS
4078          000011  HT= 11              ;;CODE FOR HORIZONTAL TAB
4079          000012  LF= 12              ;;CODE LINE FEED
4080          000015  CR= 15              ;;CODE CARRIAGE RETURN
4081          000200  CRLF= 200          ;;CODE FOR CARRIAGE RETURN-LINE FEED
4082          000007  MFPT= 7            ;;CODE FOR PROCESSOR TYPE INSTRUCTION
4083
4084          ;*GENERAL PURPOSE REGISTER DEFINITIONS
4085          ;SP=R6                        ;;STACK POINTER
4086          ;KSP=SP                      ;;KERNEL STACK POINTER
4087          000006  SSP=SP              ;;SUPERVISOR STACK POINTER
4088          000006  USP=SP              ;;USER STACK POINTER
4089          ;PC=R7                        ;;PROGRAM COUNTER
4090
4091          ;*"SWITCH REGISTER" SWITCH DEFINITIONS
4092          100000  SW15= 100000
4093          040000  SW14= 40000
4094          020000  SW13= 20000
4095          010000  SW12= 10000
4096          004000  SW11= 4000
4097          002000  SW10= 2000
4098          001000  SW9= 1000
4099          000400  SW8= 400
4100          000200  SW7= 200
4101          000100  SW6= 100
4102          000040  SW5= 40
4103          000020  SW4= 20
4104          000010  SW3= 10
4105          000004  SW2= 4
4106          000002  SW1= 2
4107          000001  SW0= 1
4108
4109          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
4110          100000  BIT15= 100000
4111          040000  BIT14= 40000
4112          020000  BIT13= 20000
4113          010000  BIT12= 10000
4114          004000  BIT11= 4000
4115          002000  BIT10= 2000
4116          001000  BIT9= 1000
4117          000400  BIT8= 400

```

```

4118      000200      BIT7= 200
4119      000100      BIT6= 100
4120      000040      BIT5= 40
4121      000020      BIT4= 20
4122      000010      BIT3= 10
4123      000004      BIT2= 4
4124      000002      BIT1= 2
4125      000001      BIT0= 1
4126
4127      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
4128      000004      ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
4129      000010      RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
4130      ;TBITVEC=14      ;; "T" BIT
4131      ;TRTVEC=      14      ;;TRACE TRAP
4132      ;BPTVEC=      14      ;;BREAKPOINT TRAP (BPT)
4133      000020      IOTVEC= 20     ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
4134      000024      PWRVEC= 24     ;;POWER FAIL
4135      000030      EMTVEC= 30     ;;EMULATOR TRAP (EMT) **ERROR**
4136      000034      TRAPVEC=34     ;; "TRAP" TRAP
4137      C00060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
4138      ;TPVEC= 64      ;;TTY PRINTER VECTOR
4139      ;LKVEC= 100     ;;LINE CLOCK (KW11-L) VECTOR
4140      000114      CACHVEC=114    ;;CACHE ERROR INTERRUPT VECTOR
4141      000114      PARVEC=CACHVEC
4142      ;PIRQVEC=240     ;;PROGRAM INTERRUPT REQUEST VECTOR
4143      000250      MMVEC= 250     ;;MEMORY MANAGEMENT VECTOR
4144      ;SBTTL DEFINE CACHE REGISTERS
4145      ;MEMERR = 177744      ;;CACHE ERROR REGISTER
4146      177746      CONTRL = 177746 ;;MEMORY CONTROL REGISTER
4147      177750      MAINT = 177750  ;;MEMORY MAINTENANCE REGISTER
4148      ;HITMIS = 177752    ;;HIT MISS REGISTER "1" IMPLIES HIT IN CACHE
4149      177754      DATARG = 177754 ;;DATA REGISTER
4150
4151      ;SBTTL DEFINE CPU REGISTERS
4152      177766      CPUERR = 177766 ;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
4153
4154      ;SBTTL DEFINE MEMORY MANAGEMENT REGISTERS
4155      ;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES
4156      177572      MMR0= 177572
4157      177574      MMR1= 177574
4158      177576      MMR2= 177576
4159      172516      MMR3= 172516
4160
4161      ;*USER "I" PAGE DESCRIPTOR REGISTERS
4162      177600      UIPDR0= 177600
4163      ;UIPDR1= 177602
4164      ;UIPDR2= 177604
4165      ;UIPDR3= 177606
4166      ;UIPDR4= 177610
4167      ;UIPDR5= 177612
4168      ;UIPDR6= 177614
4169      ;UIPDR7= 177616
4170
4171      ;*USER "D" PAGE DESCRIPTOR REGISTORS
4172      ;UDPDR0= 177620
4173      ;UDPDR1= 177622
4174      ;UDPDR2= 177624

```

```

4175          ;UDPDR3=      177626
4176          ;UDPDR4=      177630
4177          ;UDPDR5=      177632
4178          ;UDPDR6=      177634
4179          ;UDPDR7=      177636
4180
4181          ;*USER "I" PAGE ADDRESS REGISTERS
4182          177640 FASTCITY=UIPAR0
4183          177640 UIPAR0= 177640          ;PATTERN PROGRAM SPACE
4184          177642 UIPAR1= 177642          ;PATTERN PROGRAM SPACE
4185          177644 UIPAR2= 177644          ;PATTERN PROGRAM SPACE
4186          177646 UIPAR3= 177646          ;PATTERN PROGRAM SPACE
4187          177650 UIPAR4= 177650          ;PATTERN PROGRAM SPACE
4188          177652 UIPAR5= 177652          ;PATTERN PROGRAM SPACE
4189          177654 UIPAR6= 177654          ;PATTERN PROGRAM SPACE
4190          ;UIPAR7=      177656          ;PATTERN PROGRAM SPACE
4191
4192          ;*USER "D" PAGE ADDRESS REGISTERS
4193          177660 UDPARO= 177660        ;PATTERN PROGRAM SPACE
4194          ;UDPAR1=      177662          ;PATTERN PROGRAM SPACE
4195          ;UDPAR2=      177664          ;PATTERN PROGRAM SPACE
4196          ;UDPAR3=      177666          ;PATTERN PROGRAM SPACE
4197          ;UDPAR4=      177670          ;PATTERN PROGRAM SPACE
4198          ;UDPAR5=      177672          ;PATTERN PROGRAM SPACE
4199          ;UDPAR6=      177674          ;PATTERN PROGRAM SPACE
4200          177676 UDPAR7= 177676        ;PATTERN PROGRAM SPACE
4201
4202          ;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
4203          172200 SIPDR0= 172200
4204          ;SIPDR1=      172202
4205          ;SIPDR2=      172204
4206          ;SIPDR3=      172206
4207          ;SIPDR4=      172210
4208          ;SIPDR5=      172212
4209          ;SIPDR6=      172214
4210          ;SIPDR7=      172216
4211
4212          ;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
4213          ;SDPDR0=      172220
4214          ;SDPDR1=      172222
4215          ;SDPDR2=      172224
4216          ;SDPDR3=      172226
4217          ;SDPDR4=      172230
4218          ;SDPDR5=      172232
4219          ;SDPDR6=      172234
4220          ;SDPDR7=      172236
4221
4222          ;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
4223          172240 SIPAR0= 172240
4224          ;SIPAR1=      172242
4225          ;SIPAR2=      172244
4226          172246 SIPAR3= 172246          ;TEST AREA
4227          ;SIPAR4=      172250          ;TEST AREA
4228          172252 SIPAR5= 172252          ;TEST AREA
4229          172254 SIPAR6= 172254          ;TEST AREA
4230          ;SIPAR7=      172256
4231
    
```

```
4232 ;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
4233 172260 SDPAR0= 172260
4234 ;SDPAR1= 172262
4235 ;SDPAR2= 172264
4236 ;SDPAR3= 172266
4237 ;SDPAR4= 172270
4238 172272 SDPAR5= 172272
4239 172274 SDPAR6= 172274
4240 172276 SDPAR7= 172276
4241
4242 ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
4243 172300 KIPDR0= 172300
4244 ;KIPDR1= 172302
4245 ;KIPDR2= 172304
4246 ;KIPDR3= 172306
4247 ;KIPDR4= 172310
4248 ;KIPDR5= 172312
4249 ;KIPDR6= 172314
4250 ;KIPDR7= 172316
4251
4252 ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
4253 ;KDPDR0= 172320
4254 ;KDPDR1= 172322
4255 ;KDPDR2= 172324
4256 ;KDPDR3= 172326
4257 ;KDPDR4= 172330
4258 ;KDPDR5= 172332
4259 ;KDPDR6= 172334
4260 ;KDPDR7= 172336
4261
4262 ;*KERNEL "I" PAGE ADDRESS REGISTERS
4263 172340 KIPAR0= 172340
4264 ;KIPAR1= 172342
4265 ;KIPAR2= 172344
4266 ;KIPAR3= 172346
4267 172350 KIPAR4= 172350
4268 172352 KIPAR5= 172352
4269 172354 KIPAR6= 172354
4270 ;KIPAR7= 172356
4271
4272 ;*KERNEL "D" PAGE ADDRESS REGISTERS
4273 172360 KDPAR0= 172360
4274 ;KDPAR1= 172362
4275 ;KDPAR2= 172364
4276 ;KDPAR3= 172366
4277 ;KDPAR4= 172370
4278 ;KDPAR5= 172372
4279 172374 KDPAR6= 172374
4280 172376 KDPAR7= 172376
4281
```

```
4284 .SBTTL DEFINE Q-BUS MAP REGISTERS
4285 ;*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLXX'
4286 ;*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'
4287 170200 MAPL0 = 170200
4288 170202 MAPH0 = 170202
4289 170204 MAPL1 = 170204
4290 ;MAPH1 = 170206
4291 ;MAPL2 = 170210
4292 ;MAPH2 = 170212
4293 ;MAPL3 = 170214
4294 ;MAPH3 = 170216
4295 ;MAPL4 = 170220
4296 ;MAPH4 = 170222
4297 ;MAPL5 = 170224
4298 ;MAPH5 = 170226
4299 ;MAPL6 = 170230
4300 ;MAPH6 = 170232
4301 ;MAPL7 = 170234
4302 ;MAPH7 = 170236
4303 ;MAPL10 = 170240
4304 ;MAPH10 = 170242
4305 ;MAPL11 = 170244
4306 ;MAPH11 = 170246
4307 ;MAPL12 = 170250
4308 ;MAPH12 = 170252
4309 ;MAPL13 = 170254
4310 ;MAPH13 = 170256
4311 ;MAPL14 = 170260
4312 ;MAPH14 = 170262
4313 ;MAPL15 = 170264
4314 ;MAPH15 = 170266
4315 ;MAPL16 = 170270
4316 ;MAPH16 = 170272
4317 ;MAPL17 = 170274
4318 ;MAPH17 = 170276
4319 ;MAPL20 = 170300
4320 ;MAPH20 = 170302
4321 ;MAPL21 = 170304
4322 ;MAPH21 = 170306
4323 ;MAPL22 = 170310
4324 ;MAPH22 = 170312
4325 ;MAPL23 = 170314
4326 ;MAPH23 = 170316
4327 ;MAPL24 = 170320
4328 ;MAPH24 = 170320
4329 ;MAPL25 = 170324
4330 ;MAPH25 = 170326
4331 ;MAPL26 = 170330
4332 ;MAPH26 = 170332
4333 ;MAPL27 = 170334
4334 ;MAPH27 = 170336
4335 ;MAPL30 = 170340
4336 ;MAPH30 = 170342
4337 ;MAPL31 = 170344
4338 ;MAPH31 = 170346
4339 ;MAPL32 = 170350
4340 ;MAPH32 = 170352
```

4341		;MAPL33 = 170354
4342		;MAPH33 = 170356
4343		;MAPL34 = 170360
4344		;MAPH34 = 170362
4345		;MAPL35 = 170364
4346		;MAPH35 = 170366
4347		;MAPL36 = 170370
4348		;MAPH36 = 170372
4349		;MAPL37 = 170374
4350		;MAPH37 = 170376

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

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

4359		
4360		.SBTTL DEFINE PARAMETERS
4361	060000	FIRST=60000 ;START OF THE 16K TEST PATTERN AREA
4362	157776	LAST=157776 ;END OF THE 16K TEST PATTERN AREA
4363	040000	SIZE=40000 ;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)

5012		
5013	
5014		
5015		.NLIST MD ;DON'T NEED TO SEE THEM ANY MORE
5016		
5035		

Address	Value	Variable Name	Description
5083		.SBTTL VARIABLES INITIALIZED TO ZERO	
5084		; THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS	
5085		; USED IN THE PROGRAM.	
5086	002000	\$CHTAG:	; START OF COMMON TAGS
5087	002000	UFDSET:0	; USER FRIENDLY FLAG
5088	002002	SELONLY:0	; SELECT ONLY BANKS MARKED BY FIELD SERVICE MODE FLAG
5089	002004	DIAGFLAG:0	; SET FOR SHIFTING DIAGONAL TEST
5090	002006	KAMIKAZE:0	; SET FOR KAMIKAZE MODE TESTING
5091	002010	SKIPKAMI:0	; USED TO SKIP RESTORING KAMIKAZE MODE WHEN MODIFIED
5092		; NEXT TWO BYTES ARE DISPLAYED IN THE DISPLAY REGISTER	
5093	002012	\$PATMAR: .BYTE 0	; PATTERN NUMBER
5094	002013	\$BANK: .BYTE 0	; BANK & SIGN
5095	002014	\$ERFLG: .BYTE 0	; CONTAINS ERROR FLAG
5096	002015	\$ITEMB: .BYTE 0	; CONTAINS ITEM CONTROL BYTE
5097	002016	LASTERROR: .WORD 0	; NUMBER OF ERRORS ON LAST PASS
5098	002020	ERRPC: .WORD 0	; CONTAINS PC OF ERROR FOR TIMEOUT
5099	002022	BADPC: .WORD 0	; CONTAINS PC OF ERROR
5100	002024	ERRSP: .WORD 0	; CONTAINS SP OF ERROR FOR TIMEOUT
5101	002026	BADSP: .WORD 0	; CONTAINS SP OF ERROR
5102	002030	ERRPSW: .WORD 0	; CONTAINS PSW OF ERROR FOR TIMEOUT
5103	002032	BADPSW: .WORD 0	; CONTAINS PSW OF ERROR
5104	002034	ADDRESS: .WORD 0	; CONTAINS ADDRESS OF 'BAD' DATA
5105	002036	PADDRESS: .WORD 0	; ADDRESS OF PARITY ERROR
5106	002040	PHYADD: .WORD 0,0	; 22 BIT PHYSICAL ADDRESS
5107	002044	GOOD: .WORD 0	; CONTAINS 'GOOD' DATA
5108	002046	GOOD2: .WORD 0	; CONTAINS 'GOOD2' DATA
5109	002050	GOOD3: .WORD 0	; CONTAINS 'GOOD3' DATA
5110	002052	BAD: .WORD 0	; CONTAINS 'BAD' DATA
5111	002054	BAD2: .WORD 0	; CONTAINS 'BAD2' DATA
5112	002056	BAD3: .WORD 0	; CONTAINS 'BAD3' DATA
5113	002060	BAD3OR: .WORD 0	; XOR OF GOOD & BAD = BAD BITS!
5114	002062	\$AUTO: .WORD 0	; AUTOMATIC MODE INDICATOR FOR APT,ACT, & XXDP
5115	002064	FATAL#: .WORD 0	; FATAL ERROR INDICATOR
5116	002066	SKPERR: .WORD 0	; USED TO SKIP ERROR MESSAGE IN "\$ERRGEN"
5117	002070	NEMCNT: 0	; NON-EXISTANT MEMORY COUNTER (HOLES)
5118	002072	PARCNT: 0	; PARITY ERROR COUNTER
5119	002074	PATERR: 0	; PATTERN ERROR COUNTER
5120	002076	NOPAR: 0	; NO PARITY ERROR MODE INDICATOR
5121	002100	NONEM: 0	; NO NON-EXISTANT MEMORY (HOLES) MODE INDICATOR
5122	002102	BANK: 0	; MEMORY BANK UNDER TEST
5123	002104	BANKINDEX:0	; USED TO INDEX INTO CONFIG TABLE
5124	002106	CPUBIT: 0	; CONTAINS 1 BIT TO IDENTIFY CPU TO CONFIGURATION TABLE
5125	002110	MUT: 0	; MEMORY UNDER TEST FLAG
5126	002112	PATTERN:0	; PATTERN NUMBER UNDER TEST
5127	002114	KPFLAG: .WORD 0	; BANK IS PROTECTED REGION OF ECC
5128	002116	ACFLAG: .WORD 0	; BANK CAN BE ACCESSED BY THIS CPU
5129	002120	*KFLAG: .WORD 0	; IF SET INDICATES MSV11-J OR MF11S-K UNDER TEST***** ?
5130	002122	PFLAG: .WORD 0	; BANK IS IN PROGRAM SPACE
5131	002124	RRFLAG: .WORD 0	; BANK IS WHERE PROGRAM RELOCATION IS REQUIRED TO TEST
5132	002126	RLFLAG: .WORD 0	; PROGRAM IS RELOCATED FLAG
5133	002130	BMFLAG: .WORD 0	; "BANK IS IDENTIFIED AS BAD MEMORY" FLAG
5134	002132	EQFLAG: .WORD 0	; "BANK HAS EQB MEMORY" FLAG
5135	002134	TMFLAG: .WORD 0	; "TYPE OF MEMORY TO TEST" FLAG; 0 = PARITY, 1 = ECC
5136	002136	INTFLAG: .WORD 0	; "BANK IS INTERLEAVED" FLAG***** ? **
5137	002140	INT64K: .WORD 0	; "BANK IS 64K INTERLEAVED" FLAG***** ? **
5138	002142	PMEMFLG: .WORD 0	; "MEMORY UNDER TEST IS A MSV11-J" FLAG
5139	002144	ABORTFLAG: .WORD 0	; "ABORT OCCURED" FLAG

5140	002146	000000	CTKVEC: .WORD	0	; HOLDS OLD KERNAL STACK POINTER IN CASE OF CNTL/K
5141	002150	000000	CSR: .WORD	0	; DATA TO OR FROM CSR
5142	002152	000000	CSRNO: .WORD	0	; CSR ADDRESS NUMBER (4 LSB'S)
5143	002154	000000	SAVCSR: .WORD	0	; LOCATION TO SAVE CSRNO DURING FS COMMAND
5144	002156	000000	OLDCSR: .WORD	0	; OLD CSR NUMBER(USED IN INH PTR TEST)
5145					; THESE LOCATIONS STORE GPR'S DURING SUPERVISOR TESTS
5146	002160	000000	SUPDR0: .WORD	0	
5147	002162	000000	SUPDR1: .WORD	0	
5148	002164	000000	SUPDR2: .WORD	0	
5149	002166	000000	SUPDR3: .WORD	0	
5150	002170	000000	SUPDR4: .WORD	0	
5151	002172	000000	SUPDR5: .WORD	0	
5152	002174	000000	SUPDR6: .WORD	0	
5153	002176	000000	DUMMY: .WORD	0	; DUMMY LOCATION FOR ADDRESS PASSING
5154					; THESE LOCATIONS STORE GPR'S & PSW DURING DETAILED ERROR PRINTOUTS
5155	002200	000000	DETRO: .WORD	0	
5156	002202	000000	DETR1: .WORD	0	
5157	002204	000000	DETR2: .WORD	0	
5158	002206	000000	DETR3: .WORD	0	
5159	002210	000000	DETR4: .WORD	0	
5160	002212	000000	DETR5: .WORD	0	
5161	002214	000000	DETR6: .WORD	0	
5162	002216	000000	DETR7: .WORD	0	
5163	002220	000000	DETR8: .WORD	0	
5164	002222	000000	DETR9: .WORD	0	
5165	002224	000000	DETR10: .WORD	0	
5166					; DETAILED REPORT FLAG
5167	002226	000000	DETR11: .WORD	0	; CSR'S HAVE BEEN TESTED FLAG
5168	002230	000000	DETR12: .WORD	0	; 1 BIT PER EXISTING CSR, EG
5169	002232	000000	DETR13: .WORD	0	; CSR 0 REPRESENTED BY BIT 15, ETC.
5170	002234	000000	DETR14: .WORD	0	; FIRST ADDRESS UNDER CONTROL OF THIS CSR
5171	002236	000000	DETR15: .WORD	0	
5172	002240	000000	DETR16: .WORD	0	
5173	002242	000000	DETR17: .WORD	0	
5174	002246	000000	DETR18: .WORD	0	
5175	002252	000000	DETR19: .WORD	0	
5176	002256	000000	DETR20: .WORD	0	
5177	002262	000000	DETR21: .WORD	0	
5178	002264	000	DETR22: .WORD	0	
5179	002265	000	DETR23: .WORD	0	
5180	002266	000000	DETR24: .WORD	0	
5181	002270	000000	DETR25: .WORD	0	
5182	002272	000000	DETR26: .WORD	0	
5183	002274	000000	DETR27: .WORD	0	
5184	002276	000000	DETR28: .WORD	0	
5185	002300	000000	DETR29: .WORD	0	
5186	002302	000000	DETR30: .WORD	0	
5187	002304	000000	DETR31: .WORD	0	
5188	002306	000000	DETR32: .WORD	0	
5189	002310	000000	DETR33: .WORD	0	
5190	002312	000000	DETR34: .WORD	0	
5191	002314	000000	DETR35: .WORD	0	
5192	002316	000000	DETR36: .WORD	0	
5193	002320	000000	DETR37: .WORD	0	
5194	002322	000000	DETR38: .WORD	0	
5195	002324	000000	DETR39: .WORD	0	
5196	002326	000000	DETR40: .WORD	0	

;; I.L.C. ;REV

5197	002330	000000	CSRINC: .WORD	0	;VALUE TO INCREMENT ADDRESS BY TO REMAIN IN THE SAME CSR
5198	002332	000000	CSRLOOP: .WORD	0	;LOOP CONTROL FOR CSR TESTING
5199	002334	000000	SUCCESS: .WORD	0	;FLAG SET BY SUCCESSFUL TASK OR SUBROUTINE
5200	002336	000000	ZEROS: .WORD	0	;FOR AID IN "MOV" INSTRUCTIONS
5201	002340	000000	TIME: .WORD	0	;SECONDS THAT BATTERIES SHOULD LAST
5202	002342	000000	SKIPMK: .WORD	0	;FLAG TO SKIP MKCONTROL SUBROUTINE
5203	002344	000000	NULLFLAG: .WORD	0	;SET WHEN RUNNING NULL PATTERNS
5204	002346	000000	QVFLAG: 0		;FLAGS QUICK VERIFY PASS UNDER APT, ACT, OR XXDP CHAIN MODE
5205	002350	000000	ACTFLAG: 0		;FLAGS ACT AUTOMATIC MODE PROGRAMMING RULES
5206	002352	000000	APTFLAG: 0		;FLAGS APT AUTOMATIC MODE PROGRAMMING RULES
5207	002354	000000	XXDPCHAIN: 0		;FLAGS XXDP CHAIN MODE PROGRAMMING RULES
5208			;NOTE: THESE TWO BYTES MUST STAY TOGETHER		
5209	002356	000	\$NULL: .BYTE	0	::CONTAINS NULL CHARACTER FOR FILLS
5210	002357	000	\$FILLS: .BYTE	0	::CONTAINS # OF FILL CHARACTERS
5211	002360	000	\$TPFLG: .BYTE	0	::"TERMINAL NOT AVAILABLE" FLAG
5212			.EVEN		
5213	002362	000000	\$ESCAPE: 0		::ESCAPE ON ERROR ADDRESS
5214	002364	000000	EVEN: 0		;USED FOR ALTERNATE DATA PATTERNS
5215	002366	000000	STRIPES: 0		;COUNTS DIAGONAL STRIPES
5216	002370	000000	COUNT: 0		;BACKED UP COPY OF STRIPES
5217	002372	000000	NOTAB: 0		;NO TABLE BEING PRINTED - NOW
5218	002374	000000	Bsize: 0		;SIZE OF 11/45 MOS MEMORY IN K WORDS***** ? *****
5219	002376	000000	Ksize: 0		;SIZE OF MF11S-K MEMORY IN K WORDS***** ? *****
5220	002400	000000	Lsize: 0		;SIZE OF MSV11-L/P MEMORY IN K WORDS
5221	002402	000000	Msize: 0		;SIZE OF MSV11-J MEMORY IN K WORDS
5222	002404	000000	Psize: 0		;SIZE OF Q-BUS PARITY MEMORY IN K WORDS
5223	002406	000000	TOOMANY: 0		;FLAGS WHEN TOO MANY ERRORS HAVE BEEN PRINTED FOR A BANK
5224	002410	000000	READONLY: 0		;FLAG TO PATTERNS TO READ ONLY
5225	002412	000000	TESTADD: 0,0	000000	;THE ADDRESS TO RUN CSR TESTS ON
5226	002416	000000	UNITOP: 0		;HIGHEST ACCESSABLE BANK OF MEMORY THRU Q-BUS MAP
5227	002420	000000	STOPOK: 0		;FLAG TO ALLOW STOPPING WITH SWITCH REGISTER
5228	002422	000000	APTPAR: .WORD	0	;AMOUNT OF PARITY MEMORY ACCORDING TO APT
5229	002424	000000	APTECC: .WORD	0	;AMOUNT OF ECC MEMORY ACCORDING TO APT
5230	002426	000000	NOFSMODE: 0		;FLAG TO DISABLE FIELD SERVICE MODE
5231	002430	000000	NOERROR: 0		; "THIS IS NOT AN ERROR" FLAG
5232	002432	000000	LOADBANK: 0		;BANK LOADERS ARE RELOCATED TO
5233	002434	000000	TEMP: 0		;USED FOR JUNK
5234	002436	000000	QUICK: 0		;QUICK STOP FLAG FOR APT POWER FAIL
5235	002440	000000	NOSCOPE: 0		; "NO SCOPE LOOP ALLOWED" FLAG
5236	002442	000000	FSINFLAG: 0		; "FIELD SERVICE NO INTERNAL INTERLEAVE" FLAG
5237	002444	000000	APTSIZE: 0		;APT SIZING INFO FLAG
5238	002446	000000	FS7FLAG: 0		;TRUE WHEN IN FIELD SERVICE COMMAND 7
5239	002450	000000	CONFERROR: 0		;CONFIGURATION ERROR FLAG
5240	002452	000000	I: 0		;USED FOR GENERAL PURPOSE INDEXING
5241	002454	000000	NO22BIT: 0		;NO 22-BIT MODE FLAG
5242	002456	000000	NOSUPER: 0		;NO SUPERVISOR MODE FLAG
5243	002460	000000	ERRADD: .WORD	0	;HOLDS THE CSR'S ERROR ADDRESS

```

5244 002462 000000 000000 000000 CSRINFO:0.0,0.0,0.0,0.0,0.0,0.0 ;USED TO STORE INFORMATION ABOUT THE 16
      002470 000000 000000 000000
      002476 000000 000000
5245 002502 000000 000000 000000          0.0,0.0,0.0,0.0,0.0,0.0 ;POSSIBLE CSR'S
      002510 000000 000000 000000
      002516 000000 000000
5246 002522 000000          LINK1: 0 ;USED TO HOLD LINKS TO PATTERNS WHICH
5247 002524 000000          LINK2: 0 ;CAN EXECUTE IN THE PAR/PDR'S OR NOT
5248 002526 000000          CSRHOLD:0 ;USED TO STORE CSR VALUES FOR CSR TESTS
5249 002530 000000          KFLAG: 0 ;USED TO FLAG MF11S K MEMORY TO TESTS.....
5250 002532 000000 000000          PGMCSR: .WORD 0.0 ;POINTS TO PROGRAM CSR
5251 002536 000000          INHECC: .WORD 0 ;FLAGS INHIBIT ECC TESTS ON RELOCATION
5252 002540 000000          INHBANK: .WORD 0
5253 002542 000000          FULLREL: .WORD 0
5254 002544          $CMTGE: ;*END OF COMMON TAGS
    
```

				.SBTTL	VARIABLES	INITIALIZED TO NON ZERO
5257						
5258	002544	000401	000000	CACHKN:	401,0	;CACHE CONSTANT (MOVED TO CONTRL TO TURN ON CACHE)
5259	002550	001014		CACHKF:	1014	;CACHE CONSTANT (MOVED TO CONTRL TO TURN OFF CACHE)
5260	002552	040000		TESTMODE:	40000	;USED TO SELECT THE PROPER TEST MODE FOR A PATTERN RUN
5261	002554	000012		ERRMAX:	10.	;MAX # OF ERRORS PER BANK WITH SW11
5262	002556	000177		LASTBANK:	177	;HIGHEST BANK OF MEMORY
5263	002560	177000		LASTBLOCK:	177000	;HIGHEST BANK OF MEMORY*1 (IN PAR FORMAT)
5264	002562	160000		ENDADD:	160000	;ENDING ADDRESS
5265	002564	000000		ENDFLG:	0	;END FLAG
5266	002566	000000		SWRFLG:	0	;USED TO BUMP STACK ON FIRST CALL TO GTSWR
5267	002570	000000		PASCNT:	0	;PASS COUNTER
5268	002572	000031		SOBK:	25.	;SOB CONSTANT
5269	002574	002000		KSTACK:	STACK	;STACK BEGINNING
5270	002576	000001		LOADHOME:	1	;HOME BANK OF LOADERS
5271	002600	177777		WORST:	177777	;SET IF TESTING BANKS IN WORST FIRST MODE(1ST PASS)
5272	002602	176543		SEEDHI:	176543	;WORKING SEED HI (USED FOR RANDOM NUMBER GENERATOR)
5273	002604	123456		SEEDLO:	123456	;WORKING SEED LO (USED FOR RANDOM NUMBER GENERATOR)
5274	002606	176543		MSEEDH:	176543	;MASTER SEED HI (USED FOR RANDOM NUMBER GENERATOR)
5275	002610	123456		MSEEDL:	123456	;MASTER SEED LO (USED FOR RANDOM NUMBER GENERATOR)
5276	002612	177777		HEADER:	177777	;USED TO PRINT HEADINGS ONLY ONCE
5277	002614	177777		ONES:	177777	;FOR A IN "MOV" INSTRUCTIONS
5278	002616	000003		FLIPLOC:	3	;COUNTER FOR FLIPING DATA ON WORST CASE NOISE TEST
5279	002620	052525		SOFTPAT:	52525	;PATTERN FOR SOFT ERROR BACKGROUND TESTS
5280	002622	000000		\$LPADR:	.WORD 0	::CONTAINS SCOPE LOOP ADDRESS
5281	002624	000000		\$LPERR:	.WORD 0	::CONTAINS SCOPE RETURN FOR ERRORS
5282	002626	000000		RESTART:	0	;RESTART (START ADD 202) FLAG
5283	002630	000000		\$ERTTL:	.WORD 0	::CONTAINS TOTAL ERRORS
5284						
5285						;***** NOTE THESE TWO LOCATIONS MUST STAY TOGETHER *****
5286	002632	000377		BAKPAT:	.WORD 377	;BACKGROUND PATTERN *
5287	002634	177400		SWAPAT:	.WORD 177400	;SWAPPED BAKPAT *
5288						;*****
5289						
5290	002636	177570		SWR:	.WORD	DSWR
5291	002640	177570		DISPLAY:	.WORD	DDISP
5292	002642	177560		\$TKS:	177560	::TTY KBD STATUS
5293	002644	177562		\$TKB:	177562	::TTY KBD BUFFER
5294	002646	177564		\$TPS:	177564	::TTY PRINTER STATUS REG. ADDRESS
5295	002650	177566		\$TPB:	177566	::TTY PRINTER BUFFER REG. ADDRESS
5296	002652	012		\$FILLC:	.BYTE 12	::INSERT FILL CHARS. AFTER A "LINE FEED"
5297	002653	207	377	\$BELL:	.ASCIZ <207><377><377>	::CODE FOR BELL
	002656	000				
5298	002657	077		\$QUES:	.ASCII /?/	::QUESTION MARK
5299	002660	015		\$CRLF:	.ASCII <15>	::CARRIAGE RETURN
5300	002661	012	000	\$LF:	.ASCIZ <12>	::LINE FEED
5301					.EVEN	

5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326 002664 000201
5329 003670

```
.SBTTL CONFIGURATION TABLE
: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 SUCESSFULLY ACCESSED
:            BIT 2-4 RESERVED
:            BIT 5 SKIP ECC LOGIC TESTS FLAG (1=SKIP)
:            BIT 6 PROTECTED REGION OF ECC MEMORY
:            BIT 7 PROTECTED (PROGRAM SPACE)
:            BIT 8-11 CSR CODE
:            BIT 12-15 RESERVED
:      HIGH: BIT 0-7 NUMBER OF ERRORS
:            BIT 8-10 MEMORY TYPE
:            BIT 11 RESERVED
:            BIT 12 RESERVED
:            BIT 13 "BACKGROUND PATTERN VALID" FLAG
:            BIT 14 BANK SELECTED FOR TEST BY FIELD SERVICE MODE
:            BIT 15 LOADERS HOME BANK
:CONFIG: .REPT 201
CONFIEND:
```

5331
 5332 003670

```
.SBTTL ***** MAIN *****
START: SUBTST <<INITIALIZE VARIABLES TO ZERO>>
;*****
;SUBTEST INITIALIZE VARIABLES TO ZERO
;*****
```

5333
 5334 003670

```
SUBTST <<SAVEMT>>
;*****
;SUBTEST SAVEMT
;*****
;SAVEMT SAVES THE EMULATOR AND PRIORITY LOCATION
;UNDER THE NAMES OF SAV30 AND SAV32.
```

5335
 5336
 5337
 5347
 5348 003670

```
SAVEMT
;: LCP/ORION ROUTINE TO SAVE EMULATOR AND PRIORITY
```

```
003670 005737 003756 EMTSAV: TST SAV30 ;: FIRST TIME THROUGH ?
003674 001034 BNE VMKOR ;: BRANCH IF BEEN HERE ALREADY
003676 C32737 000040 000052 BIT #BIT5,#52 ;: ARE WE IN UFD MODE ?
003704 001430 BEQ VMKOR ;: LEAVE IF NOT
003706 012737 177777 003762 MOV #-1,UFDPLG ;: SET UFD FLAG
003714 032737 000100 000052 BIT #BIT6,#52 ;: ARE WE IN QUIET MODE ?
003722 001403 BEQ 1# ;: BR IF NOT
003724 012737 177777 003764 MOV #-1,UQUIET ;: SET QUIET MODE
003732 104042 1# EMT 42 ;: GET ADDRESS OF XXDP DCA TABLE
003734 005060 000042 CLR 42(R0) ;: CLR XXDP+ "DRSERR"
003740 013737 000030 003756 MOV 30,SAV30 ;: SAVE EMULATOR ADDRESS
003746 013737 000032 003760 MOV 32,SAV32 ;: SAVE EMULATOR PRIORITY LEVEL
003754 000404 BR VMKOR ;: GET AROUND TAG AREA
003756 000000 SAV30: .WORD 0 ;: PUT EMULATOR INFO HERE
003760 000000 SAV32: .WORD 0 ;: PUT PRIORITY LOCATION HERE
003762 000000 UFDPLG: .WORD 0 ;: USER FRIENDLY MODE FLAG
003764 000000 UQUIET: .WORD 0 ;: UFD QUIET MODE FLAG
003766 VMKOR:
```

5349 003766 105737 056736
 5350 003772 001001
 5351 003774 000005
 5352 003776

```
TSTB #ENV
BNE NORES
RESET
NORES: CLEAR MONFLG ;: CLEAR RETURN TO MONITOR FLAG
```

003776 005037 002276
 5353 004002 010637 002274
 5354 004006 013706 002574
 5355 004012 012700 002000
 5356 004016 005020
 5357 004020 022700 002544
 5358 004024 001374
 5359 004026 012737 000177 002556
 5360 004034

```
MOV SP,SAVMON ;: SAVE XXDP MONITOR RESTART ADDRESS
MOV KSTACK,SP ;: SETUP THE STACK POINTER
MOV #CMTAG,R0 ;: FIRST LOCATION TO BE CLEARED
1#: CLR (R0)+ ;: CLEAR MEMORY LOCATION
CMP #CMTGE,R0 ;: DONE?
BNE 1# ;: LOOP BACK IF NO
MOV #177,LASTBANK ;: RESTORE LASTBANK (THIS MUST BE DONE PRIOR TO SYSTEM SIZING)
SUBTST <<CLEAR NON-PROGRAM SPACE>>
```



```
*****  
; *SUBTEST      CLEAR NON PROGRAM SPACE  
; *****  
5361           ; THIS ATTEMPS TO GET RID OF ANY PARITY ERRORS BY WRITING INTO  
5362           ; EVERY LOCATION THAT IS NOT LOADED INTO BY THE PROGRAM OR ALLOCATED  
5363           ; TO THE XXDP LOADERS  
5364 004034 012737 000001 002076      MOV      #1,NOPAR          ; PARITY ACTION = COUNT & IGNORE  
5365 004042 005000                    CLR      R0  
5366 004044 000241                    2$:    CLC  
5367 004046 005520                    ADC      (R0),  
5368 004050 020027 160000             CMP      R0,#160000  
5369 004054 103773                    BLO     2$  
5370 004056 005037 002076             CLR      NOPAR          ; RESTORE DEFAULT PARITY ACTION  
5371
```

5373 004062

```

SUBSTST <<TYPE OF SYSTEM SIZER>>
;*****
;SUBTEST TYPE OF SYSTEM SIZER
;*****

```

```

5374 004062 000401
5375 004064 000000
5376 004066 012737 004162 000004
5377 004074 012737 004106 000010
5378 004102 000007
5379
5380
5381
5382
5383 004104 000413
5384 004106 012737 047030 000034 1#:
5385 004114 104401 071545
5386 004120 010046
5387 004122 005000
5388 004124 004737 052220
5389 004130 012600
5390 004132 000000
5391 004134 012737 000012 000010 2#:
5392 004142 110037 004064
5393 004146 004146 012737 004162 000004
5394 004154 005737 177746
5395 004160 000447
5396 004162 005037 002544 3#:
5397 004166 012737 002336 060412
5398 004174 022737 000005 004064
5399 004202 001436
5400 004204 004204 012737 004234 000004
5401 004212 005037 172516
5402 004216 052737 000020 172516
5403 004224 032737 000020 172516
5404 004232 001005
5405 004234 005237 002454 4#:
5406 004240 012737 000007 002556
5407 004246 012737 140000 002552 5#:
5408 004254 005237 002456
5409 004260 005037 060262
5410 004264 005037 060422
5411 004270 005037 060264
5412 004274 005037 060424
5413 004300 022737 000005 004064 6#:
5414 004306 001003
5415 004310 052737 000020 172516
5416 004316 004316 012737 004342 000004 22#:

```

```

BR SYSSIZ ;SKIP OVER VARIABLE LOCATION
PROTYP: .WORD 0
SYSSIZ: SET4 #3#
MOV #3#.4
.DSABL CRF
MOV #1#.10 ;TRAPS TO 10 = BAD PROCESSOR TYPE
MFPT ;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)
BR 2#
MOV #1#TYPE,34 ;LOAD TRAP VECTOR
TYPE MSG130 ;TELL THEM BAD PROCESSOR TYPE
TYPEIT ,MSG130
.DSABL CRF
MOV RO,-(SP) ;K SAVE R0
CLR RO ;K CLR +C AND +Z POSSIBILITIES
JSR PC,ABORT ;K SEE IF THIS IS A UFD ABORT SITUATION
MOV (SP)+,RO ;K IF NOT RESTORE R0 AND HALT
HALT ;NO NEED TO GO ON
MOV #12,10 ;RESTORE TRAPS TO 10
MOVB RO,PROTYP ;MOV THE CODE TO PROTYP
SET4 #3#
MOV #3#.4
.DSABL CRF
TST CONTRL ;SEE IF CACHE REGISTER RESPONDS
BR 6# ;BRANCH IF CACHE AVAILABLE
CLR CACHKN ;NO CACHE ON SYSTEM
MOV #ZEROS,DT14 ;DO NOT PRINT CONTRL ERROR MESSAGES
CMP #5,PROTYP ;IS THIS A 11/83/84
BEQ 6# ;YES - BRANCH
SET4 #4#
MOV #4#.4
.DSABL CRF
CLR MMR3 ;SEE IF THERE IS 22-BIT ADDRESSING
BIS #BIT4,MMR3
BIT #BIT4,MMR3
BNE 5#
INC NO22BIT ;BRANCH IF 22-BIT RELOCATION
MOV #7,LASTBANK ;SET FOR NO 22 BIT ADDRESSING
MOV #140000,TESTMODE ;HIGHEST BANK OF MEMORY
INC NOSUPER ;MAKE TESTMODE USER
CLR DT5+10 ;CLEAR SOME ERROR DATA TAGS
CLR DT14+10
CLR DT5+12
CLR DT14+12
CMP #5,PROTYP ;CPU TYPE = 11/83/84
BNE 22# ;NO - BRANCH
BIS #BIT4,MMR3 ;SET UP 22 BIT ADDRESSING
SET4 #7# ;TRAPS GO TO 4#
MOV #7#.4
.DSABL CRF

```

5417	004324	005037	052364		CLR	CPERRF		;CLEAR THE FLAG
5418	004330	005737	177766		TST	#177766		;IS THERE A CPU ERROR REGISTER?
5419	004334	012737	177777	052364	MOV	#-1,CPERRF		;YES-TRAPPED
5420	004342				RES4			;ENABLE TRAPS TO 4
	004342	012737	034002	000004	MOV	#TIMEOUT,4		
	004350	022737	000005	004064	CMP	#5,PROTYP		;IS THIS AN 11/83/84 ?
	004356	001002			BNE	101#		;BRANCH IF NOT
	004360	005037	177766		CLR	CPUERR		;CLEAR OUT THE CPU ERROR REGISTER BITS
	004364			101#:				;THAT A EXPECTED TRAP COULD HAVE SET
					.DSABL	CRF		

5421

5424 004364

```

SUBTST <<INITIALIZE VARIABLES TO NON ZERO>>
;*****
;#SUBTEST INITIALIZE VARIABLES TO NON ZERO
;*****

```

5425 004364 012737 177777 002600

```

SET WORST
MOV #-1,WORST

```

5426 004372 012737 000003 002616

```

MOV #3,FLIPLOC

```

5427 004400 012737 177777 002612

```

SET HEADER
MOV #-1,HEADER

```

5428 004406 012737 176543 002606

```

MOV #176543,MSEEDH

```

5429 004414 012737 123456 002610

```

MOV #123456,MSEEDL

```

5430 004422 013737 002606 002602

```

MOV MSEEDH,SEEDHI ;PRIME THE RANDOM NUMBER GENERATOR

```

5431 004430 013737 002610 ^2604

```

MOV MSEEDL,SEEDLO ;BOTH HIGH AND LOW WORDS

```

5432 004436 012737 000377 002632

```

MOV #377,BAKPAT

```

5433 004444 012737 177400 002634

```

MOV #177400,SWAPAT

```

5434 004452

```

SUBTST <<INITIALIZE VECTORS>>
;*****
;#SUBTEST INITIALIZE VECTORS
;*****

```

5435 004452 012737 051054 000020

```

MOV #SCOPE,IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE

```

5436 004460 012737 000340 000022

```

MOV #340,IOTVEC+2 ;:LEVEL 7

```

5437 004466 012737 051410 000030

```

MOV #ERROR,EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE

```

5438 004474 012737 000340 000032

```

MOV #340,EMTVEC+2 ;:LEVEL 7

```

5439 004502 012737 057034 000034

```

MOV #TRAP,TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS

```

5440 004510 012737 000340 000036

```

MOV #340,TRAPVEC+2 ;:LEVEL 7

```

5441 004516 012737 045260 000024

```

MOV #PWRDN,PWRVEC ;:POWER FAILURE VECTOR

```

5442 004524 012737 000340 000026

```

MOV #340,PWRVEC+2 ;:LEVEL 7

```

5443 004532 012737 033632 000114

```

MOV #PARITY,PARVEC ;:GET READY FOR PARITY ERRORS

```

5444 004540 012737 000340 000116

```

MOV #340,PARVEC+2

```

5445 004546 012737 034026 000010

```

MOV #PDP1105,RESVEC ;:RESERVED INSTRUCTION TRAP

```

5446 004554 012737 000340 000012

```

MOV #340,RESVEC+2

```

5447 004562 012737 034002 000004

```

MOV #TIMEOUT,ERRVEC ;:SETUP TIMEOUT ERRORS

```

5448 004570 012737 000340 000006

```

MOV #340,ERRVEC+2 ;:SET PRIORITY OF ERROR TRAPS

```

5449 004576 012737 034014 000250

```

MOV #MMTRAP,MMVEC ;:VECTOR FOR MEMORY MANAGEMENT

```

5450 004604 012737 000340 000252

```

MOV #340,MMVEC+2

```

5451 004612 104423

```

CACHON ;:TURN CACHE ON

```

5454 004614

5455
5456
5457
5458
5459 004614 012700 057004
5460 004620 012001
5461 004622 012703 016702
5462 004626 012702 000020
5463 004632 004737 004732
5464 004636 012001
5465 004640 012702 000010
5466 004644 004737 004732
5467 004650 012001
5468 004652 012703 017132
5469 004656 012702 000020
5470 004662 004737 004732
5471 004666 012001
5472 004670 012702 000010
5473 004674 004737 004732
5474 004700 012001
5475 004702 012703 017316
5476 004706 012702 000020
5477 004712 004737 004732
5478 004716 012001
5479 004720 012702 000010
5480 004724 004737 004732
5481 004730 000417
5482
5483 004732

```

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.
MOV #DDWO,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

```

5484 004732
004732 012737 000001 002452
004740
5485 004740 006001
5486 004742
004742 103402
5487 004744 012713 024106
5488 004750
004750
5489 004750 062703 000002
5490 004754
004754 005237 002452
004760 023702 002452
004764 003765
004766
5491 004766 000207

```

PATPLUG:SUBTST <<SUBR PLUG IN NULL PATTERNS>>
;*****
;SUBTEST SUBR PLUG IN NULL PATTERNS
;*****
FOR I := #1 TO R2
MOV #1,I
B0:;;;;;
ROR R1
ON.NOERROR ;IF CARRY CLEAR
BCS LO
MOV #MT0999,(R3)
END ;OF ON.ERROR
L0:;;;;;
ADD #2,R3
END ;OF FOR
INC I
CMP I,R2
BLE B0
E0:;;;;;
RETURN

```

5494 004770

SUBAAA: SUBTST <<CLEAR THE CONFIGURATION TABLE>>
;*****
;SUBTEST CLEAR THE CONFIGURATION TABLE
;*****

5495
5496
5497
5498

;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

004770 005737 002626
004774 001006
5499 004776 012700 002664
5500 005002 005020
5501 005004 022700 003670
5502 005010 001374
5503 005012
005012

10: MOV #CONFIG,RO
CLR (RO);
CMP #CONFIEND,RO
BNE 10
END ;OF IF RESTART

L1:*****

5504
5505 005012 012737 000002 002106
5506 005020

.DSABL LSB
MOV #BIT1,CPUBIT ;SET ID BIT
SUBTST <<SIZE FOR A HARDWARE SWITCH REGISTER>>

;*****
;SUBTEST SIZE FOR A HARDWARE SWITCH REGISTER
;*****

5507
5508
5509
5510

;;IF NOT FOUND OR IT IS
;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.

005020
005020 012737 005054 000004

.ENABLE LSB
SET4 #30 ;TRAPS TO 4 GOTO 30
MOV #30,4
.DSABL CRF

5511 005026 012737 177570 002636
5512 005034 012737 177570 002640
5513 005042

MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWITCH REGISTER
MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
IF #-1 EQ BSWR ;IF NO TRAP FROM REFERENCE TO BSWR AND BSWR = #-1
CMP #-1,BSWR
BNE L2

005042 022777 177777 175566
005050 001023
5514 005052 000403
5515 005054 012716 005062
5516 005060 000002
5517 005062

BR 20 ;;BRANCH IF NO TIMEOUT
30: MOV #20,(SP) ;;SET UP FOR TRAP RETURN
RTI
20: RES4 ;RESET TRAPS TO 4 TO DEFAULT

005062 012737 034002 000004
005070 022737 000005 004064
005070 001002
005100 005037 177766
005104

MOV #TIMEOUT,4
CMP #5,PROTYP ;IS THIS AN 11/83/84 ?
BNE 1010 ;BRANCH IF NOT
CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
1010: ;THAT A EXPECTED TRAP COULD HAVE SET

5518 005104 012737 000176 002636
5519 005112 012737 000174 002640
5520 005120
005120
5521

.DSABL CRF
MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
MOV #DISPREG,DISPLAY
END ;OF IF #-1

L2:*****

.DSABL LSB

```

5524 005120 SUBAAB: SUBTST <<SETUP ACT, APT, & XXDP>>
;*****
;#SUBTEST SETUP ACT, APT, & XXDP
;*****
5525 ;THIS SETS UP A BUNCH OF FLAGS TO TELL THE PROGRAM EVERYTHING
5526 ;IT CARES TO KNOW ABOUT APT, ACT, & XXDP.
5527 005120 005037 056724 CLR $PASS ;CLEAR PASS COUNT
5528 005124 132737 000040 056737 IFB #BIT5 SET.IN $ENVM BITB #BIT5,$ENVM
005124 001403 BEQ L3
5529 005134 SET $TPFLG ;INDICATE NO TERMINAL MOV #-1,$TPFLG
005134 012737 177777 002360 END ;OF IFB #BIT5 L3:~~~~~
5530 005142 IFB #BIT7 SET.IN $ENVM BITB #BIT7,$ENVM
005142 132737 000200 056737 BEQ L4
005150 001403 SET APTSIZE MOV #-1,APTSIZE
5532 005152 012737 177777 002444 END ;OF IFB #BIT7 L4:~~~~~
005160 IFB $ENV EQ #1 CMPB $ENV,#1
5534 005160 123727 056736 000001 BNE L5
005166 001023 SET APTFLAG,QVFLAG,$AUTO,QUICK MOV #-1,APTFLAG
5535 005170 012737 177777 002352 MOV #-1,QVFLAG
005176 012737 177777 002346 MOV #1,$AUTO
005204 012737 177777 002062 MOV #1,QUICK
005212 012737 177777 002436
5536 005220 012737 040626 000024 MOV #APTDOWN,PWRVEC
5537 005226 012737 056740 002636 MOV #SWREG,SWR ;USE APT SWR
5538 005234 ELSE BR L6
005234 000430 L5:~~~~~
005236 IF 42 NE #STACK AND 42 NE #0 CMP 42,#STACK
5539 005236 023727 000042 002000 BEQ L7
005244 001424 TST 42
005246 005737 000042 BEQ L7
005252 001421
5540 005254 SET QVFLAG,$AUTO MOV #-1,QVFLAG
005254 012737 177777 002346 MOV #1,$AUTO
005262 012737 177777 002062
5541 005270 IF 42 EQ #ENDAD CMP 42,#ENDAD
005270 023727 000042 013730 BNE L10
005276 001004
5542 005300 SET ACTFLAG MOV #-1,ACTFLAG
005300 012737 177777 002350 ELSE BR L11
5543 005306 000403 L10:~~~~~
005310 SET XXDPCHAIN MOV #-1,XXDPCHAIN
5544 005310 012737 177777 002354 END ;OF IF 42
5545 005316 END ;OF IF 42 L11:~~~~~
005316
5546 005316 END ;OF IFB $ENV L7:~~~~~
005316
5547 005316

```

```
005316                                     L6:iiiiii
5549 005316                               SUBST <<PROTECT PROGRAM & LOADERS>>
;.....
;SUBTEST   PROTECT PROGRAM & LOADERS
;.....
5550 005316 052737 000200 002664       BIS   #BIT7.CONFIG      ;PROTECT PROGRAM SPACE (BANK 0)
5551 005324 052737 000200 002670       BIS   #BIT7.CONFIG.4   ;PROTECT LOADER SPACE (BANK 1)
5552 005332                                IF #ENDAD NE 42        ;NOT ACT-11?
                                CMP #ENDAD,42
                                BEQ L12
5553 005342                                IF NO22BIT NE #0      ;
                                TST NO22BI
                                BEQ L13
5554 005350                                SET   MONFLG          ;RETURN TO XXDP MONITOR
                                MOV  #-1,MONFLG
5555 005356 012737 177777 002276       ERROR +64             ;ILLEGAL PROCESSOR
5556 005360 000402                                ELSE
                                BR L14
                                L13:iiiiii
5557 005362 104401 071472       TYPE   MSG000         ;TYPE PROGRAM TITLE
                                TYPEIT ,MSG000
                                .DSABL CRF
                                END
5558 005366                                ;
                                L14:iiiiii
5559 005366                                END ;OF IF #ENDAD
                                L12:iiiiii
5560
```


5561 005366

```

SUBSTST <<CHECK FOR CACHE AND MEMORY MANAGEMENT>>
;*****
;SUBTEST CHECK FOR CACHE AND MEMORY MANAGEMENT
;*****
; * THIS FIGURES OUT IF THERE IS A CACHE AND MEMORY MANAGEMENT
; * ON THE SYSTEM, AND WHETHER IT IS ENABLED OR DISABLED.
SET4 #3#
MOV #3# ,4
.DSABL CRF
TST CONTRL ;IS THERE A CONTROL REGISTER?
SET4 #1#
MOV #1# ,4
.DSABL CRF
TST MMR3 ;IS THERE A MMR3 REGISTER?
MOV 4,-(SP) ;;V SAVE OLD TIME OUT
MOV #14# ,4 ;;V PASS ON KTJ11
BIT #BIT9,@#177750 ;;V TEST FOR 1184
BEQ 14# ;;V ELSE REPORT 1183
TYPE M1184 ;;V REPORT 1184
TYPEIT ,M1184
.DSABL CRF
BIS #BIT8,@#177730 ;;V SET ABILITY TO CLEAR
BIC #77,@#177734 ;;V UNIBUS MEMORY ACCESS
TYPE NOUBMT ;;V TO UNIBUS
TYPEIT ,NOUBMT
.DSABL CRF
MOV (SP)+ ,4 ;;V AND REPORT NO UNIBUS
BR 4# ;;V MEMORY ACCESS AND CONTINUE
MOV (SP)+ ,4 ;;V ALSO RESTORE OLD TIME OUT
TYPE MSG117 ; 11/83
TYPEIT ,MSG117
  
```

5562
 5563
 5564 005366 012737 005566 000004
 5565 005374 005737 177746
 5566 005400 012737 005500 000004
 5567 005406 005737 172516
 5568 005412 013746 000004
 5569 005416 012737 005466 000004
 5570 005424 032737 001000 177750
 5571 005432 001415
 5572 005434 104401 071150
 5573 005440 052737 000400 177730 12#:
 5574 005446 042737 000077 177734
 5575 005454 104401 071620
 5576 005460 012637 000004
 5577 005464 000411
 5578 005466 012637 000004 14#:
 5579 005472 104401 071162

```

5580 005476 000404
5581 005500 18:
5582 005506 012737 177777 002276
5583 005510 052737 000014 177746 48:
5584 005516 042737 000014 177746
5585 005524 032737 000004 177746
5586 005532 001004
5587 005534 032737 000010 177746
5588 005542 001413
5589 005544 78:
005544 104401 071224
5590 005550 104424
5591 005552 013737 002544 002546
5592 005560 005037 002544
5593 005564 000404
5594 005566 38:
005566 104401 071174
5595 005572 68:
005572 104401 071203
    .DSABL CRF
    BR 48
    SET MONFLG ;PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC
    ;MOV # 1,MONFLG
    ERROR +64 ;NO MEMORY MANAGEMENT
    BIS #BIT2:BIT3,CONTRL ;SET CACHE DISABLE BITS
    BIC #BIT2:BIT3,CONTRL ;CLEAR CACHE DISABLE BITS
    BIT #BIT2,CONTRL ;IS THE BIT SET?
    BNE 78 ;BRANCH IF THE BIT IS SET
    BIT #BIT3,CONTRL ;IS THE BIT SET?
    BEQ 68 ;BRANCH IF THE BIT IS SET
    TYPE MSG121 ;CACHE BYPASSED
    TYPEIT ,MSG121
    .DSABL CRF
    CACHOFF
    MOV CACHKN,CACHKN+2 ;SAVE INFO ABOUT CACHE
    CLR CACHKN ;CACHE CANNOT BE USED - IT'S BYPASSED
    BR 88
    TYPE MSG119 ;
    TYPEIT ,MSG119 ; NO
    .DSABL CRF
    TYPE MSG120 ;CACHE AVAILABLE
    TYPEIT ,MSG120
    .DSABL CRF
    
```

```

5597 005576          SUBTST <<SETUP USER & SUPERVISOR STACK>>
;*****
;SUBTEST          SETUP USER & SUPERVISOR STACK
;*****
5598 005576 104421          DEENERGIZE          ;TURN OFF MEMORY MANAGEMENT
5599 005600 005737 002456          TST          NOSUPER          ;IS THERE A SUPERVISOR MODE?
5600 005604 001011          BNE          S#          ;NO-SKIP SUPERVISOR SETUP.
5601
5602          ;SET PREVIOUS MODE TO SUPERVISOR
5603 005606 042737 030000 177776          BIC          #BIT13!BIT12,PSW
5604 005614 052737 010000 177776          BIS          #BIT12,PSW
5605
5606 005622          PUSH          #SUPSTK
5607 005622 012746 000740          MTPI          SSP
5608
5609          ;SET PREVIOUS MODE TO USER
5610 005630 052737 030000 177776 S#:          BIS          #BIT13!BIT12,PSW
5611
5612 005636          PUSH          #USESTK
5613 005642 012746 000700          MTPI          USP
5614
5615 005644          SUBTST <<GET SOFTWARE SWITCH REGISTER IF NECESSARY>>
;*****
;SUBTEST          GET SOFTWARE SWITCH REGISTER IF NECESSARY
;*****
5616 005644          IF $AUTO IS FALSE          ;IF NOT(APT OR ACT)
5617 005652          IF SWR EQ #SWREG          ;IF SOFTWARE SWITCH REG SELECTED
5618 005662          SET SWRFLG          ;;SET FLG TO BUMP STACK
5619 005670          GTSWR          ;;GET SOFT-SWR SETTINGS
5620 005672 005037 002566          CLR SWRFLG          ;;CLEAR IT FOR REST OF PROGRAM
5621 005676          END ;OF IF SWR
5622 005676          END ;OF IF $AUTO
5623
5624 005676          SUBTST <<GET MEMORY MANAGEMENT READY>>
;*****
;SUBTEST          GET MEMORY MANAGEMENT READY
;*****
5625 005676 104422          KMAP          ;MAP KERNEL SPACE 1 TO 1
5626 005700          MAP          ;MAP SUPERVISOR SPACE (TEST AREA) 1 TO 1
5627 005712 012603          MOV          #200,R3          MOV R3,-(SP)
5628 005714 104420          CALL          MAPPER
5629 005714 004737 035604          .DSABL          CRF
5630
5631          ENERGIIZE          ;TURN ON MEMORY MANAGEMENT
    
```

5630 005716

```

005716 000004
5631
5632
5633
5634
5635
5636
5637
5638
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649 005720 005005
5650 005722 005000
5651 005724 012703 172100
5652 005730 012737 000001 002076
5653 005736
      005736 012737 006112 000004
5654 005744
      005744
5655 005744 005713
5656 005746 052705 000001
5657 005752 005004
5658 005754 042760 000006 002462
5659 005762 052760 000030 002462
5660 005770 005013
5661 005772
      005772 012713 020000
5662 005776
      005776 032713 020000
      006002 001403
5663 006004 052760 000001 002462
5664 006012
      006012
5665 006012 005013
5666 006014 004737 006306
5667 006020
      006020 026027 002462 000030
      006026 100004
5668 006030 016037 002462 002052
5669 006036 104021
5670 006040
      006040
5671 006040 062700 000002
5672 006044 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???
;*****
IF #BIT13 SET.IN (R3);IS BIT 13 SET MOV #BIT13,(R3)
;*****
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?
;*****
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

```

5673	006050	006305		ASL R5		;SHIFT CSR MAP	
5674	006052			ON.ERROR		;IS THERE A CSR 0	
	006052	103001					BCC L21
5675	006054	005204		INC R4		;YES-SET CSR PRESENT FLAG	
5676	006056			END			
	006056						L21:;;;;;;;;
5677	006056			UNTIL R0 EQ #40		;UNTIL ALL CSR'S ARE DONE	
	006056	020027	000040				CMP R0,#40
	006062	001330					BNE B1
	006064						E1:;;;;;;;;
5678	006064	006005		ROR R5		;RESYNC R5	
5679	006066	005704		TST R4		;WAS THERE A CSR 0?	
5680	006070			RNE 22#		;BRANCH IF NOT EQUAL	
	006070	001402					BEQ 22#
5681	006072	052705	100000	BIS #BIT15,R5		;YES SET IT IN CSR TABLE	
5682	006076			LET TOTCSRS := R5		;STORE CSR MAP IN TOTCSRS	
	006076	010537	002224				MOV R5,TOTCSRS
5683	006102	004737	006122	CALL CSRMAP		;PRINT CSR MAP	
5684	006106			JUMPTO CTEST			
	006106	C00137	007002				JMP CTEST
5685							
5686	006112	062706	000004	CSRBMP: ADD #4,SP		;FIX STACK POINTER FOR NON-EXISTANT CSR TRAP	
5687	006116	000137	006040	JMP NXTCSR		;GO ON TO NEXT CSR	
5688							

```

5690 006122 CSRMAP: SUBTST <<PRINT CSR REGISTER MAP>>
;*****
;SUBTEST PRINT CSR REGISTER MAP
;*****
5691 006122 005000 CLR R0 ;CLEAR CSR INFO POINTER
5692 006124 104401 071440 TYPE MSG008 ;PRINT TITLE
006124 .DSABL CRF
5693 006130 104401 066134 TYPE MSG016 ;PRINT CSR NUMBERS
006130 .DSABL CRF
5694 006134 005001 CLR R1 ;
5695 006136 REPEAT ;
006136 ; B2:;;;;;
5696 006136 010102 MOV R1,R2 ;
5697 006140 022702 000011 CMP #9.,R2 ;
5698 006144 100002 BPL 1# ;JUMP AROUND NEXT INSTRUCTION
5699 006146 062702 000007 ADD #7,R2 ;
5700 006152 062702 000060 1#: ADD #60,R2 ;MAKE IT ASCII
5701 006156 110237 066132 MOVB R2,MSG015 ;
5702 006162 104401 066132 TYPE MSG015 ;
006162 .DSABL CRF
5703 006166 104401 066130 TYPE MSG014 ;TYPE SINGLE SPACE
006166 .DSABL CRF
5704 006172 005201 INC R1 ;
5705 006174 020127 000020 UNTIL R1 EQ #16. ;
006174 001356 CMP R1,#16.
006200 001356 BNE B2
006202 E2:;;;;;
5706 006202 104401 065614 TYPE MSG009 ;TYPE MEMTYPE
006202 .DSABL CRF
5707 006206 REPEAT ;
006206 ; B3:;;;;;
5708 006206 IF CSRINFO(R0) NE #0 ;IS CSR NONEXISTANT????
006206 005760 002462 TST CSRINFO(R0)
006212 001414 BEQ L24
5709 006214 IF #BIT0 SET.IN CSRINFO(R0)
006214 032760 000001 002462 BIT #BIT0,CSRINFO(R0)
006222 001404 BEQ L25
5710 006224 112737 000105 066132 MOVB #'E,MSG015 ;IT IS A MSV11-J
5711 006232 000403 ELSE BR L26
006234 L25:;;;;;
5712 006234 112737 000120 066132 MOVB #'P,MSG015 ;IT IS A MSV11-L/P
5713 006242 END L26:;;;;;
006242 ELSE L27:;;;;;
5714 006242 000403 BR L27
006244 L24:;;;;;
5715 006244 112737 000040 066132 MOVB #' ,MSG015
5716 006252 END L27:;;;;;
006252 ;TYPE MEMORY TYPE
5717 006252 104401 066132 TYPE MSG015
006252 TYPEIT ,MSG015

```

5718	006256			.DSABL CRF		
	006256	104401	066130	TYPE MSG014	:	TYPE SPACE
				TYPEIT ,MSG014		
				.DSABL CRF		
5719	006262	000240		NOP	:	
5720	006264	062700	000002	ADD #2,R0	:	POINT TO NEXT ENTRY
5721	006270			UNTIL R0 EQ #40	:	
	006270	020027	000040			
	006274	001344				
	006276					
						CMP R0,#40
						BNE B3
5722	006276			TYPE MSG129	:	
	006276	104401	071542	TYPEIT ,MSG129		
				.DSABL CRF		
				RETURN	:	
5723	006302	000207				
5724	006304	000000		TRACE: .WORD 0		
						E3:.....

5726 006306

5727
 5728
 5729
 5730
 5731
 5732
 5733
 5734
 5735
 5736 006306
 5737 006306
 006306 010446
 006310 010546
 006312 013746 177640
 5738 006316
 006316 C10005
 5739 006320 006205
 5740 006322
 006322 010537 002152
 5741 006326
 006326 010337 002034
 5742 006332
 006332 032760 000001 002462
 006340 001403
 5743 006342
 006342 012705 017740
 5744 006346
 006346 000402
 006350
 5745 006350
 006350 012705 070032
 5746 006354
 006354
 5747 006354
 006354 012737 177777 002322
 5748 006362 040537 002322
 5749 006366
 006366 005013
 5750 006370
 006370 011304
 5751 006372 040504
 5752 006374
 006374 005704
 006376 001410
 5753 006400
 006400 005037 002044
 5754 006404
 006404 010437 002150
 5755 006410 104035
 5756 006412 042760 000010 002462
 5757 006420
 006420
 5758 006420

```

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 0
                                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                            ;
                                L31:;;;;;
    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
                                L33:;;;;;
    LET (R3) := CSR1S            ;ONES---->(R3)
    
```



```

5759 006420 013713 002322          LET R4 := (R3)          ;MASK OUT CORRECT FIELD          MOV CSR15.(R3)
006424          ;CLEAR OUT CSR          MOV (R3),R4
5760 006424 011304          CLR (R3)
5761 006426 005013          BIC R5,R4
5762 006430 040504          IF R4 NE CSR15          ;WAS PATTERN WRITTEN CORRECTLY?
006432 020437 002322          ;GOOD DATA = ALL LEGAL BITS SET          CMP R4,CSR15
006436 001411          ;BAD DATA=CSR          BEQ L34
5763 006440 013737 002322 002044          LET GOOD := CSR15          IN CSR
006440          ;BIT CLEAR ERROR          MOV CSR15,GOOD
5764 006446 010437 002150          LET CSR := R4
5765 006452 104010          ERROR +10
5766 006454 042760 000010 002462          BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
5767 006462          END ;
006462          ;SET UP LOOP COUNTER          L34:;;;;;
5768 006462 005037 002264          LET PASFLG := #0          CLR PASFLG
5769 006466 006466          REPEAT          ;REPEAT WITH A FIELD OF 1'S THROUGH 0'S
006466          ;INCREMENT LOOP COUNTER          B4:;;;;;
5770          ;USE USER PAR FOR BIT COUNTER          O'S THROUGH 1'S
5771 006466 005237 002264          INC PASFLG
5772 006472 012737 177777 177640          LET UIPARO := # 1
5773 006500          IF PASFLG EQ #1          ;PASS 1          MOV #-1,UIPARO
006500 023727 002264 000001          ;1----->FIELD OF ZEROS          CMP PASFLG,#1
006506 001003          ;PASS 2          BNE L35
5774 006510 012702 000001          LET R2 := #1
5775 006514 000402          ELSE
006514          ;0----->FIELD OF ONES          MOV #1,R2
5776 006516 012702 177776          LET R2 := #177776
006516          ;DO BITS 0-4 AND 13-15          BR L36
5777 006522          END ;          L35:;;;;;
006522          REPEAT          ;L36:;;;;;
5778 006522          INC UIPARO          ;INCREMENT BIT POINTER          B5:;;;;;
5779 006522 005237 177640          IF PASFLG EQ #1 AND #BIT0 OFF.IN CSRINFO(R0) ;
006522          ;IF THIS IS PASS 1 ON A MSV11-L/P,CLEAR BIT 14
5780 006526 023727 002264 000001          BIC #BIT14,R2
006526          ;IF THIS IS PASS 2 ON A MSV11-L/P, CLEAR ECRSRD BIT AND
006534 001006          ;          L37:;;;;;
006536 032760 000001 002462          IF PASFLG EQ #2 AND #BIT0 0-F.IN CSRINFO(R0) ;
006544 001002          ;          CMP PASFLG,#2
5781 006546 042702 040000          ;          BNE L40
5782 006552          ;          BIT #BIT0,CSRINFO(R0)
006552          ;          BNE L37
5783 006552 023727 002264 000002          IF PASFLG EQ #2 AND #BIT0 0-F.IN CSRINFO(R0) ;
006552          ;          CMP PASFLG,#2
006560 001006          ;          BNE L40
006562 032760 000001 002462          ;          BIT #BIT0,CSRINFO(R0)
006570 001002          ;          BNE L40
5784 006572 042702 040004          BIC #BIT14:BIT2,R2
5785 006576          ;          ;IF THIS IS PASS 2 ON A MSV11-L/P, CLEAR ECRSRD BIT AND
006576          ;          L40:;;;;;
5786 006576 010213          LET (R3) := R2          ;WRITE DATA
006576          ;          MOV R2.(R3)

```

```

5787 006600          LET R1 := R2          ;GET GOOD DATA AND MASK IT OUT
      006600 010201          ;MOV R2,R1
5788 006602 040501    BIC R5,R1          ;GET GOOD DATA
5789 006604          LET R4 := (R3)      ;GET DATA THAT IS READ
      006604 011304          ;MOV (R3),R4
5790 006606 040504    BIC R5,R4          ;MASK OUT CSR BITS
5791 006610          IF R1 NE R4        ;IS DATA CORRECT???
      006610 020104          ;CMP R1,R4
      006612 001416          ;BEQ L41
5792 006614          LET GOOD := R1     ;BAD DATA = CSR CONTENTS
      006614 010137 002044    ;MOV R1,GOOD
5793 006620          LET CSR := R4      ;GET GOOD DATA
      006620 010437 002150    ;MOV R4,CSR
5794 006624          IF PASFLG EQ #1    ;SELECT ERROR DEPENDING ON PASS
      006624 023727 002264 000001    ;CMP PASFLG,#1
      006632 001002          ;BNE L42
5795 006634 104035    ERROR +35        ;BIT SET ERROR
5796 006636          ELSE              ;PASS 2
      006636 000401          ;BR L43
      006640          ;L42:;;;;;;;;
5797 006640 104010    ERROR +10        ;BIT CLEAR ERROR
5798 006642          END
      006642          ;L43:;;;;;;;;
5799 006642 042760 000010 002462    BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
5800 006650          END
      006650          ;L41:;;;;;;;;
5801 006650          IF PASFLG EQ #1    ;GET DATA FOR NEXT LOOP
      006650 023727 002264 000001    ;CMP PASFLG,#1
      006656 001002          ;BNE L44
5802 006660          ASL R2            ;SHIFT 1 ACROSS 0'S
5803 006662          ELSE
      006662 000402          ;BR L45
      006664          ;L44:;;;;;;;;
5804 006664 000261    SEC              ;SET CARRY
5805 006666 006102    ROL R2          ;ROTATE A 0 ACROSS A FIELD OF ONES
5806 006670          END
      006670          ;L45:;;;;;;;;
5807 006670          UNTIL UIPARO EQ #15. ;UNTIL ALL BITS ARE DONE
      006670 023727 177640 000017    ;CMP UIPARO,#15.
      006676 001311          ;BNE B5
      006700          ;E5:;;;;;;;;
5808 006700          UNTIL PASFLG EQ #2  ;DONE WITH 2 PASSES
      006700 023727 002264 000002    ;CMP PASFLG,#2
      006706 001267          ;BNE B4
      006710          ;E4:;;;;;;;;
5809 006710          IF #BIT0 SET.IN CSRINFO(R0) THEN JUMPTO DONE ;IF MSV11-L/P DO ONE LAST WRITE
      006710 032760 000001 002462    ;BIT #BIT0,CSRINFO(R0)
      006716 001402          ;BEQ L50
      006720 000137 006766          ;JMP DONE
      006724          ;L50:;;;;;;;;
5810 006724          LET (R3) := #140005 ;WRITE ONES TO CSR WITH ECSRRO BIT ENABLED
      006724 012713 140005          ;MOV #140005,(R3)
5811 006730          LET R2 := (R3)     ;READ CSR FOR CORRECT BITS
      006730 011302          ;MOV (R3),R2
5812 006732 042702 037772    BIC #37772,R2      ;CLEAR UNWANTED BITS
5813 006736          IF R2 NE #140005   ;WAS WRITE CORRECT
      006736 020227 140005          ;CMP R2,#140005
  
```

```

006742 001411
5814 006744
006744 012737 140005 002044
5815 006752
006752 010237 002150
5816 006756 104010
5817 006760 042760 000010 002462
5818 006766
006766
5819 006766
006766 005013
5820 006770
006770 012637 177640
006774 012605
006776 012604
5821 007000 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 UIPARO,R5,R4 ;RESTORE UIPARO,R4, AND R5
RETURN

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

```

5824 ;THE FOLLOWING ROUTINE DETERMINES WHICH CSR CONTROLS PROGRAM SPACE
5825 ;
5826 007002 104424 ;CTEST: CACHOFF
5827 007004 012737 177777 002532 MOV #177777,PGMCSR
5828 007012 012737 002000 172350 MOV #2000,KIPAR4 ;SET UP MAP REGISTER
5829 007020 012701 002412 MOV #TESTADD,R1
5830 007024 012737 100000 002412 MOV #100000,TESTADD
5831 007032 012737 100002 002414 MOV #100002,TESTADD+2
5832 007040 005000 CLR R0 ;CLEAR CSR COUNTER
5833 007042 005037 002152 CLR CSRNO
5834 007046 013703 002224 MOV TOTCSRS,R3 ;OBTAIN CSR MAP
5835 007052 000240 NOP ;DEBUG AID
5836 007054 006303 4#: ASL R3 ;PUT HIGH ORDER BIT INTO C BIT
5837 007056 103407 BCS 2# ;BRANCH IF CSR EXISTS
5838 007060 062700 000002 1#: ADD #2,R0 ;UPDATE CSR COUNTER
5839 007064 010037 002152 MOV R0,CSRNO
5840 007070 005703 TST R3 ;IS MAP EMPTY?
5841 007072 001465 BEQ 3# ;BRANCH IF SO
5842 007074 000767 BR 4#
5843 007076 C00240 2#: NOP ;DEBUG AID
5844 007100 000241 CLC ;CLEAR CARRY
5845 007102 032760 000001 002462 BIT #BIT0,CSRINFO(R0) ;IS THIS PARITY MEMORY?
5846 007110 001014 BNE 5# ;BRACH IF NOT
5847 007112 052760 000004 172100 BIS #BIT2,CSRADD(R0) ;SET WRITE WRONG PARITY
5848 007120 012771 123456 000000 MOV #123456,#(R1) ;WRITE DATA
5849 007126 012771 123456 000002 MOV #123456,#2(R1)
5850 007134 005060 172100 CLR CSRADD(R0) ;RESTORE CSR
5851 007140 000414 BR 6#
5852 007142 012760 000000 172100 5#: MOV #0,CSRADD(R0) ;CLEAR THE CSR UNDER TEST
5853 007150 012771 123456 000000 MOV #123456,#(R1) ;WRITE DATA
5854 007156 012771 123456 000002 MOV #123456,#2(R1)
5855 007164 012760 020006 172100 MOV #20006,CSRADD(R0) ;SET DIAG CHECK MODE
5856 007172 005771 000000 6#: TST #(R1) ;WRITE CHECKBITS TO CSR
5857 007176 016004 172100 MOV CSRADD(R0),R4 ;WRITE CSR TO R4
5858 007202 032760 000001 002462 BIT #BIT0,CSRINFO(R0) ;PARITY MEMORY?
5859 007210 001003 BNE 7# ;BRANCH IF NOT
5860 007212 005704 TST R4 ;PARITY ERROR?
5861 007214 100412 BMI 8# ;BRACH IF SO
5862 007216 000720 BR 1# ;TRY NEXT CSR
5863 007220 000240 7#: NOP ;DEBUG AID
5864 007222 072427 177773 ASH #-5,R4
5865 007226 042704 177700 BIC #1C77,R4
5866 007232 012702 000040 MOV #40,R2 ;LOAD IN CORRECT CHECK BITS FOR MSV11-J
5867 007236 020204 CMP R2,R4 ;CORRECT CHECKBITS?
5868 007240 001307 BNE 1# ;BRANCH IF NOT
5869 007242 010037 002532 8#: MOV R0,PGMCSR
5870 007246 000240 3#: NOP ;DEBUG AID
5871 007250 104502 CLRCR ;CLEAR ALL CSR'S
5872 007252 012771 000000 000000 MOV #0,#(R1) ;RESTORE TEST LOCATIONS
5873 007260 012771 000000 000002 MOV #0,#2(R1)
5874 007266 023727 002532 177777 CMP PGMCSR,#177777
5875 007274 001402 BEQ FINT# ;IF PROGRAM CSR NOT FOUND GO TO FINT
5876 007276 000137 007320 JMP CLRMEM ;GO TO SIZING ROUTINE IF FOUND
5877 007302 012737 001000 172350 FINT#: MOV #1000,KIPAR4
5878 007310 104401 071334 TYPE MSG126
 .DSABL ,MSG126
 CRF

```

5879 007314 005037 002532

CLR PGMCSR

;SET TO DEFAULT OF 0

5881 007320

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

```

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

```

5882
 5883
 5884
 5885
 5886
 5887

```

; 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.

```

5888 007320 012737 007430 000004

CLRMEM: SET4 #CLREX ;NONEM TRAPS GO TO CLREX

5889 007326 005037 006304

MOV #CLREX,4

5890 007332 012737 000001 002076

.DSABL CRF

5891 007340 012737 002000 172350

CLR TRACE

5892 007346 012701 100000

MOV #1,NOPAR ;IGNORE PARITY ERRORS

5893 007352 020127 117776

MOV #2000,KIPAR4 ;SET UP MAP TO START AT BANK 2

5894 007356 001003

MOV #100000,R1 ;R1 MAPS TO KIPAR4

5895 007360 012737 177777 006304

1\$: CMP R1,#117776 ;WHOLE 16K BANK DONE?

5896 007366 005021

BNE 2\$;KEEP GOING IF NOT

5897 007370 005737 006304

2\$: MOV #-1,TRACE ;USE TRACE FLAG TO FLAG END OF BANK

5898 007374 001001

CLR (R1)+ ;CLEAR CONTENTS & INCREMENT

5899 007376 000765

TST TRACE ;EOB FLAG SET?

5900 007400 062737 000200 172350

BNE 3\$;GO TO NEXT BANK IF SO

5901 007406 022737 177600 172350

3\$: ADD #200,KIPAR4 ;SET MAP FOR NEXT BANK

5902 007414 001405

CMP #177600,KIPAR4 ;ARE WE AT THE PERIPHERAL PAGE

5903 007416 005037 006304

BEQ CLREX ;YES GO ON

5904 007422 012701 100000

CLR TRACE ;RESET FLAG

5905 007426 000751

MOV #100000,R1 ;RESET R1

5906 007430 000240

CLREX: NOP ;CLEAR NEXT BANK

5907 007432 005037 006304

CLR TRACE

5908 007436

RES4

007436 012737 034002 000004

MOV #TIMEOUT,4

007444 022737 000005 004064

CMP #5,PROTYP ;IS THIS AN 11/83/84 ?

007452 001002

BNE 101\$;BRANCH IF NCT

007454 005037 177766

101\$: CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS

007460

.DSABL CRF ;THAT A EXPECTED TRAP COULD HAVE SET

5911 007460

```

ANA2:  SUBST <<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.
; *

```

5936 007460 012737 010532 000004

```

SET4 #100# ;NE MEMORY TRAPS GO TO 100#
MOV #100#.4
.DSABL CRF

```

5937 007466 005037 002314
 5938 007472 012701 002412
 5939 007476 013703 002224
 5940 007502 005000
 5941 007504 005005
 5942 007506 005737 002454
 5943 007512 001403
 5944 007514 005037 002560
 5945 007520 000413
 5946 007522 022737 000177 002556 74:
 5947 007530 001407
 5948 007532 013702 002556
 5949 007536 005202
 5950 007540 072227 000011
 5951 007544 010237 002560
 5952 007550 012702 000004 14:
 5953 007554 005037 002564
 5954 007560 012737 001000 172350
 5955 007566 012737 001000 172352
 5956 007574 006303 24:
 5957 007576 103420
 5958 007600 062700 000002
 5959 007604 010037 002152
 5960 007610 005703
 5961 007612 001370
 5962 007614 012737 001000 172350

```

CLR CHECK ;CLEAR CHECK
MOV #TESTADD,R1 ;SET UP THE VIRTUAL ADDR. POINTER
MOV TOTCSRS,R3 ;MOVE CSR MAP INTO R3
CLR R0 ;CLEAR THE CSR POINTER
CLR R5 ;CLEAR THE PROGRAM CSR STATUS POINTER
TST NO22BIT ;DO WE HAVE 22 BIT ADDRESSING ?
BEQ 7# ;BRANCH IF WE DO
CLR LASTBLOCK ;ADJUST LASTBLOCK INDICATOR FOR 124K MACHINE
BR 1# ;BRANCH OVER NEXT PIECE OF CODE
CMP #177, LASTBANK ;IS THERE Q-BUS MEMORY ABOVE 17776000?
BEQ 1# ;BRANCH IF NOT
MOV LASTBANK,R2 ;SET UP A NEW LAST BLOCK INDICATOR
INC R2
ASH #9.,R2
MOV R2, LASTBLOCK
MOV #4,R2 ;R2 IS INDEX FOR CONFIG TABLE
CLR ENDFLG ;CLEAR END OF MEMORY FLAG
MOV #1000,KIPAR4 ;SET KIPAR4 FOR BANK 1
MOV #1000,KIPAR5 ;SET KIPAR5 FOR BANK 1
ASL R3 ;DOES THIS CSR EXIST?
BCS 3# ;BRANCH IF IT DOES EXIST
ADD #2,R0 ;INCREMENT THE CSR POINTER
MOV R0, CSRNO ;STORE IT IN CSRNO ALSO
TST R3 ;ARE THERE ANY MORE CSR'S TO DO?
BNE 2# ;BRANCH IF ALL CSRS NOT DONE
MOV #1000,KIPAR4 ;RESTORE KIPAR4

```

```

5963 007622 012737 001200 172352      MOV      #1200,KIPARS      ;RESTORE KIPARS
5964 007630 013706 002574      MOV      KSTACK,SP        ;RESTORE STACK
5965 007634 000137 010542      JMP      SUBAAS            ;JUMP TO SUBAAS IF ALL CSR'S ARE DONE
5966 007640 010037 002152      MOV      R0,CSRNO         ;MAKE SURE CSRNO IS UPDATED
5967 007644 104424      3#:     CACHOFF            ;TURN THE CACHE OFF
5968 007646 000240      13#:    NOP
5969 007650 012737 100000 002412 45#:    MOV      #100000,TESTADD  ;SET UP VIRTUAL ADDRESS TO KIPAR4
5970 007656 012737 120002 002414      MOV      #120002,TESTADD+2 ;SET UP VIRTUAL ADDRESS TO KIPARS
5971 007664 032762 000040 002664      BIT      #BITS.CONFIG(R2) ;IS THIS A BANK TO SKIP ECC/LOGIC TESTS?
5972 007672 001402      43#:    BEQ      43#         ;NO - BRANCH AROUND NEXT INSTRUCTION
5973 007674 000137 010430      6#      JMP      6#             ;YES - GO TO END OF BANK
5974 007700 005037 002452      43#:    CLR      I          ;CLEAR THE MEMORY CONFIGURATION COUNTER
5975 007704 005771 000000      4#      TST      @ (R1)       ;TEST TO SEE THAT THERE IS MEMORY PRESENT
5976 007710 005237 002452      INC      I                ;MEMORY PRESENT
5977 007714      PUSH    @ (R1),@2(R1)     ;SAVE THE LOCATIONS UNDER TEST
                    MOV @ (R1),-(SP)
                    MOV @2(R1),-(SP)
5978 007724 032760 000001 002462      BIT      #BIT0,CSRINFO(R0) ;IS THIS PARITY MEMORY?
5979 007732 001014      34#     BNE      34#         ;NO - BRANCH
5980 007734 052760 000004 172100      BIS      #BIT2,CSRADD(R0) ;SET WRITE WRONG PARITY
5981 007742 012771 123456 000000      MOV      #123456,@ (R1)   ;SET THE FIRST LOCATION UNDER TEST
5982 007750 012771 123456 000002      MOV      #123456,@2(R1)  ;SET THE SECOND LUT
5983 007756 005060 172100      CLR      CSRADD(R0)       ;CLEAR THE CSR
5984 007762 000411      41#     BR       41#         ;TEST LOCATIONS
5985 007764 012771 123456 000000 34#:    MOV      #123456,@ (R1)   ;SET THE FIRST LOCATION UNDER TEST
5986 007772 012771 123456 000002      MOV      #123456,@2(R1)  ;SET THE SECOND LUT
5987 010000 104503      CLR1CSR ;RESET CSR
5988 010002 104475      CB1CSR  ;SET DIAG. CHECK MODE IN CSR UNDER TEST
5989 010004 000240      NOP     ;DEBUG AID
5990 010006 005771 000000 41#:    TST      @ (R1)         ;READ THE FIRST LUT TO WRITE CKBITS. INTO CSR
5991 010012 104426      READCSR ;READ THE CSR UNDER TEST
5992 010014 000240      NOP     ;DEBUG AID
5993 010016 013704 002150      MOV      CSR,R4          ;GET THE CHECKBITS FROM THE CSR
5994 010022 000240      NO      ;DEBUG AID
5995 010024 010437 002434      MOV      R4,TEMP         ;SAVE IN TEMP FOR LATER
5996 010030 104503      CLR1CSR ;RESET CSR
5997 010032      POP     ;RESTORE LOCATIONS UNDER TEST
                    MOV (SP),@2(R1)
                    MOV (R0),@ (R1)
5998 010042 032760 000001 002462      BIT      #BIT0,CSRINFO(R0) ;IS THIS PARITY MEMORY?
5999 010050 001004      42#     BNE      42#         ;NO - BRANCH
6000 010052 005704      TST      R4              ;DID WE GET A PARITY ERROR?
6001 010054 100420      BMI     25#             ;YES - FILL IN CONFIG TABLE
6002 010056 000137 010430      6#      JMP      6#             ;NO - JUMP TO END OF BANK
6003 010062 072427 177773      42#:    ASH     @-5,R4      ;MANIPULATE THE CSR BITS
6004 010066 042704 177700      BIC     @+C77,R4        ;INTO A USABLE FORM.
6005 010072 012737 000040 002316      MOV      #40,CBITS       ;MSV11-J CHECK BITS
6006 010100 000240      NOP     ;DEBBUGGING AIDE
6007 010102 023704 002316 77#:    CMP      CBITS,R4        ;DO THE CHECKBITS COMPARE TO WHAT WAS WRITTEN?
6008 010106 000240      NOP     ;DEBBUG AIDE
6009 010110 001402      BEQ     25#             ;BRANCH IF THERE IS A MATCH
6010 010112 000137 010256      JMP     22#             ;ELSE BRANCH IF NOT THE SAME
6011      ;*
6012      ;* WE COME HERE IF THERE IS A MATCH
6013      ;*
6014 010116 010004 25#:    MOV      R0,R4          ;GET THE CSR NUMBER
6015 010120 000240      NOP     ;

```

```

6016 010122 006204          ASR      R4          ;SET IT UP FOR USE IN THE
6017 010124 000304          SWAB     R4          ;CONFIGURATION TABLE.
6018 010126 042704 170377    BIC      #170377,R4   ;CLEAR OFF EXTRANEIOUS BITS
6019 010132 050462 002664    15$:    BIS      R4,CONFIG(R2) ;PUT CSR NUMBER IN CONFIG. TABLE
6020 010136 016004 002462    MOV      CSRINFO(R0),R4 ;GET MEMORY TYPE
6021 010142 042704 177770    BIC      #+C7,R4     ;CLEAR OFF THE EXTRANEIOUS BITS
6022 010146 000304          SWAB     R4          ;MOVE INTO PROPER POSITION
6023 010150 050462 002666    BIS      R4,CONFIG+2(R2) ;SET IT INTO THE CONFIG TABLE
6024
6025          ;*
6026          ;* THIS SECTION IS EXECUTED ONLY WHEN THE BANK=1
6027 010154 022737 001000 172350 24$:    CMP      #1000,KIPAR4   ;IS THIS BANK 1 ?
6028 010162 001402          BEQ      30$          ;BRANCH IF TRUE
6029 010164 000137 010430          JMP      6$          ;ELSE JUMP TO END OF THIS BANK
6030 010170 032737 100020 002434 30$:    BIT      #BIT15!BIT4,TEMP ;WAS THERE A SBE OR DBE?
6031 010176 001417          BEQ      10$          ;BRANCH IF NOT
6032 010200 013704 C02434    MOV      TEMP,R4      ;GET CSR CONTENTS
6033 010204 072427 177767    ASH     #-9.,R4      ;MAKE ERROR ADDRESS INTO BANK #
6034 010210 022704 000001    CMP      #1,R4        ;ERROR IN BANKS 0 OR 1?
6035 010214 C03010          BGT      10$          ;BRANCH IF NOT
6036 010216 052762 000001 002664    BIS      #BIT0,CONFIG(R2) ;SET ERROR FLAG IN CONFIG TABLE
6037 010224 105262 002666    INCB    CONFIG+2(R2)  ;ADD ONE TO BANK ERROR COUNT
6038 010230          SET      CONFGERROR   ;PRINT CONFIG TABLE
6039 010236 053737 002670 002664 10$:    BIS      CONFIG+4,CONFIG ;SET UP INFORMATION IN BANK ZERO
6040 010244 053737 002672 002666    BIS      CONFIG+6,CONFIG+2
6041 010252 000240          NOP
6042 010254 000465          BR      6$          ;DEBUG AID
6043          ;*
6044          ;* IF CHECKBITS DID NOT MATCH, WE COME HERE
6045          ;*
6046 010256 032737 100020 002150 22$:    BIT      #BIT15!BIT4,CSR ;SBE OR DBE FLAGS SET?
6047 010264 001001          BNE     8$          ;BRANCH IF TRUE
6048 010266 000460          BR      6$          ;CHECK TO SEE IF IT IS MSV11-J
6049 010270 013704 002152          MOV      CSRNO,R4     ;GET CSRNO
6050 010274 042764 000006 172100    BIC     #6,CSRADD(R4) ;TURN OFF DIAG CHECK & ECC DISABLE
6051 010302          PUSH    R0,R1       ;SAVE R0 & R1
6052 010306 016401 172100          MOV     CSRADD(R4),R1 ;GET CSR INFORMATION
6053 010312 072127 177773          ASH     #-5,R1       ;SET UP ERROR ADDRESS
6054 010316 042701 177600          BIC     #+C177,R1
6055 010322 052764 040000 172100    BIS     #BIT14,CSRADD(R4) ;GET EXTENDED ERROR ADDRESS BITS
6056 010330 016400 172100          MOV     CSRADD(R4),R0 ;READ FROM CSR
6057 010334 042764 040000 172100    BIC     #BIT14,CSRADD(R4) ;TURN OFF EQB BIT
6058 010342 042700 177037          BIC     #+C740,R0    ;SET UP EXTENDED BITS
6059 010346 006300          ASL     R0
6060 010350 006300          ASL     R0
6061 010352 060001          ADD     R0,R1       ;SET UP TOTAL ERROR ADDRESS
6062 010354 010104          MOV     R1,R4       ;SAVE IN R4
6063 010356          POP     R1,R0      ;RESTORE R0 & R1
6064 010362 072427 000005          ASH     #5,R4        ;SET ERROR ADDRESS UP IN PAR NOTATION
6065 010366 020437 172350          CMP     R4,KIPAR4   ;DOES IT EQUAL KIPAR4?
6066 010372 001001          BNE     28$         ;BRANCH IF FALSE
6067 010374 000403          BR      35$         ;YES - MARK INFO IN CONFIG TABLE

```


6068	010376	020437	172352	28:	CMP	R4,KIPAR5		; DOES IT EQUAL KIPAR5?
6069	010402	001012			BNE	6:		; BRANCH IF FALSE
6070	010404	052762	000001	002664	35:	BIS	#BIT0,CONFIG(R2)	; SET BANK ERROR FLAG
6071	010412	105262	002666		INCB	CONFIG+2(R2)		; INCREMENT BANK ERROR COUNTER
6072	010416				SET	CONFGERROR		; PRINT CONFIG TABLE
6073	010424	012737	177777	002450				MOV #1,CONFGERROR
6074		000137	010116		JMP	25:		; YES - MARK INFO IN CONFIG TABLE
6075					;			
6076					;			
6077	010430	104503			;			
6078	010432	005737	002564	6:	CLR1CSR			; CLEAR THE CSR UNDER TEST
6079	010436	001021			TST	ENDFLG		; ARE WE AT TOP OF MEMORY?
6080	010440	062702	000004		BNE	70:		; IF SO THEN EXIT
6081	010444	062737	001000	172350	ADD	#4,R2		; UPDATE CONFIGURATION POINTER
6082	010452	013737	172350	172350	ADD	#1000,KIPAR4		; UPDATE KIPAR4 TO NEXT BANK
6083	010460	022737	177000	172350	MOV	KIPAR4,KIPAR5		; AND UPDATE KIPAR4
6084	010466	001005			CMP	#177000,KIPAR4		; ARE WE AT BANK 177
6085	010470				BNE	70:		; BRANCH IF NOT
	010470	C12737	177777	002564	SET	ENDFLG		; WERE AT LAST BANK
6086	010476	000137	007650					MOV #-1,ENDFLG
6087	010502	023737	002560	172350	70:	JMP	45:	; HAVE WE DONE THE WHOLE MEMORY SPACE?
6088	010510	101402			CMP	LASTBLOCK,KIPAR4		; BRANCH IF DONE ;R-C
6089	010512	000137	007650		BLOS	19:		
6090	010516	062700	000002	19:	JMP	45:		; JUMP IF NOT DONE
6091	010522	000240			ADD	#2,R0		; INCREMENT CSR POINTER
6092	010524	104423			NOP			; DEBUG AID
6093	010526	000137	007550		CACHON			; TURN ON THE CACHE
6094					JMP	1:		; JUMP TO TRY NEXT CSR
6095	010532	062706	000004	100:	ADD	#4,SP		; RESTORE STACK ;R-C
6096	010536	000137	010430		JMP	6:		; GO TO END OF BANK ROUTINE ;R-C

6098 010542 104423
6099 010544 104472
6100 010546

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

;*TEST 2 TEST BANK 0 ACCESSES

010546 000004

TST2: SCOPE
;THIS DOES A "TST" INSTRUCTION ON EVERY LOCATION IN BANK #0 TO SEE
;IF IT GETS ANY PARITY TR'PS.
;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

6101
6102
6103
6104
6105
6106
6107 010550 005037 002072
6108 010554 012737 000001 002076
6109 010562 005037 002070
6110 010566 012737 000001 002100
6111 010574
010574 012737 033736 000004

6112 010602 005000
6113 010604 012701 040000
6114 010610 104424
6115 010612 005720
6116 010614 077102
6117 010616 104423
6118
6119 010620 005737 002072
6120 010624 001403
6121 010626
010626 005237 002064
010632 104003

.DSABL CRF
CLR RO
MOV #SIZE,R1
CACHOFF ;TURN CACHE OFF
1\$: TST (RO)+ ;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

6122 010634 005737 002070
6123 010640 001401
6124 010642 162737 000002 002034
6125 010650
010650 005237 002064
010654 104004

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

6126 010656 053737 002106 002664
6127 010664
010664 012737 034002 000004
010672 022737 000005 004064
010700 001002
010702 005037 177766
010706

3\$: BIS CPUBIT,CONFIG ;SET CORRECT ACCESSED BIT ON BANK 0
RES4 ;RESET TRAPS TO 4 TO DEFAULT
MOV #TIMEOUT,4
CMP #5,PROTOP ;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

6128

6129 010706

```
      SUBTST <<ENABLE ECC FOR CORRECT TRAPS>>  
;*****  
;#SUBTEST  ENABLE ECC FOR CORRECT TRAPS  
;*****  
      IF #SWO SET.IN #SWR OR ACTFLAG IS TRUE
```

6130 010706 032777 000001 171722

```
      BIT #SWO,#SWR  
      BNE L52  
      TST ACTFLAG  
      BEQ L53
```

010714 001003
010716 005737 002350
010722 C01402
010724

```
      L52:;;;;;
```

6131 010724 104506

```
      ENASBE  
      ELSE
```

```
      ;TRAP ON SINGLE BIT ERRORS
```

6132 010726 000401

```
      BR L54
```

010728

```
      L53:;;;;;
```

6133 010730 104472

```
      ECCINIT  
      END ;OF IF #SWO
```

```
      ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
```

6134 010732
010732

```
      L54:;;;;;
```

6137 010732

010732 000004
 6138
 6139
 6140
 6141
 6142
 6143
 6144
 6145 010734 005037 002102
 6146 010740 012737 000001 002076
 6147 010746 012737 000002 002100
 6148 010754
 010754 012737 033736 000004
 6149 010762 012737 011506 002522
 6150 010770 012737 011510 002524
 6151 010776 005237 002102
 6152 011002 023737 002556 002102
 6153 011010 103451
 6154 011012 013701 002102
 6155 011016 006301
 6156 011020 006301
 6157 011022 010137 002104
 6158 011026 005037 002074
 6159 011032 005037 002072
 6160 011036 005037 002070
 6161 011042
 011042 010346
 011044 013703 002102
 011050 004737 035604
 011054 012603
 6162 011056 105761 002664
 6163 011062 100542
 6164 011064 012777 000207 171430
 6165 011072 012700 060000
 6166 011076 010004
 6167 011100 012701 040000
 6168 011104 010103
 6169 011106 005002
 6170 011110 104424
 6171 011112
 011112 053737 002552 177776
 6172 011120 004737 011502
 6173 011124 104417
 6174 011126 104423
 6175 011130 000240
 6176 011132 000416
 6177 011134 005037 002102
 6178 011140
 011140 012737 034002 000004
 011146 022737 000005 004064

```

NEW*ST <<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 #207,8LINK1 ;PUT "RETURN" INSTRUCTION AFTER WRITE ROUTINE
MOV #FIRST,R0
MOV R0,R4
MOV #SIZE,R1
MOV R1,R3
CLR R2 ;DATA IS ZEROS
CACHOFF ;TURN CACHE OFF
TESTAREA ;ENTER SUPERVISOR MODE
BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
CALL MTST3
2#: KERNEL ;ENTER KERNEL MODE
CACHON ;TURN CACHE ON
NOP
BR TAG3# ;SKIP NEXT INSTRUCTION
TAG2#: CLR BANK ;RESET TRAPS TO 4 TO DEFAULT
RES4
MOV #TIMEOUT,4
CMP #5,PROTYP ;IS THIS AN 11/83/84 ?
  
```

```

011154 001002          BNE      101#      ;BRANCH IF NOT
011156 005037 177766   CLR      CPUERR    ;CLEAR OUT THE CPU ERROR REGISTER BITS
011162          101#      ;THAT A EXPECTED TRAP COULD HAVE SET

        .DSABL CRF
6179 011162 005037 002076 CLR      NOPAR      ;INDICATE DEFAULT PARITY ACTION
6180 011166 000557          BR      SUBAAI
6181 011170 005737 002070 TAG3# : TST      NEMCNT  ;ANY TRAPS?
6182 011174 001401          BEQ      1#        ;NO - SKIP
6183 011176 000677          BR      TAG9#     ;NOW - TRY NEXT BANK
6184 011200 104424          1# : CACHOFF  ;TURN CACHE OFF
6185 011202          TESTAREA ;ENTER SUPERVISOR MODE
        011202 053737 002552 177776 BIS      TESTMODE,PSW ;GO TO SYSTEM TEST MODE
        .DSABL CRF
6186 011210 004777 171310 CALL     @LINK2     ;FINISH PATTERN
6187 011214 104417          KERNEL  ;ENTER KERNEL MODE
6188 011216 104423          CACHON   ;TURN CACHE ON
6189 011220 000240          NOP      ;DEBUG AID
6190 011222 005737 002074 TST      PATERR    ;ANY PATTERN ERRORS
6191 011226 C01032          BNE      2#        ;YES - SKIP
6192 011230 005737 002072 TST      PARCNT    ;ANY PARITY ERRORS
6193 011234 001027          BNE      2#        ;YES - SKIP
6194 011236 005737 002070 TST      NEMCNT    ;ANY NON EXISTANT MEMORY
6195 011242 001024          BNE      2#        ;YES SKIP
6196 011244 012700 060000 MOV      @FIRST,RO
6197 011250 010004          MOV      RO,R4
6198 011252 012701 040000 MOV      @SIZE,R1
6199 011256 010103          MOV      R1,R3
6200 011260 013702 002614 MOV      ONES,R2
6201 011264 012777 000240 171230 MOV      @000240,@LINK1 ;DATA IS ONES
6202 011272 104424          CACHOFF  ;PUT "NOP" INSTRUCTION BACK IN SUBROUTINE
6203 011274          TESTAREA ;TURN CACHE OFF
        011274 C53737 002552 177776 BIS      TESTMODE,PSW ;ENTER TEST MODE
        .DSABL CRF ;GO TO SYSTEM TEST MODE
        CALL     MTST3
6204 011302 004737 011502          KERNEL  ;DO IN MEMORY IF NOT
6205 011306 104417          CACHON   ;ENTER KERNEL MODE
6206 011310 104423          NOP      ;TURN CACHE ON
6207 011312 000240          MOV      BANKINDEX,RO ;DEBUG AID
6208 011314 013700 002104 2# : TST      PATERR    ;ANY PATTERN ERRORS?
6209 011320 005737 002074          BNE      3#        ;YES - SKIP
6210 011324 001006          TST      PARCNT    ;ANY PARITY ERRORS?
6211 011326 005737 002072          BNE      3#        ;YES - SKIP
6212 011332 001003          TST      NEMCNT    ;ANY HOLES?
6213 011334 005737 002070          BEQ      4#        ;NONE - SKIP
6214 011340 001406          3# : BIS      @BIT0,CONFIG(RO) ;SET ERROR BIT IN THIS BANK
6215 011342 052760 000001 002664 SET      CFGERROR  ;FORCE PRINTING OF CONFIGURATION TABLE
6216 011350          MOV      @1,CFGERROR
        011350 012737 177777 002450 4# : BIS      CPUBIT,CONFIG(RO) ;SET ACCESSED BIT
6217 011356 053760 002106 002664 JMP      TAG9#
6218 011364 000137 010776          ;TEST A PROTECTED BANK
6219          TSTBANK:PUSH R1
6220          MOV R1,(SP)
6221 011370          MOV      @1,NONEM ;SET NON-EXISTANT MEMORY TO COUNT
        011370 010146          MOV      @FIRST,RO
6222 011372 012737 000001 002100          MOV      @20000,R1
6223 011400 012700 060000
6224 011404 012701 020000

```

```

6225 011410 104424          CACHOFF          ;TURN CACHE OFF
6226 011412          TESTAREA        ;ENTER TEST MODE
      011412 053737 002552 177776    BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
      .DSABL CRF
6227 011420 005720          4#: TST (R0)+
6228 011422 077102          SOB R1,4#
6229 011424 104417          KERNEL
6230 011426 104423          CACHON
6231 011430 012737 000002 002100    MOV #2,NONEM ;ENTER KERNEL MODE
6232 011436          POP R1          ;TURN CACHE ON
      011436 012601          ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
6233 011440          IF PARCNT NE #0 ;MOV (SP)+,R1
      011440 005737 002072          TST PARCNT
      011444 001406          BEQ L55
6234 011446 052761 000001 002664    BIS #BIT0,CONFIG(R1) ;ERROR BANK
6235 011454          SET CFGERRR
      011454 012737 177777 002450    MOV #-1,CFGERRR
6236 011462          END ;OF IF PARCNT
      011462          L55:;
6237 011462          IF NEMCNT EQ #0
      011462 005737 002070          TST NEMCNT
      011466 001003          BNE L56
6238 011470 053761 002106 002664    BIS CPUBIT,CONFIG(R1) ;ACCESSED BANK
6239 011476          END ;OF IF NEMCNT
      011476          L56:;
6240 011476 000137 010776          JMP TAG9#
6241 011502 010220          MTST3: MOV R2,(R0)+
6242 011504 077102          SOB R1,MTST3
6243 011506 000240          NOP
6244 011510 012401          2#: MOV (R4)+,R1
6245 011512 020102          CMP R1,R2
6246 011514 001402          BEQ 3#
6247 011516 005237 002074          INC PATERR
6248 011522 077306          3#: SOB R3,2#
6249 011524 000207          RETURN

```

```

;WRITE MEMORY TO PATTERN
;DO TILL DONE
;THIS IS EITHER A NOP OR RETURN
;READ DATA
;DOES IT MATCH ?
;
;IF NOT COUNT ERRORS
;ANY WAY DO ALL MEMORY
;DONE...RETURN

```

6251 011526

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

;*****
 ;SUBTEST FIND SHADOW INHIBIT MODE POINTERS
 ;*****

6252
 6253
 6254

;* 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.

6255 011526 005037 002102
 6256 011532 004737 037760
 6257 011536 015700 002104
 6258 011542

SHADL1: CLR BANK ;RESET BANK TO ZERO
 CALL EXBANK ;SET BANK PARAMETERS
 MOV BANKINDEX,R0
 IF ACFLAG IS TRUE AND INTFLAG IS TRUE

011542 005737 002116
 011546 001414
 011550 005737 002136
 011554 001411

TST ACFLAG
 BEQ L57
 TST INTFLAG
 BEQ L57

6259 011556 062702 000040
 6260 011562 062737 000020 002102
 6261 011570 052760 000200 002664
 6262 011576

ADD #40,R2 ;POINT TO BANKINDEX + 8
 ADD #20,BANK ;POINT TO BANK + 16
 BIS #BIT7,CONFIG(R0) ;MAKE NEW BANK PROGRAM SPACE
 ELSE

011576 C00402
 011600
 6263 011600 005237 002102
 6264 011604

BR L60
 L57:;;;;;
 INC BANK ;GO TO NEXT BANK
 END; OF IF ACFLAG

011604
 6265 011604 023737 002556 002102
 6266 011612 002347

L60:;;;;;
 CMP LASTBANK,BANK ;HAVE WE DONE ALL THE BANKS?
 BGE SHADL1 ;BRANCH IF NOT

6269 011614

NEWST <<ECC INHIBIT MODE POINTER TEST>>

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

011614 000004
 6270
 6271
 6272
 6273
 6274
 6275
 6276
 6277
 6278
 6279
 6280
 6281
 6282
 6283
 6284
 6285
 6286
 6287 011616 104424
 6288 011620 012737 177777 002156
 6289 011626 005037 002102
 011632
 6290 011632 012701 060000
 6291 011636 004737 037760
 6292 011642 013700 002104
 6293 011646 005737 002116
 011652 001436
 6294 011654 005737 002120
 011660 001433
 6295 011662 005737 002342
 011666 001030
 6296 011670 012703 000002
 6297 011674 116002 002665
 6298 011700 006302
 6299 011702 042702 177741
 6300 011706 010237 002152
 6301 011712 023737 002152 002156
 011720 001413
 6302 011722 013737 002152 002156
 6303 011730 005737 002122
 011734 001003
 6304 011736 052760 000100 002664
 6305 011744
 011744
 6306 011744 004737 012036
 6307 011750
 011750

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 #0,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
 ; SET INDEX COUNTER
 MOV #2,R3
 MOV# 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: ;


```

6308 011750          END; OF IF SKIPMK          L63:;;;;;;;;
      011750
6309 011750          END; OF IF MKFLAG          L62:;;;;;;;;
      011750
6310 011750          END; OF IF ACFLAG          L61:;;;;;;;;
      011750
6311 011750          END; OF FOR BANK
      011750 005237 002102          INC BANK
      011754 023737 002102 002556      CMP BANK, LASTBANK
      011762 003723          BLE B6
      011764
6312 011764          MAP          ;MAP TEST SPACE TO BANK 0      E6:;;;;;;;;
      011764 010346          MOV R3, -(SP)
      011766 012703 000200      MOV #200, R3
      011772 004737 035604      CALL MAPPER
      .DSABL CRF
      011776 012603          MOV (SP)+, R3
6313 012000 005037 002102      CLR BANK
6314 012004          IF #SWO SET. IN #SWR OR ACTFLAG IS TRUE
      012004 C32777 000001 170624      BIT #SWO, #SWR
      012012 001003          BNE L66
      012014 005737 002350      TST ACTFLAG
      012020 001402          BEQ L67
      012022
6315 012022 104506          ENASBE          ;TRAP ON SINGLE BIT ERRORS      L66:;;;;;;;;
6316 012024          ELSE
      012024 000401          BR L70
      012026
6317 012026 104472          ECCINIT          ;TRAP ON DOUBLE BIT ERRORS (NORMAL)      L67:;;;;;;;;
6318 012030          END; OF IF #SWO
      012030
63.9 012030 104423          CACHON          ;TURN THE CACHE BACK ON      L70:;;;;;;;;
6320 012032 000137 012200      JMP SUBAAR      ;JUMP OVER THE SUBROUTINE
  
```

```

6322 012036 005004          IMPTEST:CLR      R4
6323 012040          MAP BANK                      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
        012040 010346          MOV      BANK,R3
        012042 013703 002102  CALL     MAPPER
        012046 004737 035604  .DSABL  CRF
                                           ;ENTER TEST MODE
                                           ;GO TO SYSTEM TEST MODE
        012052 012603          MOV (SP),R3
6324 012054 005005          CLR      R5
6325 012056 012737 020000 002150  MOV     #BIT13,CSR
6326 012064          TESTAREA
        012064 053737 002552 177776  BIS     TESTMODE,PSW
        .DSABL  CRF
        PUSH   (R1)                      ;SAVE TEST LOCATION
                                           MOV (R1), (SP)
6327 012072          ADD     R3,R1
        012072 011146          PUSH   (R1)
                                           ;INDEX TO NEXT LOCATION
                                           ;SAVE TEST LOCATION
6328 012074 060301          MOV (R1), (SP)
6329 012076          CHK1DIS
        012076 011146          MOV     R4,(R1)
6330 012100 104505          SUB     R3,R1
6331 012102 0.0411          MOV     R4,(R1)
6332 012104 160301          CLR1CSR
6333 012106 010411          TST    (R1)
6334 012110 104503          WAS1DBE
6335 012112 005711          ;CLEAR CSR
6336 012114 104501          ;READ CHECKBITS INTO REAL CSR
6337          ;WAS THERE A DOUBLE BIT ERROR
6338          ;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
6339
6340 012116          ON.NOERROR ;1
        012116 103435          BCS L71
6341 012120 012737 020000 002150  MOV     #BIT13,CSR
6342 012126 104505          CHK1DIS
6343 012130 013711 002614  MOV     ONES,(R1)
6344 012134 104503          CLR1CSR
6345 012136 005711          TST    (R1)
6346 012140 104501          WAS1DBE
6347 012142          ON.NOERROR ;2
        012142 103423          BCS L72
6348 012144 104513          ;ENABLE CHECK/SYNDROME BIT REGISTER
6349 012146 012737 023140 002150  MOV     #23140,CSR
6350 012154 104505          ;WRITE DBE'S IN CSR
6351 012156 010411          ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6352 012160 104503          CHK1DIS
6353 012162 005711          MOV     R4,(R1)
6354 012164 104501          CLR1CSR
6355 012166          TST    (R1)
        012166 103411          ;CLEAR CSR
        104513          ;WAS THERE A DOUBLE BIT ERROR
6356 012170 104513          ON.NOERROR ;3
        012170 104513          BCS L73
6357 012172 012737 023604 002150  ;ENABLE CHECK/SYNDROME BIT REGISTER
6358 012200 104505          ;WRITE DBE'S IN CSR
6359 012202 010411          ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6360 012204 104503          CHK1DIS
6361 012206 005711          MOV     R4,(R1)
6362 012210 104501          CLR1CSR
6363 012212          TST    (R1)
        012212          ;CLEAR CSR
        012212          ;WAS THERE A DOUBLE BIT ERROR
6364 012212          END ;OF ON.NOERROR ;3
        012212          L73:::
        012212          END ;OF ON.NOERROR ;2
        012212          L72:::

```

```

6365 012212          END ;OF ON.NOERROR :1
        012212
6366 012212          ON.ERROR
        012212 103001
6367 012214 005205   INC R5
6368 012216          END ;OF ON.ERROR
        012216
6369 012216 104471   ECC1DIS
6370 012220 010411   MOV R4,(R1)
6371 012222 060301   ADD R3,R1
6372 012224 010411   MOV R4,(R1)
6373 012226 104503   CLR1CSR
6374 012230 005705   TST R5
6375 012232 001405   BEQ 14
6376 012234 050560 002664 BIS R5,CONFIG(R0)
6377 012240 105260 002666 INCB CONFIG+2(R0)
6378 012244 104036   ERROR +36
6379 012246          14: POP (R1)
        012246 012611
6380 012250 160301   SUB R3,R1
6381 012252          POP (R1)
        C12252 012611
6382 012254 104417   KERNEL
6383 012256 000207   RETURN
6384
6385 012260          SUBAAR: SET STOPOK
        012260 012737 177777 002420

```

L71:

BCC L74

:IDENTIFY AS BAD BANK

L74:

:DISABLE ERROR CORRECTON

:CLEAR OUT DOUBLE BIT ERROR!

:INDEX TO SECOND WORD

:CLEAR OUT DOUBLE BIT ERROR!

:RESTORE TEST LOCATION (2ND WORD)

MOV (SP)+,(R1)

:GO BACK TO FIRST WORD

:RESTORE TEST LOCATION (1ST WORD)

MOV (SP)+,(R1)

:PROGRAM CAN NOW BE HALTED

MOV #-1,STOPOK

6388 012266

```

SUBTST <<LEGAL CONFIGURATION CHECK>>
;*****
;SUBTEST LEGAL CONFIGURATION CHECK
;*****
  
```

6389 012266 012700 000020
 6390 012272 012701 002462
 6391 012276 005021
 6392 012300 077002
 6393 012302

```

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

CLR BANK
 B7:*****

012302 005037 002102
 012306
 6394 012306 004737 037760
 6395 012312 013700 002104
 6396

```

CALL EXBANK
MOV BANKINDEX,R0
  
```

6397 012316
 012316 005737 002116
 012322 001444

IF ACFLAG IS TRUE

TST ACFLAG
 BEQ L75

6398 012324 116003 002665
 6399 012330 042703 177760
 6400 012334 C06303
 6401 012336 005263 002462
 6402 012342

```

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

TST MKFLAG
 BEQ L76

012342 005737 002120
 012346 001432

```

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

B10:*****

6403
 6404 012350
 012350

IF INTFLAG IS TRUE

TST INTFLAG
 BEQ L77

6405 012350 005737 002136
 012354 001423

```

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

CMP R3,R4
 BNE L100

6406 012356 116003 002665
 6407 012362 010304
 6408 012364 042703 177760
 6409 012370 072427 177774
 6410 012374 042704 177760
 6411 012400

```

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

BR E10

012400 020304
 012402 001007

END; OF IF R3

L100:*****

6412 012404 042760 014000 002666
 6413 012412 042760 170000 002664
 6414 012420

ELSE

BR L101

012420 000405
 6415 012422
 012422

LEAVE LEGALCSR

L77:*****

6417 012424
 012424 000403

END; OF IF INTFLAG

BR E10

6418 012426
 012426
 6419 012426

SET CONFGERROR

L101:*****

012426 012737 177777 002450
 6420 012434
 012434
 6421 012434
 012434

END LEGALCSR

MOV #-1,CONFGERROR

E10:*****

END ;OF IF MKFLAG

L76:*****

```

6422 012434          END ;OF IF ACFLAG
      012434
6423 012434          END; OF FOR BANK
      012434 005237 002102
      012440 023737 002102 002556
      012446 003717
      012450
6424 012450          PUSH    R5,R0
      012450 010546
      012452 010046
6425 012454 005000          CLR    R0
6426 012456 005001          CLR    R1
6427 012460 005005          CLR    R5
6428 012462 005037 012670          CLR    MBERR
6429 012466 022761 000177 002462 2#:  CMP    #177,CSRINFO(R1)
6430 012474 002043          BGE    5#
6431 012476 022761 000010 002462  CMP    #10,CSRINFO(R1)
6432 012504 002003          BGE    3#
6433 012506 004737 013012          CALL  ILLCSR
6434 012512 000434          BR     5#
6435 012514 016005 002664 3#:  MOV    CONFIG(R0),R5
6436 012520 032705 000002          BIT    #BIT1,R5
6437 012524 001415          BEQ    4#
6438 012526 042705 170377          BIC    #C7400,R5
6439 012532 072527 177771          ASH    #-7,R5
6440 012536 020501          CMP    R5,R1
6441 012540 001007          BNE    4#
6442 012542 032760 010000 002666  BIT    #BIT12,CONFIG+2(R0)
6443 012550 001003          BNE    4#
6444 012552 012737 000001 012670  MOV    #1,MBERR
6445 012560 062700 000004 4#:  ADD    #4,R0
6446 012564 022700 000340          CMP    #340,R0
6447 012570 001351          BNE    3#
6448 012572 005737 012670          TST   MBERR
6449 012576 001402          BEQ    5#
6450 012600 004737 013012          CALL  ILLCSR
6451 012604 005000 5#:  CLR    R0
6452 012606 005037 012670          CLR    MBERR
6453 012612 062701 000002          ADD    #2,R1
6454 012616 022701 000040          CMP    #40,R1
6455 012622 001321          BNE    2#
6456 012624          POP    R0,R5
      012624 012600
      012626 012605
6457 012630 005037 012670          CLR    MBERR
6458 012634 012700 000774          MOV    #774,R0
6459 012640 032760 000002 002664 6#:  BIT    #BIT1,CONFIG(R0)
6460 012646 001003          BNE    7#
6461 012650 162700 000004          SUB    #4,R0
6462 012654 000771          BR     6#
6463 012656 006200 7#:  ASR    R0
6464 012660 006200          ASR    R0
6465 012662 010037 002556          MOV    R0, LASTBANK
6466 012666 000402          BR     SKUJ
6467 012670 000000          MBERR: .WORD 0
6468 012672 000000          PHEBE: .WORD 0
6469 012674 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

```

6470 012676 005037 012672          CLR      PHEBE          ;CLEAR COUNTER
6471 012702 032760 000002 002664 1$: BIT      @BIT1.CONFIG(R0) ;IS THERE MEMORY PRESENT?
6472 012710 001431          BEQ      3$           ;BRANCH IF NOT
6473 012712 032760 010000 002666 BIT      @BIT12.CONFIG+2(R0) ;IS IT INTERLEAVED?
6474 012720 001005          BNE      2$           ;BRANCH IF SO
6475 012722 005237 012672          INC      PHEBE          ;INCREMENT COUNTER
6476 012726 062700 000004          ADD      @4,R0         ;INCREMENT CONFIG COUNTER
6477 012732 000763          BR       1$           ;TRY NEXT BANK
6478 012734 023727 012672 000010 2$: CMP      PHEBE,@10     ;IS THE COUNTER EQUAL TO...
6479 012742 001417          BEQ      4$           ;ONE OF THE SPECIAL VALUES.
6480 012744 023727 012672 000030          CMP      PHEBE,@30     ;IF IT IS...
6481 012752 001413          BEQ      4$           ;BRANCH TO 4$
6482 012754 023727 012672 000050          CMP      PHEBE,@50
6483 012762 001407          BEQ      4$
6484 012764 023727 012672 000070          CMP      PHEBE,@70
6485 012772 001403          BEQ      4$
6486 012774 005037 012672          3$: CLR      PHEBE          ;CLEAR INDICATOR
6487 013000 000403          BR       5$
6488 013002 012737 000001 012672 4$: MOV      @1,PHEBE      ;SET INDICATOR
6489 013010 000421          5$: BR       SUBAAP      ;BRANCH TO NEXT SUBTEST
6490 013012 010102          ILLCSR: MOV     R1,R2    ;R2 HAS CSR NUMBER
6491 013014 006202          ASR      R2           ;MAKE ACCEPTABLE FOR PRINTING
6492 013016 022702 000012          CMP      @10.,R2
6493 013022 100002          BPL      1$
6494 013024 062702 000007          ADD      @7,R2
6495 013030 062702 000060          1$: ADD      @60,R2
6496 013034 110237 071260          MOVB     R2,MSG122    ;PUT NUMBER INTO ERROR MESSAGE
6497 013040          TYPE     MSG122
        013040 104401 071244          TYPEIT   .MSG122
        .DSABL CRF
        SET    CONFGERROR
6498 013044          MOV     @1,CONFGERROR
        013044 012737 177777 002450
6499 013052 000207          RETURN

```

6502 013054

SUBAAP: SUBTST <<PRINT CONFIGURATION DETAILS>>

;SUBTEST PRINT CONFIGURATION DETAILS

6503 013054

013054 005037 002400
013060 005037 002402

CLEAR LSIZE,MSIZE

CLR LSIZE
CLR MSIZE

6504 013064

013064 013702 002556

MOV LASTBANK,R2

6505 013070

013070 006302

ASL R2

6506 013072

013072 006302

ASL R2

6507 013074

013074 005001

FOR R1 := #0 TO R2 BY #4

CLR R1

013076

013076

IF CPUBIT SET.IN CONFIG(R1)

B11:;;;;;

6508 013076

013076 033761 002106 002664

BIT CPUBIT.CONFIG(R1)
BEQ L102

013104

013104 001411

IF #BIT8 SET.IN CONFIG+2(R1)

6509 013106

013106 032761 000400 002666

BIT #BIT8.CONFIG+2(R1)
BEQ L103

013114

013114 001403

LET MSIZE := MSIZE + #1

INC MSIZE

6510 013116

013116 005237 002402

ELSE

BR L104

6511 013122

013122 000402

LET LSIZE := LSIZE + #1

L103:;;;;;

6512 013124

013124 005237 002400

END;IF BIT8

INC LSIZE

6513 013130

013130

END; OF IF CPUBIT

L104:;;;;;

6514 013130

013130

END ;OF FOR ALL BANKS IN TABLE

L102:;;;;;

6515 013130

013130 062701 000004

013134 020102

013136 003757

013140

ADD #4,R1
CMP R1,R2
BLE B11

E11:;;;;;

6516 013140

013140 005037 002452

CLR I
FOR R1 := #0 TO #10 BY #2

CLR R1

6517 013144

013144 005001

B12:;;;;;

6518 013146

013146

ASL BSIZE(R1)

6519 013152

013152 006361 002374

ASL BSIZE(R1)

6520 013156

013156 006361 002374

ASL BSIZE(R1)

6521 013162

013162 006361 002374

ASL BSIZE(R1)

;BSIZE(R1) := BSIZE(R1) * 16.
;I <- I + BSIZE(R1)

6522 013166

013166 066137 002374 002452

ADD BSIZE(R1),I

6523 013174

013174 062701 000002

END; FOR R1

ADD #2,R1
CMP R1,#10
BLE B12

6524 013200

013200 020127 000010

013204 003760

013206

E12:;;;;;

6525 013206

013206 005001

FOR R1 := #0 TO #200 BY #4

CLR R1

6526 013210

013210 033761 002106 002664

IF CPUBIT SET.IN CONFIG(R1)

B13:;;;;;

6527 013216

013216 001402

BIT CPUBIT.CONFIG(R1)
BEQ L105

6527 013220

013220

LET UNITOP := UNITOP + #1

```
013220 005237 002416                                INC UNITOP
6528 013224                                END; OF IF CPUBIT                                L105:;;;;;
013224
6529 013224                                END; OF FOR R1
013224 062701 000004                                ADD #4,R1
013230 020127 000200                                CMP R1,#200
013234 003765                                BLE B13
013236                                E13:;;;;;
6530 013236 006337 002416                                ASL UNITOP
6531 013242 006337 002416                                ASL UNITOP
6532 013246 006337 002416                                ASL UNITOP
6533 013252 006337 002416                                ASL UNITOP
6534 013256                                IF I LT UNITOP THEN LET I := UNITOP * 16.
013256 023737 002452 002416                                CMP I,UNITOP
013264 002003                                BGE L106
013266 013737 002416 002452                                MOV UNITOP,I
013274                                L106:;;;;;
6535 013274                                TYPE #CRLF
013274 104401 002660                                TYPEIT ,#CRLF
                                .DSABL CRF
6536 013300 005737 002400                                2#: TST LSIZE
6537 013304 001414                                BEQ 3#
6538 013306 022737 004000 002400                                CMP #2048.,LSIZE
6539 013314 001003                                BNE 12#
6540 013316 162737 000004 002400                                SUB #4,LSIZE ; SUBTRACT 4K FOR THE I/O PAGE
6541 013324                                TYPDEC LSIZE
013324 013746 002400                                MOV LSIZE,-(SP) ;;SAVE LSIZE FOR TYPEOUT
013330 104405                                TYPDS LSIZE,-(SP) ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                .DSABL CRF
6542 013332                                TYPE MSG112
013332 104401 071075                                TYPEIT ,MSG112
                                .DSABL CRF
6543 013336 005737 002402                                3#: TST MSIZE
6544 013342 001414                                BEQ 5#
6545 013344 022737 004000 002402                                CMP #2048.,MSIZE
6546 013352 001003                                BNE 13#
6547 013354 162737 000004 002402                                SUB #4,MSIZE ; SUBTRACT 4K FOR THE I/O PAGE
6548 013362                                TYPDEC MSIZE
013362 013746 002402                                MOV MSIZE,-(SP) ;;SAVE MSIZE FOR TYPEOUT
013366 104405                                TYPDS MSIZE,-(SP) ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                .DSABL CRF
6549 013370                                TYPE MSG113
013370 104401 071127                                TYPEIT ,MSG113
                                .DSABL CRF
```



```
6550 013374 022737 004000 002452 54:    CMP    #2048.,I
6551 013402 001003          BNE    64
6552 013404 162737 000004 002452          SUB    #4,I          ; SUBTRACT 4K FOR THE I/O PAGE
6553 013412          64:    TYPDEC I
      013412 013746 002452    MOV    I,-(SP) ;;SAVE I FOR TYPEOUT
      013416 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
                                .DSABL CRF
6554 013420          TYPE    MSG070
      013420 104401 070177    TYPEIT ,MSG070
                                .DSABL CRF
6555 013424          IF #SW6 OFF.IN @SWR
      013424 032777 000100 167204          BIT #SW6,@SWR
      013432 001002          BNE L107
6556 013434 004737 032610          CALL PCONFIG
6557 013440          END; OF IF #SW6
      013440          L107:::~::~
```

6597 013440

```
LOOP: NEWTST <<DIAGNOSTIC MODE DISPATCH ROUTINE>>
;*****
;*TEST 5 DIAGNOSTIC MODE DISPATCH ROUTINE
;*****
```

013440 000004
6598 013442 005037 002222
6599 013446 017700 167164
6600 013452 042700 177761
6601 013456 004770 013456
6602 013462 000137 013506
6603 013466 014150
6604 013470 014256
6605 013472 014364
6606 013474 014514
6607 013476 014644
6608 013500 014774
6609 013502 015146
6610 013504 015276
6611
6612 013506 C04737 014050
6613
6614 013512

```
TST5: SCOPE
      CLR CONTFLAG
      MOV @SWR,RO ;GET SWITCHES
      BIC @+C16,RO ;MASK TO ONLY MODE BITS
      CALL @DISPTBL(RO) ;DISPATCH TO ROUTINE THROUGH NEXT TABLE
      JMP MEMDONE ;GO TO NEXT TEST
DISPTBL:BAFPAF ;MODE 0;BANKS FORWARD, PATTERNS FORWARD
        BAFPAR ;MODE 1;BANKS FORWARD, PATTERNS REVERSE
        BAWPAF ;MODE 2;BANKS WORST FIRST, PATTERNS FORWARD
        BAWPAR ;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
```

013512 000004
6615
6616
6617 013514
013514 005737 002002
013520 001015
6618 013522
013522 012737 177777 002612
013530 012737 177777 002110
6619 013536 004737 021476
6620 013542
013542 012737 177777 002612
6621 013550 005037 002110
6622 013554
013554
6623 013554 004737 014050
6627
6628 013560

```
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
          SET HEADER,MUT
          TST SELONLY
          BNE L110
          MOV #-1,HEADER
          MOV #-1,MUT
          CALL MTO027
          SET HEADER
          MOV #-1,HEADER
          CLR MUT
          END ;OF IF SELONLY
          CALL DOBACK ;RESTORE BACKGROUND PATTERN
          L110:::~::~
```

6629 013560 004737 022066

```
FLUSH: SUBTST <<FLUSH OUT DBE'S>>
;*****
;*SUBTEST FLUSH OUT DBE'S
;*****
CALL MTO030
```

```

        .SBTTL END OF PASS ROUTINE
;*****
; *INCREMENT THE PASS NUMBER (#PASS)
; *INDICATE END-OF-PROGRAM AFTER EACH PASSES THRU THE PROGRAM
; *TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
; *IF THERES A MONITOR GO TO IT
; *IF THERE ISN'T JUMP TO LOOP
6632                                     $EOP: CLR     FSINFLAG
6633                                     MOV     #CONFIG+2,RO ;MOVE 2ND WORD OF CONFIG TO RO
6634                                     1$:   BIC     #BIT13,(RO) ;CLEAR BACKGROUND VALID BIT
6635                                     ADD     #4,RO ;INCREMENT TO NEXT BANK
6636                                     CMP     RO,#3620 ;DONE?
6637                                     BLE     1$ ;NO - BRANCH
6638                                     MOV     $ERTTL,LASTERROR
6639 013564 005037 002442                                     INC     $PASS ;INCREMENT THE PASS NUMBER
6640 013570 012700 002666                                     BIC     #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
6641 013574 042710 020000                                     TYPE    MSG077 ;TYPE "END PASS #"
6642 013600 062700 000004                                     TYPEIT ,MSG077
6643 013604 020027 603620                                     .DSABL CRF
6644 013610 003771                                     IF $SW11 SET.IN @SWR OR QVFLAG IS TRUE OR $PASS EQ #1
6645 013612 013737 002630 002016                                     BIT     #SW11,@SWR
6646 013620 005237 056724                                     BNE    L111
6647 013624 042737 100000 056724                                     TST    QVFLAG
6648 013632                                     BNE    L111
        013632 104401 070321                                     CMP     $PASS,#1
        013636 032777 004000 166772                                     BNE    L112
        013644 001007                                     L111:;;;;;
        013646 005737 002346                                     L112:;;;;;
        013652 001004
        013654 023727 056724 000001
        013662 001004
        013664                                     TYPE    MSG035 ;QV
        013664 104401 067343                                     TYPEIT ,MSG035
        013670 005037 002346                                     .DSABL CRF
        013674                                     CLR     QVFLAG
        013674                                     END ;OF IF SW11
        013674                                     TYPDEC $PASS
        013674 013746 056724                                     MOV     $PASS,-(SP) ;SAVE $PASS FOR TYPEOUT
        013700 104405                                     TYPDS  ;GO TYPE--DECIMAL ASCII WITH SIGN
        013702 013700 000042                                     .DSABL CRF
        013706 001456                                     MOV     42,RO ;GET MONITOR ADDRESS
        013710 022700 002000 $ZAP42: BEQ     $DOAGAIN ;BRANCH IF NO MONITOR
        013714 001453                                     CMP     #STACK,RO ;ARE WE UNDER RT11
        013716                                     BEQ     $DOAGAIN ;YES - BRANCH
        013716                                     ;WE ARE UNDER (HEAVEN HELP US) XXDP!
        013716 010046                                     PUSH    RO
        013720 004737 040544                                     CALL    SHUTUP ;MOV RO,-(SP)
        013724 012600                                     POP     RO ;MOV (SP)+,RO
        013726 000005                                     RESET
        013730 004710 $ENDAD: CALL    (RO) ;CLEAR THE WORLD
        013732 000240                                     ;GO TO MONITOR
        013734 000240                                     NOP     ;SAVE ROOM
        013736 000240                                     NOP     ;FOR
        013740 $DOAGN: ;UNDO SHUTUP STUFF ;ACT11
        ; RESTORE STACK
        ; ENERGIZE Q-BUS MAP & 22 BIT ADDRESSING
        ; ENERGIZE MEMORY MANAGEMENT
        ; PUT LOADERS BACK HOME
    
```

```
6672 013740 013706 002574      MOV      KSTACK,SP
6673 013744 005737 002454      TST      NO22BIT      ;IS THIS AN 11/83.11/23-B OR 11/23?
6674 013750 001003              BNE      1#
6675 013752 052737 000060 172516 1#:  BIS      #BITS:BIT4,MMR3
6676 013760 104420              ENERGIZE      ;TURN ON MEMORY MANAGEMENT
6677 013762 013700 002576      MOV      LOADHOME,RO  ;DESTINATION BANK
6678 013766 012701 000001      MOV      #1,R1       ;SOURCE BANK
6679 013772 004737 037350      CALL     BANKMOV
6680 013776              IF APTFLAG IS TRUE
        013776 005737 002352              TST APTFLAG
        014002 001420              BEQ L113
6681 014004              IF #USWR EQ #PASS
        014004 023737 056742 056724      CMP #USWR,#PASS
        014012 001014              BNE L114
6682 014014 012701 000050      APTHANG:  MOV #50,R1
6683 014020 077001 2#:      SOB RO,2#
6684 014022 062737 000001 056726      ADD #1,#DEVCT
6685 014030 005537 056730      ADC #UNIT
6686 014034 077107      SOB R1,2#
6687 014036 C05237 056724      INC #PASS
6688 014042 000764      BR APTHANG
6689 014044      END ;OF IF #USWR
        014044              L114:#####
6690 014044      END ;OF IF APTFLAG
        014044              L113:#####
6691 014044 000137 013440      #DOAGAIN: JMP LOOP      ;RETURN
```

6694 014050

DOBACK: SUBTST <<WRITE BACKGROUND PATTERNS>>
;*****
;SUBTEST WRITE BACKGROUND PATTERNS
;*****

6695 014050 005037 002112

6696 014054 005037 002102

014060

6697 014060 004737 037760

6698 014064 005737 002116

014070 001420

014072 005737 002124

014076 001015

6699 014100

014100 012737 177777 002612

014106 012737 177777 002110

6700 014114 004737 016772

6701 014120 005037 002110

6702 014124 012737 177777 002612

6703 014132

014132

6704 014132

014132 005237 002102

014136 023737 002102 002556

014144 003745

014146

6705 014146 000207

CLR PATTERN
FOR BANK := #0 TO LASTBANK

CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE

SET HEADER,MUT

CALL MKTEST ;CALL MJTEST WOULD ALSO WORK
CLR MUT
SET HEADER

END ;OF IF ACFLAG

END ;OF FOR BANK

RETURN

CLR BANK
B14:*****

TST ACFLAG
BEQ L115
TST RRFLAG
BNE L115

MOV #-1,HEADER
MOV #-1,MUT

MOV #-1,HEADER

L115:*****

INC BANK
CMP BANK, LASTBANK
BLE B14

E14:*****

```

6708
6709
6710 014150

6711 014150 005037 002102
6712
6713 014154 004737 037760
6714 014160 005737 002116
6715 014164 001412
6716 014166 005737 002124
6717 014172 001007
6718 014174 005037 002112
6719
6720 014200 004737 015450
6721
6722 014204 004737 040402
6723 014210 001373
6724
6725 014212 005037 002222
6726 014216 004737 040426
6727 014222 002354
6728
6729 014224 005737 002126
6730 014230 001401
6731 014232 000207
6732 014234 004737 036310
6733 014240
        014240 103001
        014242 000207
        014244

6734
6735 014244 004737 014150
6736 014250 004737 037122
6737 014254 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$: CLR CONTFLAG
        CALL INCBNK ;NEXT HIGHER BANK
        BGE 1$ ;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 L116
RTS PC
L116:::
    
```

6740 014256

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

6741 014256 005037 002102
6742
6743 014262 004737 037760
6744 014266 005737 002116
6745 014272 001412
6746 014274 005737 002124
6747 014300 001007
6748 014302 004737 040416
6749
6750 014306 004737 015450
6751
6752 014312 005337 002112
6753 014316 100373
6754
6755 014320 005037 002222
6756 014324 004737 040426
6757 014330 002354
6758
6759 014332 005737 002126
6760 014336 001401
6761 014340 000207
6762 014342 004737 036310
6763 J14346
014346 103001
014350 000207
014352
6764
6765 014352 004737 014256
6766 014356 004737 037122
6767 014362 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 L117
RTS PC
L117:::~::~

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

6770 014364

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

```
6771 014364 005037 002102
6772
6773 014370 004737 037760
6774 014374 005737 002116
6775 014400 001415
6776 014402 005737 002130
6777 014406 001412
6778 014410 005737 002124
6779 014414 001007
6780 014416 005037 002112
6781
6782 014422 004737 015450
6783
6784 014426 004737 040402
6785 014432 001373
6786
6787 014434 005037 002222
6788 014440 004737 040426
6789 014444 002351
6790
6791 014446 005137 002600
6792 014452 001003
6793
6794 014454 004737 014364
6795 014460 000207
6796 014462 005737 002126
6797 014466 001401
6798 014470 000207
6799 014472 004737 036310
6800 014476
      014476 103001
      014500 000207
      014502
6801
6802 014502 004737 014364
6803 014506 004737 037122
6804 014512 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
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
COM WORST ;IS THIS AN EVEN NUMBERED PASS?
BNE 5$ ;YES - SKIP
;**NOTE** RECURSIVE CALL
CALL BAWPAF ;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 BAWPAF ;CALL SELF
CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
RETURN
```

```
BCC L120
RTS PC
L120:::~::~
```


6807 014514

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

```
6808 014514 005037 002102
6809
6810 014520 004737 037760
6811 014524 005737 002116
6812 014530 001415
6813 014532 005737 002130
6814 014536 001412
6815 014540 005737 002124
6816 014544 001007
6817 014546 004737 040416
6818
6819 014552 004737 015450
6820
6821 014556 005337 002112
6822 014562 100373
6823
6824 014564 005037 002222
6825 014570 004737 040426
6826 014574 002351
6827
6828 014576 005137 002600
6829 014602 001003
6830
6831 014604 004737 014514
6832 014610 000207
6833 014612 005737 002126
6834 014616 001401
6835 014620 000207
6836 014622 004737 036310
6837 014626
        014626 103001
        014630 000207
        014632
6838
6839 014632 004737 014514
6840 014636 004737 037122
6841 014642 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$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
BEQ 6$ ;NO - SKIP
RETURN ;YES - RETURN
6$: CALL RELOCATE ;MOVE & MAP PROGRAM
ON.ERROR THEN $RETURN
```

```
BCC L121
RTS PC
L121:::~::~
```

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

```

6844 014644 PAFBAF: SUBTST <<PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**>>
;*****
;SUBTEST PATTERNS FORWARD,BANKS FORWARD **RECURSIVE**
;*****
6845 014644 005037 002112 CLR PATTERN ;SET PATTERN TO 0
6846 ;START OF PATTERN LOOP
6847 014650 005037 002102 1$: CLR BANK ;SET BANK TO 0
6848 ;START OF BANK LOOP
6849 014654 004737 037760 2$: CALL EXBANK ;EXAMINE BANK
6850 014660 004737 040364 CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?
6851 014664 001010 BNE 4$ ;NO - GO TO BANK LOOP TERMINATOR
6852 014666 005737 002116 TST ALFLAG ;CAN WE ACCESS THIS BANK?
6853 014672 001405 BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
6854 014674 005737 002124 TST RRFLAG ;RELOCATION REQUIRED?
6855 014700 001002 BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
6856 014702 004737 015450 CALL MTEST ;GO TEST CORRECT MEMORY
6857 ;TERMINATION OF BANK LOOP
6858 014706 005037 002222 4$: CLR CONTFLAG
6859 014712 004737 040426 CALL INCBNK ;NEXT HIGHER BANK
6860 014716 C02356 BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
6861 ;TERMINATION OF PATTERN LOOP
6862 014720 004737 040402 CALL INCRPT ;NEXT HIGHER PATTERN
6863 014724 001351 BNE 1$ ;OK - LOOP; ELSE CONTINUE
6864 ;END OF LOOPS
6865 014726 005137 002134 COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
6866 BEQ 5$ ;IS THIS AN EVEN NUMBER PASS?
6867 014732 001403 ;YES - SKIP
6868 ;**NOTE** RECURSIVE CALL
6869 014734 004737 014644 CALL PAFBAF ;CALL SELF
6870 014740 00C207 RETURN
6871 014742 005737 002126 5$: TST RLFLAG ;HAVE WE BEEN RELOCATED?
6872 014746 0014C1 BEQ 6$ ;NO - SKIP
6873 014750 000207 RETURN ;YES - RETURN
6874 014752 004737 036310 6$: CALL RELOCATE ;MOVE & MAP PROGRAM
6875 014756 ON.ERROR THEN $RETURN
014756 103001
014760 000207
014762
;*****
6876 ;**NOTE** RECURSIVE CALL
6877 014762 004737 014644 CALL PAFBAF ;CALL SELF
6878 014766 004737 037122 CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM
6879 014772 000207 RETURN
    
```

BCC L122
 RTS PC
 L122:::~::~:

6882 014774

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

6883 014774 005037 002112
 6884
 6885 015000 005037 002102
 6886
 6887 015004 004737 037760
 6888 015010 004737 040364
 6889 015014 001013
 6890 015016 005737 002116
 6891 015022 001410
 6892 015024 005737 002130
 6893 015030 001405
 6894 015032 005737 002124
 6895 015036 001002
 6896 015040 004737 015450
 6897
 6898 015044 005037 002222
 6899 015050 004737 040426
 6900 015054 002353
 6901
 6902 015056 004737 040402
 6903 015062 001346
 6904
 6905 015064 005137 002134
 6906
 6907 015070 001403
 6908
 6909 015072 004737 014774
 6910 015076 000207
 6911 015100 005137 002600
 6912 015104 001003
 6913
 6914 015106 004737 014774
 6915 015112 000207
 6916 015114 005737 002126
 6917 015120 001401
 6918 015122 000207
 6919 015124 004737 036310
 6920 015130
 015130 103001
 015132 000207
 015134
 6921
 6922 015134 004737 014774
 6923 015140 004737 037122
 6924 015144 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 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$: CLR CONTFLAG
CALL JNCBANK ;NEXT HIGHER BANK
BGE 2$ ;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
BEQ 5$ ;IS THIS AN EVEN NUMBER PASS?
;YES - SKIP
;**NOTE** RECURSIVE CALL
CALL PAFBAW ;CALL SELF
RETURN
5$: COM WORST ;4TH PASS?
BNE 6$ ;YES - SKIP
;**NOTE** RECURSIVE CALL
CALL PAFBAW ;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
```

BCC L123
 RTS PC
 L123:::~::~

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

```

6927 015146      PARBAF. SUBTST <<PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**>>
;*****
;SUBTEST      PATTERNS REVERSE,BANKS FORWARD      **RECURSIVE**
;*****
6928 015146 004737 040416      CALL      HIPAT      ;SET HIGHEST PATTERNS
6929      ;START OF PATTERN LOOP
6930 015152 005037 002102 1$:      CLR      BANK      ;SET BANK TO 0
6931      ;START OF BANK LOOP
6932 015156 004737 037760 2$:      CALL      EXBANK      ;EXAMINE BANK
6933 015162 004737 040364      CALL      BANKOK      ;CORRECT MEMORY FOR THIS BANK?
6934 015166 001010      BNE      4$      ;NO - GO TO BANK LOOP TERMINATOR
6935 015170 005737 002116      TST      ACFLAG      ;CAN WE ACCESS THIS BANK?
6936 015174 001405      BEQ      4$      ;NO - GO TO BANK LOOP TERMINATION
6937 015176 005737 002124      TST      RRFLAG      ;RELOCATION REQUIRED?
6938 015202 001002      BNE      4$      ;YES - GO TO BANK LOOP TERMINATION
6939 015204 004737 015450      CALL      MTEST      ;GO TEST CORRECT MEMORY
6940      ;TERMINATION OF BANK LOOP
6941 015210 005037 002222 4$:      CLR      CONTFLAG
6942 015214 004737 040426      CALL      INCBNK      ;NEXT HIGHER BANK
6943 015220 002356      BGE      2$      ;IF NOT DONE - LOOP ON THIS BANK
6944      ;TERMINATION OF PATTERN LOOP
6945 015222 005337 002112      DEC      PATTERN      ;NEXT LOWER PATTERN
6946 015226 100351      BPL      1$      ;OK - LOOP; ELSE CONTINUE
6947      ;END OF LOOPS
6948 015230 005137 002134      COM      TMFLAG      ;COMPLEMENT TYPE OF MEMORY
6949      ;IS THIS AN EVEN NUMBER PASS?
6950 015234 001403      BEQ      5$      ;YES - SKIP
6951      ;**NOTE** RECURSIVE CALL
6952 015236 004737 015146      CALL      PARBAF      ;CALL SELF
6953 015242 000207      RETURN
6954 015244 005737 002126 5$:      TST      RLFLAG      ;HAVE WE BEEN RELOCATED?
6955 015250 001401      BEQ      6$      ;NO - SKIP
6956 015252 000207      RETURN      ;YES - RETURN
6957 015254 004737 036310 6$:      CALL      RELOCATE      ;MOVE & MAP PROGRAM
6958 015260      ON.ERROR THEN $RETURN
        015260 103001
        015262 000207
        015264
        BCC L124
        RTS PC
        L124:::;
6959      ;**NOTE** RECURSIVE CALL
6960 015264 004737 015146      CALL      PARBAF      ;CALL SELF
6961 015270 004737 037122      CALL      UNRELOCATE      ;UNMOVE & UNMAP PROGRAM
6962 015274 000207      RETURN
    
```

6965 015276

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

6966 015276 004737 040416
 6967
 6968 015302 005037 002102
 6969
 6970 015306 004737 037760
 6971 015312 004737 040364
 6972 015316 001013
 6973 015320 005737 002116
 6974 015324 001410
 6975 015326 005737 002130
 6976 015332 001405
 6977 015334 005737 002124
 6978 015340 001002
 6979 015342 004737 015450
 6980
 6981 015346 005037 002222
 6982 015352 004737 040426
 6983 015356 002353
 6984
 6985 015360 005337 002112
 6986 015364 100346
 6987
 6988 015366 005137 002134
 6989
 6990 015372 001403
 6991
 6992 015374 004737 015276
 6993 015400 000207
 6994 015402 005137 002600
 6995 015406 001003
 6996
 6997 015410 004737 015276
 6998 015414 000207
 6999 015416 005737 002126
 7000 015422 001401
 7001 015424 000207
 7002 015426 004737 036310
 7003 015432
 015432 103001
 015434 000207
 015436
 7004
 7005 015436 004737 015276
 7006 015442 004737 037122
 7007 015446 000207

```
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 L125
 RTS PC
 L125:.....

```

7010 015450      MTEST: SUBST <<SUBR SETUP MEMORY TEST>>
;*****
;SUBTEST      SUBR  SETUP MEMORY TEST
;*****
7011 015450      SET    HEADER      ;INITIALIZE HEADER MESSAGE TYPEOUT
015450 012737 177777 002612      MOV    #-1,HEADER
7012 015456      SET    MUT          ;INDICATE THERE IS A MEMORY UNDER TEST
015456 012737 177777 002110      MOV    #-1,MUT
7013 015464      CLR    PASFLG
7014 015470      TST    MKFLAG      ;ECC?
7015 015474      BEQ    MT1          ;NO - SKIP
001413
7016 015476      BEGIN   HOLDLOOP
015476
7017 015476      IF CONTFLAG IS TRUE THEN LEAVE HOLDLOOP      B15:;;;;;
015476 005737 002222      TST CONTFLAG
015502 001005      BNE E15
7018 015504      IF SKIPMK IS FALSE
015504 005737 002342      TST SKIPMK
015510 001002      BNE L126
7019 015512      CALL    MKCONTROL
015516      END; OF IF SKIPMK
7020 015516
015516
7021 015516      END    HOLDLOOP      L126:;;;;;
015516
7022 015516      CALL    MKTEST      E15:;;;;;
004737 016772      BR     MT2          ;YES - DO ECC TESTS
7023 015522      CALL    MJTEST
000402
7024 015524      CLR    MUT          ;DO PARITY TESTS
004737 017212      SET    HEADER      ;NOW - NO MEMORY UNDER TEST
7025 015530      SET    MUT          ;ALLOW HEADERS NORMAL
005037 002110
7026 015534      MOV    #-1,HEADER
015534 012737 177777 002612
7027 015542      RETURN
000207
    
```

```

7030 015544 MKCONTROL:SUBTST      <<SUBR TEST ECC CSR LOGIC DISPATCH>>
;*****
;SUBTEST      SUBR      TEST ECC CSR LOGIC DISPATCH
;*****
7031          ;THE NEXT TWO MODULES SOLVE THE PROBLEM OF
7032          ;HOW TO RUN THE CSR TESTS ON EACH ECC MEMORY
7033          ;
7034 015544          IF SELONLY IS TRUE THEN $RETURN
          015544 005737 002002          TST SELONLY
          015550 001401          BEQ L127
          015552 000207          RTS PC
          015554          L127:;;;;;;
7035 015554          IF INHECC IS TRUE THEN $RETURN
          015554 005737 002536          TST INHECC
          015560 001401          BEQ L130
          015562 000207          RTS PC
          015564          L130:;;;;;;
7036 015564          PUSH      BANK,R0,R1,R2,R3
          015564 013746 002102          MOV BANK,-(SP)
          015570 C10046          MOV R0,-(SP)
          015572 010146          MOV R1,-(SP)
          015574 010246          MOV R2,-(SP)
          015576 010346          MOV R3,-(SP)
7037 015600 012737 060000 002232      MOV      $FIRST,CSRFBANK      ;SET FIRST TEST ADDRESS TO FIRST ADDR.
7038 015606 012737 157776 002234      MOV      $LAST,CSRLBANK
7039 015614 005037 002236          CLR      CSRINT
7040 015620 005037 002240          CLR      SPLTCSR
7041 015624 005037 002332          CLR      CSRLOOP
7042 015630 013700 002104          MOV      BANKINDEX,R0      ;AND ZERO THE LOOP COUNTER
7043 015634 016001 002664          MOV      CONFIG(R0),R1      ;GET THE BANK INDEX
7044 015640 000301          SWAB     R1      ;GET CSR NUMBER
7045 015642 042701 177760          BIC      $C17,R1
7046 015646 006301          ASL     R1
7047 015650 010137 002526          MOV      R1,CSRHOLD      ;STORE IN THE LOW BYTE
7048 015654 005737 002136          TST     INTFLAG      ;IS THIS BANK INTERLEAVED?
7049 015660 001421          BEQ     1$      ;BRANCH IF NOT INTERLEAVED
7050 015662 005237 002240          INC     SPLTCSR
7051 015666 012737 120000 002234      MOV      $120000,CSRLBANK
7052 015674 005237 002332          INC     CSRLOOP      ;WE MUST LOOP TWICE FOR AN INTERLEAVED BANK
7053 015700 005237 002236          INC     CSRINT
7054 C15704 016001 002664          MOV      CONFIG(R0),R1      ;GET THE INTERLEAVE CSR NUMBER
7055 015710 072127 177775          ASH     #-3,R1
7056 015714 042701 160777          BIC      $C17000,R1
7057 015720 050137 002526          BIS     R1,CSRHOLD      ;STORE IT IN CSRHOLD'S UPPER BYTE
7058 015724 005003          CLR     R3
7059 015726 116337 002526 002152      1$:      MKLOOP: MOVB     CSRHOLD(R3),CSRNO
7060 015734 042737 177741 002152      BIC     $C36,CSRNO      ;CLEAR ANY UNNECESSARY BITS
7061 015742          FOR MKCNT := #0 TO CSRINT
          015742 005037 016364          CLR     MKCNT
          015746          B16:;;;;;;
7062 015746          FOP CSRFIRST := CSRFBANK TO CSRLBANK BY #4000
          015746 013737 002232 002226          MOV     CSRFBANK,CSRFIRST
          015754          B17:;;;;;;
7063 015754          MAP BANK      ;MAP TEST SPACE TO BANK
          015754 010346          MOV     BANK,R3
          015756 013703 002102          CALL    MAPPER
          015762 004737 035604

```

```

.DSABL CRF
7064 015766 012603
7065 015770 104511
7066 015772 005037 002334
7067 015776 005737 002116
015776 005737 002116
016002 001503
016004 005737 002124
016010 001100
7068 016012 013737 002226 002230
7069 016020 062737 004000 002230
7070 016026 013737 002226 002412
016034
7071 016034 013737 002412 002414
7072 016042 005737 002240
7073 016046 001404
7074 016050 062737 040000 002414
7075 016056 000403
7076 016060 062737 000002 002414
7077 016066 004737 016366
7078 016072
016072 103440
7079 016074 104424
7080 016076 005037 002076
7081 016102
016102 005037 002452
016106
7082 016106
016106 012737 177777 002612
7083 016114 005037 002264
7084 016120
016120 013700 002452
7085 016124
016124 010346
7086 016126 010637 002146
7087 016132 162737 000002 002146
7088 016140 004737 016672
7089 016144
016144 012603
7090 016146
016146 005237 002452
016152 023727 002452 000027
016160 003752
016162
7091 016162 104423
7092 015164
016164 012737 177777 002334
7093 016172
016172 000407
7094 016174
016174
7095 016174
016174 062737 000004 002412
016202 023737 002412 002230

.INVALIDATE ;INVALIDATE BACKGROUND PATTERN
BEGIN CSRSTUFF
CLR SUCCESS
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
MOV CSRFIRST,CSRLAST
ADD #4000,CSRLAST
FOR TESTADD := CSRFIRST TO CSRLAST BY #4
MOV CSRFIRST,TESTADD
B20:;;;;;
MOV TESTADD,TESTADD+2
TST SPLTCR
BEQ 1#
ADD #40000,TESTADD+2
BR 2#
1#: ADD #2,TESTADD+2
2#: CALL SBTEST
ON.NOERROR
CACHOFF ;TURN CACHE OFF
CLR NOPAR ;INDICATE PARITY ACTION
FOR I := #0 TO #27
CLR I
B22:;;;;;
SET HEADER
MOV #1,HEADER
CLR PASFLG
LET R0 := I
PUSH R3 ;SAVE LOOP COUNTER
MOV SP,CTLKVEC ;SAVE VECTOR IN CSR OF *K
SUB #2,CTLKVEC
CALL CSRCASE
POP R3 ;RESTORE LOOP COUNTER
MOV (SP),R3
END ;OF FOR I
INC I
CMP I,#27
BLE B22
E22:;;;;;
CACHON ;TURN CACHE ON
SET SUCCESS
MOV #-1,SUCCESS
LEAVE CSRSTUFF
BR E20
END ;OF ON.NOERROR
L132:;;;;;
END ;OF FOR TESTADD
ADD #4,TESTADD
CMP TESTADD,CSRLAST
    
```


7121 016366

```
SBETEST:SUBTST <<CHECK FOR SBE FREE LOCATIONS>>
;*****
;SUBTEST CHECK FOR SBE FREE LOCATIONS
;*****
```

7122
7123
7124
7125
7126
7127
7128
7129
7130
7131
7132
7133
7134
7135
7136
7137
7138
7139
7140
7141
7142
7143
7144
7145
7146
7147
7148
7149
7150
7151
7152
7153
7154
7155
7156
7157
7158
7159
7160
7161
7162
7163
7164
7165

016366 010046
016366 010146
016370 010446
016372 010446
016374 013701 002412
016400 013704 002414
016404 053737 002552 177776
016412 104424
016414 104471
016416 005011
016420 005014
016422 005711
016424 001107
016426 005714
016430 001105
016432 104503
016434 005011
016436 005014
016440 005711
016442 001100
016444 005714
016446 001076
016450 104510

```
; 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)
MOV TESTADD,R1
MOV TESTADD+2,R4
TESTAREA ;ENTER TEST MODE
BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
CACHOFF ;TURN CACHE OFF
ECC1DIS ;DISABLE ECC ON 1 SELECTED CSR
CLEAR (R1),(R4) CLR (R1)
CLR (R4)
TST (R1)
BNE SBENT
TST 4)
BNE .ENT
CLR1CSR ;CLEAR 1 SELECTED CSR
CLEAR (R1),(R4) CLR (R1)
CLR (R4)
TST (R1)
BNE SBENT
TST (R4)
BNE SBENT
TSTREAD ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
```

```

7166 016452          IF #BIT15:BIT4 SET.IN CSR
      016452          032737 100020 002150          BIT #BIT15:BIT4,CSR
      016460          001415          BEQ L134
7167 016462          SET SKPERR          ;DISABLE ERRGEN'S ERROR PRINTOUT
      016462          012737 177777 002066          MOV #-1,SKPERR
7168 016470          ERRGEN
7169 016472          MOV ERRADD,RO
7170 016476          ASH #-4,RO
7171 016502          BIC #+C177,RO
7172 016506          IF BANK EQ RO THEN GOTO SBENT
      016506          023700 002102          CMP BANK,RO
      016512          001454          BEQ SBENT
7173 016514          END; OF IF #BIT15
      016514          L134:;;;;;
7174 016514          104471          ECC1DIS          ;DISABLE ECC ON 1 SELECTED CSR
7175 016516          005111          COM (R1)
7176 016520          005114          COM (R4)
7177 016522          023711 002614          CMP ONES,(R1)
7178 016526          001046          BNE SBENT
7179 016530          C23714 002614          CMP ONES,(R4)
7180 016534          001043          BNE SBENT
7181
7182 016536          104503          CLR1CSR          ;CLEAR 1 SELECTED CSR
7183 016540          005011          CLR (R1)
7184 016542          012714 100000          MOV #BIT15,(R4)
7185 016546          005711          TST (R1)
7186 016550          001035          BNE SBENT
7187 016552          022714 100000          CMP #BIT15,(R4)
7188 016556          001032          BNE SBENT
7189
7190 016560          104510          TSTREAD          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
7191 016562          IF #BIT15:BIT4 SET.IN CSR
      016562          032737 100020 002150          BIT #BIT15:BIT4,CSR
      016570          001415          BEQ L135
7192 016572          SET SKPERR          ;DISABLE ERRGEN'S FRORR PRINTOUT
      016572          012737 177777 002066          MOV #-1,SKPERR
7193 016600          ERRGEN
7194 016602          MOV ERRADD,RO
7195 016606          ASH #-4,RO
7196 016612          BIC #+C177,RO
7197 016616          IF BANK EQ RO THEN GOTC SBENT
      016616          023700 002102          CMP BANK,RO
      016622          001410          BEQ SBENT
7198 016624          END; OF IF #BIT15
      016624          L135:;;;;;
7199
7200 016624          104417          KERNEL          ;ENTER KERNEL MODE
7201 016626          104473          ECC1INIT          ;INITIALIZE 1 SELECTED CSR
7202 016630          104423          CACHON          ;TURN CACHE ON
7203 016632          POP R4,R1,RO          ;POP RO,R1 & R4 FROM STACK
      016632          012604          MOV (SP),R4
      016634          012601          MOV (SP),R1
      016636          012600          MOV (SP),RO
7204 016640          $RETURN NOERROR
      016640          000241          CLC
      016642          000207          RTS PC
7205

```

7206	016644	104503			
7207	016646		SBENT:	CLR1CSR	:CLEAR 1 SELECTED CSR
	016646	005011		CLEAR (R1),(R4)	
	016650	005014			CLR (R1)
7208	016652	104417			CLR (R4)
7209	016654	104473		KERNEL	:ENTER KERNEL MODE
7210	016656	104423		ECC1INIT	:INITIALIZE 1 SELECTED CSR
7211	016660			CACHON	:TURN CACHE ON
	016660	012604		POP R4,R1,R0	:POP R0,R1 & R4 FROM STACK
	016662	012601			MOV (SP),R4
	016664	012600			MOV (SP),R1
7212	016666				MOV (SP),R0
	016666	000261		\$RETURN ERROR	
	016670	000207			SEC
7213				.DSABL LSB	RTS PC

7216 016672

```
CSRCASE:SUBST <<CSR PATTERN CASE STATEMENT>>
;*****
;SUBTEST CSR PATTERN CASE STATEMENT
;*****
CASE R0
```

7217 016672
016672 010046
016674 006316
016676 004737 016662

```
MOV R0, -(SP)
ASL @SP
JSR PC,L136
```

7218
7219
7220
7221
7222
7223
7224
7225
7226

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

7227 016704 023712
7228 016706 020242
7229 016710 023772
7230 016712 023446
7231 016714 020344
7232 016716 023510
7233 016720 023546
7234 016722 023626
7235 016724 023662
7236 016726 024022
7237 016730 024052
7238 016732 020206
7239 016734 024106
7240 016736 024106
7241 016740 024106
7242 016742 024106
7243 016744 024106
7244 016746 024106
7245 016750 024106
7246 016752 024106
7247 016754 024106
7248 016756 024106
7249 016760 024106

```
MKCSRT: ;PAT TIME DESCRIPTION
; MT0006 ;<1 SEC 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
; MT0999 : 0 SEC NULL TEST
; MT0999 : 0 SEC NULL TEST
; MT0999 : 0 SEC NULL TEST
```

END ;OF CASE R0

7250 016762
016762 062616
016764 013646
016766 004736
7251 016770 000207

```
L136:::
ADD (SP), @SP
MOV @ (SP), -(SP)
JSR PC, @ (SP)
```

RETURN

```

7254 016772 MKTEST: SUBTST <<SUBR ECC TEST DISPATCH>>
;*****
;*SUBTEST SUBR ECC TEST DISPATCH
;*****
IF #SWO SET.IN #SWR OR ACTFLAG IS TRUE
7255 016772 032777 000001 163636 BIT #SWO,#SWR
016772 001003 BNE L137
017000 005737 002350 TST ACTFLAG
017006 001402 BEQ L140
017010 L137:*****
7256 017010 104470 ECCDIS ;DISABLE ERROR CORRECTION
7257 017012 ELSE BR L141
017012 000401 L140:*****
7258 017014 104502 CLRCSR ;CLEAR ALL CSR'S
7259 017016 END ;OF IF L141:*****
7260 017016 012737 000002 002076 MOV #2,NOPAR ;INDICATE PARITY ACTION
7261 017024 012737 000002 002326 MOV #2,PCBUMP ;TRAPS ADD 2 TO PC
7262 017032 C13700 002112 MOV PATTERN,RO ;GET PATTERN NUMBER
7263 017036 006300 ASL RO ;MAKE IT A WORD ADDRESS
7264 017040 IF MKPAT(RO) NE #MT0034 AND MKPAT(RO) NE #MT0999
017040 026027 017132 023234 CMP MKPAT(RO),#MT0034
017046 001405 BEQ L142
017050 026027 017132 024106 CMP MKPAT(RO),#MT0999
017056 001401 BEQ L142
7265 017060 104511 INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON "BANK"
7266 017062 END ;OF IF MKPAT(RO)
017062 L142:*****
7267 017062 010637 002146 MOV SP,CTLKVEC ;SAVE VECTOR IN CASE OF *K
7268 017066 162737 000002 002146 SUB #2,CTLKVEC
7269 017074 004770 017132 CALL #MKPAT(RO) ;INDEX OFF TABLE
7270 017100 IF #SWO SET.IN #SWR OR ACTFLAG IS TRUE
017100 032777 000001 163530 BIT #SWO,#SWR
017106 001003 BNE L143
017110 005737 002350 TST ACTFLAG
017114 001402 BEQ L144
017116 L143:*****
7271 017116 104506 ENASBE ;TRAP ON SINGLE BIT ERRORS
7272 017120 ELSE BR L145
017120 000401 L144:*****
017122 104472 ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7273 017122 END ;OF IF #SWO
7274 017124 L145:*****
017124 005037 002076 CLR NOPAR ;INDICATE PARITY ACTION
7276 017130 000207 RETURN
7277
7278 ;WARNING IF YOU CHANGE THIS TABLE ALSO
7279 ;CHANGE "#DDW0" - "#DDW5" (THE PATTERN BIT MAP)
7280 ;PAT TIME DISCIPTION
MKPAT: ;NOTE MT0034 MUST BE FIRST & LAST
7281 017132 MT0034 ;<1 SEC ;SOFT ERROR - BACKGROUND PATTERN TEST
7282 017132 023234 MT0017 ;<1 SEC ;HOLDING 1'S & 0'S TEST
7283 017134 020322 MT0007 ;<1 SEC ;ADDRESS BIT TEST
7284 017136 020144 MT0001 ;<1 SEC ;ADDRESS TEST
7285 017140 017432 MT0002 ;<1 SEC ;COMPLEMENT ADDRESS TEST
7286 017142 017526

```

7287	017144	017756	MT0004	: 1 SEC	;ROTATING ZEROS TEST
7288	017146	020026	MT0005	: 1 SEC	;ROTATING ONES TEST
7289	017150	020424	MT0021	: 1 SEC	;MARCHING 0'S & 1'S TEST
7290	017152	020714	MT0022	:10 SEC	;REFRESH & SHIFTING DIAGONAL TEST
7291	017154	021220	MT0026	:<1 SEC	;RANDOM DATA TEST
7292	017156	021012	MT0024	:20 SEC	;FAST GALLOPING PATTERN TEST
7293	017160	022342	MT0031	: 3 SEC	;SOB-A-LONG TEST
7294	017162	022532	MT0032	:<1 SEC	;WRITE RECOVERY TEST
7295	017164	023046	MT0033	:35 SEC	;BRANCH GOBBLE TEST
7296	017166	023234	MT0034	:<1 SEC	;SOFT ERROR - BACKGROUND PATTERN TEST
7297			;NOTE MT0034 MUST BE FIRST & LAST		
7298	017170	024106	MT0999	: 0 SEC	;NULL TEST
7299	017172	024106	MT0999	: 0 SEC	;NULL TEST
7300	017174	024106	MT0999	: 0 SEC	;NULL TEST
7301	017176	024106	MT0999	: 0 SEC	;NULL TEST
7302	017200	024106	MT0999	: 0 SEC	;NULL TEST
7303	017202	024106	MT0999	: 0 SEC	;NULL TEST
7304	017204	024106	MT0999	: 0 SEC	;NULL TEST
7305	017206	024106	MT0999	: 0 SEC	;NULL TEST
7306	017210	024106	MT0999	: 0 SEC	;NULL TEST

7309 017212

MJTEST: SUBTST <<SUBR PARITY TEST DISPATCH>>

;SUBTEST SUBR PARITY TEST DISPATCH

7310 017212 012737 000002 002076
7311 017220 012737 000002 002326
7312 017226 012737 060000 002412
7313 017234 012737 060002 002414
7314 017242 013700 002112
7315 017246 006300
7316 017250

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 L146
CMP MJPAT(RO),#MT0999
BEQ L146

017250 026027 017316 023234
017256 001405
017260 026027 017316 024106
017266 001401
7317 017270 104511

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

7319 017272 010637 002146
7320 017276 162737 000002 002146
7321 017304 004770 017316
7322 017310 005037 002076
7323 017314 000207

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

7324
7325
7326
7327
7328

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

7329 017316
7330 017316 023234
7331 017320 020110
7332 017322 020322
7333 017324 020144
7334 017326 017432
7335 017330 017526
7336 017332 017642
7337 017334 017756
7338 017336 020026
7339 017340 020424
7340 017342 023334
7341 017344 020714
7342 017346 020746
7343 017350 021220
7344 017352 021012
7345 017354 022342
7346 017356 022532
7347 017360 023046
7348 017362 023234
7349
7350 017364 024106
7351 017366 024106
7352 017370 024106
7353 017372 024106
7354 017374 024106

MJPAT: ;PAT TIME DISRIPTION
;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

7356
7357
7358
7359 017376

.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

MT0002: SUBTST <<MT0002      SETUP COMPLEMENT ADDRESS TEST>>
;*****
;*SUBTEST      MT0002      SETUP COMPLEMENT ADDRESS TEST
;*****
          MOV      #2,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
          MOV      #LAST+2,R0
          MOV      #SIZE,R1
          MOV      #FIRST,R4
          MOV      #100001,R5
          TST      NOSUPER
          BNE      2$
          CMP      SIPAR5,SIPAR6
          BNE      4$
          BR       3$
          CMP      UIPAR5,UIPAR6
          BNE      4$
          MOV      #30000,R1
          MOV      #140000,R0
          MOV      #120001,R5
          MOV      #1,R2
          MOV      R1,R3
          MOV      #MTP002,SUPDOADD ;SET UP CALLING ADDRESS
          CALL     SUPD03
    
```

7360 017376 005037 002300
7361 017402 012700 060000
7362 017406 012701 040000
7363 017412 004737 032356
7364 017416 012737 024526 002262
7365 017424 004737 024334
7366 017430 000207
7367 017432

7368 017432 012737 000001 002300
7369 017440 012700 060000
7370 017444 012701 040000
7371 017450 005737 002456
7372 017454 001005
7373 017456 023737 172252 172254
7374 017464 001007
7375 017466 000404
7376 017470 023737 177652 177654 2\$:
7377 017476 001002
7378 017500 012701 030000 3\$:
7379 017504 005002 4\$:
7380 017506 004737 032356
7381 017512 012737 024552 002262
7382 017520 004737 024334
7383 017524 000207
7384 017526

7385 017526 012737 000002 002300
7386 017534 012700 160000
7387 017540 012701 040000
7388 017544 012704 060000
7389 017550 012705 100001
7390 017554 005737 002456
7391 017560 001005
7392 017562 023737 172252 172254
7393 017570 001013
7394 017572 000404
7395 017574 023737 177652 177654 2\$:
7396 017602 001006
7397 017604 012701 030000 3\$:
7398 017610 012700 140000
7399 017614 012705 120001
7400 017620 012702 000001 4\$:
7401 017624 010103
7402 017626 012737 024604 002262
7403 017634 004737 024334

7404 017640 000207

RETURN

7407 017642

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

7408 017642 005737 002132
 017642 005737 002132
 017646 001401
 017650 000207
 017652

IF EQFLAG IS TRUE THEN \$RETURN

TST EQFLAG
 BEQ L147
 RTS PC

L147:;;

7409 017652 012737 000003 002300
 7410 017660 005037 002326
 7411 017664 004737 032366
 7412 017670 012701 060000
 7413 017674 012703 020000
 7414 017700 072327 177770
 7415 017704 012702 000004
 7416 017710 012705 000100
 7417 017714 104415
 7418 017716 012737 024636 002262
 7419 017724 004737 024334
 7420 017730 104416
 7421 017732 012737 024676 002262
 7422 017740 004737 024350
 7423 017744 022737 000003 002616
 7424
 7425 017752 001344
 7426 017754 000207
 7427

1\$:
 2\$:

MOV #3,REALPAT
 CLR PCBUMP
 CALL FLIPWARN
 MOV #FIRST,R1
 MOV #20000,R3
 ASH #8,R3
 MOV #4,R2
 MOV #64,R5
 SAVREG
 MOV #MTPA03,SUPDOADD
 CALL SUPD03
 RESREG
 MOV #MTPB03,SUPDOADD
 CALL SUPD04
 CMP #3,FLIPL0C
 BNE 1\$
 RETURN

;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
 ;TRAPS DO NOT ADD TO PC
 ;SETUP WARNING CONSTANTS & R2
 ;R1 --- STARTING ADDRESS
 ;R3 --- R3 / 256.
 ;SMALL LOOP SIZE
 ;MEDIUM LOOP SIZE

;DO IT IN MAIN MEMORY

;DONE WITH 4 PATTERNS
 ;[(0,177777);(177777,0);(401,177777);(177777,401)]?
 ;NO - LOOP

```

7428 017756          MT0004: SUBST <<MT0004      SETUP ROTATING ZEROS TEST>>
;*****
;SUBTEST            MT0004  SETUP ROTATING ZEROS TEST
;*****
7429 017756  012737  000004  002300          MOV    #4,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7430 017764  012737  000004  002326          MOV    #4,PCBUMP          ;TRAPS ADD 4 TO PC
7431 017772  013702  002614                    MOV    ONES,R2
7432 017776  004737  032516                    CALL   BACKGND            ;WRITE BACKGROUND OF ONES
7433 020002  012700  060000                    MOV    #FIRST,R0
7434 020006  012701  040000                    MOV    #SIZE,R1
7435 020012  012737  024774  002262          MOV    #MTPA04,SUPDOADD   ;SET UP LINKS
7436 020020  004737  024350                    CALL   SUPD04
7437 020024  000207                    RETURN
7438 020026          MT0005: SUBST <<MT0005      FTUP ROTATING ONES TEST>>
;*****
;SUBTEST            MT0005  SETUP ROTATING ONES TEST
;*****
7439 020026  012737  000005  002300          MOV    #5,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7440 020034  012737  000004  002326          MOV    #4,PCBUMP          ;TRAPS ADD 4 TO PC
7441 020042  005002                    CLR    R2
7442 020044  004737  032516                    CALL   BACKGND            ;WRITE BACKGROUND OF ZEROS
7443 020050  012700  060000                    MOV    #FIRST,R0
7444 020054  012701  040000                    MOV    #SIZE,R1
7445 020060  012737  025050  002262          MOV    #MTP005,SUPDOADD   ;SET UP LINKS
7446 020066  012737  025064  025046          MOV    #MTP005+14,MTPB04+16
7447 020074  004737  024350                    CALL   SUPD04
7448 020100  012737  025010  025046          MOV    #MTPA04+14,MTPB04+16 ;RESET TEST'S ORIGINAL VALUE
7449 020106  000207                    RETURN
    
```

```

7452 020110          MT0006: SUBTST <<MT0006      SETUP INITIAL DATA TEST>>
;*****
;*SUBTEST          MT0006  SETUP INITIAL DATA TEST
;*****
7453 020110  012737  000006  002300      MOV      #6,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7454 020116  012737  000004  002326      MOV      #4,PCBUMP      ;TRAPS ADD 4 TO PC
7455 020124  012701  002412                    MOV      #TESTADD,R1
7456 020130  012737  025104  002262      MOV      #MTP006,SUPDOADD
7457 020136  004737  024334      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
7458 020142  000207                    RETURN
7459 020144          MT0007: SUBTST <<MT0007      SETUP ADDRESS BIT TEST>>
;*****
;*SUBTEST          MT0007  SETUP ADDRESS BIT TEST
;*****
7460 020144  012737  000007  002300      MOV      #7,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7461 020152  005002                    CLR      R2
7462 020154  004737  032516      CALL     BACKGND        ;OF ZEROS
7463 020160  012701  060000      MOV      #FIRST,R1
7464 020164  012702  000001      MOV      #1,R2
7465 020170  C50201                    BIS      R2,R1
7466 020172  012737  025304  002262      MOV      #MTP007,SUPDOADD
7467 020200  004737  024334      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
7468 020204  000207                    RETURN
7469 020206          MT0010: SUBTST <<MT0010      SETUP BYTE ADDRESSING TEST>>
;*****
;*SUBTEST          MT0010  SETUP BYTE ADDRESSING TEST
;*****
7470 020206  012737  000010  002300      MOV      #10,REALPAT     ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7471 020214  012737  000004  002326      MOV      #4,PCBUMP      ;TRAPS ADD 4 TO PC
7472 020222  013704  002412                    MOV      TESTADD,R4
7473 020226  012737  025404  002262      MOV      #MTP010,SUPDOADD
7474 020234  004737  024334      CALL     SUPD03          ;DO IT IN SUPERVISOR MODE
7475 020240  000207                    RETURN
    
```

```

7478 020242          MTO014: SUBTST <<MTO014      SETUP BASIC DOUBLE BIT ERROR TEST>>
;*****
;SUBTEST          MTO014  SETUP BASIC DOUBLE BIT ERROR TEST
;*****
7479 020242          IF ACTFLAG IS TRUE OR APTFLAG IS TRUE
020242 005737 002350          TST ACTFLAG
020246 001003          BNE L150
020250 005737 002352          TST APTFLAG
020254 001404          BEQ L151
020256          L150:; ; ; ; ; ; ; ;
7480 020256          IF $PASS NE #0 THEN $RETURN
020256 005737 056724          TST $PASS
020262 001401          BEQ L152
020264 000207          RTS PC
020266          L152:; ; ; ; ; ; ; ;
7481 020 36          END; OF IF ACTFLAG
020266          L151:; ; ; ; ; ; ; ;
7482 020266 012737 000014 002300  MOV    #14,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7483 020274 004737 037646          CALL  MAPKERNAL        ;MAP KERNAL SPACE
7484 020300          LET R1 := #100000      ;SETUP TEST ADDRESS
020300 012701 100000          MOV #100000,R1
7485 020304 004737 032562          CALL  GETCSR          ;GET CSR INFO FROM CONFIGURATION TABLE
7486 020310 004737 025512          CALL  MTP014         ;DO BASIC DOUBLE BIT ERROR TEST
7487 020314 004737 037734          CALL  UNMAP         ;UNMAP KERNAL SPACE
7488 020320 000207          RETURN
  
```

7491
7492 020322

```
MT0017: SUBTST <<MT0017          SETUP HOLDING 1'S & 0'S>>
;*****
;#SUBTEST      MT0017  SETUP HOLDING 1'S & 0'S
;*****
7493 020322 012737 000017 002300      MOV      #17,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7494 020330 012737 025736 002262      MOV      #MTP017,SUPDOADD
7495 020336 004737 024334              CALL     SUPD03              ;DO IT IN SUPERVISOR MODE
7496 020342 000207                      RETURN
```

7499 020344

```

MT0020: SUBST <<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

```

7500 020344 005737 002350
 020350 001003
 020352 005737 002352
 020356 001404
 020360

```

      TST ACTFLAG
      BNE L153
      TST APTFLAG
      BEQ L154

```

L153:|||||

7501 020360 005737 056724
 020364 001401
 020366 000207
 020370

IF #PASS NE #0 THEN #RETURN

```

      TST #PASS
      BEQ L155
      RTS PC

```

L155:|||||

7502 020370 012737 000020 002300
 020376 004737 037646

END; OF IF ACTFLAG

L154:|||||

7503 020402 012701 100000
 7504 020406 004737 032562
 7505 020412 004737 026014
 7506 020416 004737 037734
 7507 020422 000207

```

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

```

```

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

```

```

      MOV #100000,R1

```

7506 020406 004737 032562
 7507 020412 004737 026014
 7508 020416 004737 037734
 7509 020422 000207

```

CALL    GETCSR
CALL    MTPO20
CALL    UNMAP
RETURN

```

```

;GET CSR INFO FROM CONFIGURATION TABLE
;DO SYNDROMES TO CSR ON SINGLE ERROR TEST
;UNMAP KERNAL SPACE

```

```

7511 020424          MT0021: SUBTST <<MT0021      SETUP MARCHING 0'S & 1'S TEST>>
;*****
;*SUBTEST          MT0021  SETUP MARCHING 0'S & 1'S TEST
;*****
7512 020424          SET NOSCOPE
7512 020424 012737 177777 002440          MOV    #21,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7513 020432 012737 000021 002300          MOV    BAKPAT,R2
7514 020440 013702 002632          CALL   BACKGND
7515 020444 004737 032516          MOV    R2,R3
7516 020450 010203          SWAB   R3
7517 020452 000303          MOV    #LAST+2,R1
7518 020454 012701 160000          MOV    R1,R5
7519 020460 010105          MOV    #FIRST,R4
7520 020462 012704 060000          CMP    #5,PROTYP          ;IS THIS AN 11/83
7521 020466 022737 000005 004064          BEQ    1#                  ;BRANCH IF IT IS
7522 020474 001450          CMP    #3,PROTYP          ;IS THIS AN 11/24?
7523 020476 022737 000003 004064          BEQ    3#                  ;BRANCH IF SO
7524 020504 001407          CMP    #7,BANK
7525 020506 022737 000007 002102          BNE    3#
7526 020514 C01003          MOV    #140000,R1
7527 020516 012701 140000          MOV    R1,R5
7528 020522 010105          CMP    #177,BANK
7529 020524 022737 000177 002102 3#:          BNE    5#
7530 020532 001003          MOV    #140000,R1
7531 020534 012701 140000          MOV    R1,R5
7532 020540 010105          MOV    #MTPA21,SUPDOADD
7533 020542 012737 026330 002262 5#:          CALL   SUPD03
7534 020550 004737 024334          MOV    #MTPB21,SUPDOADD
7535 020554 012737 026360 002262          CALL   SUPD04
7536 020562 004737 024350          MOV    R4,R1
7537 020566 010401          MOV    #MTPC21,SUPDOADD
7538 020570 012737 026414 002262          CALL   SUPD04
7539 020576 004737 024350          MOV    #MTPD21,SUPDOADD
7540 020602 012737 026450 002262          CALL   SUPD04
7541 020610 004737 024350          BR     2#
7542 020614 000434          CMP    #177,BANK
7543 020616 022737 000177 002102 1#:          BNE    4#
7544 020624 001003          MOV    #140000,R1
7545 020626 012701 140000          MOV    R1,R5
7546 020632 010105          MOV    #MTPA21,SUPDOADD
7547 020634 012737 026330 002262 4#:          CALL   SUPD03
7548 020642 004737 024334          MOV    #MTPB21,SUPDOADD
7549 020646 012737 026360 002262          CALL   SUPD04
7550 020654 004737 024350          MOV    R4,R1
7551 020660 010401          MOV    #MTPC21,SUPDOADD
7552 020662 012737 026414 002262          CALL   SUPD04
7553 020670 004737 024350          MOV    #MTPD21,SUPDOADD
7554 020674 012737 026450 002262          CALL   SUPD04
7555 020702 004737 024350          CLR   NOSCOPE
7556 020706 005037 002440          2#:
7557 020712 000207          RETURN
    
```



```

7559 020714          MT0022: SUBTST <<MT0022      SETUP REFRESH & SHIFTING DIAGONAL TEST>>
;*****
;SUBTEST           MT0022  SETUP REFRESH & SHIFTING DIAGONAL TEST
;*****
7560 020714  004737  024122          CALL      KAMITEST           ;CHECK FOR KAMIKAZE MODE
7561 020720          ON.ERROR THEN $RETURN          ;IF NOT IN KAMIKAZE MODE RETURN
          103001          BCC L156
          000207          RTS PC
          020724          L156:;;;;;
7562 020724  012737  000022  002300      MOV      #22,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7563 020732  012737  026500  002262      MOV      #MTP022,SUPDOADD
7564 020740  004737  024334          CALL      SUPD03          ;DO IT IN SUPERVISOR MODE
7565 020744  000207          RETURN
7566
7567 020746          MT0023: SUBTST <<MT0023      SHIFTING DIAGONAL TEST>>
;*****
;SUBTEST           MT0023  SHIFTING DIAGONAL TEST
;*****
7568 020746  004737  024122          CALL      KAMITEST           ;CHECK FOR KAMIKAZE MODE
7569 020752          ON.ERROR THEN $RETURN          ;IF NOT IN KAMIKAZE MODE RETURN
          103001          BCC L157
          000207          RTS PC
          020756          L157:;;;;;
7570 020756  012737  000023  002300      MOV      #23,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7571 020764  012737  026500  002262      MOV      #MTP022,SUPDOADD
7572 020772          SET      DIAGFLAG          ;IDENTIFY DIAGONAL TEST TO MTP022
          012737  177777  002004      MOV      #-1,DIAGFLAG
7573 021000  004737  024334          CALL      SUPD03          ;DO IT IN SUPERVISOR MODE
7574 021004  005037  002004          CLR      DIAGFLAG
7575 021010  000207          RETURN
    
```

7577 021012

```

MT0024: SUBTST <<MT0024      SETUP FAST GALLOPING PATTERN TEST>>
;*****
;SUBTEST      MT0024      SETUP FAST GALLOPING PATTERN TEST
;*****

```

7578 021012 004737 024122

```

CALL      KAMITEST      ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
                                BCC L160
                                RTS PC
                                L160:;;;;;

```

7579 021016 103001
 021020 000207
 021022

7580 021022 012737 177777 002440

```

SET      NOSCOPE
                                MOV #-1,NOSCOPE
                                ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

```

7581 021030 012737 000024 002300

```

MOV      #24,REALPAT

```

7582 021036 013702 002632

```

MOV      BAKPAT,R2

```

7583 021042 004737 032516

```

CALL     BACKGND

```

7584 021046 010203

```

MOV      R2,R3

```

7585 021050 010304

```

MOV      R3,R4

```

7586 021052 000304

```

SWAB     R4

```

7587 021054 012701 060000

```

MOV      #FIRST,R1

```

7588 021060 012705 157776

```

MOV      #LAST,R5

```

7589 021064 022737 000005 004064

```

CMP      #5,PROTYP

```

7590 021072 001426

```

BEQ      1$

```

7591 021074 022737 000003 004064

```

CMP      #3,PROTYP

```

7592 021102 001406

```

BEQ      3$

```

7593 021104 022737 000007 002102

```

CMP      #7,BANK

```

7594 021112 001002

```

BNE     3$

```

7595 021114 012705 137776

```

MOV      #137776,R5

```

7596 021120 022737 000177 002102 3\$:

```

CMP      #177,BANK

```

7597 021126 001003

```

BNE     7$

```

7598 021130 012701 137776

```

MOV      #137776,R1

```

7599 021134 010105

```

MOV      R1,R5

```

7600 021136 104415

```

SAVREG

```

7601 021140 012737 027214 002262 7\$:

```

MOV      #MTPB24,SUPDOADD

```

7602 021146 000412

```

BR       2$

```

7603 021150 022737 000177 002556 1\$:

```

CMP      #177,LASTBANK

```

7604 021156 001002

```

BNE     4$

```

7605 021160 012705 137776

```

MOV      #137776,R5

```

7606 021164 104415

```

SAVREG

```

7607 021166 012737 027214 002262 4\$:

```

MOV      #MTPB24,SUPDOADD

```

7608 021174 004737 024350 2\$:

```

CALL     SUPD04

```

7609

```

;DO IT AGAIN FOR COMPLEMENT DATA

```

7610 021200 104416

```

RESREG

```

7611 021202 000302

```

SWAB     R2

```

7612 021204 000303

```

SWAB     R3

```

7613 021206 004737 024350

```

CALL     SUPD04

```

7614 021212 005037 002440

```

CLR      NOSCOPE

```

7615 021216 000207

```

RETURN

```

```

7618 021220          MT0026: SUBTST <<MT0026      SETUP RANDOM DATA TEST>>
;*****
;SUBTEST          MT0026  SETUP RANDOM DATA TEST
;*****
7619 021220 012737 000026 002300          MOV      #26,REALPAT
7620 021226 005037 002326          CLR      PCBUMP          ;TRAPS DO NOT ADD TO THE PC
7621 021232 013703 002604          MOV      SEEDLO,R3      ;INITIALIZE RANDOM NUMBERS
7622 021236 013702 002602          MOV      SEEDHI,R2
7623 021242 010305          MOV      R3,R5
7624 021244 010204          MOV      R2,R4
7625 021246 012701 060000          MOV      #FIRST,R1
7626 021252 012700 020000          MOV      #SIZE/2,R0
7627 021256 022737 000005 004064          CMP      #5,PROTYP      ;DO WE HAVE AN 11/83 ?
7628 021264 001445          BEQ      1#             ;BRANCH IF WE DO
7629 021266 022737 000003 004064          CMP      #3,PROTYP      ;11/24?
7630 021274 001406          BEQ      3#             ;BRANCH IF SO
7631 021276 022737 000007 002102          CMP      #7,BANK
7632 021304 001002          BNE      3#
7633 021306 012700 014000          MOV      #14000,R0
7634 021312 022737 000_17 002102 3# :    CMP      #177,BANK
7635 021320 001002          BNE      7#
7636 021322 012700 014000          MOV      #14000,R0
7637 021326 104415          7# :    SAVREG
7638 021330 012737 027252 027352          MOV      #MTPA26+4,MTPD26+14
7639 021336 012737 027246 002262          MOV      #MTPA26,SUPDOADD
7640 021344 004737 024334          CALL    SUPD03
7641 021350 005037 027276          CLR      RANODD          ;FOR ERROR REPORTING
7642 021354 012737 027266 027352          MOV      #MTPB26+4,MTPD26+14 ;SET UP NEXT LINK
7643 021362 012737 027262 002262          MOV      #MTPB26,SUPDOADD
7644 021370 104416          RESREG
7645 021372 004737 024334          CALL    SUPD03
7646 021376 000432          BR      2#
7647 021400 022737 000177 002102 1# :    CMP      #177,BANK
7648 021406 001002          BNE      4#
7649 021410 012700 014000          MOV      #14000,R0
7650 021414 104415          4# :    SAVREG
7651 021416 012737 027252 027352          MOV      #MTPA26+4,MTPD26+14
7652 021424 012737 027246 002262          MOV      #MTPA26,SUPDOADD
7653 021432 004737 024334          CALL    SUPD03
7654 021436 005037 027276          CLR      RANODD          ;FOR ERROR REPORTING
7655 021442 012737 027266 027352          MOV      #MTPB26+4,MTPD26+14 ;SET UP NEXT LINK
7656 021450 012737 027262 002262          MOV      #MTPB26,SUPDOADD
7657 021456 104416          RESREG
7658 021460 004737 024334          CALL    SUPD03
7659 021464 010337 002604          2# :    MOV      R3,SEEDLO      ;UPDATE FOR NEW RANDOM NUMBERS
7660 021470 010237 002602          MOV      R2,SEEDHI
7661 021474 000207          RETURN
    
```

7664 021476

MT0027: SUBTST <<MT0027 UNIQUE BANK TEST>>

;SUBTEST MT0027 UNIQUE BANK TEST

7665

;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA

7666

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

7667 021476 012737 000027 002300

MOV #27,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY

7668 021504 104502

CLRCR ;CLEAR CSRS

7669 021506 012737 024334 002522

MOV #SUPD03,LINK1 ;SET UP LINK

7670 021514

SET NOFSMODE

021514 012737 177777 002426

MOV #-1,NOFSMODE

7671 021522

STAR27: FOR I := #1 TO #2

MOV #1,I

021522 012737 000001 002452

B23:::~::~

021530

FOR BANK := #0 TO LASTBANK

CLR BANK

7672 021530

021530 005037 002102

B24:::~::~

021534

CALL EXBANK

7673 021534 004737 037760

IF ACFLAG IS TRUE AND RRFLAG IS FALSE

7674 021540

021540 C05737 002116

TST ACFLAG

021544 001436

BEQ L161

021546 005737 002124

TST RRFLAG

021552 001033

BNE L161

7675 021554 104511

INVALIDATE

;INVALIDATE BACKGROUND PATTERN ON "BANK"

7676 021556

LET R2 := BANK

MOV BANK,R2

021556 013702 002102

7677 021562 012700 060000

MOV #FIRST,R0

7678 021566 010004

MOV R0,R4

7679 021570 012701 040000

MOV #SIZE,R1

7680 021574 010103

MOV R1,R3

7681 021576

IF I EQ #1

CMP I,#1

021576 023727 002452 000001

BNE L162

021604 001005

7682 021606 012737 027572 002262

MOV #MTP034,SUPDOADD

7683 021614 004777 160702

CALL @LINK1

7684 021620

END ;OF IF

L162:::~::~

021620

IF I EQ #2

CMP I,#2

7685 021620

021620 023727 002452 000002

BNE L163

021626 001005

7686 021630 012737 027600 002262

MOV #MTP034+6,SUPDOADD

7687 021636 004737 024334

CALL SUPD03

7688 021642

END ;OF IF

L163:::~::~

021642

END ;OF IF

L161:::~::~

7689 021642

021642

END ;OF FOR BANK

INC BANK

021642 005237 002102

CHP BANK, LASTBANK

021646 023737 002102 002556

BLE B24

021654 003727

E24:::~::~

7691 021656

END ;OF FOR I

INC I

021656 005237 002452

CMP I,#2

021662 023727 002452 000002

BLE B23

021670 003717

E23:::~::~

7692 021672

IF FS7FLAG IS TRUE

021672	005737	002446				TST FS7FLAG
7693 021676	001403					BEQ L164
7694 021700	005037	002426				
7695 021704	000207					
7696 021706						
7697 021714	012737	000001	002452			
7698 021714	013737	002556	002102			
7699 021722	004737	037760				
7700 021726	005737	002116				
7701 021732	001436					
7702 021734	005737	002124				
7703 021740	001033					
7704 021742	C13702	002102				
7705 021746	005102					
7706 021750	012700	060000				
7707 021754	010004					
7708 021756	012701	040000				
7709 021762	010103					
7710 021764	023727	002452	000001			
7711 021772	001005					
7712 021774	012737	027572	002262			
7713 022002	004777	160514				
7714 022006						
7715 022006						
7716 022006	023727	002452	000002			
7717 022014	001005					
7718 022016	012737	027600	002262			
7719 022024	004737	024334				
7720 022030						
7721 022030						
7722 022030						
7723 022030	005337	002102				
7724 022034	023727	002102	000000			
7725 022042	002327					
7726 022044						
7727 022044	005237	002452				
7728 022050	023727	002452	000002			
7729 022056	003716					
7730 022060						
7731 022060	005037	002426				
7732 022064	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,SUPD0ADD
CALL @LINK1
END ;OF IF

IF I EQ #2

MOV #MTP034+6,SUPD0ADD
CALL SUPD03
END ;OF IF

END ;OF FOR BANK

END ;OF FOR I

CLR NOFSMODE
RETURN
    
```

7721 022066
7722 022066 005037 002264
7723 022072
022072 012737 177777 002542
7724 022100 012737 000030 002300
7725 022106 012737 000001 002076
7726 022114 012737 024334 002522
7727 022122 012737 027354 002262
7728 022130 104470
7729 022132
022132 012737 177777 002426
022140 012737 177777 002440
7730 022146
022146 005037 002102
022152
7731 022152 004737 037760
7732 022156
022156 005737 002120
022162 001414
7733 022164
022164 005737 002116
022170 001411
022172 005737 002124
022176 001006
7734 022200 012701 040000
7735 022204 012700 060000
7736 022210 004777 160306
7737 022214
022214
7738 022214
022214
7739 022214
022214 005237 002102
022220 023737 002102 002556
022226 003751
022230
7740 022230
022230 005737 002264
022234 001032
7741 022236
022236 012737 177777 002264
7742 022244 104502
7743 022246 004737 036310
7744 022252
022252 103010
7745 022254 104472
7746 022256
022256 005037 002426
022262 005037 002440
022266 005037 002542
7747 022272 000207
7748 022274
022274
7749 022274 013737 002310 002102

```
MT0030: SUBTST <<MT0030      SETUP FLUSH OUT DBE'S TEST>>
;*****
;SUBTEST      MT0030      SETUP FLUSH OUT DBE'S TEST
;*****
          CLR      PASFLG
          SET      FULLREL
MTA030: MOV      #30,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
          MOV      #1,NOPAR      ;INDICATE COUNT PARITY ERRORS
          MOV      #SUPD03,LINK1
          MOV      #MTP030,SUPDOADD
          ECCDIS      ;DISABLE ERROR CORRECTION
          SET      NOFSMODE,NOSCOPE
                                MOV      #1,NOFSMODE
                                MOV      #-1,NOSCOPE
          FOR BANK := #0 TO LASTBANK
                                CLR BANK
                                B27:::
          CALL EXBANK
          IF MKFLAG IS TRUE
                                TST MKFLAG
                                BEQ L170
          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
                                TST ACFLAG
                                BEQ L171
                                TST RRFLAG
                                BNE L171
          MOV      #SIZE,R1
          MOV      #FIRST,R0
          CALL     @LINK1
          END ;OF IF ACFLAG
                                L171:::
          END ;OF IF MKFLAG
                                L170:::
          END ;OF FOR
                                INC BANK
                                CMP BANK, LASTBANK
                                BLE B27
                                E27:::
          IF PASFLG IS FALSE
                                TST PASFLG
                                BNE L172
          SET PASFLG
                                MOV      #-1,PASFLG
          CLRCSR      ;CLEAR CSRS
          CALL RELOCATE
          ON.ERROR
                                BCC L173
          ECCINIT      ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
          CLEAR      NOFSMODE,NOSCOPE,FULLREL
                                CLR      NOFSMODE
                                CLR      NOSCOPE
                                CLR      FULLREL
          RETURN
          END ;OF ON.ERROR
                                L173:::
          MOV      NEWBANK,BANK
```

```
7750 022302 004737 037760      CALL  EXBANK
7751 022306 004737 022100      CALL  MTA030
7752 022312 104472              ECCINIT      ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7753 022314 004737 037122      CALL  UNRELOCATE
7754 022320 000207              RETURN
7755 022322              END ;OF IF PASFLG
                                L172:;;;;;
7756 022322 104472              ECCINIT      ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7757 022324              CLEAR  NOFSMODE,NOSCOPE,FULLREL
                                CLR  NOFSMODE
                                CLR  NOSCOPE
                                CLR  FULLREL
                                CLR  NOFSMODE
                                CLR  NOSCOPE
                                CLR  FULLREL
7758 022340 000207              RETURN
```

```

7761 022342          MT0031: SUBTST <<MT0031      SETUP SOB-A-LONG TEST>>
;*****
;SUBTEST          MT0031 SETUP SOB-A-LONG TEST
;*****
7762 022342 004737 024122          CALL      KAMITEST          ;CHECK FOR KAMIKAZE MODE
7763 022346          ON.ERROR THEN $RETURN          ;IF NOT IN KAMIKAZE MODE RETURN
          022346 103001          BCC L174
          022350 000207          RTS PC
          022352          L174:;;;;;
7764 022352          SET      NOSCOPE
          022352 012737 177777 002440          MOV     #-1,NOSCOPE
7765 022360 012737 000031 002300          MOV     #31,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7766 022366 005037 002076          CLR     NOPAR          ;SETUP PARITY ACTION
7767 022372          MAP     BANK          ;MAP FIRST SO BLOCK MOVE WORKS
          022372 010346          MOV     R3,-(SP)
          022374 013703 002102          MOV     BANK,R3
          022400 004737 035604          CALL   MAPPER
          .DSABL CRF
          MOV     (SP)+,R3
7768 022404 012603          TESTAREA          ;ENTER TEST MODE
          022406 053737 002552 177776          BIS     TESTMODE,PSW      ;GO TO SYSTEM TEST MODE
          .DSABL CRF
7769 022414          BMOV    MTP031,FIRST,SOBLENGTH/2
          022414 004537 040732          JSR    R5,BLOCK3
          022420 000027          SOBLENGTH/2
          022422 060000          FIRST
          022424 027364          MTP031
          .DSABL CRF
          KERNEL          ;ENTER KERNEL MODE
7770 022426 104417          MOV     SOBK,R2
7771 022430 013702 002572          MOV     R2,R0
7772 022434 010200          MOV     #100776,R1
7773 022436 012701 100776          MOV     #FIRST+SOBLENGTH,R5
7774 022442 012705 060056          MOV     #FIRST+2,SUPDOADD
7775 022446 012737 060002 002262          MOV     #LAST+2,LINK1
7776 022454 012737 160000 002522          TST    NOSUPER
7777 022462 005737 002456          BNE    1$
7778 022466 001005          CMP    SIPAR5,SIPAR6
7779 022470 023737 172252 172254          BEQ   2$
7780 022476 01405          BR    3$
7781 022500 000407          CMP    UIPAR5,UIPAR6
7782 022502 023737 177652 177654 1$:          BNE   3$
7783 022510 001003          MOV     #140000,LINK1
7784 022512 012737 140000 002522 2$:          CALL  SUPD04
7785 022520 004737 024350 3$:          CLR   NOSCOPE
7786 022524 005037 002440          RETURN
7787 022530 000207
    
```


7790 022532

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

;SUBTEST MT0032 SETUP WRITE RECOVERY TEST

7791 022532 004737 024122

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

7792 022536

022536 103001
022540 000207
022542

7793 022542

SET NOSCOPE

7794 022550 012737 177777 002440

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)

7795 022556 005037 002076

7796 022562

022562 010346
022564 013703 002102
022570 004737 035604

MOV BANK,R3
CALL MAPPER
.DSABL CRF

7797 022574 012603

022576 C12700 010247

MOV #10247,R0

MOV (SP)+,R3
;OP CODE OF INSTRUCTION "MOV R2,-(PC)"
;OP CODE OF COMPLEMENT OF INSTRUCTION "JMP (R0)"
;USED FOR 1/2 BANK LOOP

7798 022602 012701 177667

MOV #177667,R1

7799 022606 012702 020000

MOV #SIZE/2,R2

7800 022612 010237 002522

MOV R2,LINK1

7801 022616 012703 060000

MOV #FIRST,R3

7802 022622 012704 160000

MOV #LAST+2,R4

7803 022626 005037 002524

CLR LINK2

7804 022632 005737 002456

TST NOSUPER

7805 022636 001005

BNE 1\$

7806 022640 023737 172252 172254

CMP SIPAR5,SIPAR6

7807 022646 001405

BEQ 2\$

7808 022650 000415

BR 3\$

7809 022652 023737 177652 177654 1\$:

CMP UIPAR5,UIPAR6

7810 022660 001011

BNE 3\$

7811 022662 012704 140000 2\$:

MOV #140000,R4

7812 022666 012702 014000

MOV #14000,R2

7813 022672 010237 002522

MOV R2,LINK1

7814 022676 012737 000001 002524

MOV #1,LINK2

7815

7816 022704

022704 053737 002552 177776 3\$:

TESTAREA ;ENTER TEST MODE
BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
;MOVE TEST TO MEMORY UNDER TEST

7817

7818 022712 010023 4\$:

MOV RO,(R3)+

7819 022714 010144

MOV R1,-(R4)

7820 022716 077203

SOB R2,4\$

7821

7822 022720 022737 000005 004064

CMP #5,PROTYP

7823 022726 001003

BNE 5\$

7824

;MOVE LAST PART OF TEST TO FASTCITY

7825 022730

022730 004537 040676

BMOV MTP032

022734 027442

JSR R5,BLOCK1
MTP032
.DSABL CRF

7826 022736 104417 5\$:

KERNEL ;ENTER KERNEL MODE

7827

7828 022740 012702 005141

MOV #5141,R2

;OP CODE OF INSTRUCTION "COM -(R1)"

7829 022744 012700 023044

MOV #10\$,RO

;ADDRESS TO RETURN TO IN RO

7830	022750	012701	160000			MOV	#LAST+2,R1	;TOP OF BANK
7831	022754	012737	060000	002262		MOV	#FIRST,SUPDOADD	
7832	022762	005737	002524			TST	LINK2	
7833	022766	001402				BEQ	6#	
7834	022770	012701	140000			MOV	#140000,R1	
7835	022774	004737	024350		6#:	CALL	SUPDO4	
7836	023000	012703	020000			MOV	#SIZE/2,R3	
7837	023004	012705	000110			MOV	#110,R5	
7838	023010	012704	060000			MOV	#FIRST,R4	
7839	023014	005737	002524			TST	LINK2	
7840	023020	001402				BEQ	7#	
7841	023022	012703	014000			MOV	#14000,R3	
7842	023026	012737	027442	002262	7#:	MOV	#MTP032,SUPDOADD	
7843	023034	004737	024350			CALL	SUPDO4	
7844	023040	005037	002440		9#:	CLR	NOSCOPE	
7845	023044	000207			10#:	RETURN		
7846								
7847								

;THIS RETURN ACTS AS A NORMAL RETURN FROM MT0032
;ALSO A RETURN FROM THE "CALL SUPDO4" ABOVE

```

7850 023046          MTO033: SUBTST <<MTO033      SETUP BRANCH GOBBLE TEST>>
;*****
;+SUBTEST          MTO033 SETUP BRANCH GOBBLE TEST
;*****
7851 023046 004737 024122          CALL      KAMITEST          ;CHECK FOR KAMIKAZE MODE
7852 023052          ON.ERROR THEN $RETURN          ;IF NOT IN KAMIKAZE MODE RETURN
          023052 103001          BCC L176
          023054 000207          RTS PC
          023056          L176:;:;:;:;
7853 023056          SET      NOSCOPE
          023056 012737 177777 002440          MOV     #-1,NOSCOPE
7854 023064 012737 000033 002300          MOV     #33,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7855 023072 005037 002076          CLR     NOPAR          ;SETUP PARITY ACTION
7856 023076          MAP     BANK          ;MAP FIRST SO THAT BLOCK MOVE WORKS
          023076 010346          MOV     R3,-(SP)
          023100 013703 002102          MOV     BANK,R3
          023104 004737 035604          CALL   MAPPER
          .DSABL CRF
          023110 012603          MOV (SP)+,R3
7857 023112          TESTAREA          ;ENTER TEST MODE
7858 023112 053737 002552 177776          BIS     TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
          .DSABL CRF
7859 023120          BMOV    MTP033,FIRST,GBLENGTH/2
          023120 004537 040732          JSR    R5,BLOCK3
          023124 000037          GBLENGTH/2
          023126 060000          FIRST
          023130 027474          MTP033
          .DSABL CRF
7860 023132 104417          KERNEL          ;ENTER KERNEL MODE
7861 023134 012705 060076          MOV     #FIRST+GBLENGTH,R5
7862 023140 012737 060004 002262          MOV     #FIRST+4,SUPDOADD
7863 023146 012701 060002          MOV     #FIRST+2,R1
7864 023152 012702 060003          MOV     #FIRST+3,R2
7865 023156 012737 160000 002522          MOV     #LAST+2,LINK1
7866 023164 005737 002456          TST     NOSUPER
7867 023170 001005          BNE     1$
7868 023172 023737 172252 172254          CMP     SIPAR5,SIPAR6
7869 023200 001405          BEQ     2$
7870 023202 000407          BR     3$
7871 023204 023737 177652 1.7654 1$:          CMP     UIPAR5,UIPAR6
7872 023212 001003          BNE     3$
7873 023214 012737 140000 002522 2$:          MOV     #140000,LINK1
7874 023222 004737 024350          3$:          CALL   SUPD04
7875 023226 005037 002440          CLR     NOSCOPE
7876 023232 000207          RETURN
7877 023234          MTO034: SUBTST <<MTO034      SOFT ERROR - BACKGROUND PATTERN TEST>>
;*****
;+SUBTEST          MTO034 SOFT ERROR - BACKGROUND PATTERN TEST
;*****
7881 023234 012737 000034 002300          MOV     #34,REALPAT
7882 023242 012700 060000          MOV     #FIRST,R0
7883 023246 012701 040000          MOV     #SIZE,R1
7884 023252 013702 002620          MOV     SOFTPAT,R2
    
```

```

7885 023256 010103          MOV      R1,R3
7886 023260 013705 002104   MOV      BANKINDEX,R5
7887 023264 010004          MOV      R0,R4
7888 023266          IF #BIT13 SET.IN CONFIG+2(R5)
                                BIT #BIT13,CONFIG+2(R5)
                                BEQ L177
7889          ;BACKGROUND PATTERN IS VALID
7890 023276 012737 027600 002262   MOV      #MTPO34+6,SUPDOADD
7891 023304 004737 024334          CALL    SUPD03          ;READ IT
7892 023310          ELSE
                                BR L200
                                L177:#####
                                023310 000410
7893          ;BACKGROUND PATTERN HAS BEEN INVALIDATED
7894 023312 012737 027572 002262   MOV      #MTPO34,SUPDOADD
7895 023320 004737 024334          CALL    SUPD03
7896 023324 052765 020000 002666   BIS      #BIT13,CONFIG+2(R5) ;VALIDATE IT
7897 023332          END ;OF IF #BIT13
                                L200:#####
                                023332
7898 023332 000207          RETURN
7899
7900 023334          MT0035: SUBST <<MT0035          SETUP WORST CASE NOISE PARITY TEST>>
                                ;*****
                                ;*SUBTEST          MT0035 SETUP WORST CASE NOISE PARITY TEST
                                ;*****
7901 023334 012737 000035 002300   MOV      #35,REALPAT          ;SET UP TEST NUMBER FOR DISPLAY
7902 023342 013703 002104          MOV      BANKINDEX,R3
7903 023346 016301 002664          MOV      CONFIG(R3),R1
7904 023352 000301          SWAB    R1
7905 023354 042701 177760          BIC      #C17,R1
7906 023360 006301          ASL     R1
7907 023362 010137 002152          MOV      R1,CSRNO
7908 023366 023737 002152 002532   CMP      CSRNO,PGMCSR
7909 023374 001001          BNE     1:
7910 023376 000207          RETURN
7911 023400 012702 052524          1: MOV      #52524,R2
7912 023404 004737 032516          CALL    BACKGND          ;WRITE BACKGROUND OF ALMOST ALT. 1'S AND 0'S
7913 023410 012737 027616 002262   MOV      #MTPO35,SUPDOADD
7914 023416 004737 024334          CALL    SUPD03
7915 023422          IF QVFLAG IS TRUE THEN RETURN
                                TST QVFLAG
                                BEQ L201
                                RTS PC
                                L201:#####
7916 023432 005102          COM     R2
7917 023434 004737 032516          CALL    BACKGND          ;WRITE COMPLEMENT PATTERN INTO MUT
7918 023440 004737 024350          CALL    SUPD04
7919 023444 000207          RETURN
    
```

7921 023446

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

;SUBTEST MT0036 SETUP CORRECTION CODE TEST

7922 023446 012737 000036 002300
7923 023454 004737 032562
7924 023460 005037 002264
7925 023464 005000
7926 023466 012701 100000
7927 023472 004737 037646
7928 023476 004737 027760
7929 023502 004737 037734
7930 023506 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

7932 023510

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

;*SUBTEST MT0037 SETUP ECC DISABLE TEST

7933 023510 012737 000037 002300
7934 023516 012701 100000
7935 023522 005000
7936 023524 004737 037646
7937 023530 004737 032562
7938 023534 004737 030204
7939 023540 004737 037734
7940 023544 000207

MOV #37,REALPAT ;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY
MOV #100000,R1 ;SET UP TEST ADDRESS
CLR R0 ;CLEAR DATA TO BE WRITTEN
CALL MAPKERNAL ;MAP THIS TEST TO KERNEL SPACE
CALL GETCSR ;GET CSRINFO FROM CONFIG TABLE
CALL MTP037 ;CHECK ECC DISABLE
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

7942 023546

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

;*****

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

;*****

7943 023546 012737 000041 002300

MOV #41,REALPAT ;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY

7944 023554 004737 032562

CALL GETCSR ;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE

7945 023560

LET SUPDUADD := #MTP041 ;SET UP TEST ADDRESS

7946 023566 012737 030256 002262

LET R1 := #FIRST ;SET UP FIRST ADDRESS

7947 023572 012701 060000

IF BANK EQ #177 ;ARE WE AT BANK 177?

7948 023572 023727 002102 000177

MOV #41,INST,R1

7949 023600 001004

CMP BANK,#177

7948 023602 012737 000014 002570

BNE L202

7949 023610 000403

LET PASCNT := #12.

7950 023612 012737 000020 002570

ELSE

7951 023620 000207

BR L203

7950 023612 012737 000020 002570

L202:;;;;;

7951 023620 000207

LET PASCNT := #16.

7952 023620 004737 024334

MOV #16.,PASCNT

7953 023624 000207

END

CALL SUPD03 ;EXECUTE ADDRESS TO CSR TEST IN SUPERVISOR MODE

RETURN ;

7955 023626

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

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

7956 023626 012737 000042 002300
7957 023634 012701 100000
7958 023640 004737 037646
7959 023644 004737 032562
7960 023650 004737 030430
7961 023654 004737 037734
7962 023660 000207

MOV #42,REALPAT ;SETUP PATTERN AND NUMBER FOR TYPEOUT AND DISPLAY
MOV #100000,R1 ;SET UP TEST ADDRESS
CALL MAPKERNAL ;MAP TO KERNEL SPACE
CALL GETCSR ;SET UP CSRINFO FROM CONFIGURATION TABLE
CALL MTP042 ;CHECK EXTENDED Q-BUS ADDRESS TO CSR
CALL UNMAP ;REMAP KERNEL SPACE
RETURN


```

7964 023662          MT0043: SUBTST <<MT0043          SETUP WRITE BYTE CLEARS SBE TEST>>
;*****
;*SUBTEST          MT0043 SETUP WRITE BYTE CLEARS SBE TEST
;*****
7965 023662 012737 000043 002300          MOV    #43,REALPAT          ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7966 023670 004737 037646          CALL   MAPKERNAL          ;MAP TO KERNEL SPACE
7967 023674          LET R1 := #100000          ;SET UP TEST ADDRESS
7968 023700 012701 100000          CALL   MTP043              ;PERFORM WRITE BYTE TEST
7969 023704 004737 030716          CALL   UNMAP              ;REMAP KERNEL SPACE
7970 023710 000207          RETURN
7971 023712          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
;*****
7972 023712 012737 000044 002300          MOV    #44,REALPAT          ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7973 023720 004737 032562          CALL   GETCSR             ;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE
7974 023724          LET SUPDOADD := #MTP044          ;SET UP TEST ADDRESS          ;:IL
7975 023732 012737 031112 002262          LET R1 := #FIRST          ;SET UP FIRST ADDRESS          ;:IL
7976 023736 012701 060000          IF BANK EQ #177          ;ARE WE AT BANK 177?
7977 023746          LET ENDADD := #120000
7978 023754 000403          ELSE
7979 023756          LET ENDADD := #160000
7980 023764 012737 160000 002562          END
7981 023764 004737 024334          CALL   SUPD03             ;EXECUTE ADDRESS TO CSR TEST IN SUPVISIOR MODE
7982 023770 000207          RETURN
7983 023772          MT0045: SUBTST <<MT0045          SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR>>
;*****
;*SUBTEST          MT0045 SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR
;*****
7984 023772 012737 000045 002300          MOV    #45,REALPAT          ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7985 024000 004737 037646          CALL   MAPKERNAL          ;MAP TO KERNEL SPACE
7986 024004          LET R1 := #100000          ;SET UP TEST ADDRESS
7987 024010 012701 100000          CALL   MTP045              ;PERFORM SYNDROMES TO CSR ON DOUBLE BIT ERROR
7988 024014 004737 031434          CALL   UNMAP              ;REMAP KERNEL SPACE
7989 024020 000207          RETURN
7990 024022          MT0046: SUBTST <<MT0046          SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET>>
;*****
;*SUBTEST          MT0046 SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET
;*****
7991 024022 012737 000046 002300          MOV    #46,REALPAT          ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7992 024030 004737 037646          CALL   MAPKERNAL          ;MAP TO KERNEL SPACE
7993 024034          LET R1 := #100000          ;SET UP TEST ADDRESS
7994 024040 012701 100000          CALL   MTP046              ;PERFORM TRAPS DETECTED ON SBE WITH ECC DISABLED TE
7995 024044 004737 031622          CALL   UNMAP              ;REMAP KERNEL SPACE
7996 024050 000207          RETURN
    
```

7997 024052

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

7998 024052 012737 000047 002300
7999 024060 004737 037646
8000 024064
024064 012701 100000
8001 024070
024070 012702 120000
8002 024074 004737 032162
8003 024100 004737 037734
8004 024104 000207

MOV #47,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
CALL MAPKERNAL ;MAP TO KERNEL SPACE
LET R1 := #100000 ;SET UP TEST ADDRESS
MOV #100000,R1
LET R2 := #120000 ; " " SECOND TEST ADDRESS
MOV #120000,R2
CALL MTP047 ;PERFORM NO UPDATE TO CSR ON SBE WITH DBF
CALL UNMAP ;REMAP KERNEL SPACE
RETURN

```

8007 024106          MT0999: SUBTST <<MT0999          SETUP NULL TEST>>
;*****
;#SUBTEST          MT0999 SETUP NULL TEST
;*****
8008 024106 005037 002300          CLR          REALPAT
8009 024112          SET          NULLFLAG
      024112 012737 177777 002344          MOV          #-1,NULLFLAG
8010 024120 000207          RETURN
8011
8012 024122          KAMITEST:SUBTST <<CHECK FOR KAMIKAZE MODE>>
;*****
;#SUBTEST          CHECK FOR KAMIKAZE MODE
;*****
      IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE
8013 024122          TST          KAMIKAZE
      024122 005737 002006          BNE          L206
      024126 001006          TST          ACTFLAG
      024130 005737 002350          BNE          L206
      024134 001003          TST          APTFLAG
      024136 005737 002352          BNE          L207
      024142 C01403          BEQ          L207
      024144          L206:;;;;;
8014 024144          $RETURN NOERROR          ;RUN THE TEST          CLC
      024144 000241          ;RTS PC
      024146 000207          ELSE
8015 024150          BR          L210
      024150 000402          L207:;;;;;
      024152          $RETURN ERROR          ;DON'T RUN THE TEST          SEC
      024152 000261          ;RTS PC
      024154 000207          END ;OF IF KAMIKAZE
8017 024156          L210:;;;;;
      024156
    
```

```
8020 024156 SUPD01: SUBTST <<SUBR EXECUTE PATTERN IN SUPERVISOR>>
;*****
;SUBTEST SUBR EXECUTE PATTERN IN SUPERVISOR
;*****
8021 024156 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
024156 010346 MOV R3,-(SP)
024160 013703 002102 CALL BANK,R3
024164 004737 035604 .DSABL MAPPER
CRF
MOV (SP)+,R3
8022 024170 012603 SUPD02: CALL GETDIS
024172 004737 051336 PUSH $LPERR,$LPADR
MOV $LPERR,-(SP)
MOV $LPADR,(SP)
8023 024176 013746 002624
024202 013746 002622
8024 024206 010037 002160 MOV R0,SUPDR0
8025 024212 012700 002162 MOV #SUPDR1,R0
8026 024216 010120 MOV R1,(R0)+
8027 024220 010220 MOV R2,(R0)+
8028 024222 010320 MOV R3,(R0)+
8029 024224 C10420 MOV R4,(R0)+
8030 024226 010520 MOV R5,(R0)+
8031 024230 010620 MOV SP,(R0)+
8032 024232 013700 002160 MOV SUPDR0,R0
8033 024236 012737 024252 002622 MOV #TAG4+,$LPADR
8034 024244 013737 002622 002624 MOV $LPADR,$LPERR
8035 024252 012700 002176 TAG4+: MOV #SUPDR6+2,R0
8036 024256 014006 MOV -(R0),SP
8037 024260 014005 MOV -(R0),R5
8038 024262 014004 MOV -(R0),R4
8039 024264 014003 MOV -(R0),R3
8040 024266 014002 MOV -(R0),R2
8041 024270 014001 MOV -(R0),R1
8042 024272 014000 MOV -(R0),R0
8043 024274 SUPERVISOR ;ENTER SUPERVISOR MODE
024274 052737 040000 177776 BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
.DSABL CRF
MOV #SUPSTK,SSP
8044 024302 012706 000740 CACHOFF ;TURN CACHE OFF
8045 024306 104424 CALL ;CALL TO THE USER INSTRUCTION PAR'S
8046 024310 004737 177640 CACHON ;TURN CACHE ON
8047 024314 104423 KERNEL ;ENTER KERNEL MODE
8048 024316 104417 SCOPE
8049 024320 000004 POP $LPADR,$LPERR
MOV (SP)+,$LPADR
MOV (SP)+,$LPERR
8050 024322 012637 002622
024326 012637 002624
8051 024332 000207 RETURN
```

```

8054 024334 010346 002102 SUPD03: MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      024334 013703 035604 MOV BANK,R3
      024336 004737 035604 CALL MAPPER
      .DSABL CRF
      MOV (SP)+,R3
8055 024350 004737 051336 SUPD04: CALL GETDIS
8056 024354 013746 002624 PUSH $LPERR,$LPADR
      MOV $LPERR,-(SP)
      MOV $LPADR,-(SP)
8057 024364 010037 002160 MOV R0,SUPDR0
8058 024370 012700 002162 MOV #SUPDR1,R0
8059 024374 010120 MOV R1,(R0)+
8060 024376 010220 MOV R2,(R0)+
8061 024400 010320 MOV R3,(R0)+
8062 024402 010420 MOV R4,(R0)+
8063 024404 010520 MOV R5,(R0)+
8064 024406 010620 MOV SP,(R0)+
8065 024410 013700 002160 MOV SUPDR0,R0
8066 024414 C12737 024430 002622 MOV #TBG4$, $LPADR
8067 024422 J13737 002622 002624 $LPADR,$LPERR
8068 024430 012700 002176 TBG4$: MOV #SUPDR6+2,R0
8069 024434 014006 MOV -(R0),SP
8070 024436 014005 MOV -(R0),R5
8071 024440 014004 MOV -(R0),R4
8072 024442 014003 MOV -(R0),R3
8073 024444 014002 MOV -(R0),R2
8074 024446 014001 MOV -(R0),R1
8075 024450 014000 MOV -(R0),R0
8076 024452 053737 002552 177776 TESTAREA ;ENTER SUPERVISOR MODE
      BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      TST NOSUPER
      BEQ 1$
      MOV #USESTK,USP
      BR 2$
      MOV #SUPSTK,SSP
      CACHOFF ;TURN CACHE OFF
      CALL @SUPDOADD
      CACHON ;TURN CACHE ON
      KERNEL ;ENTER KERNEL MODE
      SCOPE
      POP $LPADR,$LPERR
      MOV (SP)+,$LPADR
      MOV (SP)+,$LPERR
8088 024524 000207 RETURN
    
```

```

8091
8092
8093
8094
8095
8096
8097
8098
8099
8100
8101 024526

8102 024526 010220
8103 024530 077102
8104 024532 000240
8105 024534 012401
8106 024536 020102
8107 024540 101402
8108 024542 104430
8109 024544 000240
8110 024546 077306
8111 024550 000207
8112 024552

8113 024552 010220
8114 024554 062702 000002
8115 024560 077104
8116 024562 000240
8117 024564 012400
8118 024566 020005
8119 024570 001401
8120 024572 104427
8121 024574 062705 000002
8122 024600 077307
8123 024602 000207
8124 024604

8125 024604 010540
8126 024606 062705 000002
8127 024612 077104
8128 024614 000240
8129 024616 162702 000002
8130 024622 012401
8131 024624 020102
8132 024626 001401
8133 024630 104430
8134 024632 077307
8135 024634 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

```



```

8179 024736      MTPC03: SUBST <<MTPC03      TEST DATA SUBPROGRAM>>
;*****
;*SUBTEST      MTPC03 TEST DATA SUBPROGRAM
;*****
8180 024736 020421      CMP      R4,(R1)+      ;V172360
8181 024740 001401      BEQ      1$          ;V172362
8182 024742 104431      PERRO3          ;V172364
8183 024744 005141      1$:      COM      -(R1)      ;V172366
8184 024746 005111      COM      (R1)        ;V172370
8185 024750 000137 024754  JMP      @MTPD03      ;V172372      GO TO V172260
8186
8187 024754      MTPD03: SUBST <<MTPD03      TEST DATA SUBSUBPROGRAM>>
;*****
;*SUBTEST      MTPD03 TEST DATA SUBSUBPROGRAM
;*****
8188 024754 020421      CMP      R4,(R1)+      ;V172260
8189 024756 001401      BEQ      1$          ;V172262
8190 024760 104431      PERRO3          ;V172264
8191 024762 005127      1$:      COM      (PC)+      ;V172266
8192 024764 C00000      O              ;V172270
8193 024766 001363      BNE      MTPC03      ;V172272      GO TO V172360
8194 024770 000137 024702  JMP      @MTPB03+4    ;V172274      GO TO V177644

```


8197 024774

MTPA04: SUBTST <<MTPA04 ROTATING ZEROS TEST>>
;*****
;*SUBTEST MTPA04 ROTATING ZEROS TEST
;*****

8198 024774 012705 000010
8199 025000 010504
8200 025002 000241
8201 025004 000137 025030
8202 025010 016004 177776
8203 025014 103402
8204 025016 020204
8205 025020 001401
8206 025022 104432
8207 025024 077115
8208 025026 000207
8209
8210 025030

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
;*****

8211 025030 106110
8212 025032 077502
8213 025034 106120
8214 025036 106110
8215 025040 077402
8216 025042 106120
8217 025044 000137 025010
8218
8219 025050

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
;*****

8220 025050 012705 000010
8221 025054 010504
8222 025056 000261
8223 025060 000137 025030
8224 025064 016004 177776
8225 025070 103002
8226 025072 020204
8227 025074 001401
8228 025076 104432
8229 025100 077115
8230 025102 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

8233 025104

```

MTP006: SUBSTST <<MTP006          INITIAL DATA TEST>>
;*****
; *SUBTEST      MTP006  INITIAL DATA TEST
;*****
;              THIS TEST CHECKS THE DI/DO LINES BY
;              SHIFTING A 1 THROUGH THE WORD.
;              ;SET THE FIRST TEST BIT
8234          ;MOV      <1,DATBUF
8235          ;CLR      DATBUF+2      ;CLEAR 2ND WORD
8236 025104 012737 000001 002242      MOV      <1,DATBUF      ;SET THE FIRST TEST BIT
8237 025112 005037 002244              CLR      DATBUF+2      ;CLEAR 2ND WORD
8238 025116 013771 002242 000000 1$:  MOV      DATBUF,&(R1)    ;WRITE TEST WORD 1
8239 025124 013771 002244 000002      MOV      DATBUF+2,&2(R1) ;AND TEST WORD 2
8240 025132 017102 000000              MOV      &(R1),R2
8241 025136 023702 002242              CMP      DATBUF,R2      ;NOW READ THEM
8242 025142 001401              BEQ      2$              ;BR IF FIRST 16 OK
8243 025144 104433              PERR07      ;ERROR TRAP
8244
8245 025146 017102 000002      2$:  MOV      &2(R1),R2
8246 025152 023702 002244              CMP      DATBUF+2,R2    ;NOW READ SECOND WORD
8247 025156 001401              BEQ      3$              ;BR IF OK
8248 025160 104434              PERR10      ;ERROR TRAP
8249
8250 025162 005737 002244      3$:  TST      DATBUF+2      ;HAS LAST BIT BEEN TESTED ?
8251 025166 100405              BMI      4$              ;MINUS MEANS BIT 31
8252 025170              DLEFT     DATBUF
8253          ROL      DATBUF      ;NO, SHIFT TEST BIT LEFT
8254          ROL      DATBUF+2
8255          .DSABL   CRF
8256          BR       1$
8257          ;GO WRITE NEW TEST DATA
8258          ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
8259 025200 000746              MOV      &177776,DATBUF ;PUT A 0 IN BIT 0
8260 025202 012737 177776 002242 4$:  MOV      &-1,DATBUF+2    ;AND 1'S IN ALL OTHERS
8261 025210 012737 177777 002244      MOV      DATBUF,&(R1)    ;WRITE THE DATA
8262 025216 013771 002242 000000 5$:  MOV      DATBUF+2,&2(R1) ;2 WORDS WORTH
8263 025224 013771 002244 000002      MOV      &(R1),R2
8264 025232 017102 000000              CMP      DATBUF,R2      ;NOW READ FIRST WORD
8265 025236 023702 002242              BEQ      6$              ;BR IF OK
8266 025242 001401              PERR07
8267 025244 104433
8268
8269 025246 017102 000002      6$:  MOV      &2(R1),R2
8270 025252 023702 002244              CMP      DATBUF+2,R2    ;NOW, READ SECOND WORD
8271 025256 001401              BEQ      7$              ;BR IF OK
8272 025260 104434              PERR10
8273
8274 025262 005737 002244      7$:  TST      DATBUF+2      ;TESTED BIT 31 YET?
8275 025266 100005              BPL      8$              ;BR IF YES, WE'RE DONE
8276 025270              DLEFT     DATBUF
8277          ROL      DATBUF
8278          ROL      DATBUF+2
8279          .DSABL   CRF
8280          BR       5$
8281          ;KEEP GOING
8282 025300 000746              BR
8283 025302 000207      8$:  RETURN
  
```

8276 025304

```

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.
;
;
MOV B (R1),R0
TST B R0 ;READ AND COMPARE FOR ZEROS
BEQ 1$ ;BR IF OK
PERR11
;
1$: COM B (R1) ;COMPLEMENT THE BYTE
MOV B (R1),R0
TST B R0 ;READ FOR NON ZEROS
BNE 2$ ;BR IF OK
PERR12
;
2$: BIC R2,R1 ;MASK OFF THE ASSERTED BIT
ASL R2 ;SHIFT R2 FOR NEXT BIT
BIS R2,R1 ;SET THE NEW BIT INTO R1
MOV (R1),R0 ;READ THE NEW ADDRESS
TST R0 ;READ FOR ZEROS
BEQ 3$
PERR13
;
3$: COM (R1) ;COMPL THE WORD
MOV (R1),R0 ;READ IT AGAIN
TST R0
BNE 4$
PERR14
;
4$: CMP #100000,R2
BEQ 5$
CMP #10000,R2 ;CHECK FOR MSB IN 4K BANK
BNE 2$ ;NOT LAST BIT, BRANCH
ASL R2
MOV #160000,R1
BR 2$
5$: RETURN
  
```

8277
 8278
 8279
 8280
 8281 025304 111100
 8282 025306 105700
 8283 025310 001401
 8284 025312 104435
 8285
 8286 025314 105111
 8287 025316 111100
 8288 025320 105700
 8289 025322 001001
 8290 025324 104436
 8291
 8292 025326 C40201
 8293 025330 006302
 8294 025332 050201
 8295 025334 011100
 8296 025336 005700
 8297 025340 001401
 8298 025342 104437
 8299
 8300 025344 005111
 8301 025346 011100
 8302 025350 005700
 8303 025352 001001
 8304 025354 104440
 8305
 8306 025356 022702 100000
 8307 025362 001407
 8308 025364 022702 010000
 8309 025370 001356
 8310 025372 006302
 8311 025374 012701 160000
 8312 025400 000752
 8313 025402 000207

8316 C25404

```

MTP010: SUBST <<MTP010      BYTE ADDRESSING TEST>>
;*****
;SUBTEST      MTP010  BYTE ADDRESSING TEST
;*****

```

```

8317
8318
8319 025404 010402
8320 025406 010403
8321 025410 062702 000004
8322 025414 012713 177777
8323 025420 012763 177777 000002
8324 025426 105013
8325 025430 010401
8326 025432 020201
8327 025434 001420
8328 025436 020301
8329 025440 001007
8330 025442 111100
8331
8332 025444 C22700 000000
8333 025450 001401
8334 025452 104435
8335
8336 025454 005201
8337 025456 000765
8338 025460 111100
8339 025462 122700 177777
8340 025466 001401
8341 025470 104436
8342
8343 025472 005201
8344 025474 000756
8345 025476 112713 177777
8346 025502 005203
8347 025504 020302
8348 025506 001347
8349 025510 000207
8350

```

```

;TEST 3 THIS 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
;WARNING IF YOU OPTOMIZE CHANGE THE PCBUMP FOR THIS ERRGR INCASE OF TRAPS
;IT IS, COMPARE FOR ZEROS
;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

```

8353 025512

```
MTP014: SUBTST <<MTP014 BASIC DOUBLE BIT ERROR TEST>>
;*****
;SUBTEST MTP014 BASIC DOUBLE BIT ERROR TEST
;*****
```

8354
8355
8356
8357
8358

```
; THIS TEST CHECKS THAT A DOUBLE ERROR WILL BE DETECTED
; A BYTE WRITE WITH A DOUBLE ERROR ON A MSV11-P
; WILL BE ABOTRED.
```

8359 025512 104424
8360 025514 005037 002072
8361 025520 012737 000001 002076
8362 025526 012737 060000 002034
8363 025534 104513
8364 025536 012737 003145 002150
8365 025544 104425
8366 025546 012737 103145 002044
8367 025554 005011
8368 025554 005711
8369 025560 023727 002072 000001
8370 025570 104055
8371 025572 104426
8372 025572 104426
8373 025574 042737 020000 002150
8374 025602 023737 002150 002044
8375 025610 001407
8376 025612 012737 177777 002612
8377 025620 013737 002150 002052
8378 025626 104065
8379 025630 104473
8380 025632 005037 002266
8381 025636 104473
8382 025636 104473
8383 025640 005237 002266
8384 025644 005037 002072
8385 025650 112711 000377
8386 025654 105711
8387 025656 023727 002072 000001
8388 025664 001411
8388 025666

```
CACHOFF ;TURN OFF CACHE
LET PARCNT := #0 ;CLEAR PARCNT CLR PARCNT

LET NOPAR := #1 ;SET PARITY ACTION MOV #1,NOPAR

LET ADDRESS := #FIRST ;SET ADDRESS FOR ERROR REPORT MOV #FIRST,ADDRESS

CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
LET CSR := #3145 ;DBE CHECK BITS FOR CSR MOV #3145,CSR

LOADCSR ;WRITE DBE CHECK BITS TO CSR
LET GOOD := #103145 ;GOOD DATA MOV #103145,GOOD

LET (R1) := #0 ;WRITE ZEROS AND DBL ERROR CHK BITS A=0
; CLR (R1)

TST (R1) ;READ A=0 TO GET DOUBLE BIT ERROR
IF PARCNT NE #1 ;WAS BUSPBL ASSERTED????

ERROR +55 ;ERROR CALL ;;MISSED EXPECTED TRAP
END ;

READCSR ;READ CSR FOR CORRECT CHECK BITS AND DBE INDICATOR
BIC #BIT13,CSR ;CLEAR INHIBIT MODE POINTER FROM DATA IF IT EXISTS1
IF CSR NE GOOD THEN ;CHECK IF DOUBLE ERROR BIT IS SET
CMP CSR,GOOD
BEQ L212

SET HEADER ;
MOV #-1,HEADER

LET BAD := CSR ;BAD DATA
MOV CSR,BAD

ERROR +65 ;
END ;

L211:::
ECC1INIT ;ENABLE BUSPBL
CLR PASSNO ;CLEAR LOOP COUNTER
REPEAT ;

L212:::
B30:::
ECC1INIT ;ENABLE BUSPBL
INC PASSNO ;INCREMENT LOOP COUNTER
CLR PARCNT ;CLEAR PARITY ACTION COUNTER
LET (R1) :B= #377 ;WRITE BYTE SHOULD BE ABORTED
MOV B #377,(R1)

TSTB (R1) ;READ R1 TO SEE IF IT IS STILL 0
IF PARCNT NE #1 ;WAS WRITE ABORTED???
CMP PARCNT,#1
BEQ L213

SET HEADER ;
```

8389	025666	012737	177777	002612				MOV #1,HEADER
	025674				LET GOOD := #0	:GOOD DATA		CLR GOOD
8390	025674	005037	002044					
	025700				LET BAD := #377	:BAD DATA		MOV #377,BAD
8391	025700	012737	000377	002052				
8392	025706	104056			ERROR +56	:		
	025710				END	:		
8393	025710	005201					L213:::::	
8394	025710				INC R1	:AND REPEAT ON HIGH BYTE		
	025712				UNTIL PASSNO EQ #2	:		
	025712	023727	002266	000002				CMP PASSNO,#L
	025720	001346						BNE B30
	025722						E30:::::	
8395	025722	005041			CLR (R1)	:CLEAR LUT		
8396	025724	104503			CLR1CSR	:CLEAR CSR		
8397	025726	005037	002072		CLR PARCNT	:CLEAR PARITY TRAP COUNTER		
8398	025732	104423			CACHON	:TURN ON CACHE		
8399	025734	000207			RETURN	:		
8400								

8403 025736

```

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$:     MOVB   R0,(R1)+ ;WRITE A BYTE
        MOVB   R3,(R1)+ ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
        CMP    R1,R5    ;COMPARE TEST LOC TO TOP + 2
        BLO   1$      ;BRANCH IF LOWER
2$:     MOV    -(R1),R2
        CMP    R0,R2    ;TEST THE MEMORY TO SEE IF IT CONTAINS
                        ;THE WORD STORED IN BAKPAT
        BEQ   3$
        PERR2
3$:     CMP    R1,R4    ;KEEP ON TESTING THE MEMORY UNTIL
        BHI   2$      ;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

```

8404
8405
8406
8407
8408
8409
8410 025736 012701 060000
8411 025742 010104
8412 025744 012705 160000
8413 025750 012700 000377
8414 025754 010003
8415 025756 000303
8416 025760 110021
8417 025762 110321
8418 025764 020105
8419 025766 103774
8420
8421 025770 014102
8422 025772 020002
8423
8424 025774 001401
8425 025776 104446
8426
8427 026000 020104
8428 026002 101372
8429 026004 000303
8430 026006 000300
8431 026010 001763
8432
8433 026012 000207

8437 026014

MTP020: SUBTST <<MTP020 SYNDROMES TO CSR ON SINGLE BIT ERROR TEST>>
;*****
;*SUBTEST MTP020 SYNDROMES TO CSR ON SINGLE BIT ERROR TEST
;*****

8438
8439
8440
8441
8442
8443 026014 104424
8444 026016 005000
8445 026020 105037 002264
8446 026024 104513
8447 026026
8448 026026 105237 002264
8449 026032 012704 177777
8450 026036 005037 002324
8451 026042 123727 002264 000001
8452 026052 001003
8453 026052 012705 000001
8454 026056 000402
8455 026060
8456 026060 012705 177776
8457 026064 005237 002324
8458 026070 005204
8459 026072 116402 030164
8460 026076 072227 000005
8461 026102 052702 000004
8462 026106 010237 002150
8463 026112 104425
8464 026114 010011
8465 026116 104503
8466 026120 005711
8467 026122 104426
8468 026124 042737 177757 002150
8469 026132 023727 002150 000020
8470 026142 001407
8471 026150 012737 000020 002044

; 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 ;
; B31:;;;;;
LET PASFLG :B= PASFLG + #1 ;INCREMENT LOOP COUNTER INCB PASFLG
LET R4 := #-1 ;INDEX TO SINGLE BIT ERROR TABLE MOV #-1,R4
LET BITNO := #0 ;CLEAR INNER LOOP COUNTER CLR BITNO
IFB PASFLG EQ #1 ;SELECT DATA TO BE CORRECTED BY PASSNO CLR BITNO
;DATA=0;BIT TO BE CORRECTED IS A ONE CMPB PASFLG,#1
LET R5 := #1 ;BNE L215
MOV #1,R5
ELSE ;
BR L216
; L215:;;;;;
LET R5 := #177776 ;DATA=177776;BIT TO BE CORRECTED IS A ZERO
MOV #177776,R5
; L216:;;;;;
; B32:;;;;;
END ;INCREMENT BIT POINTER
REPEAT ;POINT TO NEXT SET OF CHECK BITS
INC BITNO INC R4
LET R4 := R4 + #1 ;GET NEXT SET OF CHECK BITS
LET R2 :B= PTABLE(R4) MOVB PTABLE(R4),R2
ASH #5,R2 ;SHIFT TO LINE UP IN CSR
BIS #BIT2,R2 ;ENABLE DIAG MODE
LET CSR := R2 ;GET CHECK BITS TO BE WRITTEN
MOV R2,CSR
LOADCSR ;LOAD CSR WITH DATA
LET (R1) := R0 ;WRITE DATA TO TEST ADDRESS
MOV R0,(R1)
CLR1CSR ;CLEAR CSR
TST (R1) ;CORRECT SBE
READCSR ;READ CSR FOR CORRECT SBE BIT AND SYNDROMES
BIC #C20,CSR ;CLEAR ALL BUT SBE INDICATOR
IF CSR NE #20 ;WAS DATA CORRECTED???
CMP CSR,#20
BEQ L217
LET GOOD := #20 ;
LET BAD := CSR ;
MOV #20,GOOD


```

      026150 013737 002150 002052                                MOV CSR,BAD
8472 026156 104060                                ERROR +60                                ;NO ERROR
8473 026160                                END                                        ;
      026160                                SYNREG                                L217:;;;;;
8474 026160 104514                                READCSR                                ;ENABLE SYNDROME BIT REGISTER
8475 026162 104426                                BIC #C3744,CSR                        ;GET SYNDROMES FROM CSR
8476 026164 042737 174033 002150                    LET R3 :B= SBESYN(R4)                ;MASK SYNDROME BITS
8477 026172                                ;GET GOOD SYNDROMES
      026172 116403 026310                                MOV# SBESYN(R4),R3
8478 026176 072327 000005                                ASH #5,R3                            ;SHIFT INTO POSITION
8479 026202 052703 000004                                BIS #BIT2,R3                          ;SET DIAG MODE IN DATA
8480 026206                                IF R3 NE CSR                          ;DO SYNDROME BITS AGREE
      026206 020337 002150                                CMP R3,CSR
      026212 001411                                BEQ L220
8481 026214                                SET HEADER                            ;
      026214 012737 177777 002612                                MOV #-1,HEADER
8482 026222                                LET GOOD := R3                        ;
      026222 010337 002044                                LET BAD := CSR                        ;
8483 026226                                ;GET GOOD SYNDROMES
      026226 C13737 002150 002052                                MOV CSR,BAD
8484 026234 104042                                ERROR +42
8485 026236                                END
      026236                                CLR (R1)                                L220:;;;;;
8486 026236 005011                                IFB PASFLG EQ #1                    ;CLEAR LUT
8487 026240                                ;SHIFT NEW DATA DEPENDING ON PASFLG
      026240 123727 002264 000001                                CMPB PASFLG,#1
      026246 001002                                BNE L221
8488 026250 006305                                ASL R5                                ;SHIFT BITNO TO THE LEFT
8489 026252                                ELSE
      026252 000402                                BR L222
      026254                                L221:;;;;;
8490 026254 000261                                SEC                                    ;SET CARRY BIT AND.....
8491 026256 006105                                ROL R5                                ;ROTATE LEFT
8492 026260                                END
      026260                                UNTIL BITNO EQ #16.                L222:;;;;;
8493 026260                                ;UNTIL ALL BITS ARE DONE
      026260 023727 002324 000020                                CMP BITNO,#16.
      026266 001276                                BNE B32
      026270                                E32:;;;;;
8494 026270 005100                                COM R0                                ;COMPLEMENT DATA AND REPEAT
8495 026272                                UNTILB PASFLG EQ #2                ;UNTIL 2 PASSES ARE COMPLETE!
      026272 123727 002264 000002                                CMPB PASFLG,#2
      026300 001252                                BNE B31
      026302                                E31:;;;;;
8496 026302 104503                                CLR1CSR                                ;CLEAR CSR
8497 026304 104423                                CACHON                                ;TURN CACHE
8498 026306 000207                                RETURN
8499
8500 ;
8501 ;
8502 026310 016 013 023 SBESYN: .BYTE 16,13,23,25,26,31,32,34,43,45,46,51,52,54,61,64
      026313 025 026 031
      026316 032 034 043
      026321 045 046 051
      026324 052 054 061
      026327 064
8503

```

8505 026330

MTPA21: SUBTST <<MTPA21 MARCHING 1'S & 0'S PATTERN TEST>>
 ;*****
 ;*SUBTEST MTPA21 MARCHING 1'S & 0'S PATTERN TEST
 ;*****

8506
 8507 026330 014100
 8508 026332 020200
 8509 026334 001401
 8510 026336 104443
 8511
 8512 026340 000311
 8513 026342 011100
 8514 026344 020300
 8515 026346 001401
 8516 026350 104444
 8517
 8518 026352 020401
 8519 026354 001365
 8520 026356 000207

;READ,BYTESWAP-MODIFY,READ,DOWN
 1#: MOV -(R1),R0 ;V177640
 CMP R2,R0 ;V177642
 BEQ 2# ;V177644
 PERR17 ;V177646
 2#: SWAB (R1) ;V177650
 MOV (R1),R0 ;V177652
 CMP R3,R0 ;V177654
 BEQ 3# ;V177656
 PERR20 ;V177660
 3#: CMP R4,R1 ;V177662 ;DONE?
 BNE 1# ;V177664 ;NO - LOOP
 RETURN ;V177666 ;YES - RETURN

8521
 8522 026360
 8523 026360 011100
 8524 026362 020300
 8525 026364 001401
 8526 026366 104444
 8527
 8528 026370 000311
 8529 026372 011100
 8530 026374 020200
 8531 026376 001401
 8532 026400 104443

MTPB21: ;READ,BYTESWAP-MODIFY,READ,UP
 1#: MOV (R1),R0 ;V177640
 CMP R3,R0 ;V177642
 BEQ 2# ;V177644
 PERR20 ;V177646
 2#: SWAB (R1) ;V177650
 MOV (R1),R0 ;V177652
 CMP R2,R0 ;V177654
 BEQ 3# ;V177656
 PERR17 ;V177660

8533
 8534 026402 062701 000002
 8535 026406 020501
 8536 026410 001363
 8537 026412 000207

3#: ADD #2,R1 ;V177662
 CMP R5,R1 ;V177666 ;DONE?
 BNE 1# ;V177670 ;NO - LOOP
 RETURN ;V177672 ;YES - RETURN

8538
 8539 026414
 8540 026414 011100
 8541 026416 020200
 8542 026420 001401
 8543 026422 104443
 8544
 8545 026424 000311
 8546 026426 011100
 8547 026430 020300
 8548 026432 001401
 8549 026434 104444

MTPC21: ;READ,BYTESWAP-MODIFY,READ,UP
 1#: MOV (R1),R0 ;V177640
 CMP R2,R0 ;V177642
 BEQ 2# ;V177644
 PERR17 ;V177646
 2#: SWAB (R1) ;V177650
 MOV (R1),R0 ;V177652
 CMP R3,R0 ;V177654
 BEQ 3# ;V177656
 PERR20 ;V177660

8550
 8551 026436 062701 000002
 8552 026442 020501
 8553 026444 001363
 8554 026446 000207

3#: ADD #2,R1 ;V177662
 CMP R5,R1 ;V177666 ;DONE?
 BNE 1# ;V177670 ;NO - LOOP
 RETURN ;V177672 ;YES - RETURN

```

8557 026450
8558 026450 014100
8559 026452 020300
8560 026454 001401
8561 026456 104444
8562
8563 026460 000311
8564 026462 011100
8565 026464 020200
8566 026466 001401
8567 026470 104443
8568
8569 026472 020401
8570 026474 001365
8571 026476 000207
8572

MTPD21: ;READ.BYTESWAP-MODIFY.READ.DOWN
1$: MOV -(R1),R0 ;V177640
CMP R3,R0 ;V177642
BEQ 2$ ;V177644
PERR20 ;V177646

2$: SWAB (R1) ;V177650
MOV (R1),R0 ;V177652
CMP R2,R0 ;V177654
BEQ 3$ ;V177656
PERR17 ;V177660

3$: CMP R4,R1 ;V177662 ;DONE?
BNE 1$ ;V177664 ;NO - LOOP
RETURN ;V177666 ;YES - RETURN
```

```

8575 026500 MTP022: SUBST <<MTP022 REFRESH & SHIFTING DIAGONAL TEST>>
;*****
;SUBTEST MTP022 REFRESH & SHIFTING DIAGONAL TEST
;*****
; (1) WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
; (2) IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
; (3) WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
; (WITH CACHE OFF).
;
KDIAG=8. ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
FOR EVEN := #1 TO #2 ;FOR DATA & COMPLEMENT DATA
MOV #1,EVEN
B33:;;;;;
IF EVEN EQ #1
CMP EVEN,#1
BNE L225
LET R2 := ZEROS
MOV ZEROS,R2
LET R3 := ONES
MOV ONES,R3
ELSE
BR L226
L225:;;;;;
MOV ONES,R2
LET R2 := ONES
MOV ONES,R2
LET R3 := ZEROS
MOV ZEROS,R3
END ;OF IF EVEN
L226:;;;;;
FOR STRIPES := #0 TO #KDIAG-1 ;FOR THE NUMBER OF STRIPES
CLR STRIPES
B34:;;;;;
;WRITE LOOP
CACHON ;TURN CACHE ON
LET COUNT := STRIPES
MOV STRIPES,COUNT
LET R1 := #FIRST
MOV #FIRST,R1
WHILE R1 LOS #LAST
B35:;;;;;
CMP R1,#LAST
BHI L227
IF COUNT LT #0 THEN LET COUNT := #KDIAG-1
TST COUNT
BGE L230
MOV #KDIAG-1,COUNT
L230:;;;;;
IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1
BIT #374,R1
BNE L231
DEC COUNT
L231:;;;;;
IF COUNT NE #0
TST COUNT
BEQ L232
LET (R1) := R2
MOV R2,(R1)

```

8600	026624			LET 2(R1) := R2		MOV R2,2(R1)
8601	026624	010261	000002	ELSE		BR L233
	026630	000403				L232:
8602	026632			LET (R1) := R3		MOV R3,(R1)
8603	026632	010311		LET 2(R1) := R3		MOV R3,2(R1)
	026634	010361	000002	END ;OF IF COUNT		L233:
8604	026640			LET COUNT := COUNT - #1		DEC COUNT
8605	026640	005337	002370	LET R1 := R1 + #4		ADD #4,R1
8606	026644	062701	000004	END ;OF WHILE		BR B35
8607	026650	000743				L227:
	026652					E35:
8608				;END OF WRITE LOOP		
8609				IF DIAGFLAG IS FALSE THEN #CALL REFRESH		
8610	026652	005737	002004			TST DIAGFLAG
	026656	001002				BNE L234
	026660	004737	027054			JSR PC,REFRESH
	026664					L234:
8611				;READ LOOP		
8612	026664	013737	002366 002370	LET COUNT := STRIPES		MOV STRIPES,COUNT
8613	026672	012701	060000	LET R1 := #FIRST		MOV #FIRST,R1
8614	026676	104424		CACHOFF ;TURN CACHE OFF		

8616	026700			WHILE R1 LOS #LAST	B36:::~::~
	026700				CMP R1,#LAST
	026700	020127	157776		BHI L235
	026704	101046			
8617	026706			IF COUNT LT #0 THEN LET COUNT := #KDIAG-1	TST COUNT
	026706	005737	002370		BGE L236
	026712	002003			MOV #KDIAG-1,COUNT
	026714	012737	000007		L236:::~::~
	026722				#1
8618	026722			IF #374 OFF.IN R1 THEN LET COUNT := COUNT	BIT #374,R1
	026722	032701	000374		BNE L237
	026726	001002			DEC COUNT
	026730	005337	002370		L237:::~::~
	026734				
8619	026734			IF COUNT NE #0	TST COUNT
	026734	005737	002370		BEQ L240
	026740	001412			
8620	026742			LET R0 := (R1)	MOV (R1),R0
	026742	011100			
8621	026744			IF R2 NE R0	CMP R2,R0
	026744	020200			BEQ L241
	026746	001401			
8622	026750	104443		PERR17	
8623	026752			END ;OF IF R2	L241:::~::~
	026752				
8624	026752			LET R0 := 2(R1)	MOV 2(R1),R0
	026752	016100	000002		
8625	026756			IF R2 NE R0	CMP R2,R0
	026756	020200			BEQ L242
	026760	001401			
8626	026762	104443		PERR17	
8627	026764			END ;OF IF R2	L242:::~::~
	026764				
8628	026764			ELSE	BR L243
	026764	000411			L240:::~::~
	026766				
8629	026766			LET R0 := (R1)	MOV (R1),R0
	026766	011100			
8630	026770			IF R3 NE R0	CMP R3,R0
	026770	020300			BEQ L244
	026772	001401			
8631	026774	104444		PERR20	
8632	026776			END ;OF IF R3	L244:::~::~
	026776				
8633	026776			LET R0 := 2(R1)	MOV 2(R1),R0
	026776	016100	000002		
8634	027002			IF R3 NE R0	CMP R3,R0
	027002	020300			BEQ L245
	027004	001401			
8635	027006	104444		PERR20	
8636	027010			END ;OF IF R3	L245:::~::~
	027010				
8637	027010			END ;OF IF COUNT	L243:::~::~
	027010				
8638	027010			LET COUNT := COUNT #1	DEC COUNT
	027010	005337	002370		
8639	027014			LET R1 := R1 + #4	

```

8640 027014 062701 000004          END ;OF WHILE          ADD #4,R1
      027020          BR B36
      027020 000727          L235:;;;;;;;;
      027022          E36:;;;;;;;;
      027022          ;END OF READ LOOP
8641          ;END OF FOR STRIPES
8642          END ;OF FOR STRIPES          INC STRIPES
8643 027022 005237 002366          CMP STRIPES,#KDIAG-1
      027026 023727 002366 000007          BLE B34
      027034 003643          E34:;;;;;;;;
      027036          END ;OF FOR EVEN          INC EVEN
8644 027036 005237 002364          CMP EVEN,#2
      027042 023727 002364 000002          BLE B33
      027050 003616          E33:;;;;;;;;
      027052          RETURN
8645 027052 000207          REFRESH:SUBTST <<SUBR REFRESH DELAY>>
8646          ;*****
8647 027054          ;*SUBTEST SUBR REFRESH DELAY
          ;*****
          ;DISTURB EACH ROW FOR > 3.2 MS
          FOR RO := #FIRST TO #FIRST+374 BY #4
8648          MOV #FIRST,RO
8649 027054 012700 060000          B37:;;;;;;;;
      027054 004737 027124          CALL REFSUB
      027060          END ;OF FOR RO
8650 027064 062700 000004          ADD #4,RO
8651 027064 020027 060374          CMP RO,#FIRST+374
      027070 003771          BLE B37
      027076          E37:;;;;;;;;
      027076          LET RO := #FIRST+BIT14
          MOV #FIRST+BIT14,RO
8652 027076 012700 120000          WHILE RO LOS #LAST+BIT14+374
8653 027102          B40:;;;;;;;;
      027102 020027 020372          CMP RO,#LAST+BIT14+374
      027106 101005          BHI L246
8654 027110 004737 027124          CALL REFSUB
8655 027114          LET RO := RO + #4
          ADD #4,RO
8656 027120 062700 000004          END ;OF WHILE          BR B40
      027120 000770          L246:;;;;;;;;
      027122          E40:;;;;;;;;
      027122          RETURN
8657 027122 000207          REFSUB: MOV #640,R4
8658 027124 012704 000640          ADD #2,RO
8659 027130 062700 000002          1$: COM -(RO)
8660 027134 005140          COM (RO)+
8661 027136 005120          COM (RO)
8662 027140 005110          COM (RO)
8663 027142 005110          SOB R4,1$
8664 027144 077405          SUB #2,RO
8665 027146 162700 000002          RETURN
8666 027152 000207
    
```

8669 027154

MTPA24: SUBTST <<MTPA24 FAST GALLOPING PATTERN TEST>>
;*****
;*SUBTEST MTPA24 FAST GALLOPING PATTERN TEST
;*****

8670
8671
8672
8673
8674
8675
8676
8677
8678
8679
8680
8681
8682
8683
8684
8685
8686
8687
8688
8689
8690
8691
8692
8693
8694
8695 027154 011100
8696 027156 020004
8697 027160 001401
8698 027162 104447
8699
8700 027164 011200
8701 027166 020003
8702 027170 001401
8703 027172 104450
8704
8705 027174 062702 000400
8706 027200 020205
8707 027202 101764
8708
8709 027204 062701 000002
8710 027210 000137 027214

;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 5-8 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
1\$: MOV (R1),R0 ;V177640 ;READ 'A'
CMP R0,R4 ;V177642 ;CHECK 'A'
BEQ 2\$;V177644 ;BR IF OK
PERR23 ;V177646 ;REPORT ERROR
2\$: MOV (R2),R0 ;V177650 ;READ 'B'
CMP R0,R3 ;V177652 ;CHECK 'B'
BEQ 3\$;V177654 ;BR IF OK
PERR24 ;V177656 ;REPORT ERROR
3\$: ADD #400,R2 ;V177660 ;BUMP 'B'
CMP R2,R5 ;V177664 ;AT END YET?
BLOS 1\$;V177666 ;BR IF NO
ADD #2,R1 ;V177670 ;BUMP 'A'
JMP @MTPB24 ;V177674 ;GOTO V177260

8713 027214

```
MTPB24: SUBTST <<MTPB24 FAST GALLOP PART B>>
;*****
;SUBTEST MTPB24 FAST GALLOP PART B
;*****
```

8714

8715 027214 010411
 8716 027216 020105
 8717 027220 001001
 8718 027222 000207
 8719 027224 000137 027230
 8720
 8721 027230

```
;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
;*****
```

8722

8723 027230 010102
 8724 027232 011100
 8725 027234 020004
 8726 027236 C01401
 8727 027240 104447
 8728 027242 000137 027174
 8729

```
;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
```

8732 027246

```
MTPA26: SUBTST <<MTPA26 RANDOM DATA (WRITE)>>
;*****
; *SUBTEST MTPA26 RANDOM DATA (WRITE)
;*****
1$: JMP @MTPC26 ;V177640 GOTO V172360
MOV R2,(R1)+ ;V177644
MOV R3,(R1)+ ;V177646
SOB RO,1$ ;V177650
RETURN ;V177652
```

8733 027246 000137 027316
8734 027252 010221
8735 027254 010321
8736 027256 077005
8737 027260 000207
8738
8739 027262

```
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$: COM RANODD ;V177664
SOB RO,1$ ;V177670
RETURN ;V177672
.DSABL LSB
.ENABL AMA
```

8740
8741
8742 027262 000137 027316
8743 027266 020221
8744 027270 001401
8745 027272 104451
8746 027274 005127
8747 027276 000000
8748 027300 020321
8749 027302 001401
8750 027304 104451
8751 027306 005167 177764
8752 027312 077015
8753 027314 000207
8754
8755
8756
8757 027316

```
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
```

8758
8759
8760
8761
8762
8763 027316 073427 000007
8764 027322 060305
8765 027324 005504
8766 027326 060204
8767 027330 062705 001057
8768 027334 000240
8769
8770 027336

```
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
```

8771 027336 005504
8772 027340 062704 047401
8773 027344 010503
8774 027346 010402
8775 027350 000137 027252

8778 027354

```

MTP030: SUBTST <<MTO030      FLUSH OUT DBE'S>>
;*****
;*SUBTEST      MTO030  FLUSH OUT DBE'S
;*****
1$:  MOV      (R0),R2          ;V177640
      MOV      R2,(R0)+       ;V177642
      SOB      R1,1$          ;V177644
      RETURN                     ;V177646
    
```

8779 027354 011002
8780 027356 010220
8781 027360 077103
8782 027362 000207
8783
8 J4 027364

```

MTP031: SUBTST <<MTP031      SOB-A-LONG TEST>>
;*****
;*SUBTEST      MTP031  SOB-A-LONG TEST
;*****
    
```

8785
8786 027364 000000
8787 027366 077001
8788 027370 005167 177772
8789 027374 020167 177766
8790 027400 001403
8791 027402 104454
8792 027404 010167 177756
8793 027410 005167 177752
8794 027414 010200
8795
8796 027416 010503
8797 027420 005725
8798 027422 010504
8799 027424 020537 002522
8800 027430 001001
8801 027432 000207
8802
8803 027434 014344
8804 027436 001376
8805 027440 000752
8806 000056
8807

```

      .DSABL  AMA
      0
1$:  SOB      R0,1$          ;MOVE TERMINATOR
      COM      1$           ;SOB TILL R0 UNDERFLOWS
      CMP      R1,1$       ;WRITE COMPLEMENT OF SOB
      BEQ      2$           ;READ & CHECK FOR NOT "SOB R0, DOT"
      PERR30
      MOV      R1,1$
2$:  COM      1$           ;CORRECT SOB INSTRUCTION
      MOV      R2,R0       ;REINITIALIZE SOB CONSTANT
      ;UPDATE MOVE REGISTERS
      MOV      R5,R3
      TST      (R5)+       ;BUMP (SAFELY) BY 2
      MOV      R5,R4
      CMP      R5,@LINK1  ;DONE?
      BNE      3$         ;NO - SKIP
      RETURN                    ;YES
3$:  MOV      -(R3),-(R4)
      BNE      3$
      BR       1$
SOBLENGTH=.-MTP031
      .ENABL  AMA
    
```

8835 027442

MTP032: SUBTST <<MTP032 WRITE RECOVERY TEST>>
;*****
;*SUBTEST MTP032 WRITE RECOVERY TEST
;*****

8836
8837
8838
8839
8840
8841 027442 012401
8842 027444 020102
8843 027446 001401
8844 027450 104430
8845 027452 077305
8846 027454 013703 002522
8847 027460 012400
8848 027462 020005
8849 027464 001401
8850 027466 104427
8851 027470 C77305
8852 027472 000207

;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.
1\$: MOV (R4)+,R1 ;V177640 ;GET DATA FROM LOWER 1/2 BANK
CMP R1,R2 ;V177642 ;IS IT #5141?
BEQ 2\$;V177644 ;YES - SKIP
PERRO2 ;V177646 ;NO - TAKE ERROR TRAP
2\$: SOB R3,1\$;V177650 ;LOOP FOR 1/2 BANK
MOV @LINK1,R3 ;V177652 ;RESTORE LOOP SIZE
3\$: MOV (R4)+,R0 ;V177656 ;GET DATA FROM UPPER 1/2 BANK
CMP R0,R5 ;V177660 ;IS IT #110?
BEQ 4\$;V177662 ;YES - SKIP
PERRO1 ;V177664 ;NO- TAKE ERROR TRAP
4\$: SOB R3,3\$;V177666 ;LOOP FOR 1/2 BANK
RETURN

8855 027474

MTP033: SUBTST <<MTP033 BRANCH GOBBLE TEST>>
;*****
;*SUBTEST MTP033 BRANCH GOBBLE TEST
;*****

8856
8857 027474 000000
8858 027476 000000
8859 027500 000261
8860 027502 105511
8861 027504 100402
8862 027506 105212
8863 027510 000773
8864
8865
8866 027512 102401
8867 027514 104461
8868
8869 027516 000242
8870 027520 105212
8871 027522 103402
8872 027524 102001
8873 027526 100401
8874 027530 104461
8875
8876
8877
8878 027532 010701
8879 027534 162701 000036
8880 027540 010102
8881 027542 005202
8882
8883
8884 027544 010503
8885 027546 005725
8886 027550 010504
8887
8888
8889 027552 020537 002522
8890 027556 001001
8891 027560 000207
8892
8893
8894 027562 014344
8895 027564 001376
8896 027566 005011
8897 027570 000743
8898 000076
8899

.DSABL AMA
0 ;MOVE TERMINATOR
BGTEST: 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 ;CLEAR V-BIT
INCB (R2) ;INCREMENT HIGH BYTE OF "BGTEST" ONCE MORE
BCS 3# ;BR IF C-BIT SET (SHOULD NOT BE)
BVC 3# ;BR IF V-BIT CLEAR (SHOULD NOT BE)
BMI 4# ;BR IF N-BIT SET (SHOULD BE)
3#: PERR35 ;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)+ ;BUMP (SAFELY) BY 2
MOV R5,R4

;DONE?
CMP R5,#LINK1 ;DONE?
BNE 6# ;NO - SKIP
RETURN ;YES - RETURN

;MOVE CODE 1 LOCATION
6#: MOV -(R3),-(R4)
BNE 6#
CLR (R1) ;CLEAR TEST WORD "BGTEST"
BR BRGOBB ;RUN MOVED CODE AGAIN

GBLENGTH=-MTP033
.ENABL AMA

8901 027572

8902 027572 010220
8903 027574 077102
8904 027576 000207
8905 027600 012401
8906 027602 020102
8907 027604 001402
8908 027606 104430
8909 027610 000240
8910 027612 077306
8911 027614 000207

```
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
```

```

8913 027616 MTP035:SUBTST <<MTP035 WORST CASE NOISE PARITY TEST>>
;*****
;SUBTEST MTP035 WORST CASE NOISE PARITY TEST
;*****
8914 027616 012737 000003 002076 MOV #3,NOPAR ;SET PARITY TRAPS TO RETURN TO "PARTHERE"
8915
8916 027624 FOR RO := #FIRST TO #LAST BY #4000 MOV #FIRST,RO
027624 012700 060000 B41:;;;;;
027630
8917 027630 012737 000005 002150 MOV #BIT2!BIT0,CSR ;SET WRITE WRONG PARITY & PAR. TRAPS INTO CSR
8918 027636 104425 LOADCSR
8919 027640 012737 027674 002304 MOV #1#,PARTHERE
8920 027646 011010 MOV (RO),(RO) ;WWP TEST LOCATION
8921 027650 005710 TST (RO)
8922 027652 010037 002034 MOV RO,ADDRESS
8923 027656 104050 ERROR +50
8924 027660 004737 050354 CALL PERBNK
8925 027664 032763 002000 002666 BIT #BIT10,CONFIG+2(R3)
8926 027672 001002 BNE 2#
8927 027674 104426 1#: READCSR
8928 027676 104512 ERGEN
8929
8930 027700 104503 2#: CLR1CSR
8931 027702 011010 MOV (RO),(RO) ;CLEAR WRONG PARITY IN MEMORY
8932 027704 012737 000001 002150 MOV #BIT0,CSR
8933 027712 104425 LOADCSR
8934 027714 012737 027726 002304 MOV #3#,PARTHERE
8935 027722 005710 TST (RO)
8936 027724 000405 BR 4#
8937 027726 010037 002034 3#: MOV RO,ADDRESS
8938 027732 104050 ERROR +50
8939 027734 004737 050354 CALL PERBNK
8940 027740 4#: END; OF FOR
027740 062700 004000 ADD #4000,RO
027744 020027 157776 CMP RO,#LAST
027750 003727 BLE B41
027752 E41:;;;;;
8941
8942 027752 005037 002076 CLR NOPAR ;RESET PARITY TRAP ACTION
8943 027756 000207 RETURN

```

8945 027760

```

MTP036: SUBSTST      <<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
; CLRB PASFLG     ;CLEAR PASFLG
; CBREG          ;ENABLE CHECK/SYNDROME BIT REGISTER
; REPEAT          ;
;
; B42:;;;;;
; 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 L247
; DATA=0;BIT TO BE CORRECTED IS A ONE
; MOV #1,R5
;
; ELSE
;
; BR L250
;
; L247:;;;;;
; DATA=177776;BIT TO BE CORRECTED IS A ZERO
; MOV #177776,R5
;
;
; L250:;;;;;
;
; B43:;;;;;
; 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 SBE
; WAS DATA CORRECTED???
; CMP (R1),R5
; BEQ L251
; MOV #60000,ADDRESS
; MOV R2,CHECK
; MOV R5,TSTDAT
;
; LET PASFLG :B= PASFLG + #1
; LET R4 := #-1
; LET BITNO := #0
; IFB PASFLG EQ #1
;
; LET R5 := #1
;
; LET R5 := #177776
; END
; REPEAT
;
; INC BITNO
; LET R4 := R4 + #1
; LET R2 :B= PTABLE(R4)
; ASH #5,R2
; BIS #BIT2,R2
; LET CSR := R2
; LOADCSR
; LET (R1) := R0
; TST (R1)
; IF (R1) NE R5
;
; LET ADDRESS := #60000
; LET CHECK := R2
; LET TSTDAT := R5
; LET TSTDAT+2 := (R1)
  
```

8946
 8947
 8948
 8949
 8950 027760 104424
 8951 027762 105037 002264
 8952 027766 104513
 8953 027770
 8954 027770 105237 002264
 8955 027774 012704 177777
 8956 030000 005037 002324
 8957 030004 123727 002264 000001
 8958 030012 001003
 8959 030014 012705 000001
 8960 030020 000402
 8961 030022
 8962 030022 012705 177776
 8963 030026
 8964 030026 005237 002324
 8965 030032 005204
 8966 030034 116402 030164
 8967 030040 072227 000005
 8968 030044 052702 000004
 8969 030050 010237 002150
 8970 030054 104425
 8971 030056 010011
 8972 030060 005711
 8973 030062 021105
 8974 030066 001412 060000 002034
 8975 030074 012737 002314
 8976 030100 010237 002246
 8977 030104


```

      030104 011137 002250
8978 030110 104052
8979 030112
      030112
8980 030112 005011
8981 030114
      030114 123727 002264 000001
      030122 001002
8982 030124 006305
8983 030126
      030126 000402
      030130
8984 030130 000261
8985 030132 006105
8986 030134
      030134
8987 030134
      030134 023727 002324 000020
      030142 001331
      030144
8988 030144 005100
8989 030146
      030146 123727 002264 000002
      030154 001305
      030156
8990 030156 104503
8991 030160 104423
8992 030162 000207
8993
8994
8995
8996 030164 002 007 037
      030167 031 032 025
      030172 026 020 057
      030175 051 052 045
      030200 046 040 075
      030203 070

      ERROR +52 ;NO ERROR
      END ;
      CLR (R1) ;CLEAR LUT
      IFB PASFLG EQ #1 ;SHIFT NEW DATA DEPENDING ON PASFLG
      ;CMPB PASFLG.#1
      ;ASL R5 ;SHIFT BITNO TO THE LEFT
      ELSE ;
      ;BR L253
      ;L252:;;;;;
      SEC ;SET CARRY BIT AND.....
      ROL R5 ;ROTATE LEFT
      END ;
      UNTIL BITNO EQ #16. ;UNTIL ALL BITS ARE DONE
      ;L253:;;;;;
      ;CMP BITNO.#16.
      ;BNE B43
      ;E43:;;;;;
      COM R0 ;COMPLEMENT DATA AND REPEAT
      UNTILB PASFLG EQ #2 ;UNTIL 2 PASSES ARE COMPLETE!
      ;CMPB PASFLG.#2
      ;BNE B42
      ;E42:;;;;;
      CLR1CSR ;CLEAR CSR
      CACHON ;TURN CACHE
      RETURN ;

      ;
      ; MSV11-P SINGLE BIT ERROR CHECK BIT TABLE
      ;
      ;PTABLE: .BYTE 2,7,37,31,32,25,26,20,57,51,52,45,46,40,75,70

```

8998 030204

```

MTP037: SUBST <<MTP037      CHECK ECC DISABLE TEST>>
;*****
;SUBTEST      MTP037 CHECK ECC DISABLE TEST
;*****
  
```

```

8999      ; THIS TEST CHECKS THAT ECC CAN BE DISABLED AND THAT
9000      ; NO CORRECTION TAKES PLACE WITH ECC DISABLED.
9001      ;
9002      ;
9003 030204 104424      CACHOFF      ;TURN OFF CACHE
9004 030206      LET GOOD := #0      ;GOOD DATA FOR ERROR PRINT OUT      CLR GOOD
          030206 005037 002044
9005 030212      LET CHECK := #0      ;CLEAR CHECK BIT FIELD
          030212 005037 002314      CLR CHECK
9006 030216 104475      CB1CSR      ;ENABLE SYNDROME/CHECK BIT REGISTER
9007 030220      LET CHECK := #100      ;SBE CHECK BITS      MOV #100,CHECK
          030220 012737 000100 002314
9008 030226 104475      CB1CSR      ;WRITE CHECK BITS TO CB REGISTER
9009 030230      LET (R1) := #0      ;WRITE CHECK BITS TO MEMORY      CLR (R1)
          030230 005011
9010 030232      IF (R1) NE #0      ;WAS CORRECTION MADE????      TST (R1)
          030232 005711      BEQ L256
          030234 001406
          LET BAD := (R1)      ;YES IT WAS.....ERROR      MOV (R1),BAD
          030236 011137 002052
          LET ADDRESS := #60000 ;
          030242 012737 060000 002034      MOV #60000,ADDRESS
9013 030250 104037      ERROR +37      ;
9014 030252      END      ;
          030252
9015 030252 104423      CACHON      ;TURN ON CACHE
9016 030254 000207      RETURN      ;
          L256:;:;:;:;
  
```

9019 030256

MTP041: SUBTST <<MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST>>
;*****
;SUBTEST MTP041 ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
;*****

9020
9021
9022
9023

; THIS TEST CHECKS TO SEE IF THE CORRECT ADDRESS APPEARS
; IN CSR BITS 5-11 ON A DOUBLE ERROR.

9024 030256 013704 002102
030256 072427 000011
9025 030262 072427 000011
9026 030266 042704 170037
9027 030272 012700 177740
9028 030276 012701 060000
030276 162701 004000
030302 162701 004000
9029 030306 105037 002264
030306 105037 002264
9030 030312 05037 002314
9031 030316 104475
9032 030320 030320
9033 030320 105237 002264
9034 030324 062700 000040
030324 062701 004000
9035 030330 062701 004000
030330 062701 004000
9036 030334 012737 001340 002314
030334 104475
9037 030342 005011
030344 104503
9039 030346 104503
9040 030350 005711
9041 030352 104426

LET R4 := BANK ;GET STARTING BANK NUMBER
MOV BANK,R4
ASH #9,R4 ;SHIFT INTO POSITION TO MATCH ADDRESS IN CSR
BIC #C7740,R4 ;CLEAR OFF EXTRANEOUS 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 := #0 ;INIT PASFLG
CLR PASFLG
CLR CHECK ;CLEAR CHECK BIT FIELD TO BE LOADED
CB1CSR ;ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT
B44:;;;;
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)
CLR CSR ;CLEAR CSR
TST (R1) ;READ ADDRESS TO GET DOUBLE ERROR
READCSR ;READ CSR FOR CORRECT ADDRESS

```

9042 030354          LET R5 := CSR          ;
      030354 013705 002150          ;                               MOV CSR,R5
9043 030360 042705 170037          BIC #C7740,R5          ;
9044 030364          LET R2 := R0          ;GET CORRECT ADDRESS
      030364 010002          ;                               MOV R0,R2
9045 030366 060402          ADD R4,R2          ;ADD STARTING BANK TO DOUBLE BIT ADDRESS
9046 030370 000240          NOP          ;DEBUG AIDE
9047 030372          IF R2 NE R5          ;DO ADRESSES AGREE?
      030372 020205          ;                               CMP R2,R5
      030374 001405          ;                               BEQ L257
9048 030376          LET BAD := R2          ;
      030376 010237 002052          ;                               MOV R2,BAD
9049 030402          LET GOOD := R5          ;
      030402 010537 002044          ;                               MOV R5,GOOD
9050 030406 104455          PERR31          ;NO ERROR
9051 030410          END          ;
      030410          ;                               L257:::::
9052 030410          LET (R1) := #0          ;
      030410 C05011          ;                               CLR (R1)
9053 030412 104475          CB1CSR          ;ENABLE CHECK/SYNDROME BIT REGISTER
9054 030414          UNTILB PASFLG EQ PASCNT ;DO 16K AT A TIME
      030414 123737 002264 002570          ;                               CMPB PASFLG,PASCNT
      030422 001336          ;                               BNE B44
      030424          ;                               E44:::::
9055 030424 104503          CLR1CSR          ;
9056 030426 000207          RETURN          ;
9057

```

```

9060 030430 MTP042: SUBTST <<MTP042 EXTENDED ADDRESS TO CSR ON ERROR TEST>>
;*****
;SUBTEST MTP042 EXTENDED ADDRESS TO CSR ON ERROR TEST
;*****
;
; THIS TESTS THE EXTENDED Q-BUS ADDRESS IN THE
; CSR BY CAUSING A SINGLE ERROR, ENABLING BIT # 14, THEN CHECKING
; FOR THE PROPER ADDRESS IN THE CSR.
;
9061
9062
9063
9064
9065
9066 030430 104424 CACHOFF ;TURN OFF CACHE MEMORY
9067 030432 LET R4 := BANK ;GET BANK NUMBER TO FIGURE OUT EXTENDED ADDRESS
030432 013704 002102 MOV BANK,R4
9068 030436 IF BANK EQ #177
030436 023727 002102 000177 CMP BANK,#177
030444 001004 BNE L261
9069 030446 LET PASCNT := #1 ;DO ONCE IN LAST BANK
030446 012737 000001 002570 MOV #1,PASCNT
9070 030454 ELSE
030454 000403 BR L262
030456 L261:::::
9071 030456 LET PASCNT := #2
030456 012737 000002 002570 MOV #2,PASCNT
9072 030464 END
030464 L262:::::
9073 030464 042704 177607 BIC #C170,R4 ;CLEAR OFF LOWER BITS
9074 030470 072427 000002 ASH #2,R4 ;SHIFT TO LINE UP WITH CSR
9075 030474 052704 040000 BIS #BIT14,R4 ;SET EXTENDED ADDRESS BIT
9076 030500 062737 000400 172352 ADD #400,KIPARS ;SET UP PAR TO POINT TO TOP OF A BANK
9077 030506 LET PASFLG := #0 ;INIT LOOP COUNTER
030506 105037 002264 CLR PASFLG
9078 030512 LET R5 := BANK ;R5 GETS THE BANK NUMBER
030512 013705 002102 MOV BANK,R5
9079 030516 BIC #C7,R5 ;CLEAR ALL BUT THE LOWER BITS
9080 030522 072527 000011 ASH #9,R5 ;ROTATE INTO POSITION
9081 030526 052705 000020 BIS #BIT4,R5 ;SET UP SBE INDICATOR ;;DATA TO BE EXPECTED
9082 030532 104513 CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
9083 030534 REPEAT
030534 B45:::::
9084 030534 105237 002264 INCB PASFLG ;INCR LOOP COUNTER
9085 030540 LET CSR := #104 ;WRITE CHECK BITS TO CSR WITH DIAG MODE
030540 012737 000104 002150 MOV #104,CSR
9086 030546 104425 LOADCSR ;LOAD CSR WITH DATA
9087 030550 LET (R1) := #0 ;WRT ZEROS AT A=0 AND SINGLE ERROR BITS
030550 005011 CLR (R1)
9088 030552 104503 CLR1CSR ;CLEAR CSR
9089 030554 005711 TST (R1) ;READ A=0;DATA BIT 0 SHOULD BE CORRECTED TO A 1
9090 030556 104426 READCSR ;READ CSR FOR DATA
9091 030560 042737 020000 002150 BIC #BIT13,CSR ;CLEAR POSSIBLE INHIBIT MODE IN DATA "CSR"
9092 030566 IF CSR NE R5 THEN ;HAS SINGLE ERROR BITS SET IN CSR?
030566 023705 002150 CMP CSR,R5
030572 001406 BEQ L263
9093 030574 LET BAD := CSR ;
030574 013737 002150 002052 MOV CSR,BAD
9094 030602 LET GOOD := R5 ;
030602 010537 002044 MOV R5,GOOD
9095 030606 104023 ERROR +23 ;
9096 030610 END ;

```

```

          030610
9097 030610          LET CSR := #40000 ;WRITE EQB BIT TO CSR          L263:;;;;;;;;
          030610 012737 040000 002150          MOV #40000,CSR
9098 030616 104425
9099 030620 104426
9100 030622 042737 020000 002150
          LOADCSR ;
          READCSR ;READ FOR CORRECT EXTENDED Q-BUS ADDRESS
9101 030630 023704 002150          BIC #BIT13,CSR ;CLEAR INHIBIT MODE POINTER IN DATA
          IF CSR NE R4 THEN ;READ EQB ADDRESS
          030634 001411          CMP CSR,R4
          BEQ L264
9102 030636          LET BAD := CSR ;
          030636 013737 002150 002052          MOV CSR,BAD
9103 030644          LET GOOD := R4 ;
          030644 010437 002044          MOV R4,GOOD
9104 030650          SET HEADER ;
          030650 012737 177777 002612          MOV #1,HEADER
9105 030656 104023
          ERROR +23 ;
9106 030660          END ;
          030660          L264:;;;;;;;;
9107 030660          LET (R1) := #0 ;CLEAR LUT
          030660 C05011          CLR (R1)
9108 030662          LET R1 := #137776 ;SET UP NEW ADDRESS
          030662 012701 137776          MOV #137776,R1
9109 030666 062705 000740
          ADD #740,R5 ;ADD TO GET NEW ADDRESS
9110 030672 052704 001000          BIS #BIT9,R4 ;SET BIT 9 SINCE WE ARE ASSERTING A1 ON PASS2
9111 030676 104513          CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
9112 030700          UNTILB PASFLG EQ PASCNT ;LOOP 2 TIMES
          030700 123737 002264 002570          CMPB PASFLG,PASCNT
          030706 001312          BNE B45
          030710          E45:;;;;;;;;
9113 030710 104503          CLR1CSR ;CLEAR CSR
9114 030712 104423          CACHON ;TURN ON CACHE
9115 030714 000207          RETURN

```

9117 030716

```
MTP043: SUBTST <<MTP043 WRITE BYTE CLEARS SINGLE BIT ERROR TEST>>
;*****
;SUBTEST      MTP043  WRITE BYTE CLEARS SINGLE BIT ERROR TEST
;*****
```

9118
9119
9120
9121
9122

```
; 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.
```

9123 030716 104424
9124 030720 104513
9125 030722 105037 002264
9126 030726

```
;
;CACHOFF          ;TURN OFF CACHE
CBREG             ;ENABLE CHECK/SYNDROME BIT REGISTER
CLR B PASFLG     ;CLEAR LOOP COUNTER
LET R2 := R1 + #1 ;R2 POINTS TO HIGH BYTE
```

030726 010102
030730 005202
9127 030732
030732 012704 000001
9128 030736

```
MOV R1,R2
INC R2
LET R4 := #1      ;INITIAL DATA = 1
REPEAT           ;
MOV #1,R4
```

030736 105237 002264
9129 030736 105237 002264
9130 030742

```
INC B PASFLG     ;INCREMENT LOOP COUNTER
LET CSR := #604  ;WRITE CHECK BITS CORRESPONDING TO DATA OF 0
MOV #604,CSR
```

030742 012737 000604 002150
9131 030750 104425
9132 030752

```
LOADCSR         ;WRITE CSR
LET (R1) := R4   ;WRITE DATA OF 1 CREATING A SINGLE BIT ERROR
MOV R4,(R1)
```

030752 010411
9133 030754 104503
9134 030756

```
CLR1CSR         ;WRITE CSR TO NORMAL MODE
LET (R2) :B= #377 ;WRITE BYTE OF WORD
MOV B #377,(R2)
```

030756 112712 000377
9135 030762 104426
9136 030764 042737 177757 002150
9137 030772

```
READCSR        ;READ CSR
BIC #C20,CSR   ;SEE IF SBE INDICATOR IS SET
IF CSR NE #20  ;IS SBE SET?????
```

030772 023727 002150 000020
031000 001407
9138 031002
031002 012737 000020 002044
9139 031010
031010 013737 002150 002052

```
LET GOOD := #20 ;
LET BAD := CSR  ;
MOV #20,GOOD
MOV CSR,BAD
```

9140 031016 104060
9141 031020
031020
9142 031020 104513
9143 031022 005711
9144 031024 104426

```
ERROR +60      ;
END             ;
;
;L266:::
CBREG          ;WRITE CSR TO DIAG MODE
TST (R1)       ;READ SAO FOR CORRECT CHECK BITS
READCSR        ;READ CSR
```

9145	031026	042737	174037	002150	BIC #+C3740,CSR	;MASK OUT CHECK BIT FIELD	
9146	031034				IF CSR NE #300	;WERE CORRECT CHECK BITS GENERATED?????	
	031034	023727	002150	000300			CMP CSR,#300
	031042	001412					BEQ L267
9147	031044				SET HEADER	:	
	031044	012737	177777	002612			MOV #-1,HEADER
9148	031052				LET GOOD := #300	:	
	031052	012737	000300	002044			MOV #300,GOOD
9149	031060				LET BAD := CSR	:	
	031060	013737	002150	002052			MOV CSR,BAD
9150	031066	104061			ERROR +61	:	
9151	031070				END	:	
	031070						L267:::~::~
9152	031070	005302			DEC R2	;POINT TO HIGH BYTE AND REPEAT	
9153	031072				LET R4 := #400	;BIT 0 OF HIGH BYTE	
	031072	012704	000400				MOV #400,R4
9154	031076				UNTILB PASFLG EQ #2	;DO HIGH AND LOW BYTE	
	031076	123727	002264	000002			CMPB PASFLG,#2
	031104	C01314					BNE B46
	031106						E46:::~::~
9155	031106	104423			CACHON	;TURN ON CACHE	
9156	031110	000207			RETURN	:	

9158 031112
9159
9160
9161
9162
9163
9164
9165 031112
031112
9166 031112 104424
9167 031114 105037 002264
031114
9168 031120 012705 174037
031120
9169 031124 104475
9170 031126 C12702 000046
031126
9171 031132
031132
9172 031132 105237 002264
031132
9173
9174
9175 031136 005037 002266
031136
9176 031142
031142
9177 031142 005237 002266
031142
9178 031146 010204
031146
9179 031150 010237 002150
031150
9180 031154 104425
9181 031156 005011
031156
9182 031160 005105
9183 031162 010546
9184 031164 040416
9185 031166 040504
9186 031170 052604
9187 031172 010437 002150
031172
9188 031176 104425
9189 031200 104426
9190 031202 013703 002150
031202
9191 031206 042703 020000
031206
9192 031212 020304
031212
031214 001412
9193 031216 010137 002034
031216
9194 031222

```

MTP044: SUBSTST <<MTP044      SHIFTING CHECK BITS THROUGH THE CSR TEST>>
;*****
;*SUBSTEST      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
                                        B47:;;;;;
CACHOFF                                ;TURN OFF CACHE
LET PASFLG := #0                       ;INIT PASFLG
                                        CLR B PASFLG
LET R5 := #174037                       ;CHECK BIT MASK FOR CSR
                                        MOV #174037,R5
CB1CSR                                  ;ENABLE CHECK/SYNDROME BIT REGISTER
LET R2 := #46                           ;SET UP INITIAL CSR DATA
                                        MOV #46,R2
REPEAT
                                        B50:;;;;;
LET PASFLG := PASFLG + #1              ;INC LOOP COUNTER
                                        INCB PASFLG
                                        ;CHK BITS = 1
                                        ;DISABLE ECC;DIAG CHK SET
LET PASSNO := #0                       ;INIT PASSNO(INNER LOOP COUNTER)
                                        CLR PASSNO
REPEAT
                                        B51:;;;;;
LET PASSNO := PASSNO + #1              ;INC LOOP COUNTER
                                        INC PASSNO
LET R4 := R2                            ;COPY R2 TO R4
                                        MOV R2,R4
LET CSR := R2                           ;GET CSR DATA TO BE WRITTEN
                                        MOV R2,CSR
LOADCSR                                  ;WRITE SBE CHECK BITS TO CSR
LET (R1) := #0                          ;WRITE DATA AND CHECK BITS AT A=0
                                        CLR (R1)
COM R5                                    ;COMPLEMENT MASK
MOV R5,-(SP)                             ;SAVE R5 ON STACK
BIC R4,(SP)                               ;CREATE AN XOR FUNCTION
BIC R5,R4
BIS (SP)+,R4
LET CSR := R4
                                        ;
                                        ;
                                        MOV R4,CSR
LOADCSR                                  ;LOAD CSR WITH COMPLEMENT CHECK BITS
READCSR                                  ;READ CSR FOR COMPLEMENT CHECK BITS
LET R3 := CSR                             ;COPY CSR DATA TO R3
                                        MOV CSR,R3
BIC #BIT13,R3                            ;CLEAR ANY POSSIBLE INHIBIT MODE POINTER
IF R3 NE R4 THEN                         ;READ CSR FOR PROPER CHECK BITS
                                        CMP R3,R4
                                        BEQ L271
LET ADDRESS := R1                        ;
                                        MOV R1,ADDRESS
LET GOOD := R4                            ;

```

9195	031222	010437	002044		LET BAD := R3	:	MOV R4,GOOD
9196	031226	010337	002052		SET HEADER	:	MOV R3,BAD
9197	031232	012737	177777	002612	PERR37	:ERROR CALL	MOV #-1,HEADER ;:ILC::REVB
9198	031240	104463			END		
9199	031242	005105			COM R5		L271:;;;;;
9200	031244	005711			TST (R1)		;COMPLEMENT MASK
9201	031246	000240			NOP		;READ CHECK BITS AT A=0 INTO CSR
9202	031250	104426			READCSR		;READ CSR FOR CORRECT CHECK BITS
9203	031252	040537	002150		BIC R5,CSR		;MASK OUT CHECK BIT FIELD
9204	031256				LET R4 := R2		;GET CHECK BITS THAT WERE WRITTEN MOV R2,R4
9205	031260	010204			BIC R5,R4		;MASK OUT CHECK BIT FIELD
9206	031262	040504			IF R4 NE CSR		;ARE CHECK BITS THE SAME?
9207	031266	020437	002150				CMP R4,CSR BEQ L272
9208	031270	010437	002044		LET GOOD := R4	:	MOV R4,GOOD
9209	031274	013737	002150	002052	LET BAD := CSR	:	MOV CSR,BAD
9210	031302	010137	002034		LET ADDRESS := R1	:	MOV R1,ADDRESS
9211	031306	012737	177777	002612	SET HEADER	:	MOV #-1,HEADER ;:ILC::REVB
9212	031314	104464			PERR40	:ERROR CALL	
9213	031316				END		
9214	031316	040502			BIC R5,R2		L272:;;;;;
9215	031320	123727	002264	000001	IFB PASFLG EQ #1		;SHIFT CHECK BITS AND CREATE NEW DATA FOR CSR ;SELECT FUNCTION
9216	031326	001002					CMPB PASFLG.#1 BNE L273
9217	031330	006302			ASL R2		;DO A FIELD OF ZEROS---->ONES
9218	031332	000413			ELSE		;SHIFT CHECK BITS
9219	031334						;DO A FIELD OF ONES ---->ZEROS BR L274
9220	031336	005105			COM R5		L273:;;;;;
9221	031338	010546			MOV R5,-(SP)		;TAKE OUT CHECK BIT FIELD
9222	031340	040216			BIC R2,(SP)		
9223	031342	040502			BIC R5,R2		
9224	031344	052602			BIS (SP)+,R2		
9225	031346	006302			ASL R2		;SHIFT CHECK BITS
9226	031350	010546			MOV R5,-(SP)		;PUT BACK CHECK BIT FIELD
9227	031352	040216			BIC R2,(SP)		
9228	031354	040502			BIC R5,R2		
9229	031356	052602			BIS (SP)+,R2		
9230	031360	005105			COM R5		;COMPLEMENT DATA PATTERN
9231	031362				END		
9232	031362	062702	000006		LET R2 := R2 + #6		L274:;;;;;
9233	031366	123727	002266	000006	UNTILB PASSNO EQ #6		;ADD 6 SO THAT WRITE ON CSR WILL ENABLE DIAG MODE ADD #6,R2
9234	031374	001262					;DO ALL CHECK BITS CMPB PASSNO,#6 BNE B51

```

9232 031376          LET R2 := #3706          ;REPEAT WITH FIELD OF ONES
031376 012702 003706          ;MOV #3706,R2
9233 031402          UNTILB PASFLG EQ #2
031402 123727 002264 000002          ;CMPB PASFLG,#2
031410 001250          ;BNE B50
031412          E50:;
9234 031412 104503          CLR1CSR          ;
9235 031414 005011          CLR (R1)          ;
9236 031416 062701 000100          ADD #100,R1          ;ALL 256 TO ADDRESS          ;ILC;;REVB
9237 031422          UNTIL R1 EQ ENDADD          ;          ;ILC;;REVB
031422 020137 002562          ;CMP R1,ENDADD
031426 001231          ;BNE B47
031430          E47:;
9238 031430 104423          CACHON          ;TURN ON CACHE
9239 031432 000207          RETURN          ;
```

```

9241 031434 MTP045: SUBTST <<MTP045 SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST>>
;*****
;*SUBTEST MTP045 SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST
;*****
9242 ;
9243 ; THIS TEST CHECKS TO SEE IF THE DOUBLE BIT ERROR INDICATOR IS SET
9244 ; ON A DOUBLE BIT ERROR AND THE CORRECT SYNDROMES ARE LATCHED INTO THE
9245 ; CSR. THIS TEST IS THEN REPEATED WITH MULTIPLE ERROR CHECK/SYNDROME BITS
9246 ;
9247 031434 104424 CACHOFF ;TURN OFF CACHE
9248 031436 104513 CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
9249 031440 LET PASSNO := #0 ;CLEAR LOOP COUNTER
031440 005037 002266 CLR PASSNO
9250 031444 LET GOOD := #3744 ;GOOD DATA
031444 012737 003744 002044 MOV #3744,GOOD
9251 031452 LET CSR := #3144 ;DBE CHECK BITS FOR CSR
031452 012737 003144 002150 MOV #3144,CSR
9252 031460 REPEAT ;
031460 ; B52:;;;;;
9253 031460 005237 002266 INC PASSNO ;
9254 031464 104425 LOADCSR ;WRITE DBE CHECK BITS TO CSR
9255 031466 LET (R1) := #0 ;WRITE ZEROS AND DBL ERROR CHK BITS A=0
031466 005011 CLR (R1)
9256 031470 CLR1CSR ;CLEAR CSR OUT
9257 031472 005711 TST (R1) ;READ A=0 TO GET DOUBLE BIT ERROR
9258 031474 104426 READCSR ;WAS UNCORRECABLE ERROR BIT SET???
9259 031476 IF #BIT15 OFF.IN CSR ;
031476 032737 100000 002150 BIT #BIT15.CSR
031504 001007 BNE L300
9260 031506 SET HEADER ;
031506 012737 177777 002612 MOV #-1,HEADER
9261 031514 LET BAD := CSR ;
031514 013737 002150 002052 MOV CSR,BAD
9262 031522 ERROR +63 ;BIT NOT SET
9263 031524 END ;
031524 ; L300:;;;;;
9264 031524 104514 SYNREG ;ENABLE SYNDROME BIT REGISTER
9265 031526 104426 READCSR ;READ CSR FOR CORRECT SYNDROME BITS

```

```

9266 031530 000240      NOP      ;DEBUG AIDE
9267 031532 042737 174033 002150    BIC #+C3744,CSR ;MASK SYNDROMES OUT
9268 031540      IF CSR NE GOOD THEN ;CHECK IF DOUBLE ERROR BIT IS SET
      031540 023737 002150 002044      CMP CSR,GOOD
      031546 001407      BEQ L301
9269 031550      LET BAD := CSR ;BAD DATA
      031550 013737 002150 002052      MOV CSR,BAD
9270 031556      SET HEADER ;
      031556 012737 177777 002612      MOV #-1,HEADER
9271 031564 104042      ERROR +42 ;
9272 031566      END ;
      031566      L301:;;;;;
9273 031566 005011      CLR (R1) ;CLEAR LUT
9274 031570      LET GOOD := #3604 ;REPEAT WITH MULTIPLE ERROR SYNDROMES
      031570 012737 003604 002044      MOV #3604,GOOD
9275 031576      LET CSR := #3004 ;MULTIPLE ERROR CHECK BITS
      031576 012737 003004 002150      MOV #3004,CSR
9276 031604      UNTIL PASSNO EQ #2 ;
      031604 C23727 002266 000002      CMP PASSNO,#2
      031612 001322      BNE B52
      031614      E52:;;;;;
9277 031614 104503      CLR1CSR ;
9278 031616 104423      CACHON ;
9279 031620 000207      RETURN ;
  
```

```

9281 031622      MTP046: SUBTST <<MTP046      CHECK SINGLE BIT ERRORS WITH ECC DISABLED>>
;*****
;SUBTEST      MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED
;*****
;
; THIS TEST CHECKS TO SEE THAT FOR EACH BIT OF A DATA WORD THAT A SBE
; IS TREATED LIKE A UNCORRECTABLE ERROR WITH ECC DISABLED AND TRAPS
; ARE DETECTED.
;
9282
9283
9284
9285
9286
9287 031622 005037 002266      CLR PASSNO      ;CLEAR OUTER LOOP COUNTER
9288 031626 104424      CACHOFF        ;TURN OFF CACHE
9289 031630      REPEAT
;
;                               B53:; ; ; ; ;
9290 031630      LET PASSNO := PASSNO + #1      ;
;                               INC PASSNO
9291 031630 005237 002266      CLR RO          ;CLEAR DATA
9292 031636 105037 002264      CLR PASFLG      ;CLEAR PASFLG
9293 031642 104513      CBREG          ;ENABLE CHECK/SYNDROME BIT REGISTER
9294 031644      REPEAT
;
;                               B54:; ; ; ; ;
9295 031644      LET PASFLG := PASFLG + #1      ;INCREMENT LOOP COUNTER
;                               INCB PASFLG
9296 031650 105237 002264      LET R4 := #-1      ;INDEX TO SINGLE BIT ERROR TABLE
;                               MOV #-1,R4
9297 031654 012704 177777      LET NOPAR := #1      ;ENABLE PARITY ACTION
;                               MOV #1,NOPAR
9298 031662 012737 000001 002076      LET BITNO := #0      ;CLEAR INNER LOOP COUNTER
;                               CLR BITNO
9299 031666 005037 002324      IFB PASFLG EQ #1      ;SELECT DATA TO BE CORRECTED BY PASSNO
;                               CMPB PASFLG,#1
;                               BNE L303
9300 031674 001003      LET R5 := #1      ;DATA=0;BIT TO BE CORRECTED IS A ONE
;                               MOV #1,R5
9301 031702 000402      ELSE
;
;                               BR L304
;                               L303:; ; ; ; ;
9302 031704 012705 177776      LET R5 := #177776      ;DATA=177776;BIT TO BE CORRECTED IS A ZERO
;                               MOV #177776,R5
9303 031710      END
;
;                               L304:; ; ; ; ;
9304 031710      REPEAT
;
;                               B55:; ; ; ; ;
9305 031710      LET PARCNT := #0      ;CLEAR PARITY COUNTER
;                               CLR PARCNT
9306 031714 005037 002072      INC BITNO      ;INCREMENT BIT POINTER
9307 031720 005237 002324      LET R4 := R4 + #1      ;POINT TO NEXT SET OF CHECK BITS
;                               INC R4
9308 031722 005204      LET R2 := PTABLE(R4)      ;GET NEXT SET OF CHECK BITS
;                               MOVB PTABLE(R4),R2
9309 031726 116402 050164      ASH #5,R2      ;SHIFT TO LINE UP IN CSR
9310 031732 072227 000005      BIS #BIT2:BIT1,R2      ;ENABLE DIAG MODE
9311 031736 052702 000006      LET CSR := R2      ;GET CHECK BITS TO BE WRITTEN
;                               MOV R2,CSR
9312 031742 010237 002150      LOADCSR      ;LOAD CSR WITH DATA
9313 031744 104425      LET (R1) := RO      ;WRITE DATA TO TEST ADDRESS
;                               MOV RO,(R1)
9314 031746      IF PASSNO EQ #1      ;WRITE CSR

```

	031746	023727	002266	000001					CMP PASSNO.#1 BNE L305
9315	031754	001002							
	031756	104471				ECC1DIS			;FIRST PASS ;ECC DISABLE,NO ^BL
9316	031760					ELSE			:
	031760	000401							BR L306
	031762								L305:
9317	031762	104507				ENALSBE			;SECOND PASS ;ECC DISABLE,PBL ENABLED
9318	031764					END			:
	031764								L306:
9319	031764	005711				TST (R1)			;CORRECT SBE
9320	031766	004737	032106			CALL CHKTRP			;CHECK FOR CORRECT TRAP
9321	031772	104426				READCSR			;READ THE CSR FOR UNCORRECTABLE ERROR
9322	031774					IF #BIT15 OFF.IN CSR			;IS UNCORRECTABLE ERROR BIT SET????
	031774	032737	100000	002150					BIT #BIT15.CSR
	032002	001007							BNE L307
9323	032004						LET BAD := CSR		:
	032004	013737	002150	002052					MOV CSR,BAD
9324	032012						SET HEADER		:
	032012	012737	177777	002612					MOV #-1,HEADER
9325	032020	104045					ERROR +45		:
9326	032022					END			:
	032022								L307:
9327	032022	104503				CLR1CSR			:
9328	032024	005011				CLR (R1)			;CLEAR LUT
9329	032026					IFB PASFLG EQ #1			;SHIFT NEW DATA DEPENDING ON PASFLG
	032026	123727	002264	000001					CMPB PASFLG.#1
	032034	001002							BNE L310
9330	032036	006305					ASL R5		;SHIFT BITNO TO THE LEFT
9331	032040						ELSE		:
	032040	000402							BR L311
	032042								L310:
9332	032042	000261					SEC		;SET CARRY BIT AND.....
9333	032044	006105					ROL R5		;ROTATE LEFT
9334	032046					END			:
	032046								L311:
9335	032046					UNTIL BITNO EQ #16.			;UNTIL ALL BITS ARE DONE
	032046	023727	002324	000020					CMP BITNO.#16.
	032054	001315							BNE B55
	032056								E55:
9336	032056	005100				COM R0			;COMPLEMENT DATA AND REPEAT
9337	032060					UNTILB PASFLG EQ #2			;UNTIL 2 PASSES ARE COMPLETE!
	032060	123727	002264	000002					CMPB PASFLG.#2
	032066	001266							BNE B54
	032070								E54:
9338	032070					UNTIL PASSNO EQ #2			:
	032070	023727	002266	000002					CMP PASSNO.#2
	032076	001254							BNE B53
	032100								E53:
9339	032100	104503				CLR1CSR			;TURN CACHE
9340	032102	104423				CACHON			:
9341	032104	000207				RETURN			:
9342									
9343									
9344	032106					CHKTRP: IF PASSNO EQ #1			;PASS 1 CHECK FOR NO TRAP
	032106	023727	002266	000001					CMP PASSNO.#1
	032114	001011							BNE L315
9345	032116					IF PARCNT EQ #1			:

	032116	023727	002072	000001					CMP PARCNT.#1
	032124	001004							BNE L316
9346	032126				SET HEADER	:			
	032126	012737	177777	002612					MOV #-1.HEADER
9347	032134	104057			ERROR +57	:			
9348	032136				END	:			
	032136							L316:	
9349	032136				ELSE	:			BR L317
	032136	000410						L315:	
	032140								CMP PARCNT.#1
9350	032140				IF PARCNT NE #1	:			BEQ L320
	032140	023727	002072	000001					
	032146	001404							MOV #-1.HEADER
9351	032150				SET HEADER	:			
	032150	012737	177777	002612					
9352	032156	104064			ERROR +64	:			
9353	032160				END	:			
	032160							L320:	
9354	032160				END	:			
	032160							L317:	
9355	032160	000207			RETURN	:			

9357 032162

MTP047: SUBTST <<MTP047 NO CSR UPDATE ON SBE WITH EXSISTING DBE>>
;*****
;SUBTEST MTP047 NO CSR UPDATE ON SBE WITH EXSISTING DBE
;*****

```
9358  
9359  
9360  
9361  
9362  
9363 032162 104424  
9364 032164 013704 002102  
9365 032170 072427 000011  
9366 032174 042104 170037  
9367 032200 052704 100000  
9368 032204 104513  
9369 032206 012737 003144 002150  
9370 032214 104425  
9371 032216 005011  
9372 032220 012737 000104 002150  
9373 032226 104425  
9374 032230 005012  
9375 032232 104503  
9376 032234 005711  
9377 032236 104426  
9378 032240 042737 020000 002150  
9379 032246 023704 002150  
9380 032252 001411  
9381 032254 013737 002150 002052  
9382 032262 010437 002044  
9383 032266 012737 177777 002612  
9384 032274 104063  
9385 032276 052704 000020  
9386 032302 005712  
9387 032304 104426
```

```
;  
; 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  
ASH #9,R4  
BIC #C7740,R4  
BIS #BIT15,R4  
CBREG  
LET CSR := #3144  
LOADCSR  
LET (R1) := #0  
LET CSR := #104  
LOADCSR  
LET (R2) := #0  
CLR1CSR  
TST (R1)  
READCSR  
BIC #BIT13,CSR  
IF CSR NE R4  
;  
;TURN OFF CACHE  
;GET BANK NUMBER  
MOV BANK,R4  
;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  
;  
;WRITE DBE CHECK BITS  
CLR (R1)  
;WRITE SBE CHECK BITS  
MOV #104,CSR  
;  
;WRITE SBE CHECK BITS AT ADDRESS + 4K  
CLR (R2)  
;  
;CLEAR CSR  
;READ DBE LOCATION  
;READ FOR CSR DBE INDICATOR  
;CLEAR INHIBIT MODE POINTER  
;  
CMP CSR,R4  
BEQ L321  
;  
MOV CSR,BAD  
;  
MOV R4,GOOD  
;  
MOV #-1,HEADER  
;  
;  
L321:::;  
;SET BIT IN GOOD DATA  
;READ SBE  
;READ CSR FOR NO CHANGE
```

```
9388 032306 042737 020000 002150      BIC #BIT13,CSR      ;CLEAR INHIBIT MODE POINTER
9389 032314      IF CSR NE R4      ;
      032314 023704 002150      ;                               CMP CSR,R4
      032320 001411      ;                               BEQ L322
9390 032322      LET BAD := CSR      ;
      032322 013737 002150 002052      ;                               MOV CSR,BAD
9391 032330      LET GOOD := R4      ;
      032330 010437 002044      ;                               MOV R4,GOOD
9392 032334      SET HEADER      ;
      032334 012737 177777 002612      ;                               MOV #1,HEADER
9393 032342 104051      ERROR +51      ;
9394 032344      END      ;
      032344      ;                               L322:::~::~
9395 032344 104503      CLR1CSR      ;CLEAR 1 CSR
9396 032346 005011      CLR (R1)      ;
9397 032350 005012      CLR (R2)      ;
9398 032352 104423      CACHON      ;TURN ON CACHE
9399 032354 000207      RETURN
```

```

          .SBTTL  MISC SUBROUTINES
9401
9402
9403 032356  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
;*****
9404 032356  010004  MOV      R0,R4
9405 032360  010103  MOV      R1,R3
9406 032362  010205  MOV      R2,R5
9407 032364  000207  RETURN
9408
9409 032366  FLIPWARN:SUBTST  <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
;*****
; *SUBTEST  FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
;*****
9410 032366  010046  PUSH     R0
9411 032366  005237  002616  INC      FLIPLOC
9412 032374  042737  177774  002616  BIC      @C3,FLIPLOC
9413 032402  C22737  000001  002616  CMP      @1,FLIPLOC
9414 032410  001414  BEQ      1$
9415 032412  022737  000002  002616  CMP      @2,FLIPLOC
9416 032420  001413  BEQ      2$
9417 032422  022737  000003  002616  CMP      @3,FLIPLOC
9418 032430  001414  BEQ      3$
9419 032432  005000  CLR      R0
9420 032434  013704  002614  MOV      ONES,R4
9421 032440  000414  BR       4$
9422 032442  1$:     CLEAR  R0,R4
9423 032446  005000  CLR     R0
9424 032446  005004  CLR     R4
9425 032446  000411  BR       4$
9426 032450  012700  000401  2$:     MOV     @401,R0
9427 032454  013704  002614  MOV     ONES,R4
9428 032460  000404  BR       4$
9429 032462  012700  000401  3$:     MOV     @401,R0
9430 032466  012704  000401  MOV     @401,R4
9431 032472  010037  024650  4$:     MOV     R0,WARN2
9432 032476  010037  024664  MOV     R0,WARN3
9433 032502  010037  024710  MOV     R0,WARN4
9434 032506  010037  024724  MOV     R0,WARN5
9435 032512  012600  POP     R0
9436 032514  000207  MOV     (SP),R0
          RETURN

```

9436 032516

9437
9438 032516 104415
9439 032520 012700 060000
9440 032524 012701 040000
9441 032530 012737 000207 024532
9442 032536 012737 024526 002262
9443 032544 004737 024334
9444 032550 012737 000240 024532
9445 032556 104416
9446 032560 000207
9447

```
BACKGND:SUBTST <<SUBR WRITE BACKGROUND>>  
;*****  
;SUBTEST SUBR WRITE BACKGROUND  
;*****  
;WRITES DATA FROM R2  
SAVREG  
MOV #FIRST,R0  
MOV #SIZE,R1  
MOV #207,MTP000+4 ;WARNING PUTTING "RETURN" AFTER WRITE  
MOV #MTP000,SUPDOADD  
CALL SUPD03  
MOV #240,MTP000+4 ;RESTORE 'NOP' AFTER WRITE  
RESREG  
RETURN
```

9449 032562

```
GETCSR: SUBTST <<SUBR GET CSR INFORMATION FROM CONFIGURATION TABLE>>  
;*****  
;SUBTEST SUBR GET CSR INFORMATION FROM CONFIGURATION TABLE  
;*****
```

9450

;INPUTS : NONE

9451

;OUTPUT : CSRNO = CSR NUMBER

9452

9453

9454 032562 013702 002104

MOV BANKINDEX,R2 ;GET INDEX INTO CONFIG TABLE

9455 032566 016203 002664

MOV CONFIG(R2),R3 ;MOV IT INTO R3

9456 032572 000303

SWAB R3 ;

9457 032574 006303

ASL R3 ;

9458 032576 042703 177741

BIC #C36,R3 ;CLEAR OFF SOME BITS

9459 032602 010337 002152

MOV R3,CSRNO ;SAVE CSR NUMBER

9460 032606 000207

RETURN

```

9463 032610          PCONFIG:SUBTST <<SUBR PRINT CONFIGURATION MAP>>
;*****
;+SUBTEST          SUBR PRINT CONFIGURATION MAP
;*****
          PUSH          TKVEC,TKVEC+2,R0
9464 032610          MOV          SP,PCONFS          ;SAVE LAST GOOD SP
          032610 013746 000060          MOV          #PCONF2,TKVEC
          032614 013746 000062          MOV          #340,TKVEC+2
          032620 010046          MOV          #TKB,R0          ;KILL ANY OLD INTERRUPT
9465 032622 010637 033222          BIC          #BIT7,PSW          ;LOWER CPU PRIORITY TO 140
9466 032626 012737 033170 000060          BIS          #BIT6,#TKS          ;ENABLE KEYBOARD INTERRUPTS
9467 032634 012737 000340 000062
9468 032642 017700 147776
9469 032646 042737 000200 177776
9470 032654 052777 000100 147760
9471
9472 032662          TYPE          MSG001
          032662 104401 065173          TYPEIT         ,MSG001
          .DSABL         CRF
9473 032666          TYPE          MSG002
          032666 104401 065255          TYPEIT         ,MSG002
          .DSABL         CRF
9474 032672          TYPE          MSG003
          032672 104401 065332          TYPEIT         ,MSG003
          .DSABL         CRF
9475 032676 022737 000060 002556          CMP          #60,LASTBANK
9476 032704 002006          BGE          NOOJ
9477
9478 032706          ;IF FAT PAPER ON TERMINAL GOTO 1#
          032706 032777 000020 147722          IF #SW4 SET.IN #SWR THEN JUMPTO PCONF1
          032714 001402          BIT          #SW4,#SWR
          032716 000137 033130          BEQ          L323
          032722          JMP          PCONF1
          L323:::~::~
9479 032722 012700 000074          NOOJ: MOV          #60,R0
9480 032726 010004          MOV          R0,R4
9481 032730          CLEAR          R1,R3
          032730 005001          CLR          R1
          032732 005003          CLR          R3
9482 032734          TYPE          MSG004
          032734 104401 065437          TYPEIT         ,MSG004
          .DSABL         CRF
9483 032740 004737 033224          CALL          TCONFIG          ;GO TYPE CONFIGURATION (1ST HALF)
9484 032744 022737 000060 002556          CMP          #60,LASTBANK
9485 032752 002106          BGE          PCONF2
9486 032754          TYPE          #CRLF
          032754 104401 002660          TYPEIT         ,#CRLF
          .DSABL         CRF
9487 032760          TYPE          MSG017          ;PRINT SPACE(S)
          032760 104401 066146          TYPEIT         ,MSG017
          .DSABL         CRF
9488 032764          TYPE          MSG11A
          032764 104401 065640          TYPEIT         ,MSG11A
          .DSABL         CRF
9489 032770          TYPE          #CRLF
          032770 104401 002660          TYPEIT         ,#CRLF
          .DSABL         CRF
9490 032774          TYPE          MSG017          ;PRINT SPACE(S)
          032774 104401 066146          TYPEIT         ,MSG017
  
```

9491	033000			.DSABL	CRF		
	033000	104401	065726	TYPE	MSG011		
				TYPEIT	,MSG011		
9492	033004			.DSABL	CRF		
	033004	104401	002660	TYPE	,\$CRLF		
				TYPEIT	,,\$CRLF		
9493	033010			.DSABL	CRF		
	033010	104401	066146	TYPE	MSG017		;PRINT SPACE(S)
				TYPEIT	,MSG017		
9494	033014			.DSABL	CRF		
	033014	104401	066014	TYPE	MSG012		
				TYPEIT	,MSG012		
9495	033020	012701	000360	.DSABL	CRF		
9496	033024	010103		MOV	#60,*2*2,R1		
9497	033026	004737	033224	MOV	R1,R3		
9498	033032	022737	000170	CALL	TCONFIG		
9499	033040	002053		CMP	#170, LASTBANK		
9500	033042			BGE	PCONF2		
	033042	104401	002660	TYPE	,\$CRLF		
				TYPEIT	,,\$CRLF		
9501	033046			.DSABL	CRF		
	033046	104401	066146	TYPE	MSG017		;PRINT SPACE(S)
				TYPEIT	,MSG017		
9502	033052			.DSABL	CRF		
	033052	104401	066111	TYPE	MSG11B		
				TYPEIT	,MSG11B		
9503	033056			.DSABL	CRF		
	033056	104401	002660	TYPE	,\$CRLF		
				TYPEIT	,,\$CRLF		
9504	033062			.DSABL	CRF		
	033062	104401	066146	TYPE	MSG017		;PRINT SPACE(S)
				TYPEIT	,MSG017		
9505	033066			.DSABL	CRF		
	033066	104401	066113	TYPE	MSG11C		
				TYPEIT	,MSG11C		
9506	033072			.DSABL	CRF		
	033072	104401	002660	TYPE	,\$CRLF		
				TYPEIT	,,\$CRLF		
9507	033076			.DSABL	CRF		
	033076	104401	066146	TYPE	MSG017		;PRINT SPACE(S)
				TYPEIT	,MSG017		
9508	033102			.DSABL	CRF		
	033102	104401	066115	TYPE	MSG11D		
				TYPEIT	,MSG11D		
9509	033106	012701	000740	.DSABL	CRF		
9510	033112	010103		MOV	#740,R1		
9511	033114	012700	000010	MOV	R1,R3		
9512	033120	010004		MOV	#8.,R0		
9513	033122	004737	033224	MOV	R0,R4		
9514	033126	000420		CALL	TCONFIG		
9515				BR	PCONF2		
9516	033130	012700	000170	PCONF1:	MOV	#120.,R0	
9517	033134	010004			MOV	R0,R4	
9518	033136	005001			CLEAR	R1,R3	
	033140	005003					CLR R1 CLR R3

9519	033142					TYPE	MSG014		;SPACE
	033142	104401	066130			TYPEIT	,MSG014		
						.DSABL	CRF		
9520	033146					TYPE	MSG011		
	033146	104401	065726			TYPEIT	,MSG011		
						.DSABL	CRF		
9521	033152					TYPE	MSG004		
	033152	104401	065437			TYPEIT	,MSG004		
						.DSABL	CRF		
9522	033156					TYPE	MSG012		
	033156	104401	066014			TYPEIT	,MSG012		
						.DSABL	CRF		
9523	033162	004737	033224			CALL	TCONFIG		
9524	033166	000721				BR	N00J1		
9525									
9526	033170	013706	033222		PCONF2:	MOV	PCONFS,SP		;RESTORE STACK
9527	033174	042777	000100	147440		BIC	#BIT6,#TKS		
9528	033202	117700	147436			MOVB	#TKB,R0		;READ CHAR TO KILL FLAG
9529	033206					POP	R0,TKVEC+2,TKVEC		
	033206	C12600							MOV (SP)+,R0
	033210	012637	000062						MOV (SP)+,TKVEC+2
	033214	012637	000060						MOV (SP)+,TKVEC
9530	033220	000207				RETURN			
9531									
9532	033222	000000			PCONFS:	0			;STACK SAVED HERE!

9535 033224

```

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
;*****
  
```

9536
 9537
 9538
 9539
 9540
 9541
 9542
 9543

9544
 9545
 9546

9547
 9548 033224 012737 000340 177776
 9549 033232 104401 065545
 9550 033236 032761 000001 002664
 9551 033244 001403
 9552 033246 104401 066126
 9553 033252 000402
 9554 033254 104401 066130
 9555 033260 062701 000004
 9556 033264 077014
 9557 033266 010400
 9558 033270 010301

```

;*****
;** ERROR **
;*****
TCONFIG:MOV #340,PSW ;DISABLE INTERUPTS
TYPE MSG005
TYPEIT ,MSG005
.DSABL CRF
1$: BIT #BIT0,CONFIG(R1) ;ERROR ON THIS BANK?
BEQ 2$ ;NO - SKIP
TYPE MSG013 ;PRINT "X"
TYPEIT ,MSG013
.DSABL CRF
BR 3$
2$: TYPE MSG014 ;PRINT SPACE
TYPEIT ,MSG014
.DSABL CRF
3$: ADD #4,R1 ;BUMP POINTER
SOB R0,1$ ;LOOP UNTIL DONE
MOV R4,R0
MOV R3,R1
  
```

```

9561 ;*****
9562 ;** INTERLEAVE **
9563 ;*****
9564 ;THIS IS AN ENTRY POINT FROM ERROR REPORTS
9565 033272 012737 000340 177776 TCFIG1: MOV #340,PSW ;DISABLE INTERUPTS
9566 033300 112737 000040 066132 MOVB #' ,MSG015 ;MOVE A BLANK IN TO BE PRINTED
9567 033306 TYPE MSG015
033306 104401 066132 TYPEIT ,MSG015
;*****
;** MEMORY TYPE **
;*****
ENABL LSB
TYPE MSG009
TYPEIT ,MSG009
;*****
9577 033332 033761 002106 002664 TCFIG2: BIT CPUBIT,CONFIG(R1)
9578 033340 001432 BEQ 17#
9579 033342 016105 002666 MOV CONFIG+2(R1),R5
9580 033346 000305 SWAB R5 ;GET MEMORY TYPE
9581 033350 042705 177770 BIC #C7,R5 ;CLEAR NON INTERESTING BITS
9582 033354 020527 000003 CMP R5,#3 ;IS IT A LEGAL MEMORY TYPE
9583 033360 003022 BGT 17# ;IF IF SO BRANCH!!!!!!!
9584 033362 IF #BIT0 SET.IN R5 ;IS IT AN ECC MEMORY????
033362 032705 000001 BIT #BIT0,R5
033366 001413 BEQ L325
9585 033370 IF #BIT1 SET.IN R5 ;IS IT A MSV11-P OR A MSV11-J???
033370 032705 000002 BIT #BIT1,R5
033374 001404 BEQ L326
9586 033376 112737 000120 066132 MOVB #'P,MSG015 ;IT IS A MSV11-P
9587 033404 ELSE ;
033404 000403 ; BR L327
033406 L326:;;;;;;
9588 033406 112737 000105 066132 MOVB #'E,MSG015 ;IT IS A MSV11-J
9589 033414 END ;
033414 L327:;;;;;;
9590 033414 ELSE ; BR L330
033414 000403 L325:;;;;;;
9591 033416 112737 000120 066132 MOVB #'P,MSG015 ;IT IS A MSV11-L/P
9592 033424 END ;
033424 L330:;;;;;;
9593 033424 000403 BR 8#
9594 033426 112737 000040 066132 17#: MOVB #' ,MSG015
9595 033434 104401 066132 8#: TYPE MSG015
TYPEIT ,MSG015
;*****
;** MEMORY TYPE **
;*****
ENABL CRF
IF NOTAB NE #0 THEN $RETURN
TST NOTAB
BEQ L324
RTS PC
L324:;;;;;;

```

```

                                BEQ L331
                                RTS PC
                                L331:|||||
9597 033444 001401
033446 000207
033450
9597 033450 062701 000004
9598 033454 077052
9599 033456 010400
9600 033460 010301
9601
9602
9603
9604
9605
9606 033462
033462 104401 066134
9607 033466 112737 000040 066132 TCFIG3:
9608 033474 016105 002664
9609 033500 032705 000002
9610 033504 001414
9611 033506 042705 170377
9612 033512 000305
9613 033514 022705 000011
9614 033520 100002
9615 033522 062705 000007
9616 033526 062705 000060 10:
9617 033532 110537 066132 16:
9618 033536 104401 066132
9619 033542
033542 005737 002372
033546 001401
033550 000207
033552
9620 033552 062701 000004
9621 033556 077035
9622 033560 010400
9623 033562 010301
9624

ADD #4,R1 ;BUMP POINTER
SOB R0,TCFIG2 ;LOOP UNTIL DONE
MOV R4,R0
MOV R3,R1
.DSABL LSB

;*****
;** CSR **
;*****
TYPE MSG016
TYPEIT ,MSG016
.DSABL CRF
MOV #',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 ;MAKE ASCII
MOV R5,MSG015 ;PLUG INTO MEMORY
TYPE MSG015
TYPEIT ,MSG015
.DSABL CRF
IF NOTAB NE #0 THEN $RETURN

                                TST NOTAB
                                BEQ L332
                                RTS PC
                                L332:|||||
;BUMP POINTER
                                L332:|||||

```

```
9625 ;*****  
9626 ;** PROTECTED **  
9627 ;*****  
9628 033564 104401 065626 TYPE MSG010  
033564 .DSABL CRF  
9629 033570 105761 002664 11+: TSTB CONFIG(R1) ;BANK PROTECTED?  
9630 033574 100006 BPL 14: ;NO - SKIP  
9631 033576 112737 000120 066132 MOVB @'P,MSG015  
9632 033604 104401 066132 13+: TYPE MSG015  
033604 .DSABL CRF  
9633 033610 000402 BR 15: ;PRINT SPACE  
9634 033612 104401 066130 14+: TYPE MSG014  
033612 .DSABL CRF  
9635 033616 062701 000004 15+: ADD @4,R1 ;BUMP POINTER  
9636 033622 C77016 SOB R0,11: ;LOOP UNTIL DONE  
9637 033624 010400 MOV R4,R0  
9638 033626 010301 MOV R3,R1  
9639 033630 000207 RETURN
```

```

9642          .SBTTL TRAP PARITY ERROR HANDLER
9643          ;*****
9644          ;VECTOR TO HERE FROM TRAPS TO 114
9645          ;IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
9646          ;*****
9647          ;
9648          ; CODE ACTION
9649          ; - 0 - PRINT UNEXPECTED PARITY TRAP
9650          ; 1 COUNT ERROR
9651          ; 2 SET "ABORT" / SETUP "BADPC" / RETURN VIA PCBUMP
9652          ; 3 RETURN VIA "PARTHERE"
9653          ;
9654          ;
9655 033632 022737 000001 002076 PARITY: CMP #1,NOPAR ;COUNTING PARITY ERRORS?
9656 033640 001003 BNE 1$ ;NO - SKIP
9657 033642 005237 002072 INC PARCNT ;PARITY ERROR COUNTER + 1
9658 033646 000002 RTI
9659 033650 022737 000002 002076 1$: CMP #2,NOPAR ;ACTION CODE = 2 ?
9660 033656 001013 BNE 2$ ;NO SKIP
9661 033660 SET ABORTFLAG ;YES
9662 033666 012737 177777 002144 MOV #-1,ABORTFLAG
9663 033672 004737 034040 CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
9664 033676 063716 002326 ADD PCBUMP,(SP) ;UPDATE RETURN PC
9665 033704 042766 000004 000002 BIC #BIT2,2(SP) ;SHOW FAILURE BY .NE.
9666 033706 000002 RTI
9667 033706 022737 000003 002076 2$: CMP #3,NOPAR ;ACTION CODE = 3 ?
9668 033714 001003 BNE 3$ ;NO - SKIP
9669 033716 013716 002304 MOV PARTHERE,(SP)
9670 033722 000002 RTI
9671 033724 004737 034040 3$: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
033730 005237 002064 INC FATAL$ ;SET FATAL INDICATOR
033734 104032 ERROR +32
.DSABL CRF

```

```

9674 .SBTTL TRAP NON-EXISTANT MEMORY (HOLES) HANDLER
9675 ;*****
9676 ;VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
9677 ; CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
9678 ; 1) IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
9679 ; 2) TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
9680 ;*****
9681
9682 033736 022737 000001 002100 NONEXIST: CMP #1,NONEM ;COUNTING NON-EXISTANT MEMORY ERRORS?
9683 033744 001011 BNE 2$ ;NO - SKIP
9684 033746 005237 002070 INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
9685 033752 022737 000001 002070 CMP #1,NEMCNT ;FIRST ERROR?
9686 033760 001002 BNE 1$ ;NO - SKIP
9687 033762 010037 002034 MOV RO,ADDRESS ;ASSUME RO CONTAINS THE ADDRESS ACCESSED
9688 033766 000002 1$: RTI
9689 033770 005237 002070 2$: INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
9690 033774 012701 000001 MOV #1,R1 ;DUMMY UP R1 FOR A FORCED SOB EXIT
9691 034000 000002 RTI
9692
9693 ;*****
9694 .SBTTL TRAP TIMEOUT (TRAP TO 4) HANDLER
9695 034002 004737 034040 TIMEOUT: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
9696 034006 005237 002064 FATAL 6 ;SET FATAL INDICATOR
034006 104006 ERROR +6
.DSABL CRF
9697 ;*****
9698 .SBTTL TRAP MEMORY MANAGEMENT (TRAP TO 250) HANDLER
9699 034014 004737 034040 MMTRAP: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
9700 034020 005237 002064 FATAL 7 ;SET FATAL INDICATOR
034020 104007 INC FATAL$
ERROR +7
.DSABL CRF
9701 .SBTTL TRAP RESERVED INSTRUCTION HANDLER
9702 034026 004737 034040 PDP1105: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
9703 034032 005237 002064 FATAL 5 ;SET FATAL INDICATOR
034032 104005 INC FATAL$
ERROR +5
.DSABL CRF
9704
9710
9711 034040 BADSTACK: SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
;*****
;*SUBTEST FIND BAD SP, PC, & PSW FROM STACK
;*****
9712 034040 010637 002026 MOV SP,BADSP
9713 034044 062737 000002 002026 ADD #2,BADSP
9714 034052 016637 000002 002022 MOV 2(SP),BADPC
9715 034060 016637 000004 002032 MOV 4(SP),BADPSW
9716 034066 000207 RETURN
  
```

```
9719          .SBTTL TRAP   KERNEL TRAP HANDLER
9720          ;*****
9721          ;KERNEL IS A TRAP THAT COMES HERE
9722          ;*****
9723          $KERNEL:      BIC      @140000,2(SP)
9724 034070 042766 140000 000002
9725 034076 000002
9726          ;*****
9727          .SBTTL TRAP   ENERGIZE TRAP HANDLER
9728 034100 052737 000001 177572 $ENERGIZE: BIS      @BITO,MMRO
9729 034106 000002
9730          ;*****
9731          .SBTTL TRAP   DEENERGIZE TRAP HANDLER
9732 034110 042737 000001 177572 $DEENERGIZE: BIC @BITO,MMRO
9733 034116 000002
9734          ;*****
9735          .SBTTL TRAP   CACHON TRAP HANDLER
9736 034120 005737 002544 $CACHN: TST      CACHKN      ;IS THERE A CACHE
9737 034124 001406
9738 034126 013737 002544 177746      BEQ      1$              ;NO - RETURN
9739 034134 052737 000001 177746      MOV      CACHKN,CONTRL ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
9740 034142 000002
9741          1$:      RTI
9742          ;*****
9743          .SBTTL TRAP   CACHOFF TRAP HANDLER
9744 034144 005737 002544 $CACHF: TST      CACHKN      ;IS THERE A CACHE?
9745 034150 001403
9746          BEQ      1$              ;NO RETURN
9747          ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
9746 034152 053737 002550 177746      BIS      CACHKF,CONTRL
9747 034160 000002
9747          1$:      RTI
```

```

9750          .SBTTL TRAP LOAD CSR TRAP HANDLER
9751          ;LOAD CORRECT CSR WITH DATA IN CSR
9752          ;PROGRAM CSR'S ASSERT INHIBIT MODE POINTER WHEN LOADED
9753          $LOADC: PUSH    RO,R1          ;SAVE REGISTERS
                                MOV RO,-(SP)
                                MOV R1,-(SP)
9754          034162 010046
9755          034164 010146
9756          034166 013700 002152      MOV    CSRNO,RO          ;CREATE CSR ADDRESS
9757          034172          002536      IF INMECC IS TRUE THEN GOTO 3$ ;DON'T WANT INH. MODE POINTER ON
                                TST INMECC
9758          034176 005737 002536      BNE 3$
9759          034200 005737 002532      TST    PGMCSR          ;PROGRAM IN INTERLEAVED SPACE?
9760          034204 100007          BPL    1$              ;BRANCH IF NOT
9761          034206 113701 002533      MOVB   PGMCSR+1,R1     ;CHECK SECOND CSR
9762          034212 042701 177740      BIC    #C37,R1        ;CLEAR UNNECESSARY BITS
9763          034216 020137 002152      CMP    R1,CSRNO       ;IS THIS THE CURRENT CSR?
9764          034222 001404          BEQ    2$              ;BRANCH IF IT IS
9765          034224 123737 002532 002152 1$: CMPB   PGMCSR,CSRNO    ;IS THIS THE CURRENT CSR?
9766          034232 001003          BNE    3$              ;BRANCH IF NOT
9767          034234 052737 020000 002150 2$: BIS    #BIT13,CSR     ;SET THE INHIBIT MODE POINTER TO 1ST 16K
9768          034242 C13760 002150 172100 3$: MOV    CSR,CSRADD(RO) ;LOAD THE CSR
9769          034250          012601      POP    R1,RO          ;RESTORE REGISTERS
9770          034252 012600          MOV (SP)+,R1
9771          034254 000002          MOV (SP)+,RO
9772          .SBTTL TRAP READ CSR TRAP HANDLER
9773          ;READ THE CORRECT CSR INTO LOCATIONS CSR
9774          $READC: PUSH    RO
                                MOV RO,(SP)
9775          034256 010046
9776          034260 013700 002152      MOV    CSRNO,RO
9777          034264 016037 172100 002150  MOV    CSRADD(RO),CSR ;READ IT
9778          034272 012600          POP    RO
9779          034274 000002          MOV (SP)+,RO
9780          RTI

```



```

9777          .SBTTL TRAP TEST (R1) & READ CSR CAREFULLY
9778 034276          #TSTRD: PUSH R0,R2,R3
          034276 010046
          034300 010246
          034302 010346
9779 034304 012700 172100 MOV #CSRADD,R0 ;CREATE CSR ADDRESS
9780 034310 063700 002152 ADD CSRNO,R0
9781 034314 005002 CLR R2
9782 034316 005737 002532 TST PGMCSR
9783 034322 100007 BPL 1#
9784 034324 113703 002533 MOVB PGMCSR+1,R3
9785 034330 042703 000200 BIC #BIT7,R3
9786 034334 020337 002152 CMP R3,CSRNO
9787 034340 001404 BEQ 2#
9788 034342 123737 002532 002152 1#: CMPB PGMCSR,CSRNO
9789 034350 001002 BNE 3#
9790 034352 012702 020000 2#: MOV #BIT13,R2
9791 034356 004737 034430 3#: CALL TSTRD1
9792          ;IF SINGLE BIT ERROR ONLY - SET CARRY BIT
9793          5#: POP R3,R2,R0
          034362
          034362 012603
          034364 012602
          034366 012600
          IF #BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
9794 034370
          034370 032737 000020 002150
          034376 001410
          034400 032737 100000 002150
          034406 001004
9795 034410 052766 000001 000002 BIS #BIT0,2(SP)
9796 034416 ELSE
          034416 000403
          034420
          BR L334
9797 034420 042766 000001 000002 BIC #BIT0,2(SP)
9798 034426 END ;OF IF #BIT4
          034426
          L333:;;;;;;
9799 034426 000002 RTI
          034430
          L334:;;;;;;
9800          TSTRD1: MOV R2,(R0) ;
9801 034430 010210 TESTAREA ;
9802 034432 053737 002552 177776 BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
          .DSABL CRF
          TSTB (R1) ;
9803 034440 105711 BIC #BIT15!BIT14,PSW ;
9804 034442 042737 140000 177776 MOV (R0),CSR ;
9805 034450 011037 002150 RETURN ;
9806 034454 000207
  
```

```

9809          .SBTTL TRAP ECC DISABLE ALL CSR'S TRAP HANDLER
9810 034456 012737 000002 002150 #ECCDIS:MOV #BIT1,CSR
9811 034464 004737 035202          CALL CSRROUT
9812 034470 000002          RTI
9813          .SBTTL TRAP ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
9814 034472 012737 000002 002150 #ECC1DIS:MOV #BIT1,CSR
9815 034500 104425          LOADCSR
9816 034502 000002          RTI
9817          .SBTTL TRAP INITIALIZE ALL CSR'S TRAP HANDLER
9818 034504 012737 000001 002150 #ECCINIT:MOV #BIT0,CSR
9819 034512 004737 035202          CALL CSRROUT
9820 034516 000002          RTI
9821          .SBTTL TRAP INITIALIZE 1 SELECTED CSR TRAP HANDLER
9822 034520 012737 000001 002150 #ECC1INIT:MOV #BIT0,CSR
9823 034526 104425          LOADCSR
9824 034530 000002          RTI
9825          .SBTTL TRAP ENABLE SBE PARITY TRAPS ON ALL CSR'S
9826 034532 012737 000003 002150 #ENASBE:MOV #BIT0!BIT1,CSR
9827 034540 004737 035202          CALL CSRROUT
9828 034544 000002          RTI
9829          .SBTTL TRAP ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
9830 034546 012737 000003 002150 #ENA1SBE:MOV #BIT0!BIT1,CSR
9831 034554 104425          LOADCSR
9832 034556 000002          RTI
9833          .SBTTL TRAP WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
9834 034560 013737 002314 002150 #CBCSR:MOV CHECK,CSR ;BITS 11-5
9835 034566 052737 000006 002150          BIS #BIT1!BIT2,CSR ;CHECK MODE
9836 034574 004737 035202          CALL CSRROUT
9837 034600 000002          RTI
9838          .SBTTL TRAP WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
9839 034602 013737 002314 002150 #CB1CSR:MOV CHECK,CSR ;BITS 11-5
9840 034610 052737 000006 002150          BIS #BIT1!BIT2,CSR ;CHECK MODE
9841 034616 104425          LOADCSR
9842 034620 000002          RTI
  
```

```

9845          .SBTTL TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER
9846 034622    $WASSBE:PUSH R1,R4
          MOV R1,-(SP)
          MOV R4,(SP)
          034622 010146
          034624 010446
9847 034626 013701 002224    MOV TOTCSRS,R1 ;GET CSR'S BYTE
9848 034632 005004    CLR R4
9849 034634    BEGIN LWSBE
          B56:;
          034634    FOR CSRNO := #0 TO #36 BY #2
9850 034634 005037 002152
          CLR CSRNO
          034640
          B57:;
9851 034640 006301    ASL R1
9852 034642    ON.ERROR
          BCC L335
          034642 103010
9853 034644 104426    READCSR
9854 034646    IF #BIT4 SET.IN CSR
          BIT #BIT4,CSR
          034646 032737 000020 002150    BEQ L336
          034654 001403
9855 034656    SET R4
          MOV #-1,R4
          034656 C12704 177777    LEAVE LWSBE
          BR E56
9856 034662    END ;OF IF #BIT4
          L336:;
9857 034664    END ;OF ON.ERROR
          L335:;
9858 034664    IF R1 EQ #0 THEN LEAVE LWSBE
          TST R1
          034664 005701    BEQ E56
          034666 001407
9860 034670    END ;OF FOR CSRNO
          ADD #2,CSRNO
          034670 062737 000002 002152    CMP CSRNO,#36
          034676 023727 002152 000036    BLE B57
          034704 003755
          E57:;
9861 034706    END LWSBE
          E56:;
          034706
9862 034706 006004    ROR R4 ;SET C BIT FOR ERROR
9863 034710    POP R4,R1
          MOV (SP)+,R4
          034710 012604    MOV (SP)+,R1
          034712 012601
9864 034714    ON.ERROR
          BCC L337
          034714 103004
9865 034716 052766 000001 000002    BIS #BIT0,2(SP)
9866 034724    ELSE
          BR L340
          034724 000403
          L337:;
          034726
9867 034726 042766 000001 000002    BIC #BIT0,2(SP)
9868 034734    END ;OF ON.ERROR
          L340:;
          034734
9869 034734 000002    RTI
  
```

```
9870 .SBTTL TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
9871 ;ON RETURN IF CARRY IS SET THERE WAS A SBE
9872 034736 104426 #WAS1SBE:READCSR
9873 034740 042766 000001 000002 BIC #BIT0,2(SP) ;CLR C BIT ON STACK
9874 034746 032737 000020 002150 BIT #BIT4,CSR
9875 034754 001403 BEQ 1#
9876 034756 052766 000001 000002 BIS #BIT0,2(SP) ;SET C BIT ON STACK
9877 034764 000002 1# RTI
```

```

9880          .SBTTL  TRAP  WAS THERE A DBE ON ANY CSR TRAP HANDLER
9881 034766          $WASDBE:PUSH  R1,R4
                                MOV R1,-(SP)
                                MOV R4,-(SP)
9882 034772 010146          MOV  TOTCSRS,R1      ;GET CSR'S BYTE
9883 034776 010446          CLR   R4
9884 035000 013701 002224    BEGIN LWDBE
                                B60:;;;;;;;;
9885 035000 005037 002152    FOR CSRNO := #0 TO #36 BY #2
                                CLR CSRNO
9886 035004 006301          ASL   R1
9887 035006 103010          ON.ERROR
                                B61:;;;;;;;;
                                BCC L341
9888 035010 104426          READCSR
9889 035012 032737 100000 002150  IF #BIT15 SET.IN CSR
                                BIT #BIT15,CSR
9890 035022 001403          SET   R4
                                BEQ L342
9891 035026 012704 177777    LEAVE LWDBE
                                MOV #-1,R4
9892 035030 000411          END ;OF IF #BIT4
                                BR E60
9893 035030 000000          END ;OF ON.ERROR
                                L342:;;;;;;;;
9894 035030 000000          IF R1 EQ #0 THEN LEAVE LWDBE
                                L341:;;;;;;;;
9895 035032 005701          END ;OF FOR CSRNO
                                TST R1
9896 035034 001407          BEQ E60
9897 035036 062737 000002 002152  ADD #2,CSRNO
9898 035042 023727 002152 000036  CMP CSRNO,#36
9899 035050 003755          BLE B61
9900 035052          E61:;;;;;;;;
9901 035052          END LWDBE
                                E60:;;;;;;;;
9902 035054 006004          ROR   R4
9903 035056 012604          POP   R4,R1      ;SET C BIT FOR ERROR
9904 035060 012601          ON.ERROR
                                MOV (SP),R4
9905 035062 103004          MOV (SP),R1
9906 035064 052766 000001 000002  BCC L343
9907 035070 000403          ELSE
                                BR L344
9908 035072 042766 000001 000002  BIS #BIT0,2(SP)
9909 035100 000002          ELSE
                                L343:;;;;;;;;
9910 035100 000002          BIC #BIT0,2(SP)
9911 035100 000002          END ;OF ON.ERROR
                                L344:;;;;;;;;
9912 035100 000002          RTI
    
```

```
9905 .SBTTL TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
9906 ;ON RETURN IF CARRY IS SET THERE WAS A DBE
9907 035102 104426 $WAS1DBE:READCSR
9908 035104 005737 002150 TST CSR ;DBE?
9909 035110 100004 BPL 3# ;NO - SKIP
9910 035112 052766 000001 000002 BIS #BIT0,2(SP) ;SET C BIT ON STACK
9911 035120 000002 RTI
9912 035122 042766 000001 000002 3#: BIC #BIT0,2(SP) ;CLR C BIT ON STACK
9913 035130 000002 RTI
```

```

TRAP CLEAR ALL ECC CSR'S TRAP HANDLER
9916 .SBTTL TRAP CLEAR ALL ECC CSR'S TRAP HANDLER
9917 035132 #CLRCSR: CLEAR CSR CLR CSR
      035132 005037 002150
9918 035136 004737 035202 CALL CSROUT
9919 035142 000002 RTI
9920 .SBTTL TRAP CLEAR 1 SELECTED CSR TRAP HANDLER
9921 035144 #CLR1CSR: CLEAR CSR CLR CSR
      035144 005037 002150
9922 035150 104425 LOADCSR
9923 035152 000002 RTI
9924 .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
9925 ;CHECKBITS ALREADY IN LOC "CSR"
9926 035154 052737 000006 002150 #CHKDIS: BIS #BIT1!BIT2.CSR ;ECC DISABLE & DIAG CHECK MODE
9927 035162 004737 035202 CALL CSROUT
9928 035166 000002 RTI
9929 .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
9930 ;CHECKBITS ALREADY IN LOC "CSR"
9931 035170 052737 000006 002150 #CHK1DIS: BIS #BIT1!BIT2.CSR ;ECC DISABLE & DIAG CHECK MODE
9932 035176 104425 LOADCSR
9933 035200 C00002 RTI
    
```

```

9936 035202 CSROUT: SUBTST <<SUBR WRITE IN ALL CSR'S>>
;*****
;#SUBTEST SUBR WRITE IN ALL CSR'S
;*****
9937 035202 PUSH R1 MOV R1,-(SP)
035202 010146
9938 035204 013701 002224 MOV TOTCSRS,R1 ;GET CSR'S BYTE
9939 035210 BEGIN LCSROUT
035210 B62:|||||
9940 035210 005037 002152 FOR CSRNO := #0 TO #36 BY #2 CLR CSRNO
035214 B63:|||||
9941 035214 006301 ASL R1
9942 035216 ON.ERROR BCC L345
035216 103001
9943 035220 104425 LOADCSR
9944 035222 END ;OF ON.ERROR
035222 L345:|||||
9945 035222 IF R1 EQ #0 THEN LEAVE LCSROUT
035222 C05701 TST R1
035224 001407 BEQ E62
9946 035226 062737 000002 002152 END ;OF FOR CSRNO ADD #2,CSRNO
035234 023727 002152 000036 CMP CSRNO,#36
035242 003764 BLE B63
035244 E63:|||||
9947 035244 END LCSROUT
035244 E62:|||||
9948 035244 POP R1 MOV (SP)+,R1
035244 012601
9949 035246 000207 RETURN
9950
9951 035250 #INVALID: SUBTST <<TRAP INVALIDATE BACKGROUND PATTERN>>
;*****
;#SUBTEST TRAP INVALIDATE BACKGROUND PATTERN
;*****
9952 035250 PUSH RO,R1 MOV RO,-(SP)
035250 010046 MOV R1,-(SP)
035252 010146
9953 035254 013701 002102 MOV BANK,R1
9954 035260 006301 ASL R1
9955 035262 006301 ASL R1
9956 035264 042761 020000 002666 BIC #BIT13,CONFIG+2(R1)
9957 035272 POP R1,RO MOV (SP)+,R1
035272 012601 MOV (SP)+,RO
035274 012600
9958 035276 000002 RTI
  
```



```

9960 035300          $ERRGEN:          SUBTST<<TRAP  GENERATE AND TEST ERROR ADDRESS>>
;*****
;SUBTEST          TRAP  GENERATE AND TEST ERROR ADDRESS
;*****
9961 035300          PUSH          R0,R1,R2,R3
;*****
035300 010046          MOV          R0,-(SP)
035302 010146          MOV          R1,-(SP)
035304 010246          MOV          R2,-(SP)
035306 010346          MOV          R3,-(SP)
9962 035310 013703 002104      MOV          BANKINDEX,R3
9963 035314 005737 002456      TST          NOSUPER
9964 035320 001003          BNE          6:
9965 035322 013700 172246      MOV          SIPAR3,R0          ;GENERATE WHAT ERROR ADDR SHOULD BE
9966 035326 000402          BR          7:
9967 035330 013700 177646      6: MOV          UIPAR3,R0
9968 035334 072027 177773      7: A: M          #-5,R0
9969 035340 005737 002132      TST          EQFLAG
9970 035344 001002          BNE          1:
9971 035346 042700 177600      BIC          #+C177,R0          ;GET CURRENT ADDRESS BITS 11 AND 12
9972 035352 C00301          1: SWAB          R1
9973 035354 006201          ASR          R1
9974 035356 006201          ASR          R1
9975 035360 006201          ASR          R1
9976 035362 042701 177775      BIC          #+C2,R1
9977 035366 060100          ADD          R1,R0          ;ADD THEM TO THE ADJUSTED PAR VALUE
;GET ERROR ADDRESS FROM CSR UNDER TEST
9978 035370 013701 002150      MOV          CSR,R1
9979 035374 072127 177773      ASH          #-5,R1
9980 035400 042701 177600      BIC          #+C177,R1
9981 035404 005737 002454      TST          NO22BIT          ;IS THIS AN 11/83,11/23-B OR 11/23 ?
9982 035410 001024          BNE          2:          ;BRANCH IF NOT NECESSARY
9983 035412 005737 002132      TST          EQFLAG          ;IS IT EQB?
9984 035416 001421          BEQ          2:          ;BRANCH IF NOT
9985 035420          PUSH          R0          ;SAVE GENERATED ERROR ADDRESS
9986 035420 010046          MOV          R0,-(SP)
9987 035422 013702 002152          MOV          CSRN0,R2          ;GET CSR NUMBER
9988 035426 052762 040000 172100      BIS          #BIT14,CSRADD(R2) ;TURN ON EQB BIT CAREFULLY
9989 035434 016200 172100      MOV          CSRADD(R2),R0 ;GET CSR CONTENTS
9990 035440 042762 040000 172100      BIC          #BIT14,CSRADD(R2) ;TURN OFF EQB BIT CAREFULLY
9991 035446 042700 177037          BIC          #+C740,R0          ;CLEAR EVERYTHING BUT ERROR ADDR
9992 035452 006300          ASL          R0
9993 035454 006300          ASL          R0          ;SHIFT ADDR BITS 18-21 INTO POSITION
9994 035456 060001          ADD          R0,R1          ;ADD TO CURRENT ERROR ADDRESS
9995 035460          POP          R0
9996 035460 012600          MOV          (SP),R0
9997 035462 020001          2: CMP          R0,R1          ;COMPARE REAL AND GENERATED ERR. ADDR.
9998 035464 001420          BEQ          3:          ;BRANCH IF THEY ARE THE SAME
9999 035466 005737 002136          TST          INTFLAG          ;INTERLEAVED?
10000 035472 001411          BEQ          3:          ;NO - WE HAVE AN ERROR
10001 035474 062700 000100          ADD          #100,R0
10002 035500 005737 002140          TST          INT64K          ;64K INTERLEAVED MEMORY?
10003 035506 062700 000100          BNE          4:
10004 035512 020001          4: ADD          #100,R0
10005 035514 001404          CMP          R0,R1
10006 035516 005737 002066          BEQ          5:
10007 035522 001001          5: TST          SKPERR          ;ARE WE SUPPOSED TO SKIP ERROR P.O.?
;YES - SKIP ERROR PRINTOUT
  
```

```

10008 035524 104462          PERR36          ;ELSE PRINT ERROR ADDRESS ERROR
10009 035526 010137 002460  S#:  MOV      R1,ERRADD ;SAVE CSR'S ERROR ADDRESS
10010 035532 005037 002066  CLR      SKPERR   ;ENABLE THE ERROR PRINTOUT AGAIN
10011 035536          POP      R3,R2,R1,R0 ;RESTORE REGISTERS
      035536 012603          MOV (SP),R3
      035540 012602          MOV (SP),R2
      035542 012601          MOV (SP),R1
      035544 012600          MOV (SP),R0
10012 035546 000002          RTI
10013
10014 035550          ;CBREG: SUBTST <<TRAP      ENAP .E CHECKBIT REGISTER>>
      ;*****
      ;*SUBTEST      TRAP      ENABLE CHECKBIT REGISTER
      ;*****
10015 035550 005037 002150          CLR      CSR      ;
10016 035554 052737 000004 002150  BIS      #BIT2,CSR ;ENABLE DIAGNOSTIC MODE
10017 035562 104425          LOADCSR ;LOAD CSR REGISTER
10018 035564 000002          RTI      ;
10019
10020 035566          ;SYNREG: SUBTST <<TRAP      ENABLE SYNDROME BIT REGISTER>>
      ;*****
      ;*SUBTEST      TRAP      ENABLE SYNDROME BIT REGISTFR
      ;*****
10021 035566 005037 002150          CLR      CSR      ;
10022 035572 052737 040004 002150  BIS      #BIT14!BIT2,CSR ;ENABLE DIAGNOSTIC MODE
10023 035600 104425          LOADCSR ;LOAD CSR REGISTER
10024 035602 000002          RTI      ;
10025

```



```

10077 036004 062702 000004          ADD    #4,R2
10078 036010 022737 000001  G02240      CMP    #1,SPLTCSR
10079 036016 001403          BEQ    10#
10080 036020 010200          MOV    R2,R0
10081 036022 010102          MOV    R1,R2
10082 036024 010001          MOV    R0,R1
10083 036026 012122          10#:  MOV    (R1)+,(R2)+
10084 036030 011112          MOV    (R1),(R2)
10085 036032 013700 002104          MOV    BANKINDEX,R0
10086 036036 005737 002140          TST    INT64K
10087 036042 001403          BEQ    11#
10088 036044 012700 004000          MOV    #4000,R0
10089 036050 000402          BR     12#
10090 036052 012700 010000          11#:  MOV    #10000,R0
10091 036056 005737 002456          12#:  TST    NOSUPER
10092 036062 001403          BEQ    13#
10093 036064 012701 177652          MOV    #UIPAR5,R1
10094 036070 000402          BR     14#
10095 036072 012701 172252          13#:  MOV    #SIPAR5,R1
10096 036076 060021          14#:  ADD    R0,(R1)+
10097 036100 060011          ADD    R0,(R1)
10098          ;IF WE ONLY HAVE AN 124K SYSTEM, WE DON'T WANT TO TEST THE
10099          ;LAST 4K, WHERE THE Q-BUS DEVICE PAGE IS. INSTEAD, THE
10100          ;PROGRAM WILL REMAP THE LAST 4K TO 8-12K. ALSO, IF THERE
10101          ;IS A BANK 177 ON AN 11/83, THE PROGRAM WILL REMAP THE LAST
10102          ;4K TO 8-12K FOR THE SAME REASON.
10103 036102 022737 000007 002556  9#:  CMP    #7, LASTBANK
10104 036110 001010          BNE    7#
10105 036112 005737 002454          TST    NO22BIT          ;11/83,11/23-B OR 23?
10106 036116 001423          BEQ    3#              ;BRANCH IF SO
10107 036120 022737 000007 002102          CMP    #7,BANK        ;BANK 7?
10108 036126 001017          BNE    3#              ;NO - BRANCH
10109 036130 000404          BR     8#
10110 036132 022737 000177 002102  7#:  CMP    #177,BANK
10111 036140 001012          BNE    3#
10112 036142 005737 002456          8#:  TST    NOSUPER
10113 036146 001404          BEQ    6#
10114 036150 013737 177652 177654          MOV    UIPAR5,UIPAR6
10115 036156 000403          BR     3#
10116 036160 013737 172252 172254  6#:  MOV    SIPAR5,SIPAR6
10117 036166          3#:  POP    R5,R4,R2,R1,R0
          MOV (SP)+,R5
          MOV (SP)+,R4
          MOV (SP)+,R2
          MOV (SP)+,R1
          MOV (SP)+,R0
10118 036200 000207          RETURN
10119          .SBTTL TRAP MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
10120 036202          $KMAP: PUSH    R0,R1,R2,R3,R4
          MOV R0,-(SP)
          MOV R1,-(SP)
          MOV R2,-(SP)
          MOV R3,-(SP)
          MOV R4,(SP)
10121 036214 013737 172354 036306          MOV    KIPAR6,SVKPAR6
10122 036222 005000          CLR    R0              ;1ST AREA TO MAP TO
10123 036224 012701 172340          MOV    #KIPAR0,R1      ;FIRST ADDRESS
    
```

10124	036230	012702	077406		MOV	#77406,R2		;CONSTANT FOR 4K PAGE,UP,R/W
10125	036234	012703	172300		MOV	#KIPDR0,R3		;1ST PAGE DESCRIPTOR REGISTER
10126	036240	012704	000010		MOV	#8,R4		;COUNTER
10127	036244	010021		1#:	MOV	R0,(R1)+		;PUT IN KERNEL ADDRESS
10128	036246	010223			MOV	R2,(R3)+		;PUT IN KERNEL DISCRIPTOR
10129	036250	062700	000200		ADD	#200,R0		;ADD ADDRESS CONSTANT FOR 4K CHANGE
10130	036254	077405			SQB	R4,1#		;LOOP TILL DONE
10131	036256	012741	177600		MOV	#177600,-(R1)		;THE PERIPHERALS PAGE TO KIPAR7
10132	036262	01741	177400		MOV	#177400,-(R1)		;AND NEXT LOWER PAGE TO KIPAR6
10133	036266	01711	036306		MOV	SVKPAR6,(R1)		
10140	036272				POP	R4,R3,R2,R1,R0		
	036272	012604						MOV (SP)+,R4
	036274	012603						MOV (SP)+,R3
	036276	012602						MOV (SP)+,R2
	036300	012601						MOV (SP)+,R1
	036302	012600						MOV (SP)+,R0
10141	036304	000002			RTI			
10142	036306	000000			SVKPAR6:.WORD	0		::K SAVE THE XXDP V2 KPAR6 POINTER HERE

```

10145 036310 RELOCATE:SUBTST <<RELOCATE PROGRAM>>
;*****
;SUBTEST RELOCATE PROGRAM
;*****
10146 036310 032777 010000 144320 IF #SW12 SET.IN #SWR THEN #RETURN ERROR
036310 032777 010000 144320 BIT #SW12,#SWR
036316 001402 BEQ L346
036320 000261 SEC
036322 000207 RTS PC
036324 L346:;;;;;
10147 036324 IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
036324 005737 002352 TST APTFLAG
036330 001003 BNE L347
036332 005737 002350 TST ACTFLAG
036336 001405 BEQ L350
036340 L347:;;;;;
10148 036340 IF $PASS NE #0 THEN #RETURN ERROR
036340 005737 056724 TST $PASS
036344 001402 BEQ L351
036346 000261 SEC
036350 000207 RTS PC
036352 L351:;;;;;
10149 036352 END; OF IF APTFLAG
036352 L350:;;;;;
10150 036352 BEGIN LOADERBANK
036352 B64:;;;;;
10151 036352 FOR BANK := #1 TO LASTBANK
036352 012737 000001 002102 MOV #1,BANK
036360 B65:;;;;;
10152 036360 004737 037760 CALL EXBANK
10153 036364 IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE
036364 005737 002116 TST ACFLAG
036370 001431 BEQ L352
036372 005737 002122 TST PFLAG
036376 001026 BNE L352
036400 005737 002130 TST BMFLAG
036404 001023 BNE L352
10154 036406 013700 002102 MOV BANK,RO
10155 036412 010037 002432 MOV RO,LOADBANK
10156 036416 013701 002576 MOV LOADHOME,R1
10157 036422 004737 037350 CALL BANKMOV
10158 036426 004737 037702 CALL NEWLOAD ;MAP NEW LOADER BANK IN KERNEL
10159 036432 013701 002104 MOV BANKINDEX,R1
10160 036436 052761 100000 002666 BIS #BIT15,CONFIG+2(R1) ;MARK LOADER
10161 036444 042761 020000 002666 BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
10162 036452 LEAVE LOADERBANK
036452 000416 BR E64
10163 036454 END ;OF IF ACFLAG
036454 L352:;;;;;
10164 036454 END ;OF FOR BANK
036454 005237 002102 INC BANK
036460 023737 002102 002556 CMP BANK,LASTBANK
036466 003734 BLE B65
036470 E65:;;;;;
10165 036470 IF #SW13 OFF.IN #SWR
036470 032777 020000 144140 BIT #SW13,#SWR
036476 001002 BNE L353

```

```

10166 036500 104401 070246          TYPE          MSG075          ;RELOCATION NOT POSSIBLE
      036500          .DSABL CRF          TYPEIT ,MSG075
10167 036504          END ;OF IF #SW13          .DSABL CRF
      036504          $RETURN ERROR          END ;OF IF #SW13
10168 036504          000261          $RETURN ERROR          L353:;;;;;
      036506          000207          SEC
10169 036510          END LOADERBANK          RTS PC
      036510          BEGIN FINDBANK          E64:;;;;;
10170 036510          013702 002556          MOV LASTBANK,R2          B66:;;;;;
      036510          006302          ASL R2
10171 036510          006302          ASL R2          ;R2 <- R2 * 4
10172 036514          012701 000004          FOR R1 := #2*2 TO R2 BY #4
      036516          012701 000004          MOV #2*2,R1
10173 036520          032761 000201 002664          IF #BIT7!BIT0 OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT
      036522          001035          B67:;;;;;
10174 036524          032761 100000 002666          IF #BIT15 OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK
      036526          001031          BIT #BIT7!BIT0,CONFIG(R1)
10175 036534          032761 002106 002664          IF CPUBIT SET.IN CONFIG(R1) ;IF ACCESSABLE
      036536          001425          BNE L354
10176 036544          032761 000400 002666          IF #BIT8 OFF.IN CONFIG+2(R1) THEN LEAVE FINDBANK ;IF PARITY
      036546          001460          BIT #BIT15,CONFIG+2(R1)
10177 036554          032761 000100 002664          IF #BIT6 SET.IN CONFIG(R1) AND #BIT7 OFF.IN CONFIG(R1)
      036556          001405          BIT CPUBIT,CONFIG(R1)
10178 036564          032761 000200 002664          ;IF 1ST PROTECTABLE ECC BANK
      036602          001001          LEAVE FINDBANK
10180 036604          000447          END ;OF IF #BIT6          BR E66
10181 036606          IF INHECC IS FALSE          L357:;;;;;
10182 036606          005737 002536          SET INHECC          TST INHECC
10183 036612          012737 177777 002536          MOV R1,INHBANK          BNE L360
10184 036614          010137 002540          END; OF IF INHECC          MOV #-1,INHECC
10185 036622          END ;OF IF CPUBIT          L360:;;;;;
10186 036626          END ;OF IF #BIT15          L356:;;;;;
10187 036626          END ;OF IF #BIT7          L355:;;;;;
10188 036626          END ;OF FOR          L354:;;;;;
10189 036626          062701 000004          ADD #4,R1
      036632          020102          CMP R1,R2

```

```

036634 003733
036636
10191 036636 IF FULLREL IS FALSE
036636 005737 002542
036642 001012
10192 036644 IF INHECC IS TRUE
036644 005737 002536
036650 001407
10193 036652 013701 002540
10194 036656 023727 002300 000030
10195 036664 001421
10196 036666 000420
10197 036670
036670
10198 036670
036670
10199 036670 005037 002536
10200 036674 032777 020000 143734
036674 001006
036702 C01006
10201 036704 023727 002300 000030
10202 036712 001402
10203 036714 104401 070246
036714
10204 036720
036720
10205 036720 SKUB: $RETURN ERROR
036720 000261
036722 000207
10206 036724
036724
10207 036724
036724 005037 002536
10208 036730 042761 020000 002666
10209 036736 005000
10210 036740 071027 000004
10211 036744
036744 010037 002310
10212 036750 013737 002532 002534
10213 036756 004737 037516
10214 036762
036762 052737 140000 177776
10215 036770
036770 004537 040732
036774 040000
036776 100000
037000 000000
10216 037002 104417
10217 037004 042737 000001 177572
10218 037012 004737 037600
10219 037016 013700 002310
10220 037022 006300
10221 037024 006300
10222 037026 016002 002664

```

```

BLE B67
E67:;;;;;;;;
TST FULLREL
BNE L361
TST INHECC
BEQ L362
MOV INHBANK,R1
CMP REALPAT,#30
BEQ RELENT1
BR RELENT1
END; OF IF INHECC
END; OF IF FULLREL
CLR INHECC
IF #SW13 OFF.IN @SWR
;IS THIS PATTERN 30?
;YES - SKIP MESSAGE
L362:;;;;;;;;
L361:;;;;;;;;
;MAKE SURE FLAG IS TURNED OFF!
BIT #SW13,@SWR
BNE L363
CMP REALPAT,#30
BEQ SKUB
TYPE MSG075
TYPEIT ,MSG075
.DSABL CRF
END ;OF IF #SW13
L363:;;;;;;;;
SEC
RTS PC
E66:;;;;;;;;
;IF WE RELOCATED PROPERLY, THIS SHOULD BE OFF!
CLR INHECC
;INVALIDATE BACKGROUND PATTERN
MOV RO,NEWBANK
;SAVE CURRENT PGM. CSR
;MAP NEWBANK TO USER PAR
;ENTER USER MODE
;GO TO USER MODE
;MOVE PROGRAM
JSR R5,BLOCK3
SIZE
100000
0
.DSABL CRF
KERNEL
;ENTER KERNEL MODE
;DEENERGIZE MEMORY MANAGEMENT
JMPRL1: BIC #BIT0,MMRO
CALL NEWKERNEL
MOV NEWBANK,RO
ASL RO
ASL RO
;RO <- RO * 4
MOV CONFIG(RO),R2

```


10223	037032	000302			SWAB	R2		
10224	037034	042702	177760		BIC	#C17,R2		
10225	037040	006302			ASL	R2		
10226	037042	052737	000001	177572	BIS	#BIT0,MMRO		;ENERGIZE MEMORY MANAGEMENT
10227	037050	010237	002532		MOV	R2,PGMCSR		;PUT NEW PGM. CSR INTO PGMCSR
10228	037054	032760	010000	002666	BIT	#BIT12,CONFIG+2(R0)		;IS THE NEW BANK INTERLEAVED?
10229	037062	001412			BEQ	1#		;BRANCH IF NOT INTERLEAVED
10230	037064	016002	002664		MOV	CONFIG(R0),R2		
10231	037070	042702	007777		BIC	#C170000,R2		
10232	037074	072227	177775		ASH	#-3,R2		
10233	037100	052702	100000		BIS	#BIT15,R2		
10234	037104	050237	002532		BIS	R2,PGMCSR		
10235	037110				SET	RLFLAG		
	037110	012737	177777	002126				MOV # 1,RLFLAG
10236	037116				\$RETURN	NOERROR		
	037116	000241						CLC
	037120	000207						RTS PC

```

10239 037122 UNRELOCATE:SUBTST <<UNRELOCATE PROGRAM>>
;*****
;SUBTEST UNRELOCATE PROGRAM
;*****
10240 ;RESTORE LOADERS
10241 037122 PUSH RO MOV RO, (SP)
037122 010046
10242 037124 013701 002432 MOV LOADBANK,R1
10243 037130 013700 002576 MOV LOADHOME,R0
10244 037134 004737 037350 CALL BANKMOV
10245 037140 004737 037702 CALL NEWLOAD ;MAP NEW LOADER BANK IN KERNEL SPACE
10246 037144 PUSH BANK MOV BANK,-(SP)
037144 013746 002102
10247 037150 013737 002432 002102 MOV LOADBANK,BANK
10248 037156 004737 037760 CALL EXBANK
10249 037162 013701 002104 MOV BANKINDEX,R1
10250 037166 042761 100000 002666 BIC #BIT15,CONFIG+2(R1) ;CLEAR LOADER FLAG
10251 037174 013737 002576 002102 MOV LOADHOME,BANK
10252 037202 004737 037760 CALL EXBANK
10253 037206 013701 002104 MOV BANKINDEX,R1
10254 037212 042761 020000 002666 BIC #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
10255 037220 POP BANK
037220 012637 002102 MOV (SP)+,BANK
10256 037224 CLEAR INHECC ;MAKE SURE ECC TESTS ARE NOT INHIBITED!
037224 005037 002536 CLR INHECC
10257 ;RESTORE BANK 0
10258 BIC #BIT13,CONFIG+2 ;INVALIDATE BACKGROUND PATTERN
10259 037230 042737 020000 002666 LET NEWBANK := #0
10260 037236 005037 002310 CLR NEWBANK
037236 004737 037516 CALL USERMAP ;MAP NEWBANK TO USER PAR
10261 037242 USER ;ENTER USER MODE
10262 037246 052737 140000 177776 BIS #BIT15:BIT14,PSW ;GO TO USER MODE
.DSABL CRF
BMOV 0,100000,SIZE ;MOVE PROGRAM
JSR R5,BLOCK3
SIZE
100000
0
.DSABL CRF
10264 037266 104417 KERNEL ;ENTER KERNEL MODE
10265 037270 042737 000001 177572 BIC #BIT0,MMRO ;DEENERGIZE MEMORY MANAGEMENT
10266 037276 004737 037600 CALL NEWKERNEL
10267 037302 013737 002534 002532 MOV PGMCSR+2,PGMCSR ;RESTORE PREVIOUS PGM. CSR
10268 037310 052737 000001 177572 BIS #BIT0,MMRO ;ENERGIZE MEMORY MANAGEMENT
10269 037316 005037 002126 CLR RLFLAG
10270 037322 012700 002666 1#: MOV #CONFIG+2,R0 ;MOVE 2ND WORD OF CONFIG TO R0
10271 037326 042710 020000 2#: BIC #BIT13,(R0) ;CLEAR BACKGROUND VALID BIT
10272 037332 062700 000004 ADD #4,R0 ;INCREMENT TO NEXT BANK
10273 037336 020027 003620 CMP R0,#3620 ;DONE?
10274 037342 003771 BLE 2# ;NO - BRANCH
10275 037344 POP RO MOV (SP)+,R0
037344 012600
10276 037346 000207 RETURN
10277

```

10280 037350

BANKMOV:SUBTST <<MOVE BANKS>>
;*****
;*SUBTEST MOVE BANKS
;*****

10281
10282
10283
10284
10285 037350 104415
10286 037352 004737 037516
10287 037356 104416
10288 037360 104415
10289 037362 072027 000011
10290 037366 072127 000011
10291 037372 012702 177650
10292 037376 012703 000200
10293
10294 037402 010122
10295 037404 060301
10296 037406 C10122
10297 037410 060301
10298
10299 037412 010022
10300 037414 060300
10301 037416 010022
10302 037420 060300
10303
10304 037422
037422 052737 140000 177776
10305 037430
037430 004537 040732
037434 020000
037436 140000
037440 100000

;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 CPF

```

10306 037442 104417          KERNEL          ;ENTER KERNEL MODE
10307
10308 037444 012702 177650    MOV      #UIPAR4,R2
10309
10310 037450 010122          MOV      R1,(R2)+          ;MAP 2ND HALF BANK
10311 037452 060301          ADD      R3,R1            ;BUMP BY 4K
10312 037454 010122          MOV      R1,(R2)+
10313 037456 060301          ADD      R3,R1
10314
10315 037460 010022          MOV      R0,(R2)+
10316 037462 060300          ADD      R3,R0
10317 037464 010022          MOV      R0,(R2)+
10318 037466 060300          ADD      R3,R0
10319
10320 037470          USER
      037470 052737 140000 177776    BIS      #BIT15!BIT14,PSW          ;GO TO USER MODE
      .DSABL CRF
10321 037476          BMOV     100000,140000,SIZE/4      ;MOV 3RD FOURTH OF BANK
      037476 C04537 040732          JSR     R5,BLOCK3
      037502 010000          SIZE/4
      037504 140000          140000
      037506 100000          100000
      .DSABL CRF
10322 037510 104417          KERNEL          ;ENTER KERNEL MODE
10323
10324 037512 104416          RESREG
10325 037514 000207          RETURN

```

```

10328 037516      USERMAP:SUBTST  <<SUBR  MAP USER TO NEW BANK>>
;*****
;SUBTEST        SUBR  MAP USER TO NEW BANK
;*****
10329 037516 012701 177640      MOV    #UIPAR0,R1      ;COPY KERNEL PAR'S & PDR'S (0-3)
10330 037522 012702 172340      MOV    #KIPAR0,R2
10331 037526 012703 177600      MOV    #UIPDR0,R3
10332 037532 012704 172300      MOV    #KIPDR0,R4
10333 037536 012705 000004      MOV    #4,R5
10334 037542 012221          1#:  MOV    (R2)+,(R1)+
10335 037544 011423          MOV    (R4),(R3)+
10336 037546 077503          SOB    R5,1#
10337
10338 037550 013700 002310      MOV    NEWBANK,R0
10339 037554 072027 000011      ASH    #9.,R0          ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000
10340
10341 037560 012705 000004      MOV    #4,R5
10342 037564 010021          2#:  MOV    R0,(R1)+        ;SETUP UIPAR(4-7)
10343 037566 062700 000200      ADD    #200,R0        ;BUMP ADDRESS #K
10344 037572 C11423          MOV    (R4),(R3)+    ;SETUP UIPDR(4-7)
10345 037574 077505          SOB    R5,2#
10346 037576 000207          RETURN
10347
10348 037600      NEWKERNEL:SUBTST  <<SUBR  SETUP KERNEL PAR'S FOR NEW BANK>>
;*****
;SUBTEST        SUBR  SETUP KERNEL PAR'S FOR NEW BANK
;*****
10349 037600          PUSH   R0,R1,R5
;*****
037600          MOV    R0,-(SP)
037602          MOV    R1,-(SP)
037604          MOV    R5,-(SP)
10350 037606 012700 172340      MOV    #KIPAR0,R0
10351 037612 013701 002310      MOV    NEWBANK,R1
10352 037616 072127 000011      ASH    #9.,R1          ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000
10353
10354 037622 012705 000004      MOV    #4,R5
10355 037626 010120          1#:  MOV    R1,(R0)+        ;SETUP KIPAR(0-3)
10356 037630 062701 000200      ADD    #200,R1
10357 037634 077504          SOB    R5,1#
10358 037636          POP    R5,R1,R0
;*****
037636          MOV    (SP)+,R5
037640          MOV    (SP)+,R1
037642          MOV    (SP)+,R0
10359 037644 000207          RETURN
10360
10361 037646      MAPKERNAL:SUBTST <<SUBR  MAP KERNAL PARS 4 AND 5 TO A BANK>>
;*****
;SUBTEST        SUBR  MAP KERNAL PARS 4 AND 5 TO A BANK
;*****
10362
10363 037646 013705 002102      MOV    BANK,R5          ;MOV BANK NUMBER TO R5
10364 037652 072527 000011      ASH    #9.,R5          ;R5 ENTERS 100000 LESS SHIFT TO CREATE MAPPING
10365 037656 013737 172350      MOV    KIPAR4,SAVPAR   ;SAVE OLD PAR
002272          MOV    R5,KIPAR4      ;GET NEW PAR'S
10366 037664 010537 172350      ADD    #200,R5
10367 037670 062705 000200      MOV    R5,KIPAR5
10368 037674 010537 172352      MOV    R5,KIPAR5
10369 037700 000207          RETURN

```

```

10370
10371 037702      MEMLOAD:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
;*****
;#SUBTEST      SUBR      SETUP KERNEL PAR'S FOR NEW LOADER BANK
;*****
10372
10373 037702      ;RO CONTAINS THE DESTINATION BANK
037702 010046      PUSH      RO,R1
037704 010146      MOV      RO,-(SP)
037706 012701 172350  MOV      #KIPAR4,R1
037712 072027 000011  ASH      #9,R0      ;BANK 1 STARTS AT 100000 LESS 6 LSB'S (1000)
037716 010021  MOV      RO,(R1)+   ;SETUP KIPAR4
037720 062700 000200  ADD      #200,R0
037724 010021  MOV      RO,(R1)+   ;SETUP KIPARS5
037726 012601  POP      R1,R0
037730 012600  MOV      (SP)+,R1
037732 000207  MOV      (SP)+,RO
10380 037732      RETURN
10381
10382 037734      UNMAP: SUBTST <<SUBR UNMAP KERNAL PAR'S 4 AND 5>>
;*****
;#SUBTEST      SUBR      UNMAP KERNAL PAR'S 4 AND 5
;*****
10383 037734 013737 002272 172350  MOV      SAVPAR,KIPAR4      ;RESTORE KIPAR4
10384 037742 062737 000200 002272  ADD      #200,SAVPAR      ;ADD 200 FOR NEXT PAR
10385 037750 013737 002272 172352  MOV      SAVPAR,KIPARS5     ;RESTORE KIPARS5
10386 037756 000207      RETURN
;

```

10389 037760

```

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.
;

```

10390
 10391
 10392
 10393
 10394
 10395
 10396
 10397
 10398
 10399
 10400
 10401
 10402
 10403
 10404
 10405
 10406
 10407
 10408

```

10409 037760          PUSH      R0,R1,R2                                MOV R0,-(SP)
      037760 010046          MOV R1,-(SP)
      037762 010146          MOV R2,-(SP)
      037764 010246
10410 037766          CLEAR     MKFLAG,KPFLAG                          CLR MKFLAG
      037766 005037 002120          CLR KPFLAG
      037772 005037 002114          SET      ACFLAG
10411 037776          CLEAR     PFLAG,RRFLAG,BMFLAG                    MOV #-1,ACFLAG
      037776 012737 177777 002116          CLR PFLAG
10412 040004          CLEAR     INTFLAG,INT64K,SKIPMK                 CLR RRFLAG
      040004 005037 002122          CLR BMFLAG
      040010 005037 002124
      040014 005037 002130
10413 040020          MOV      BANK,R1
      040020 005037 002136          ASL     R1
      040024 005037 002140          ASL     R1 ;R1 <- R1 * 4
      040030 005037 002342          MOV     R1,BANKINDEX
10414 040034 013701 002102          BIT    #BIT6,CONFIG(R1) ;PROTECTED REGION OF ECC MEMORY?
10415 040040 006301          BEQ    1$ ;NO SKIP
10416 040042 006301
10417 040044 010137 002104          SET    KPFLAG
10418 040050 032761 000100 002664
10419 040056 001403
10420 040060          MOV     #-1,KPFLAG
      040060 012737 177777 002114
10421 040066 012700 000002 1$: MOV #BIT1,R0
10422 040072          IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)
      040072 030037 002106          BIT R0,CPUBIT
      040076 001405          BEQ L364
      040100 030061 002664          BIT R0,CONFIG(R1)
      040104 001002          BNE L364
10423 040106 005037 002116          CLR ACFLAG
10424 040112          END ;OF IF R0
      040112

```

L364:::1:::

```

10425 040112 005737 002116      TST      ACFLAG      ;ACTIVE MEMORY?
10426 040116 001415      BEQ      2$          ;BRANCH IF NOT
10427 040120 016102 002666      MOV      CONFIG+2(R1),R2
10428 040124 000302      SWAB     R2
10429 040126 042702 177770      BIC      #1C7,R2     ;ISOLATE MEM TYPE BITS
10430 040132 020227 000003      CMP      R2,#3       ;IS THIS AN ILLEGAL MEM TYPE?
10431 040136 003405      BLE     2$          ;BRANCH IF NOT
10432 040140      SET      BMFLAG     ;SET BAD BANK FLAG
10433 040146 012737 177777 002130      JMP      ENEXBK     ;JUMP OVER REST OF FLAG TESTS
10434 040152 032761 040354 002666 2$:      BIT      #BIT8,CONFIG+2(R1) ;IS THERE ECC THERE?
10435 040160 001403      BEQ      3$          ;NO - SKIP
10436 040162      SET      MKFLAG     ;YES SET MKFLAG
10437 040170 012737 177777 002120      MOV     #-1,MKFLAG
10438 040176 032761 00020, 002664 3$:      BIT      #BIT7,CONFIG(R1) ;BANK = PROGRAM SPACE?
10439 040200      BEQ      5$          ;NO - SKIP
10439 040200      SET      PFLAG,RRFLAG
10440 040214 012737 177777 002122      MOV     #-1,PFLAG
10440 040214 012737 177777 002124      MOV     #-1,RRFLAG
10441 040220 005737 002126 5$:      TST      RLFLAG     ;IS PROGRAM RELOCATED?
10442 040222 001402      BEQ      6$          ;NO - SKIP
10443 040226 005137 002124 002664 6$:      COM      RRFLAG     ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
10444 040234 032761 000001 002664 6$:      BIT      #BIT0,CONFIG(R1) ;ERRORS PRESENT IN THIS BANK?
10445 040236 001403      BEQ      8$          ;NO - SKIP
10445 040236      SET      BMFLAG
10446 040244 012737 177777 002130      MOV     #1,BMFLAG
10447 040250 005737 002600 8$:      TST      WORST     ;IS THIS A WORST FIRST PASS?
10448 040252 001002      BNE     9$          ;YES - SKIP
10449 040256 005137 002130 9$:      COM      BMFLAG     ;NO - COMPLEMEN, BAD MEMORY FLAG
10449 040256      IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
10449 040256 005737 002002      TST SELONLY
10449 040262 001407      BEQ L365
10449 040264 032761 040000 002666      BIT #BIT14,CONFIG+2(R1)
10449 040272 001003      BNE L365
10450 040274      SET RRFLAG
10451 040274 012737 177777 002124      MOV #1,RRFLAG
10451 040302      END ;OF IF SELONLY
10452 040302 032761 010000 002666      BIT #BIT12,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED?
10453 040310 001421      BEQ ENEXBK ;BRANCH IF IT IS NOT
10454 040312      SET INTFLAG
10455 040312 012737 177777 002136      MOV #1,INTFLAG
10455 040320 032761 004000 002666      BIT #BIT11,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED WITH 64K BOARDS?
10456 040326 001403      BEQ 10$          ;BRANCH IF IT IS NOT
10457 040330      SET INT64K
10458 040330 012737 177777 002140      MOV #-1,INT64K
10459 040336 032761 000040 002664 10$:      BIT #BIT5,CONFIG(R1) ;SHOULD THIS BANK BE TESTED?
10460 040344 001403      BEQ ENEXBK ;BRANCH IF IT SHOULD
10461 040346 012737 177777 002342      SET SKIPMK
10461 040354 012737 177777 002342      ENEXBK: POP R2,R1,R0 ;RESTORE REGISTERS
10461 040354 012602      MOV (SP),R2
10461 040356 012601      MOV (SP),R1
10462 040360 012600      MOV (SP),R0
10462 040362 000207      RETURN
    
```



```

10465 040364 BANKOK: SUBTST <<SUBR BANK OK?>>
;*****
;*SUBTEST SUBR BANK OK?
;*****
10466 ;TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
10467 ;IS OF THE TYPE WE ARE TESTING "TMFLAG".
10468 ;RESULT IS RETURNED IN THE CONDITION CODES (OK = (-0)).
10469 040364 013700 002134 MOV TMFLAG,R0
10470 040370 005100 COM RO
10471 040372 013701 002120 MOV MKFLAG,R1
10472 040376 074001 XOR RO,R1
10473 040400 000207 RETURN ;OK = (-OK)
10474
10475 040402 INCRPT:
10476 040402 INCPAT: SUBTST <<SUBR INCREMENT PATTERN TESTING >>
;*****
;*SUBTEST SUBR INCREMENT PATTERN TESTING
;*****
10477 ;INCREMENT THE PATTERN & SET UP THE CONDITION CODES
10478 ;RESULT - Z BIT SET INDICATES OVERFLOW
10479 040402 005237 002112 INC PATTERN
10480 040406 022737 000030 002112 CMP #30,PATTERN ;SET UP CONDITION CODES
10481 040414 000207 RETURN ;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)
10482
10483 040416 SETPAT:
10484 040416 HIPAT: SUBTST <<SUBR SET HIGHEST PATTERN TESTING TYPE>>
;*****
;*SUBTEST SUBR SET HIGHEST PATTERN TESTING TYPE
;*****
10485 040416 012737 000027 002112 MOV #27,PATTERN ;SET HIGHEST PATTERN
10486 040424 000207 RETURN
10487
10488 040426 INCBNK: SUBTST <<SUBR INCREMENT BANK & TEST>>
;*****
;*SUBTEST SUBR INCREMENT BANK & TEST
;*****
10489 ;RESULTS RETURNED IN CONDITION CODES
10490 040426 005237 002102 INC BANK
10491 040432 023737 002556 002102 CMP LASTBANK,BANK ;TOO FAR?
10492 040440 000207 RETURN

```

```
10495 040442      QUIT:  SUBTST  <<QUIT ROUTINE>>
;*****
;*SUBTEST      QUIT ROUTINE
;*****
10496             ;INITIALIZE ALL CSR'S
10497             ;UNRELOCATE IF NECESSARY
10498             ;FLUSH OUT ANY DBE'S
10499             ;TURN OFF MEMORY MANAGEMENT
10500             ;TURN OFF THE Q-BUS MAP
10501             ;HALT
10502 040442 104472  ECCINIT          ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
10503 040444          SET4   #QUIT1      ;TRAPS TO 4 GOTO QUIT1
          040444 012737 040506 000004  MOV     #QUIT1.4
          ;DSABL CRF
          IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
10504 040452          TST RLFLAG
          040452 005737 002126          BEQ L366
          040456 001402          JSR PC,UNRELOCATE
          040460 004737 037122          L366:::
          040464          ;FLUSH OUT DBE'S
10505 040464 C04737 022066          DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
10506 040470 104421          TST     NO22BIT ;IS THIS AN 11/83,11/23-B OR 11/23?
10507 040472 005737 002454          BNE     QUIT1
10508 040476 001003          BIC     #BITS,MMR3 ;TURN OFF THE Q-BUS MAP
10509 040500 042737 000040 172516  QUIT1: RESET
10510 040506 000005          HALT
10511 040510 000000
10512
```

```
10515 040512          EXIT:  SUBTST  <<HALT PROGRAM>>
;*****
;*SUBTEST          HALT PROGRAM
;*****
10516 040512 004737 040544          CALL  SHUTUP
10517 040516          EXIT2:  IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
      040516 005737 002352          TST  APTFLAG
      040522 001003          BNE  L367
      040524 005737 002350          TST  ACTFLAG
      040530 001402          BEQ  L370
      040532          L367:::
10518 040532 000777          BR
10519 040534          ELSE
      040534 000403          BR  L371
      040536          L370:::
10520 040536 000000          $EXHALT: HALT
10521 040540 000137 003670          JMP  START
10522 040544          END ;OF IF APTFLAG
      040544          L371:::
10523
10524 040544          SHUTUP: SUBTST  <<SHUTDOWN DIAGNOSTIC>>
;*****
;*SUBTEST          SHUTDOWN DIAGNOSTIC
;*****
10525          ;INITIALIZE ALL CSR'S
10526          ;UNRELOCATE
10527          ;FLUSH OUT DBE'S
10528          ;RESTORE LOADERS
10529          ;TURN OFF MEMORY MANAGEMENT
10530          ;UNMAP THE Q-BUS MAP
10534 040544 104472          ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
10535 040546          IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
      040546 005737 002126          TST  RLFLAG
      040552 001402          BEQ  L372
      040554 004737 037122          JSR  PC,UNRELOCATE
      040560          L372:::
10536 040560          IF QUICK IS FALSE
      040560 005737 002436          TST  QUICK
      040564 001002          BNE  L373
10537 040566 004737 022066          CALL  MT0030 ;FLUSH OUT DBE'S
10538 040572          END ;OF IF QUICK
      040572          L373:::
10539 040572 012700 000001          MOV   #1,R0 ;DESTINATION BANK
10540 040576 013701 002576          MOV   LOADHOME,R1 ;SOURCE BANK
10541 040602 004737 037350          CALL  BANKMOV
10542 040606 104421          DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
10543 040610 005737 002454          TST  NO22BIT ;DOES THIS PDP-11 HAVE 22-BIT ADDR?
10544 040614 001003          BNE  1$ ;BRANCH IF NOT
10545 040616 042737 0J0040 172516 BIC   #BITS.MMR3 ;TURN OFF Q-BUS MAP
10549 040624 000207          1$:  RETURN
10550
10551 040626          APTDOWN:SUBTST <<APT SHUTDOWN SEQUENCE>>
;*****
;*SUBTEST          APT SHUTDOWN SEQUENCE
;*****
10552 040626 010346          MAP   #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
      040626          MOV  R3,-(SP)
```

	040630	012703	000000		MOV	#0,R3		
	040634	004737	035604		CALL	MAPPER		
					.DSABL	CRF		
	040640	012603						MOV (SP),R3
10553	040642				TESTAREA		;ENTER TEST MODE	
	040642	053737	002552	177776	BIS	TESTMODE,PSW		;GO TO SYSTEM TEST MODE
					.DSABL	CRF		
10554	040650	012737	040626	060024	MOV	#APTDOWN,FIRST+24		
10555	040656	012737	000340	060026	MOV	#340,FIRST+26		
10556	040664	012737	000000	120626	MOV	#0,FIRST+APTDOWN		
10557	040672	104417			KERNEL		;ENTER KERNEL MODE	
10558	040674	000000			APTHLT: HALT			

```

10561 040676          SUBTST  <<BLOCK MOVE SUBROUTINE>>
;*****
;SUBTEST      BLOCK MOVE SUBROUTINE
;*****
;BLOCK3 HAS 3 ARGUEMENTS
;BLOCK2 HAS 2 ARGUEMENTS
;BLOCK1 HAS 1 ARGUEMENTS
;
;ALL ARE CALLED BY THE BMOV MACRO
.ENABL  LSB
BLOCK1: PUSH  R0,R1,R2
                                MOV R0, (SP)
                                MOV R1, (SP)
                                MOV R2, -(SP)
10562 040676 010046
10563 040700 010146
10564 040702 010246
10565 040704 012702 177640
10566 040710 012701 000020
10567 040714 000413
10568 040716
                                MOV #FASTCITY,R2
                                MOV #16.,R1
                                BR 3#
10569 040716
BLOCK2: PUSH  R0,R1,R2
                                MOV R0, (SP)
                                MOV R1, (SP)
                                MOV R2, (SP)
10570 040716 C10046
10571 040720 010146
10572 040722 010246
10573 040724 012701 000020
10574 040724 012701 000020
10575 040730 000404
                                MOV #16.,R1
                                BR 2#
10576 040732
BLOCK3: PUSH  R0,R1,R2
                                MOV R0, (SP)
                                MOV R1, -(SP)
                                MOV R2, -(SP)
10577 040732 010046
10578 040734 010146
10579 040736 010246
10580 040740 012501
                                MOV (R5)+,R1
                                MOV (R5)+,R2
10581 040742 012502
10582 040744 012500
                                MOV (R5)+,R0
10583 040746 012022
10584 040750 077102
                                MOV (R0)+,(R2)
                                SOB R1,1#
                                POP R2,R1,R0
10585 040752 012602
10586 040754 012601
                                MOV (SP)+,R2
                                MOV (SP)+,R1
                                MOV (SP)+,R0
10587 040756 012600
10588 040760 000205
                                RTS R5
                                .DSABL LSB
10589

```

```

10588                                     .SBTTL FIELD SERVICE MODE
10589
10590 040762    FIELDSERVICE:SUBTST    <<SUBR FIELD SERVICE COMMAND MODE>>
;*****
;SUBTEST      SUBR    FIELD SERVICE COMMAND MODE
;*****
10591 040762    104415    SAVREG
10592 040764    104401    066166    TYPE      MSG020      ;FIELD SERVICE COMMAND MODE
                                TYPEIT    ,MSG020
                                .DSABL    CRF
10593
10594 040770    IF RLFLAG IS TRUE OR NOFSMODE IS TRUE
                                TST RLFLAG
                                BNE L374
                                TST NOFSMODE
                                BEQ L375
                                L374:;;;;;
10595 041004    104401    067627    TYPE      MSG048      ;NOT AVAILABLE NOW - TRY LATER!
                                TYPEIT    ,MSG048
                                .DSABL    CRF
                                RESREG
                                RETURN
                                END ;OF IF RLFLAG
                                L375:;;;;;
10596 041010    104416
10597 041012    000207
10598 041014
10599 041014    005737    002544    TST      CACHKN
10600 041020    001402    BEQ      1#
10601 041022    013746    177746    PUSH     CONTRL      ;SAVE CACHE STATUS
                                MOV CONTRL,-(SP)
10602 041026    013746    002152    1#:     PUSH     CSRNO,KAMIKAZE ;SAVE CSR & KAMIKAZE STATUS
                                MOV CSRNO,-(SP)
                                MOV KAMIKAZE,(SP)
                                041026    013746    002006
10603 041036    104424    CACHOFF  ;TURN CACHE OFF
10604 041040    012737    177777    002006    SET      KAMIKAZE
                                MOV #1,KAMIKAZE
10605 041046    104401    067057    FS1:    TYPE      MSG026
                                TYPEIT    ,MSG026
                                .DSABL    CRF
                                ;COMMAND:
10606 041052    104414    RDDEC    ;READ A DECIMAL NUMBER
10607 041054    012600    POP      RO          ;COMMAND --> RO
                                MOV (SP)+,RO
10608 041056    020027    000022    CMP      RO,#18.
10609 041062    101403    BLOS    1#
10610 041064    104401    066207    TYPE      MSG021
                                TYPEIT    ,MSG021
                                .DSABL    CRF
10611 041070    000766    BR      FS1
10612 041072    010046    1#:     CASE RO
                                MOV RO,(SP)
                                ASL @SP
                                JSR PC,L376
                                041072    006316
                                041074    004737    041150
10613 041102    041160    FSCMD0    ;EXIT FIELD SERVICE COMMANDS
10614 041104    041262    FSCMD1    ;READ CSR
10615 041106    041372    FSCMD2    ;LOAD CSR
10616 041110    041540    FSCMD3    ;EXAMINE MEMORY
10617 041112    042014    FSCMD4    ;MODIFY MEMORY
10618 041114    042334    FSCMD5    ;SELECT BANK & PATTERN
10619 041116    043252    FSCMD6    ;TYPE CONFIGURATION MAP

```

10620	041120	043260	FSCMD7	;SOB A-LONG TEST
10621	041122	043552	FSCMD8	;ERROR SUMMARY
10622	041124	044000	FSCMD9	;REFRESH TEST
10623	041126	044272	FCMD10	;SET FILL COUNT
10624	041130	044320	FCMD11	;ENTER KAMIKAZE MODE
10625	041132	044342	FCMD12	;EXIT KAMIKAZE MODE
10626	041134	044362	FCMD13	;TURN CACHE OFF
10627	041136	044404	FCMD14	;TURN CACHE ON
10628	041140	044422	FCMD15	;TEST ONLY SELECTED BANKS
10629	041142	044506	FCMD16	;RESUME TESTING ALL BANKS
10630	041144	044550	FCMD17	;ENABLE TRACE
10631	041146	044564	FCMD18	;DISABLE TRACE
10632	041150		END ;OF CASE	
	041150			
	041150	062616		
	041152	013646		
	041154	004736		
10633	041156	000733	BR FS1	

L376:;;;;;
ADD (SP)+,0SP
MOV 0(SP)+,-(SP)
JSR PC,0(SP)+

```

10636 041160 FSCMD0: SUBTST <<COMMAND 0 EXIT>>
;*****
;*SUBTEST COMMAND 0 EXIT
;*****
10637 041160 TYPE MSG103 ;LEAVING FIELD SERVICE MODE
041160 104401 070605 TYPEIT ,MSG103
10638 041164 062706 000002 .DSABL CRF
10639 C11170 ADD #2,SP
041170 005737 002010 IF SKIPKAMI IS TRUE TST SKIPKAMI
041174 001405 BEQ L377
10640 041176 062706 000002 ADD #2,SP ;THROW AWAY OLD KAMIKAZE FLAG
10641 041202 005037 002010 CLR SKIPKAMI
10642 041206 000402 ELSE BR L400
041210 L377:;;;;;
10643 041210 POP KAMIKAZE ;RESTORE OLD KAMIKAZE FLAG
041210 012637 002006 MOV (SP)+,KAMIKAZE
10644 041214 END ;OF IF SKIPKAMI L400:;;;;;
10645 041214 POP CSRNO MOV (SP)+,CSRNO
041214 012637 002152 TST CACHKN
10646 041220 005737 002544 BEQ RESO ;IF CACHE IS OFF
10647 041224 001414 IF CACHKN EQ CACHKF CMP CACHKN,CACHKF
041226 023737 002544 002550 BNE L401
10649 041236 062706 000002 ADD #2,SP ;THROW AWAY CACHE STATUS
10650 041242 000405 ELSE BR L402
041244 L401:;;;;;
10651 041244 005737 002544 TST CACHKN
10652 041250 001402 BEQ RESO ;RESTORE CACHE STATUS
10653 041252 012637 177746 POP CONTRL MOV (SP)+,CONTRL
041252 012637 177746 END ;OF IF CACHKN L402:;;;;;
10654 041256 RESO: RESREG
041256 104416 RETURN
10655 041256 104416
10656 041260 000207
10657
10658 041262 FSCMD1: SUBTST <<FS COMMAND 1 READ CSR>>
;*****
;*SUBTEST FS COMMAND 1 READ CSR
;*****
10659 041262 004737 044576 CALL WHICHCSR
10660 041266 010637 002306 MOV SP,FSSTACK
10661 041272 012737 041336 000004 SET4 #RES1 ;TRAPS TO 4 GOTO RES1
041272 012737 041336 000004 MOV #RES1.4
.DSABL CRF
10662 041300 104426 READCSR
10663 041302 012737 177777 002430 SET NOERROR MOV #-1,NOERROR
041302 012737 177777 002430 ERROR +26 ;USE ERROR ROUTINE FOR PRINTOUT
10664 041310 104026 RES4 ;RESET TRAPS TO 4 TO DEFAULT
10665 041312 012737 034002 000004 MOV #TIMEOUT,4
041312 022737 000005 004064 CMP #5,PROTYP ;IS THIS AN 11/83/84 ?
041326 001002 BNE 101$ ;BRANCH IF NOT
  
```



```

041330 005037 177766          CLR    CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
041334                                101#: ;THAT A EXPECTED TRAP COULD HAVE SET

                                .DSABL CRF
10666 041334 000207          RETURN
10667 041336                                TYPE    MSG025      ;THIS CSR DOES NOT EXIST
                                041336 104401 067033      TYPEIT  ,MSG025
                                .DSABL CRF
10668 041342 013706 002306    MOV    FSSTACK,SP  ;RESET TRAPS TO 4 TO DEFAULT
10669 041346                                RES4
                                041346 012737 034002 000004  MOV    @TIMEOUT,4
                                041354 022737 000005 004064  CMP    @5,PROTYP  ;IS THIS AN 11/83/84 ?
                                041362 001002                                BNE    101#        ;BRANCH IF NOT
                                041364 005037 177766          CLR    CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                041370                                101#: ;THAT A EXPECTED TRAP COULD HAVE SET

                                .DSABL CRF
10670 041370 000207          RETURN
  
```

```

10673 041372          FSCMD2: SUBTST <<FS  COMMAND 2  LOAD CSR>>
;*****
;SUBTEST  FS  COMMAND 2  LOAD CSR
;*****
10674 041372 004737 044576          CALL  WHICHCSR
10675 041376 010637 002306          MOV   SP,FSSTACK
10676 041402 012737 041504 000004  SET4  #RES2          ;TRAPS TO 4 GOTO RES2
                                MOV   #RES2.4
                                .DSABL CRF
10677 041410 104426          READCSR
10678 041412 104401 067071          TYPE  MSG027
                                TYPEIT ,MSG027
                                .DSABL CRF
10679 041416 012737 177777 002430  SET   NOERROR
                                MOV   #-1,NOERROR
10680 041424 104026          ERROR  +26          ;USE ERROR ROUTINE FOR PRINTOUT
10681 041426 012737 034002 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
                                .DSABL CRF
10682 041450 104401 067017          TYPE  MSG023          ;FIRST CSR WORD
                                TYPEIT ,MSG023
                                .DSABL CRF
10683 041454 104413          RDOCT
10684 041456 012637 002150  POP   CSR          ;READ AN OCTAL NUMBER
                                ;PUT IN IN LOC "CSR"
                                MOV   (SP)+,CSR
10685 041462 104425          LOADCSR
10686 041464 104426          READCSR
10687 041466 104401 067106          TYPE  MSG028
                                TYPEIT ,MSG028
                                .DSABL CRF
10688 041472 012737 177777 002430  SET   NOERROR
                                MOV   #1,NOERROR
10689 041500 104026          ERROR  +26          ;USE FOR PRINTOUT - NOT AN ERROR
10690 041502 000207          RETURN
10691 041504 104401 067033  RES2: TYPE  MSG025          ;THIS CSR DOES NOT EXIST
                                TYPEIT ,MSG025
                                .DSABL CRF
10692 041510 013706 002306          MOV   FSSTACK,SP
10693 041514 012737 034002 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
                                .DSABL CRF
10694 041536 000207          RETURN
    
```

```

10697 041540          FSCMD3: SUBST  <<FS  COMMAND 3  EXAMINE MEMORY>>
;*****
;SUBTEST  FS  COMMAND 3  EXAMINE MEMORY
;*****
10698 041540          PUSH  BANK,NOPAR,PARTHERE,4
041540 013746 002102          MOV BANK,-(SP)
041544 013746 002076          MOV NOPAR,-(SP)
041550 013746 002304          MOV PARTHERE,-(SP)
041554 013746 000004          MOV 4,-(SP)
10699 041560 012737 000002 002076  MOV  #2,NOPAR ;INDICATE PARITY ACTION
10700 041566 104401 067122  TYPE  MSG029 ;EXAMINE MEMORY
;TYPEIT ,MSG029
;DSABL CRF
10701 041572 104401 067161 14:  TYPE  MSG031 ;PHYSICAL ADDRESS (0-17775776)??
;TYPEIT ,MSG031
;DSABL CRF
10702 041576 104413          RDOCT ;READ OCTAL NUMBER ONTO STACK & #HIOCT
10703 041600 013737 056216 002102  MOV  #HIOCT,BANK ;PUT MSB'S IN BANK
10704 041606 012600          POP  RO ;PUT LSB'S IN RO
;MOV (SP),.RO
10705 041610 000241          CLC
10706 041612 006100          ROL  RO
10707 041614 006137 002102  ROL  BANK
10708 041620 000241          CLC
10709 041622 000000          ROR  RO
10710 041624 023737 002102 002556  CMP  BANK, LASTBANK ;CHECK FOR BANK TOO HIGH
10711 041632 003357          BGT  14 ;BRANCH IF TRUE
10712 041634 062700 060000  ADD  #FIRST,RO
10713 041640 032700 000001  BIT  #BIT0,RO ;CHECK FOR ODD ADDRESS
10714 041644 001352          BNE  14 ;BRANCH IF ODD ADDRESS
10715 041646 020027 157776  CMP  RO,#LAST ;CHECK FOR ADDRESS OVER 16K
10716 041652 101347          BHI  14 ;BRANCH IF OVER 16K
10717 041654 012737 041726 002304  MOV  #31,PARTHERE ;INCASE OF ABORTS
10718 041662 012737 041734 000004  SET4 #41,4 ;TRAPS TO 4 GOTO 41
;MOV
;DSABL CRF
;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
10719 041670 010346          MAP  BANK ;MOV R3,-(SP)
041670 013703 002102  MOV  BANK,R3
041672 004737 035604  CALL MAPPER
;DSABL CRF
;MOV (SP),.R3
10720 041702 012603          TESTAREA ;ENTER TEST MODE
041704 053737 002552 177776  BIS  TESTMODE,PSW ;GO TO SYSTEM TEST MODE
;DSABL CRF
10721 041712 011001          MOV  (RO),R1
10722 041714 104417          KERNEL ;ENTER KERNEL MODE
10723 041716 010146          TYPOCS R1
041716 041720 104403          MOV  R1,-(SP) ;SAVE R1 FOR TYPEOUT
;TYPOS ;GO TYPE--OCTAL ASCII
;BYTE 6 ;TYPE 6 DIGITS
;BYTE 0 ;SUPPRESS LEADING ZEROS
;DSABL CRF
10724 041724 000410          BR  EXCMD3
10725
10726 041726 104401 067221 30:  TYPE  MSG032 ;PARITY ABORT
;TYPEIT ,MSG032
    
```

```

10727 041732 000405          .DSABL CRF
10728 10728          BR          EXCMD3
10729 041734 062706 000004 4: ADD #4,SP          ;FIX STACK
10730 041740 104401 067240 TYPEIT MSG033        ;TIMEOUT TRAP
                                .DSABL CRF
                                BR          EXCMD3
10731 041744 000400
10732 10732
10733 041746 104417 EXCMD3: KERNEL      ;ENTER KERNEL MODE
10734 041750 0126J7 000004 POP          4,PARTHERE,NOPAR,BANK
                                MOV (SP),.4
                                MOV (SP),.PARTHERE
                                MOV (SP),.NOPAR
                                MOV (SP),.BANK
10735 041770 RES4          ;RESET TRAPS TO 4 TO DEFAULT
041770 012737 034002 000004 MOV #TIMEOUT,4
041776 022737 000005 004064 CMP #5,PROTYP    ;IS THIS AN 11/83/84 ?
042004 001002 BNE 101#        ;BRANCH IF NOT
042006 C05037 177766 CLR CPUERR      ;CLEAR OUT THE CPU ERROR REGISTER BITS
042012 101#:          ;THAT A EXPECTED TRAP COULD HAVE SET

10736 042012 000207          .DSABL CRF
                                RETURN
  
```

```

10739 042014 FSCMD4: SUBTST <<FS COMMAND 4 MODIFY MEMORY>>
;.....
;SUBTEST FS COMMAND 4 MODIFY MEMORY
;.....
10740 042014 PUSH BANK,NOPAR,PARTHERE,4
042014 013746 002102 MOV BANK, -(SP)
042020 013746 002076 MOV NOPAR, (SP)
042024 013746 002304 MOV PARTHERE, (SP)
042030 013746 000004 MOV 4, (SP)
10741 042034 012737 000003 002076 MOV #3,NOPAR ;INDICATE PARITY ACTION
10742 042042 104401 067346 TYPE MSG036 ;MODIFY MEMORY
.DSABL CRF
10743 042046 104401 067161 1$: TYPE MSG031 ;PHYSICAL ADDRESS (0 17775776)??
TYPEIT ,MSG031
.DSABL CRF
10744 042052 104413 RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
10745 042054 013737 056216 002102 MOV $HIOCT,BANK ;PUT MSB'S IN BANK
10746 042062 012600 POP RO ;PUT LSB'S IN RO
MOV (SP),.RO
10747 042064 000241 CLC
10748 042066 006100 ROL RO
10749 042070 006137 002102 ROL BANK
10750 042074 000241 CLC
10751 042076 006000 ROR RO
10752 042100 IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
042100 023737 002102 002556 CMP BANK, LASTBANK
042106 003357 BGT 1$
10753 042110 ADD #FIRST,RO
10754 042114 IF #BIT0 SET.IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
042114 032700 000001 BIT #BIT0,RO
042120 001352 BNE 1$
10755 042122 IF RO HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER 16K
042122 020027 157776 CMP RO,#LAST
042126 101347 BHI 1$
10756 042130 012737 042176 002304 MOV #3$,PARTHERE ;INCASE OF ABORTS
10757 042136 012737 042204 000004 SET4 #4$ ;TRAPS TO 4 GOTO 4$
MOV #4$,4
.DSABL CRF
MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
MOV R3, (SP)
1075J 042144 010346
042144 013703 002102 MOV BANK,R3
042152 004737 035604 CALL MAPPER
.DSABL CRF
MOV (SP),.R3
10759 042156 012603 INVALIDATE
10760 042160 104511 TESTAREA ;ENTER TEST MODE
10760 042162 053737 002552 177776 BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
10761 042170 011001 MOV (RO),R1
10762 ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
10763 042172 104417 KERNEL ;ENTER KERNEL MODE
10764 042174 000410 BR 5$
10765
10766 042176 104401 067221 3$: TYPE MSG032 ;PARITY ABORT
TYPEIT ,MSG032
.DSABL CRF

```

```

10767 042202 000431          BR      EXCMD4          ;EXIT
10768
10769 042204 062706 000004    4$:   ADD      #4,SP          ;FIX STACK
10770 042210          TYPEIT   MSG033        ;TIMEOUT TRAP
      042210 104401 067240    .DSABL  CRF
      .DSABL  CRF
10771 042214 000424          BR      EXCMD4          ;EXIT
10772
10773 042216          TYPE      MSG037        ;OLD DATA WAS
      042216 104401 067365    TYPEIT   ,MSG037
      .DSABL  CRF
      TYPOCS R1          ;PRINT IT
10774 042222          MOV      R1, -(SP)      ;;SAVE R1 FOR TYPEOUT
      042222 010146    TYPOS    6          ;;GO TYPE--OCTAL ASCII
      042224 104403    .BYTE    0          ;;TYPE 6 DIGITS
      042226 006      .DSABL  CRF          ;;SUPPRESS LEADING ZEROS
      042227 000      TYPE      MSG039        ;INPUT NEW DATA
10775 042230          TYPEIT   ,MSG039
      042230 104401 067422    .DSABL  CRF
10776 042234 104413          RDOCT          ;READ ON OCTAL NUMBER ONTO THE STACK
10777 042236          POP      R1          ;GET NEW NUMBER
      042236 012601    TESTAREA ;ENTER TEST MODE          MOV (SP),R1
10778 042240          BIS      TESTMODE,PSW ;GO TO SYSTEM TEST MODE
      042240 053737 002552 177776 .DSABL  CRF
10779 042246          MOV      R1,(R0)        ;PUT IT IN MEMORY
10780 042250          MOV      (R0),R1      ;READ IT AGAIN
10781 042252 104417    KERNEL    ;ENTER KERNEL MODE
10782 042254          TYPE      MSG038        ;DATA IS NOW
      042254 104401 067404    TYPEIT   ,MSG038
      .DSABL  CRF
10783 042260          TYPOCS R1          ;PRINT IT
      042260 010146    MOV      R1, -(SP)      ;;SAVE R1 FOR TYPEOUT
      042262 104403    TYPOS    6          ;;GO TYPE--OCTAL ASCII
      042264 006      .DSABL  CRF          ;;TYPE 6 DIGITS
      042265 000      .DSABL  CRF          ;;SUPPRESS LEADING ZEROS
10784
10785 042266 104417    EXCMD4: KERNEL ;ENTER KERNEL MODE
10786 042270          POP      4,PARTHERE,NOPAR,BANK
      042270 012637 000004    MOV (SP),.4
      042274 012637 002304    MOV (SP),.PARTHERE
      042300 012637 002076    MOV (SP),.NOPAR
      042304 012637 002102    MOV (SP),.BANK
10787 042310          RES4          ;RESET TRAPS TO 4 TO DEFAULT
      042310 012737 034002 000004 MOV      #TIMEOUT,4
      042316 022737 000005 004064 CMP      #5,PROTYP
      042324 001002          BNE     101$
      042326 005037 177766    CLR     CPUERR
      042332          ;CLEAR OUT THE CPU ERROR REGISTER BITS
      ;THAT A EXPECTED TRAP COULD HAVE SET
10788 042332 000207          .DSABL  CRF
      RETURN
    
```

```

10791 042334          FSCMD5: SUBTST <<FS  COMMAND 5  SELECT BANK & PATTERN>>
;*****
;SUBTEST           FS  COMMAND 5  SELECT BANK & PATTERN
;*****
10792 042334          PUSH      BANK,PATTERN,TESTADD,PCBUMP,TKVEC,TKVEC*2
      042334 013746 002102          MOV BANK,-(SP)
      042340 013746 002112          MOV PATTERN,-(SP)
      042344 013746 002412          MOV TESTADD,-(SP)
      042350 013746 002326          MOV PCBUMP,(SP)
      042354 013746 000060          MOV TKVEC,-(SP)
      042360 013746 000062          MOV TKVEC*2,-(SP)
10793 042364 010637 002306          MOV      SP,FSSTACK          ;SAVE LAST GOOD STACK POINTER
10794 042370          TYPE      MSG040          ;SELECT BANK & PATTERN TEST
      042370 104401 067444          TYPEIT   ,MSG040
      .DSABL CRF
10795 042374          1$:      TYPE      MSG030          ;BANK(0-177)?
      042374 104401 067142          TYPEIT   ,MSG030
      .DSABL CRF
10796 042400 104413          RDOCT           ;READ AN OCTAL NUMBER ONTO THE STACK
10797 042402          POP      BANK          ;PUT IT IN BANK
      042402 012637 002102          MOV (SP)+,BANK
10798 042406          IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
      042406 023737 002102 002556          CMP BANK, LASTBANK
      042414 003367          BGT 1$
10799
10800 042416 013701 002102          MOV      BANK,R1
10801 042422 006301          ASL      R1
10802 042424 006301          ASL      R1
10803 042426          IF CPUBIT OFF.IN CONFIG(R1)
      042426 033761 002106 002664          BIT CPUBIT,CONFIG(R1)
      042434 001003          BNE L403
10804 042436          TYPE      MSG041          ;BANK NOT ACCESSABLE
      042436 104401 067470          TYPEIT   ,MSG041
      .DSABL CRF
      GOTO 1$
10805 042442          END :OF IF
      042442 000754          BR 1$
10806 042444          L403:::
      042444
10807
10808 042444          2$:      TYPE      MSG042          ;PATTERN(0-45)?
      042444 104401 067515          TYPEIT   ,MSG042
      .DSABL CRF
10809 042450 104413          RDOCT           ;READ AN OCTAL NUMBER ONTO THE STACK
10010 042452          POP      PATTERN          ;PUT IT IN PATTERN
      042452 012637 002112          MOV (SP)+,PATTERN
10811 042456          IF PATTERN GT #47 THEN GOTO 2$ ;CHECK FOR PATTERN TO HIGH
      042456 023727 002112 000047          CMP PATTERN,#47
      042464 003367          BGT 2$
10812 042466          IF PATTERN EQ #0
      042466 005737 002112          TST PATTERN
      042472 001004          BNE L404
10813 042474          TYPE      MSG043          ;PATTERN 0 DATA IS?
      042474 104401 067533          TYPEIT   ,MSG043
      .DSABL CRF
10814 042500 104413          RDOCT           ;READ AN OCTAL NUMBER ONTO THE STACK
10815 042502          POP      R2          ;PUT IT IN R2
      042502 012602          MOV (SP)+,R2
  
```

```

10816 042504          END ;OF IF
10817 042504          L404:;;;;;
10818
10819 042504          MAP      BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      042504 010346          MOV      BANK,R3          MOV R3,-(SP)
      042506 013703 002102  CALL     MAPPER
      042512 004737 035604  .DSABL  CRF
      042516 012603          MOV (SP),R3
10820 042520 104511  INVALIDATE
10821 042522 004737 037760  CALL     EXBANK          ;SET NEW MARGINS
10822 042526          IF RRFLAG IS TRUE
      042526 005737 002124          TST RRFLAG
      042532 001404          BEQ L405
10823 042534          TYPE MSG049          ;BANK REQUIRES RELOCATION
      042534 104401 067667  TYPEIT ,MSG049
      .DSABL CRF
      JMP CMD5C
10824 042540 000137 043154  END ;OF IF RRFLAG
10825 042544
      042544          L405:;;;;;
10826 042544          TYPE MSG046          ;TO ESCAPE TYPE ANY KEY!
      042544 104401 067555  TYPEIT ,MSG046
      .DSABL CRF
10827 042550 013737 002152 002154  MOV      CSRNO,SAVCSR  ;SAVE OLD CSR NUMBER
10828 042556 013702 002102          MOV      BANK,R2
10829 042562 072227 000002          ASH     #2,R2          ;GENERATE INDEX INTO CONFIGURATION TABLE
10830 042566 016203 002664          MOV     CONFIG(R2),R3 ;R3 = LOW WORD OF CONFIGURATION TABLE FOR THIS BANK
10831 042572 072327 177770          ASH     #-10,R3       ;POSITION CSR CODE IN BITS 0-3
10832 042576 042703 177760          BIC     #+C17,R3      ;CLEAR ALL BUT THE CSR CODE
10833 042602 006303          ASL     R3            ;ADJUST CSR NUMBER
10834 042604 010337 002152          MOV     R3,CSRNO
10835 042610 012737 043154 000060  MOV     #CMD5C,TKVEC
10836 042616 012737 000340 000062  MOV     #340,TKVEC 2
10837 042624 017700 140014          MOV     #TKB,RO
10838 042630 042737 000200 177776  BIC     #BIT7,PSW     ;KILL ANY OLD INTERRUPT
10839 042636 052777 000100 137776  BIS     #BIT6,#TKS    ;LOWER CPU PRIORITY TO 140
10840                                     ;ENABLE KEYBOARD INTERRUPTS
10841
10842 042644          SET     HEADER,MUT
      042644 012737 177777 002612          MOV     #-1,HEADER
      042652 012737 177777 002110          MOV     #-1,MUT
10843 042660 013701 002102          CMD5B: MOV     BANK,R1
10844 042664 006301          ASL     R1
10845 042666 006301          ASL     R1
10846 042670 005037 002240          CLR     SPLTCSR
10847 042674 005037 002264          CLR     PASFLG
10848 042700 012737 060000 002412  MOV     #FIRST,TESTADD
10849 042706 012737 060002 002414  MOV     #FIRST+2,TESTADD+2
10850 042714          IF #BIT12 SET.IN CONFIG+2(R1)
      042714 032761 010000 002666          BIT     #BIT12,CONFIG+2(R1)
      042722 001413          BEQ L406
10851 042724 005237 002240          INC     SPLTCSR
10852 042730          MAP     BANK
      042730 010346          MOV R3,(SP)
      042732 013703 002102  MOV     BANK,R3
      042736 004737 035604  CALL     MAPPER
  
```


10853	042742	012603			.DSABL	CRF		MOV (SP)+,R3
10854	042744	012737	120000	002414	MOV	#120000,TESTADD+2		
	042752				END; OF IF	#BIT12		L406:::~:~:~
10855	042752				IF	#SWO SET.IN #SWR		
	042752	032777	000001	137656				BIT #SWO,#SWR
10856	042760	001402						BEQ L407
10857	042762	104470			ECCDIS		:DISABLE ERROR CORRECTION	
	042764				ELSE			BR L410
	042764	000405						L407:::~:~:~
10858	042766				PUSH	CSRNO		
10859	042766	013746	002152					MOV CSRNO,-(SP)
10860	042772	104502			CLRCR		:CLEAR CSRS	
	042774				POP	CSRNO		
10861	042774	012637	002152					MOV (SP)+,CSRNO
	043000				END ;OF IF			L410:::~:~:~
10862	043000	C12737	000002	002076	MOV	#2,NOPAR	:PARITY ACTION	
10863	043006	012737	000002	002326	MOV	#2,PCBUMP	:TRAPS ADD 2 TO PC	
10864	043014	013700	002112		MOV	PATTERN,RO		
10865	043020	006300			ASL	RO		
10866	043022	004770	043034		CALL	#FSPAT(RO)		
10867	043026	005037	002076		CLR	NOPAR		
10868	043032	000712			BR	CMD5B	:LOOP TILL KEYBOARD INTERRUPT	
10869								
10870	043034	017376			FSPAT: MT0000	:<1 SEC	DATA PATTERN TEST	
10871	043036	017432			MT0001	:<1 SEC	ADDRESS TEST	
10872	043040	017526			MT0002	:<1 SEC	COMPLEMENT ADDRESS TEST	
10873	043042	017642			MT0003	: 1 SEC	3 XOR 9 WORST CASE NOISE TEST	
10874	043044	017756			MT0004	: 1 SEC	ROTATING ZEROS TEST	
10875	043046	020026			MT0005	: 1 SEC	ROTATING ONES TEST	
10876	043050	020110			MT0006	:<1 SEC	INITIAL DATA TEST	
10877	043052	020144			MT0007	:<1 SEC	ADDRESS BIT TEST	
10878	043054	020206			MT0010	:<1 SEC	BYTE ADDRESSING TEST	
10879	043056	024106			MT0999	:<1 SEC	NULL TEST TO KEEP TABLE ORDER	
10880	043060	024106			MT0999	:<1 SEC	NULL TEST TO KEEP TABLE ORDER	
10881	043062	024106			MT0999	:<1 SEC	NULL TEST TO KEEP TABLE ORDER	
10882	043064	020242			MT0014	: 1 SEC	BASIC DOUBLE BIT ERROR TEST	
10883	043066	024106			MT0999	:<1 SEC	NULL TEST TO KEEP TABLE ORDER	
10884	043070	024106			MT0999	:<1 SEC	NULL TEST TO KEEP TABLE ORDER	
10885	043072	020322			MT0017	:<1 SEC	HOLDING 1'S & 0'S TEST	
10886	043074	020344			MT0020	: 1 SEC	SYNDROMES TO CSR ON SBE TEST	
10887	043076	020424			MT0021	: 1 SEC	MARCHING 0'S & 1'S TEST	
10888	043100	020714			MT0022	:10 SEC	REFRESH & SHIFTING DIAGONAL TEST	
10889	043102	020746			MT0023	:10 SEC	SHIFTING DIAGONAL TEST	
10890	043104	021012			MT0024	:20 SEC	FAST GALLOPING PATTERN TEST	
10891	043106	024106			MT0999	:<1 SEC	NULL TEST TO KEEP TABLE ORDER	
10892	043110	021220			MT0026	:<1 SEC	RANDOM DATA TEST	
10893	043112	021476			MT0027	: 1 SEC	UNIQUE BANK TEST	
10894	043114	022066			MT0030	: 1 SEC	FLUSH OUT DBE'S TEST	
10895	043116	022342			MT0031	: 3 SEC	SOB-A-LONG TEST	
10896	043120	022532			MT0032	:<1 SEC	WRITE RECOVERY TEST	
10897	043122	023046			MT0033	:35 SEC	BRANCH GOBBLE TEST	
10898	043124	023234			MT0034	: 1 SEC	SOFT ERROR TEST	
10899	043126	023334			MT0035	:<1 SEC	WORST CASE NOISE PARITY TEST	

10900	043130	023446			MT0036	; 1 SEC	CORRECTION CODE TEST
10901	043132	023510			MT0037	; <1 SEC	ECC DISABLE TEST
10902	043134	024106			MT0999	; <1 SEC	NULL TEST TO KEEP TABLE ORDER
10903	043136	023546			MT0041	; 1 SEC	ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
10904	043140	023626			MT0042	; <1 SEC	EXTENDED Q-BUS ADDRESS TEST
10905	043142	023662			MT0043	; 1 SEC	WRITE BYTE CLEARS SBE TEST
10906	043144	023712			MT0044	; 5 SEC	SHIFTING 1/0'S THROUGH THE CHECK BITS
10907	043146	023772			MT0045	; 1 SEC	SYNDROMES TO CSR ON DBE TEST
10908	043150	024022			MT0046	; 1 SEC	CHECK SINGLE BIT ERROR WITH ECC DISABLED TEST
10909	043152	024052			MT0047	; <1 SEC	NO CSR UPDATE WITH SBE ON DBE TEST
10910							
10911	043154	013706	002306		CMD5C: MOV	FSSTACK, SP	; RECOVER OLD STACK POINTER
10912	043160	042777	000100	137454	BIC	#BIT6, #TKS	
10913	043166				POP	TKVEC+2, TKVEC	
		043166	012637	000062			MOV (SP)+, TKVEC+2
		043172	012637	000060			MOV (SP)+, TKVEC
10914	043176	117700	137442		MOVB	#TKB, R0	; GET CHARACTER TO GET RID OF FLAG
10915	043202				POP	PCBUMP, TESTADD	
		043202	012637	002326			MOV (SP)+, PCBUMP
		043206	012637	002412			MOV (SP)+, TESTADD
10916	043212				POP	PATTERN, BANK	
		043212	012637	002112			MOV (SP)+, PATTERN
		043216	012637	002102			MOV (SP)+, BANK
10917	043222				MAP	BANK	; REMAP OLD BANK
		043222	010346				MOV R3, (SP)
		043224	013703	002102	MOV	BANK, R3	
		043230	004737	035604	CALL	MAPPER	
					.DSABL	CRF	
		043234	012603				MOV (SP)+, R3
10918	043236	004737	037760		CALL	EXBANK	
10919	043242	013737	002154	002152	MOV	SAVCSR, CSRNO	; RESTORE CSRNO.
10920	043250	000207			RETURN		
10922	043252				FSCMD6: SUBTST	<<FS COMMAND 6 TYPE CONFIGURATION MAP>>	
					;*****		
					;*SUBTEST	FS COMMAND 6 TYPE CONFIGURATION MAP	
					;*****		
10923	043252	004737	032610		CALL	PCONFIG	
10924	043256	000207			RETURN		
10925							

```

10928 043260          FSCMD7: SUBTST <<FS  COMMAND 7  SOB-A LONG TEST>>
;*****
;SUBTEST  FS  COMMAND 7  SOB-A-LONG TEST
;*****
10929 043260          PUSH  BANK,PATTERN,TKVEC,TKVEC*2,NOPAR
      043260 013746 002102          MOV BANK,-(SP)
      043264 013746 002112          MOV PATTERN,-(SP)
      043270 013746 000060          MOV TKVEC,-(SP)
      043274 013746 000062          MOV TKVEC*2,-(SP)
      043300 013746 002076          MOV NOPAR,-(SP)
10930 043304 010637 002306          MOV SP,FSSTACK          ;SAVE LAST GOOD STACK POINTER
10931 043310 104401 067741          TYPE MSG055          ;SOB-A-LONG TEST
      043310 104401 067741          TYPEIT ,MSG055
      .DSABL CRF

10932
10933 043314          IF #SWO SET.IN @SWR
      043314 032777 000001 137314          BIT #SWO,@SWR
      043322 001402          BEQ L411
10934 043324 104470          ECCDIS          ;DISABLE ERROR CORRECTION
10935 043326          ELSE
      043326 000401          BR L412
      043330          L411:;;;;;
10936 043330 104502          CLRCSR          ;CLEAR CSRS
10937 043332          END ;OF IF
      043332          L412:;;;;;
10938 043332          TYPE MSG056          ;BELL = EACH PASS COMPLETE
      043332 104401 067762          TYPEIT ,MSG056
      .DSABL CRF

10939
10940 043336          TYPE MSG046          ;TO ESCAPE TYPE ANY KEY!
      043336 104401 067555          TYPEIT ,MSG046
      .DSABL CRF
10941 043342 012737 043466 000060          MOV #CMD7C,TKVEC
10942 043350 012737 000340 000062          MOV #340,TKVEC*2
10943 043356 017700 137262          MOV #TKB,RO
10944 043362 042737 000200 177776          BIC #BIT7,PSW
10945 043370 052777 000100 137244          BIS #BIT6,#TKS
10946
10947
10948 043376          SET  HEADER,MUT
      043376 012737 177777 002612          MOV #-1,HEADER
      043404 012737 177777 002110          MOV #-1,MUT

10949
10950 043412          CMD7B: FOR BANK := #0 TO LASTBANK
      043412 005037 002102          CLR BANK
      043416          B70:;;;;;
10951 043416 004737 037760          CALL EXBANK
10952 043422          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
      043422 005737 002116          TST ACFLAG
      043426 001406          BEQ L413
      043430 005737 002124          TST RRFLAG
      043434 001003          BNE L413
10953 043436 104511          INVALIDATE
10954 043440 004737 022342          CALL MT0031
10955 043444          END ;OF IF ACFLAG
      043444          L413:;;;;;
10956 043444          END ;OF FOR BANK
  
```

```

043444 005237 002102
043450 023737 002102 G02556
043456 003757
043460
10957 043460 TYPE #BELL ;RING BELL E70:;;;;;
043460 TYPEIT ,#BELL
.DSABL CRF
GOTO CMD7B
10958 043464 BR CMD7B
043464 000752
10959
10960 043466 013706 002306 CMD7C: MOV FSSTACK,SP ;RECOVER OLD STACK POINTER
10961 043472 042777 000100 137142 BIC #BIT6,#TKS
10962 043500 117700 137140 MOVB #TKB,R0 ;READ CHAR TO KILL FLAG
10963 043504 POP NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
043504 012637 002076 MOV (SP),NOPAR
043510 012637 000062 MOV (SP),TKVEC+2
043514 012637 000060 MOV (SP),TKVEC
043520 012637 002112 MOV (SP),PATTERN
043524 012637 002102 MOV (SP),BANK
10964 043530 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
043530 010346 MOV R3,-(SP)
043532 013703 002102 MOV BANK,R3
043536 004737 035604 CALL MAPPER
.DSABL CRF
10965 043542 012603 MOV (SP),R3
10965 043544 004737 037760 CALL EXBANK
10966 043550 000207 RETURN
  
```

```

10969 043552          FSCMD8: SUBTST <<FS  COMMAND 8  ERROR SUMMARY>>
;*****
;SUBTEST  FS  COMMAND 8  ERROR SUMMARY
;*****
10970 043552          PUSH  R0,R2,R3,BANK
043552 010046          MOV  R0,-(SP)
043554 010246          MOV  R2,-(SP)
043556 010346          MOV  R3,-(SP)
043560 013746 002102 002434          MOV  BANK,-(SP)
10971 043564 013737 056724 002434          MOV  $PASS,TEMP
10972 043572 005337 002434          DEC  TEMP
10973 043576 013746 002434          TYPDEC TEMP
043602 104405          MOV  TEMP,-(SP) ;;SAVE TEMP FOR TYPEOUT
;;GO TYPE--DECIMAL ASCII WITH SIGN
10974 043604 013737 172350 002270          .DSABL CRF
10975 043612 012737 001000 172350          MOV  KIPAR4,SAV4 ;;I.L.C.;REV B
10976 043620 104401 071312          MOV  #1000,KIPAR4 ;;I.L.C.;REV B
043620 104401 071312          TYPE MSG125 ;;PASSES COMPLETED
TYPEIT ,MSG125
.DSABL CRF
10977 043624 013746 002630          TYPDEC $ERTTL
043624 013746 002630          MOV  $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
043630 104405          TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
.DSABL CRF
10978 043632 104401 070335          TYPE MSG079 ;ERROR(S) DETECTED
TYPEIT ,MSG079
.DSABL CRF
10979 043636 005737 002630          IF $ERTTL NE #0
043636 005737 002630          TST $ERTTL
043642 001445          BEQ L414
10980 043644 005037 002334          CLR  SUCCESS
10981 043650 005037 002102          FOR BANK := #0 TO LASTBANK
043654          CLR  BANK
043654          B71:;;;;;
10982 043654 013703 002102          MOV  BANK,R3
10983 043660 070327 000004          MUL  #4,R3
10984 043664 105763 002666          IFB CONFIG+2(R3) NE #0
043664 105763 002666          TSTB CONFIG+2(R3)
043670 001424          BEQ L415
10985 043672 005737 002334          IF SUCCESS IS FALSE
043672 005737 002334          TST SUCCESS
043676 001005          BNE L416
10986 043700 104401 070300          TYPEIT TYPE MSG076 ;BANK ERRORS
.DSABL ,MSG076
CRF
10987 043704 012737 177777 002334          SET SUCCESS
043704 012737 177777 002334          MOV  #-1,SUCCESS
10988 043712          END ;OF IF SUCCESS
043712          L416:;;;;;
10989 043712 013746 002102          MOV  TYPOCS BANK,3
043712 013746 002102          BANK,-(SP) ;;SAVE BANK FOR TYPEOUT
;;3
043716 104403          TYPOS ;;GO TYPE--OCTAL ASCII
043720 006          .BYTE 6 ;;TYPE 6 DIGITS
043721 000          .BYTE 0 ;;SUPPRESS LEADING ZEROS
.DSABL CRF
10990 043722 116300 002666          MOV  CONFIG+2(R3),R0
  
```



```

11002 044000      FSCMD9: SUBTST <<FS  COMMAND 9  REFRESH TEST>>
;*****
;SUBTEST      FS  COMMAND 9  REFRESH TEST
;*****
11003 044000      PUSH  BANK,PATTERN,TKVEC,TKVEC*2,NOPAR
044000 013746 002102      MOV BANK,-(SP)
044004 013746 002112      MOV PATTERN,-(SP)
044010 013746 000060      MOV TKVEC,-(SP)
044014 013746 000062      MOV TKVEC*2,-(SP)
044020 013746 002076      MOV NOPAP,-(SP)
11004 044024 010637 002306      MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
11005 044030      TYPE MSG073 ;REFRESH TEST
044030 104401 070230      TYPEIT ,MSG073
;DSABL CRF

11006
11007 044034      IF #SWO SET.IN #SWR
044034 032777 000001 136574      BIT #SWO,#SWR
044042 001402      BEQ L417

11008 044044 104470      ECCDIS ;DISABLE ERROR CORRECTION
11009 044046      ELSE
044046 000401      BR L420
044050      L417:;;;;;

11010 044050 104502      CLRCRS ;CLEAR CSRS
11011 044052      END ;OF IF      L420:;;;;;

11012 044052      TYPE MSG056 ;BELL = EACH PASS COMPLETE
044052 104401 067762      TYPEIT ,MSG056
;DSABL CRF

11013
11014 044056      TYPE MSG046 ;TO ESCAPE TYPE ANY KEY!
044056 104401 067555      TYPEIT ,MSG046
;DSABL CRF

11015 044062 012737 044206 000060      MOV #CMD9C,TKVEC
11016 044070 012737 000340 000062      MOV #340,TKVEC*2
11017 044076 017700 136542      MOV #TKB,RO ;KILL ANY OLD INTERRUPT
11018 044102 042737 000200 177776      BIC #BIT7,PSW ;LOWER CPU PRIORITY TO 140
11019 044110 052777 000100 136524      BIS #BIT6,#TKS ;ENABLE KEYBOARD INTERRUPTS
11020
11021 044116      SET HEADER,MUT
044116 012737 177777 002612      MOV #-1,HEADER
044124 012737 177777 002110      MOV #-1,MUT

11022
11023 044132      CMD9B: FOR BANK := #0 TO LASTBANK
044132 005037 002102      CLR BANK
044136      B72:;;;;;

11024 044136 004737 037760      CALL EXBANK
11025 044142      IF ACFLAG IS TRUE AND RRFLAG IS FALSE
044142 005737 002116      TST ACFLAG
044146 001406      BEQ L421
044150 005737 002124      TST RRFLAG
044154 001003      BNE L421

11026 044156 104511      INVALIDATE
11027 044160 004737 020714      CALL MTO022
11028 044164      END ;OF IF ACFLAG
044164      L421:;;;;;

11029 044164      END ;OF FOR BANK
044164 005237 002102      INC BANK
    
```

```

044170 023737 002102 002556             CMP BANK.LASTBANK
044176 003757                             BLE B72
11030 044200                                E72:|||||||
044200 104401 002653             TYPE      #BELL                ;RING BELL
                                TYPEIT    ,#BELL
                                .DSABL    CRF
                                GOTO      CMD9B
11031 044204                                BR CMD9B
044204 000752
11032 044206 013706 002306             CMD9C: MOV      FSSTACK,SP                ;RECOVER OLD STACK POINTER
11033 044212 042777 000100             BIC      #BIT6,#TKS
11035 044220 117700 136420             MOV      #TKB,RO                ;READ CHAR TO KILL FLAG
11036 044224 012637 002076             POP      NOPAR,TKVEC+2,TKVEC,PATTERN,BANK
                                MOV (SP)+,NOPAR
                                MOV (SP)+,TKVEC+2
                                MOV (SP)+,TKVEC
                                MOV (SP)+,PATTERN
                                MOV (SP)+,BANK
                                MOV R3,-(SP)
044224 012637 000062
044230 012637 000060
044234 012637 002112
044240 012637 002102
044244 012637 002102
11037 044250             MAP      BANK                ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
044250 C10346
044252 013703 002102             MOV      BANK,R3
044256 004737 035604             CALL    MAPPER
                                .DSABL    CRF
                                MOV (SP)+,R3
044262 012603
11038 044264 004737 037760             CALL    EXBANK
11039 044270 000207
11040
  
```



```

11043 044272          FCMD10: SUBTST <<FS  COMMAND 10  SET FILL COUNT>>
;*****
;*SUBTEST          FS  COMMAND 10  SET FILL COUNT
;*****
11044 044272          PUSH      RO
044272 010046
11045 044274          TYPE      MSG085          ;FILL COUNT(OCTAL)?
044274 104401 070361  TYPEIT   ,MSG085
          .DSABL  CRF
11046 044300          RDOCT
104413
11047 044302          POP      RO
044302 012600          MOV      RO,-(SP)
11048 044304          BIC      #C17,RO
042700 177760
11049 044310          MOV      RO,#FILLS
11050 044314          POP      RO
044314 012600          MOV      (SP),RO
044314 000207
11051 044316          RETURN
11052
11053 044320          FCMD11: SUBTST <<FS  COMMAND 11  ENTER KAMIKAZE MODE>>
;*****
;*SUBTEST          FS  COMMAND 11  ENTER KAMIKAZE MODE
;*****
11054 044320          TYPE      MSG101          ;ENTERING KAMIKAZE MODE
044320 104401 070526  TYPEIT   ,MSG101
          .DSABL  CRF
11055 044324          SET      KAMIKAZE,SKIPKAMI
044324 012737 177777 002006          MOV      #-1,KAMIKAZE
044332 012737 177777 002010          MOV      #-1,SKIPKAMI
11056 044340          RETURN
000207
11057
11058 044342          FCMD12: SUBTST <<FS  COMMAND 12  EXIT KAMIKAZE MODE>>
;*****
;*SUBTEST          FS  COMMAND 12  EXIT KAMIKAZE MODE
;*****
11059 044342          TYPE      MSG102          ;LEAVING KAMIKAZE MODE
044342 104401 070556  TYPEIT   ,MSG102
          .DSABL  CRF
11060 044346          CLR      KAMIKAZE
005037 002006
11061 044352          SET      SKIPKAMI
044352 012737 177777 002010          MOV      # 1,SKIPKAMI
11062 044360          RETURN
000207
11063
    
```

```

11064 044362          FCMD13: SUBTST <<FS  COMMAND 13  TURN CACHE OFF>>
;*****
;*SUBTEST           FS  COMMAND 13  TURN CACHE OFF
;*****
11065 044362          TYPE      MSG106
044362 104401 070704  TYPEIT   .MSG106
                                .DSABL   CRF
                                CACHOFF
11066 044366 104424                                ;TURN CACHE OFF
11067 044370 013737 002544 002546  MOV      CACHKN,CACHKN+2
11068 044376 005037 002544  CLR      CACHKN
11069 044402 000207                                ;SAVE OLD CACHE ON STATE
11070                                ;KEEP CACHE OFF
11071 044404          FCMD14: SUBTST <<FS  COMMAND 14  TURN CACHE ON>>
;*****
;*SUBTEST           FS  COMMAND 14  TURN CACHE ON
;*****
11072 044404          TYPE      MSG107
044404 104401 070722  TYPEIT   .MSG107
                                .DSABL   CRF
                                MOV      CACHKN+2,CACHKN
11073 044410 013737 002546 002544  CACHON
11074 044416 104423                                ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
11075 044420 000207                                ;RESTORE OLD CACHE ON STATE
11076                                ;TURN CACHE ON

```

```

11089
11090 044422 FCMD15: SUBTST <<FS COMMAND 15 TEST ONLY SELECTED BANKS>>
;*****
;SUBTEST FS COMMAND 15 TEST ONLY SELECTED BANKS
;*****
11091 044422 104401 070631 TYPE MSG105 ;ENTER BANKS IN OCTAL USE NUMBER OUTSIDE RANGE TO TERMINAT
044422 .DSABL CRF ;ERASE OLD SELECTIONS
11092 044426 004737 044516 CALL CMD16A ;ERASE OLD SELECTIONS
11093 044432 BEGIN CMD16LOOP
11094 044432 REPEAT B73:;;;;;
044432 TYPE MSG030 ;BANK(0 177)? B74:;;;;;
11095 044432 104401 067142 TYPEIT ,MSG030 ;READ AN OCTAL NUMBER ONTO THE STACK
044432 .DSABL CRF ;PUT IT IN R1 MOV (SP),R1
11096 044436 104413 RDOCT
11097 044440 012601 POP R1
11098 044442 IF R1 GT #177 OR R1 LT #0 CMP R1,#177
044442 020127 000177 BGT L422
044446 003002 TST R1
044450 005701 BGE L423
044452 002001 L422:;;;;;
044454 LEAVE CMD16LOOP BR E73
11099 044454 000406 END ;OF IF R1 L423:;;;;;
11100 044456 ASL R1 ;R1 <- R1 * 4
11101 044456 006301 ASL R1
11102 044460 006301 BIS #BIT14,CONFIG+2(R1)
11103 044462 052761 040000 002666 END ;OF REPEAT BR B74
11104 044470 000760 E74:;;;;;
044472 END CMD16LOOP E73:;;;;;
11105 044472 TYPE MSG110 ;ONLY SELECTED BANKS WILL BE TESTED
044472 TYPEIT ,MSG110
11106 044472 104401 070777 .DSABL CRF
044472 SET SELONLY
11107 044476 012737 177777 002002 MOV #-1,SELONLY
044476 000207 RETURN
11108 044504
11109

```

```

11110 044506          FCMD16: SUBTST <<FS  COMMAND 16  RESUME TESTING ALL BANKS>>
;*****
;SUBTEST          FS  COMMAND 16  RESUME TESTING ALL BANKS
;*****
11111 044506          TYPE          MSG111
044506 104401 071043  TYPEIT        .MSG111
;*****
11112 044512 005037 002002  .DSABL        CRF
;*****
11113
11114          CLR          SELONLY
;*****
11115 044516 013702 002556  CMD16A: MOV          LASTBANK,R2
11116 044522 006302          ASL          R2
11117 044524 006302          ASL          R2
11118 044526          FOR R1 := #0 TO R2 BY #4
044526 005001
044530          CLR R1
11119 044530 042761 040000 002666          BIC #BIT14,CONFIG+2(R1)
11120 044536          END ;OF FOR R1
044536 062701 000004          ADD #4,R1
044542 020102          CMP R1,R2
044544 003771          BLE B75
044546          E75:;;;;;
11121 044546 000207          RETURN
  
```

```
11124 044550 FCMD17: SUBTST <<FS COMMAND 17 ENABLE TRACE>>
;*****
;*SUBTEST FS COMMAND 17 ENABLE TRACE
;*****
11125 044550 104401 071401 TYPE MSG127
044550 TYPEIT ,MSG127
11126 044554 012737 177777 006304 .DSABL CRF
11127 044562 000207 MOV @-1,TRACE
RETURN

11130 044564 FCMD18: SUBTST <<FS COMMAND 18 DISABLE TRACE>>
;*****
;*SUBTEST FS COMMAND 18 DISABLE TRACE
;*****
11131 044564 104401 071420 TYPE MSG128
044564 TYPEIT ,MSG128
11132 044570 005037 006304 .DSABL CRF
11133 044574 000207 CLR TRACE
RETURN
```

```

11136 044576          WHICHCSR:SUBTST <<SUBR DETERMINE CORRECT CSR>>
;*****
;*SUBTEST          SUBR DETERMINE CORRECT CSR
;*****
11137 044576 013700 002224          MOV TOTCSRS,RO ;GET CSR'S FLAG
11138 044602 022700 100000          CMP #BIT15,RO ;CSR 0?
11139 044606 001003                   BNE 1$ ;NO - SKIP
11140 044610 005037 002152          CLR CSRNO ;YES - SET IT UP
11141 044614 000207          RETURN
11142
11143 044616          1$: TYPE MSG022 ;WHICH CSR(O-F)
044616 104401 066775          TYPEIT ,MSG022
.DSABL CRF
RDLIN ;GET CHARACTER
POP RO ;PUT IN RO
MOV (SP),.RO
11144 044622 104412          MOV (RO),RO ;PUT CHAR IN RO
11145 044624          CMP RO,#106 ;CHECK LIMIT
044624 012600                   BHI 1$ ;IF BAD LOOP TILL HE TYPES IT RIGHT
11146 044626 011000                   CMP #A,RO
11147 044630 020027 000106          BHIS 2$
11148 044634 101370                   SUB #7,RO
11149 044636 C22700 000101          SUB #60,RO
11150 044642 103002                   ASL RO
11151 044644 162700 000007          MOV RO,CSRNO
11152 044650 162700 000060          RETURN
11153 044654 006300
11154 044656 010037 002152
11155 044662 000207

```

```

11621          .SBTTL  ERROR DATA (SUPERVISOR) SETUP STUFF
11622 044664          $PER25: LET ADDRESS := R1 #2
11622 044664 010137 002034          MOV R1,ADDRESS
11622 044670 162737 000002 002034          SUB #2,ADDRESS
11623 044676          IF ABORTFLAG IS FALSE
11623 044676 005737 002144          TST ABORTFLAG
11623 044702 001007          BNE L424
11624 044704          TESTAREA          ;ENTER TEST MODE
11624 044704 053737 002552 177776          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
11624          .DSABL CRF
11624          LET BAD := -2(R1)
11625 044712          KERNEL          ;ENTER KERNEL MODE
11625 044712 016137 177776 002052          MOV -2(R1),BAD
11626 044720 104417          END ;OF IF ABORTFLAG
11627 044722          L424:::
11627 044722          IF 177654 EQ #0
11628 044722 005737 177654          TST 177654
11628 044726 001003          BNE L425
11629 044730          LET GOOD := R2
11629 044730 C10237 002044          MOV R2,GOOD
11630 044734          ELSE
11630 044734 000402          BR L426
11631 044736          LET GOOD := R3
11631 044736 010337 002044          MOV R3,GOOD
11632 044742          END ;OF IF
11632 044742          L425:::
11633 044742 000137 050120          JMP PERRAW
11633          L426:::
11634
11635 044746          PERRA3: SUBTST <<DATA WAS 3 WORDS>>
11635          ;*****
11635          ;*SUBTEST DATA WAS 3 WORDS
11635          ;*****
11636 044746          IF BADPC EQ #0 THEN $CALL BADSTACK
11636 044746 005737 002022          TST BADPC
11636 044752 001002          BNE L427
11636 044754 004737 034040          JSR PC,BADSTACK
11636 044760          L427:::
11637 044760          PUSH R0
11637 044760 010046          MOV R0,-(SP)
11638 044762 005037 002150          CLR CSR          ;MAKE SURE CSR BIT HOLDER IS CLEAR
11639 044766 104505          CHK1DIS          ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
11640 044770          TESTAREA
11640 044770 053737 002552 177776          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
11640          .DSABL CRF
11641 044776 005711          TST (R1)          ;READ LOCATION TO READ CHECKBITS INTO CSR
11642 045000 104417          KERNEL
11643 045002 104426          READCSR          ;GET CSR CONTENTS
11644 045004 013700 002150          MOV CSR,R0          ;SAVE CSR CONTENTS IN R0
11645 045010 104503          CLR1CSR          ;RETURN CSR TO NORMAL MODE
11646 045012 072027 177773          ASH #-5,R0          ;MOVE CHECK BITS TO BOTTOM OF WORD
11647 045016 042700 177600          BIC #+C177,R0          ;CLEAR OFF EXTRANEIOUS GARBAGE
11648 045022          LET ADDRESS := R1          ;SAVE VIRTUAL ADDRESS FOR PRINTOUT
11649 045022 010137 002034          MOV R1,ADDRESS
11649 045026 005037 002044          CLR GOOD          ;FIRST TEST WORD WRITTEN SHOULD ALWAYS BE ZERO
11650 045032          TESTAREA          ;ENTER TEST MODE
11650 045032 053737 002552 177776          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
    
```

11651	045040	011137	002052	.DSABL	CRF				
11652	045044	011437	002054	MOV	(R1),BAD			;GET BAD DATA FROM MUT	FIRST WORD
11653	045050	104417		MOV	(R4),BAD2			;AND SECOND WORD	
11654	045052	110037	002056	KERNEL				;ENTER KERNEL MODE	
11655	045056	105037	002057	MOVB	RO,BAD3			;MOVE BAD CHECKBITS FOR PRINTOUT	
11656	045062	004737	050354	CLRB	BAD3+1			;CLEAR OFF THE OTHER UNUSED BITS	
11657	045066	104033		CALL	PERBNK			;MARK BANK AS BAD IN CONFIG TABLE	
11658	045070			ERROR	+33				
	045070	012600		POP	RO			;RESTORE RO	
11659	045072								MOV (SP)+,RO
	045072	032777	000001	IF	#SWO SET.IN @SWR				BIT #SWO,@SWR
	045100	001402							BEQ L430
11660	045102	104506		ENASBE				;TRAP ON SINGLE BIT ERRORS	
11661	045104			ELSE					
	045104	000401							BR L431
	045106								L430::::::::::
11662	045106	104472		ECCINIT				;TRAP ON UNCORRECTABLE ERRORS	
11663	045110			END; OF IF #SWO					
	045110								L431::::::::::
11664	045110	000002		RTI					


```

11667 045112          $PER30: LET GOOD := R1
      045112 010137 002044
11668 045116          LET ADDRESS := (SP) - 16
      045116 011637 002034
      045122 163737 000016 002034
11669 045130          IF ABORTFLAG IS FALSE
      045130 005737 002144
      045134 001007
11670 045136          TESTAREA          ;ENTER TEST MODE
      045136 053737 002552 177776     BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      LET BAD := @ADDRESS
11671 045144          MOV @ADDRESS,BAD
      045144 017737 134664 002052
11672 045152          KERNEL          ;ENTER KERNEL MODE
      104417
11673 045154          END ;OF IF ABORTFLAG
      045154
11674 045154          JMP PERRAW
      000137 050120
11675
11676 045160          GETDATA:SUBTST <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
      ;*****
      ;*SUBTEST          GET DATA FROM ABORTED AREA IF POSSIBLE
      ;*****
      PUSH          RO,4,114
11677 045160          MOV RO,-(SP)
      045160 010046          MOV 4,-(SP)
      045162 013746 000004          MOV 114,-(SP)
      045166 013746 000114
11678 045172          MOV SP,GETDA1
      010637 045256
11679 045176          MOV @1$,4
      012737 045236 000004
11680 045204          MOV @1$,114
      012737 045236 000114
11681 045212          MOV ADDRESS,RO
      013700 002034
11682 045216          TESTAREA
      045216 053737 002552 177776     BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      MOV (RO),BAD
11683 045224          KERNEL
      011037 002052
11684 045230          CLR ABORTFLAG
      104417
11685 045232          MOV GETDA1,SP
      005037 002144
11686 045236          POP 114,4,RO          ;RESTORE KNOWN GOOD STACK POINTER
      013706 045256
11687 045242          MOV (SP)+,114
      045242 012637 000114          MOV (SP)+,4
      045246 012637 000004          MOV (SP)+,RO
      045252 012600
11688 045254          RETURN
      000207
11689 045256          GETDA1: 0
      000000

```

```

11692          .SBTTL POWER FAIL AUTO RESTART
11693          .SBTTL ROUTINE POWER DOWN AND UP
11694          ;*****
11695          ;POWER DOWN ROUTINE
11696 045260     $PWRDN:
11704          ;SAVE CACHE STATUS
11705 045260 005737 002544     TST CACHKN
11706 045264 001403           BEQ 5$
11707 045266           PUSH  CONTRL
11708 045266 013746 177746           MOV  CONTRL,-(SP)
11708 045272 104423           CACHON
11709 045274 012737 046206 000024 5$:  MOV  @#ILLUP,PWRVEC ;TURN CACHE ON
11710 045302 012737 000340 000026     MOV  @340,PWRVEC+2 ;SET FOR FAST UP
11711 045310           PUSH  R0,R1,R2,R3,R4,R5,CSRNO ;PRIO:7
11711 045310 010046           MOV  R0,-(SP)
11711 045312 010146           MOV  R1,-(SP)
11711 045314 010246           MOV  R2,-(SP)
11711 045316 010346           MOV  R3,-(SP)
11711 045320 010446           MOV  R4,-(SP)
11711 045322 C10546           MOV  R5,-(SP)
11711 045324 013746 002152           MOV  CSRNO,-(SP)
11712          ;SAVE USER PAR'S & PDR7
11713 045330 012700 177700     MOV  @177700,R0
11714 045334 012701 000021     MOV  @17.,R1
11715 045340           1$:  PUSH  -(R0)
11716 045342 014046           MOV  -(R0),(SP)
11716 045342 077102           SOB  R1,1$
11717          ;SAVE SUPERVISOR PAR'S
11718 045344 005737 002456     TST  NOSUPER
11719 045350 001013           BNE  PD1
11720 045352 012700 172300     MOV  @172300,R0
11721 045356 012701 000020     MOV  @16.,R1
11722 045362           2$:  PUSH  -(R0)
11723 045362 014046           MOV  -(R0),-(SP)
11723 045364 077102           SOB  R1,2$
11724 045366           IF RLFLAG IS TRUE THEN $CALL WOOPS
11724 045366 005737 002126     TST  RLFLAG
11724 045372 001402           BEQ  L433
11724 045374 004737 046214     JSR  PC,WOOPS
11725          L433: :::::
11725          ;COPY KERNEL MAP TO USER & SUPERVISOR
11726 045400 012700 172300     PD1:  MOV  @KIPDR0,R0
11727 045404 012701 177600     MOV  @UIPDR0,R1
11728 045410 012702 172200     MOV  @SIPDR0,R2
11729 045414 012703 000040     MOV  @32.,R3
11730 045420 011021           3$:  MOV  (R0),(R1)+
11731 045422 012022           MOV  (R0)+,(R2)+
11732 045424 077303           SOB  R3,3$

```

11734						;SAVE USER & SUPERVISOR STACK POINTERS		
11735	045426					USER		
	045426	052737	140000	177776		BIS #BIT15:BIT14,PSW		;GO TO USER MODE
						.DSABL CRF		
11736	045434	010600				MOV USP,R0		
11737	045436	104417				KERNEL		;ENTER KERNEL MODE
11738	045440					PUSH R0		
	045440	010046						MOV R0,-(SP)
11739	045442	005737	002456			TST NOSUPER		
11740	045446	001006				BNE 7#		
11741	045450					SUPERVISOR		;ENTER SUPERVISOR MODE
	045450	052737	040000	177776		BIS #BIT14,PSW		;GO TO SUPERVISOR MODE
						.DSABL CRF		
11742	045456	010600				MOV SSP,R0		
11743	045460	104417				KERNEL		;ENTER KERNEL MODE
11744	045462					PUSH R0		
	045462	010046						MOV R0,(SP)
11745						;SAVE ECC REGISTERS		
11746	045464	013701	002224		7#:	MOV TOTCSRS,R1		;GET CSR'S
11747	045470					BEGIN LCSRSAVE		
	045470							B76:::~::~
11748	045470	005037	002152			FOR CSRNO := #0 TO #36 BY #2		CLR CSRNO
	045474							B77:::~::~
11749	045474	006301				ASL R1		
11750	045476					ON.ERROR		
	045476	103003						BCC L434
11751	045500	104426				READCSR		
11752	045502					PUSH CSR		
	045502	013746	002150					MOV CSR,(SP)
11753	045506					END ;OF ON.ERROR		
	045506							L434:::~::~
11754	045506					IF R1 EQ #0 THEN LEAVE LCSRSAVE		
	045506	005701						TST R1
	045510	001407						BEQ E76
11755	045512					END ;OF FOR CSRNO		
	045512	062737	000002	002152				ADD #2,CSRNO
	045520	023727	002152	000036				CHP CSRNO,#36
	045526	003762						BLE B77
	045530							E77:::~::~
11756	045530					END LCSRSAVE		
	045530							E76:::~::~
11757						;SAVE MMR0,1,2,3		
11758	045530					PUSH MMR0,MMR1,MMR2		
	045530	013746	177572					MOV MMR0,-(SP)
	045534	013746	177574					MOV MMR1,-(SP)
	045540	013746	177576					MOV MMR2,-(SP)
11759	045544	005737	002456			TST NOSUPER		
11760	045550	001002				BNE 8#		
11761	045552					PUSH MMR3		
	045552	013746	172516					MOV MMR3,(SP)
11762						;SAVE KERNEL PAR'S		
11763	045556	012700	172400		8#:	MOV #172400,R0		
11764	045562	012701	000020		4#:	MOV #16.,R1		
11765	045566					PUSH -(R0)		
	045566	014046						MOV -(R0),(SP)
11766	045570	077102				SQB R1,4#		

```
11767                                     ;SAVE Q-BUS MAP REGISTERS
11768 045572 022737 000005 004064      CMP      #5,PROTYP      ;IS THIS AN 11/83 ?
11769 045600 001004                                     BNE      9#             ;BRANCH IF NOT
11770 045602                                     PUSH     MAPHO,MAPLO
                                045602 013746 170202      MOV MAPHO,-(SP)
                                045606 013746 170200      MOV MAPLO,(SP)
11771                                     ;SAVE POSSIBLE SOFTWARE SWITCH REGISTER
11772 045612 017746 135020      9#:      PUSH     BSWR
                                045612 017746 135020      MOV BSWR,-(SP)
11773                                     ;SAVE STACK POINTER
11774 045616 010637 046212      MOV      SP,#SAVR6     ;;SAVE SP
11775                                     ;NOW SET UP REAL VECTOR
11776 045622 012737 045634 000024      MOV      #PWRUP,PWRVEC ;;SET UP VECTOR
11777 045630 000000                                     HALT
11778 045632 000776      $DOWN:  BR      $DOWN     ;;HANG UP
```

```

11781 ;*****
11782 ;POWER UP ROUTINE
11783 045634 ;PWRUP:
11787 045634 012737 046206 000024      MOV    #0ILLUP,PWRVEC ;SET FOR FAST DOWN
11788 ;RESTORE STACK POINTER
11789 045642 013706 046212      MOV    #SAVR6,SP      ;GET SP
11790 045646 005037 046212      CLR    #SAVR6        ;WAIT LOOP FOR THE TTY
11791 045652 005237 046212      10:   INC    #SAVR6        ;WAIT FOR THE INC
11792 045656 001375 ;BNE    10           ;OF A WORD
11793 ;RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
11794 045660      POP    BSMR
11795 045660 012677 134752      MOV    (SP)+,BSMR
11796 045664 012700 172340      100:  MOV    #172340,R0
11797 045670 012702 172300      MOV    #KIPDR0,R2
11798 045674 012701 000020      MOV    #16,R1
11799 045700 012620 ;POP    (R0)+
11799 045702 012722 077406      MOV    #77406,(R2)+
11800 045706 077104      SOB    R1,60
11801 ;RESTORE MMR3,2,1,0
11802 045710 005737 002456      TST    NOSUPER
11803 045714 001002      BNE    110
11804 045716 012637 172516      POP    MMR3
11805 045722 012637 177576      110:  POP    MMR2,MMR1,MMR0
11806 045722 012637 177574      MOV    (SP)+,MMR2
11807 045726 012637 177574      MOV    (SP)+,MMR1
11808 045732 012637 177572      MOV    (SP)+,MMR0
11806 ;RESTORE ECC REGISTERS
11807 045736 013701 002224      MOV    TOTCSRS,R1 ;GET CSR'S
11808 045742 042701 177400      BIC    #177400,R1
11809 045746 ;BEGIN LCSRRESTORE
11810 045746 ;FOR CSRNO := #36 DOWNT0 #0 BY #2
11811 045746 012737 000036 002152      ASR    R1
11812 045754 006201 ;ON.ERROR
11813 045756 103003 ;POP    CSR
11814 045760 012637 002150 ;LOADCSR
11815 045764 104425 ;END ;OF ON.ERROR
11816 045766 ;IF R1 EQ #0 THEN LEAVE LCSRRESTORE
11817 045766 005701 ;TST R1
11818 045770 001407 ;BEQ E100
11817 045772 ;END ;OF FOR CSRNO
11818 045772 162737 000002 002152      SUB    #2,CSRNO
11819 046000 023727 002152 000000      CMP    CSRNO,#0
11820 046006 002362 ;BGE B101
11818 046010 ;E101:
11819 046010 ;E100:

```

11819					;COPY KERNEL MAP TO USER & SUPERVISOR
11820	046010	012700	172300		MOV #KIPDR0,R0
11821	046014	012701	177600		MOV #UIPDR0,R1
11822	046020	012702	172200		MOV #SIPDR0,R2
11823	046024	012703	000040		MOV #32,R3
11824	046030	011021		34:	MOV (R0),(R1)+
11825	046032	012022			MOV (R0)+,(R2)+
11826	046034	077303			S0B R3,34

```

11828                                     ;RESTORE SUPERVISOR & USER STACK POINTERS
11829 046036 005737 002456               TST     NOSUPER
11830 046042 001006                       BNE     13$
11831 046044                               POP     R0
                                           MOV (SP),R0
11832 046046 02737 040000 177776         SUPERVISOR ;ENTER SUPERVISOR MODE
046046 02737 040000 177776             BIS     #BIT14,PSW ;GO TO SUPERVISOR MODE
                                           .DSABL CRF
11833 046054 010006                       MOV     R0,SSP
11834 046056 104417                       KERNEL ;ENTER KERNEL MODE
11835 046060 012600 13$:                 POP     R0
                                           MOV (SP),R0
11836 046062 052737 140000 177776         USER
046062 052737 140000 177776             BIS     #BIT15!BIT14,PSW ;GO TO USER MODE
                                           .DSABL CRF
11837 046070 010006                       MOV     R0,USP
11838 046072 104417                       KERNEL ;ENTER KERNEL MODE
11839                                     ;RESTORE SUPERVISOR PAR'S
11840 046074 012700 172240               MOV     #172240,R0
11841 046100 012701 000020               MOV     #16,R1
11842 046104 012620 7$:                 POP     (R0)
                                           MOV (SP),(R0)
11843 046106 077102                       SOB     R1,7$
11844                                     ;RESTORE USER PAR S & PDR7
11845 046110 012700 177636               MOV     #177636,R0
11846 046114 012701 000021               MOV     #17,R1
11847 046120 012620 8$:                 POP     (R0)
                                           MOV (SP),(R0)
11848 046122 077102                       SOB     R1,8$
11849                                     ;RESTORE POSSIBLE SOFTWARE DISPLAY REGISTER
11850 046124 013777 002012 134506         MOV     $PATMAR,@DISPLAY
11851 046132 013737 002012 000174         MOV     $PATMAR,DISPREG
11852 046140 012637 002152               POP     CSRNO,R5,R4,R3,R2,R1,R0
                                           MOV (SP),CSRNO
                                           MOV (SP),R5
                                           MOV (SP),R4
                                           MOV (SP),R3
                                           MOV (SP),R2
                                           MOV (SP),R1
                                           MOV (SP),R0
11853 046160 012737 045260 000024         MOV     #PWRDN,PWRVEC ;;SET UP THE POWER DOWN VECTOR
11854 046166 104401 067722               TYPE    MSG051 ;REPORT THE POWER FAILURE
046166 104401 067722               TYPEIT  ,MSG051
                                           .DSABL CRF
11855                                     ;RESTORE CACHE STATUS
11856 046172 005737 002544               TST     CACHKN
11857 046176 001402                       BEQ     9$
11858 046200 012637 177746               POP     CONTRL
                                           MOV (SP),CONTRL
11859 046204 000002 9$:                 RTI
11860 046206 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
11861 046210 000776 BR $ILLUP ;; BEFORE THE POWER DOWN WAS COMPLETE
11862 046212 000000 $SAVR6: 0 ;;PUT THE SP HERE
11863                                     .EVEN

```

```

11875 046214      WOOPS: SUBTST <<POWER FAIL WHILE RELOCATED>>
;*****
;SUBTEST      POWER FAIL WHILE RELOCATED
;*****
11876 046214      PUSH      BANK
11877 046214      013746 002102      MOV BANK,-(SP)
11878 046220      005037 002102      CLR      BANK
11878 046224      MAP      BANK      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
11878 046224      010346      MOV R3,-(SP)
11878 046226      013703 002102      MOV      BANK,R3
11878 046232      004737 035604      CALL     MAPPER
11878 046232      .DSABL   CRF
11878 046236      012603      MOV (SP),R3
11878 046240      SUPERVISOR ;ENTER SUPERVISOR MODE
11878 046240      052737 040000 177776      BIS      #BIT14,PSW ;GO TO SUPERVISOR MODE
11878 046240      .DSABL   CRF
11880 046246      013737 060024 046612      MOV      FIRST+PWRVEC,WOOPSAV
11881 046254      013737 060026 046614      MOV      FIRST+PWRVEC+2,WOOPSAV+2
11882 046262      BMOV     FIRST+WOOPUP,WOOPSAV+4,WOOPEND-WOOPUP/2+12.
11882 046262      C04537 040732      JSR R5,BLOCK3
11882 046266      000105      WOOPEND-WOOPUP/2+12.
11882 046270      046616      WOOPSAV+4
11882 046272      126400      FIRST+WOOPUP
11882 046272      .DSABL   CRF
11883 046274      012737 046400 060024      MOV      #WOOPUP,FIRST+PWRVEC
11884 046302      012737 000340 060026      MOV      #340,FIRST+PWRVEC+2
11885 046310      BMOV     WOOPUP,FIRST+WOOPUP,WOOPEND-WOOPUP/2
11885 046310      004537 040732      JSR R5,BLOCK3
11885 046314      000071      WOOPEND-WOOPUP/2
11885 046316      126400      FIRST+WOOPUP
11885 046320      046400      WOOPUP
11885 046320      .DSABL   CRF
11886 046322      012700 172340      MOV      #KIPARO,R0
11887 046326      012701 126562      MOV      #FIRST+WOOPEND,R1
11888 046332      012702 000010      MOV      #8,R2
11889 046336      012021      1#:      MOV      (R0),R1
11890 046340      077202      SOB      R2,1#
11891 046342      005737 002456      TST      NOSUPER
11892 046346      001002      BNE      2#
11893 046350      013721 172516      MOV      MMR3,(R1)
11894 046354      013721 177576      2#:      MOV      MMR2,(R1)
11895 046360      013721 177574      MOV      MMR1,(R1)
11896 046364      013721 177572      MOV      MMRO,(R1)
11897 046370      104417      KERNEL
11898 046372      POP      BANK      ;ENTER KERNEL MODE
11898 046372      012637 002102      MOV (SP),BANK
11899 046376      000207      RETURN

```


11902 046400

WOOPUP: SUBST <<POWER UP FROM BANK 0 TO RELOCATION>>
;*****
;SUBTEST POWER UP FROM BANK 0 TO RELOCATION
;*****

11903 046400 012700 046562
11904 046404 012701 172340
11905 046410 012703 172300
11906 046414 012702 000010
11907 046420 012021
11908 046422 012723 077406
11909 046426 077204
11910 046430 005737 002456
11911 046434 001002
11912 046436 012037 172516
11913 046442 012037 177576
11914 046446 012037 177574
11915 046452 012037 177572
11916 046456 013706 046212
11917 046462
046462 C13746 002102
11918 046466 005037 002102
11919 046472
046472 010346
046474 013703 002102
046500 004737 035604

046504 012603
11920 046506
046506 052737 040000 177776

11921 046514 013737 046612 060024
11922 046522 013737 046614 060026
11923
11924
11925 046530 012700 046616
11926 046534 012701 000105
11927 046540 012702 126400
11928 046544 012022
11929 046546 077102
11930
11931 046550 104417
11932 046552
046552 012637 002102
11933 046556 000137 045634
11934 046562 000014
11937 046612 000107

```

MOV #WOOPEND,R0
MOV #KIPARO,R1
MOV #KIPDRO,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 $SAVR6,SP
PUSH BANK
MOV BANK,-(SP)
CLR BANK
MAP BANK ,MAP SUPERVISOR SPACE (TEST AREA) TO BANK
MOV R3,-(SP)
MOV BANK,R3
CALL MAPPER
.DSABL CRF
MOV (SP)+,R3
SUPERVISOR ;ENTER SUPERVISOR MODE
BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
.DSABL CRF
MOV WOOPSAV,FIRST+PWRVEC
MOV WOOPSAV+2,FIRST+PWRVEC+2
;SIMULATE THE FOLLOWING BLOCK MOV BUT WITH NO STACK ACCESSES
;BMOV WOOPSAV+4,FIRST+WOOPUP,WOOPEND-WOOPUP/2+12.
MOV #WOOPSAV+4,R0
MOV #WOOPEND-WOOPUP/2+12.,R1
MOV #FIRST+WOOPUP,R2
2$: 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
    
```

```

11942          .SBTTL IO SUBROUTINES
11943
11944          .SBTTL ROUTINE TYPE
11945
11946          ;*****
11947          ;*ROUTINE TO TYPE ASCIZ MESSAGE MESSAGE MUST TERMINATE WITH A 0 BYTE.
11948          ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
11949          ;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
11950          ;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
11951          ;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
11952          ;*
11953          ;*CALL:
11954          ;*1) USING A TRAP INSTRUCTION
11955          ;*      TYPE      MESADR          ;.MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
11956          ;*OR
11957          ;*      TYPE
11958          ;*      MESADR
11959          ;*
11960
11961 047030 105737 002360 $TYPE: TSTB $TPFLG          ;;IS THERE A TERMINAL?
11962 047034 100407 BMI 6$          ;BR IF NO
11963 047036 010046 1$: MOV RO,-(SP)          ;;SAVE RO
11964 047040 017600 000002 MOV @2(SP),RO          ;;GET ADDRESS OF ASCIZ STRING
11965 047044 112046 4$: MOV (RO)+,-(SP)          ;;PUSH CHARACTER TO BE TYPED ONTO STACK
11966 047046 001005 BNE 7$          ;;BR IF IT ISN'T THE TERMINATOR
11967 047050 005726 TST (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
11968 047052 012600 5$: MOV (SP)+,RO          ;;RESTORE RO
11969 047054 062716 000002 6$: ADD @2,(SP)          ;;ADJUST RETURN PC
11970 047060 000002 RTI          ;;RETURN
11971 047062 122716 000011 7$: CMPB @HT,(SP)          ;BRANCH IF NOT <HT>
11972 047066 001002 BNE 11$          ;
11973 047070 112716 000040 MOVB #' ,(SP)          ;REPLACE TAB WITH SPACE
11974 047074 122716 000200 11$: CMPB @CRLF,(SP)          ;;BRANCH IF NOT <CRLF>
11975 047100 001006 BNE 8$          ;
11976 047102 005726 TST (SP)+          ;;POP <CR><LF> EQUIV
11977 047104 047104 104401 TYPE          ;;TYPE A CR AND LF
11978 047106 002660 .DSABL CRF
11979 047110 105037 047342 CLRB $CHARCNT          ;;CLEAR CHARACTER COUNT
11980 047114 000753 BR 4$          ;;GET NEXT CHARACTER
11981 047116 004737 047156 8$: CALL $TYPEC          ;;GO TYPE THIS CHARACTER
11982 047122 123726 002652 9$: CMPB $FILLC,(SP)+          ;;IS IT TIME FOR FILLER CHARS.?
11983 047126 001346 BNE 4$          ;;IF NO GO GET NEXT CHAR.
11984 047130 013746 002356 MOV $NULL,-(SP)          ;;GET # OF FILLER CHARS. NEEDED
11985          ;;AND THE NULL CHAR.
11986 047134 105366 000001 10$: DECB 1(SP)          ;;DOES A NULL NEED TO BE TYPED?
11987 047140 002770 BLT 9$          ;;BR IF NO--GO POP THE NULL OFF OF STACK
11988 047142 004737 047156 CALL $TYPEC          ;;GO TYPE A NULL
11989 047146 105337 047342 DECB $CHARCNT          ;;DO NOT COUNT AS A COUNT
11990 047152 000770 BR 10$          ;;LOOP
11991 047154 000000 XOCHAR: .WORD 0
11992 047156 047156 $TYPEC: PUSH R1          MOV R1,-(SP)
11993 047160 116601 000004 MOVB 4(SP),R1
11994 047164 005737 002544 TST CACHKN
11995 047170 001402 BEQ 2$

```

```

ROUTINE TYPE
11996 047172          PUSH   CONTRL
11997 047172 013746 177746          2$:  PUSH   RO                MOV CONTRL,-(SP)
      047176 010046          3$:  CACHOFF ;TURN CACHE OFF
      047176 104424          TSTB  @#TPS ;WAIT UNTIL PRINTER IS READY
11998 047200 104424          BPL   3$
12023 047202 105777 133440          CLR   XOCHAR
12024 047206 100375          TSTB  @#TKS ;CHECK FOR XOFF
12025 047210 005037 047154          BPL   NC ;SKIP IF NO CHARACTER
12026 047214 105777 133422          MOVB  @#TKB,XOCHAR ;SAVE THE CHARACTER
12027 047220 100032          BIC  @#C177,XOCHAR ;STRIP OFF ASCII
12028 047222 117737 133416 047154          CMP  XOCHAR,#023 ;WAS IT A CONTROL S?
12029 047230 042737 177600 047154          BNE  NC ;BRANCH IF NOT
12030 047236 023727 047154 000023          CONTS3: TSTB @#TKS ;WAIT FOR CHARACTER
12031 047244 001020          BPL  CONTS3
12032 047246 105777 133370          MOVB  @#TKB,XOCHAR ;GET CHARACTER
12033 047252 100375          BIC  @#C177,XOCHAR ;STRIP OFF ASCII
12034 047254 117737 133364 047154          IF XOCHAR EQ #21 ; IF IT IS A +Q
12035 047262 042737 177600 047154          CMP  XOCHAR,#21
12036 047270          BNE  L436
      047270 C23727 047154 000021          BR   NC
      047276 001002          ELSE
12037 047300 000402          BR   L437
12038 047302          BR   L436
      047302 000401          L436:;;;;;
      047304          BR   L437
12039 047304 000760          END ;OF IF XOCHAR
12040 047306          L437:;;;;;
12041 047306 110177 133336          NC:  MOVB  R1,@#TPB ;LOAD CHAR TO BE TYPED INTO DATA REG.
12045 047312 122766 000015 000002          CMPB @CR,2(SP) ;IS CHARACTER A CARRIAGE RETURN?
12046 047320 001003          BNE  1$ ;BRANCH IF NO
12047 047322 105037 047342          CLRB $CHARCNT ;YES--CLEAR CHARACTER COUNT
12048 047326 000406          BR   $TYPEX ;EXIT
12049 047330 122766 000012 000002          1$:  CMPB @LF,2(SP) ;IS CHARACTER A LINE FEED?
12050 047336 001402          BEQ  $TYPEX ;BRANCH IF YES
12051 047340 105227          INCB (PC)+ ;COUNT THE CHARACTER
12052 047342 000000          $CHARCNT: .WORD 0 ;CHARACTER COUNT STORAGE
12053 047344          $TYPEX: POP  RO
      047344 012600          MOV  (SP)+,RO
12054 047346 005737 002544          TST  CACHKN ;IS THERE A CACHE?
12055 047352 001402          BEQ  2$ ;BRANCH IF NOT
12056 047354          POP  CONTRL ;POP CACHE STATUS
      047354 012637 177746          MOV  (SP)+,CONTRL
12057 047360          2$:  POP  R1
      047360 012601          MOV  (SP)+,R1
12058 047362 000207          RETURN
12059 047364          SUPLIMIT:;!!!!!!!!!!!!!!!!THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE

```

```

12737      .SBTTL  ERROR DATA SETUP
12738      :
12739      : USE THIS   IF THIS CONDITION DISCRIBES THE ERROR
12740      :
12741      : PERR01    TRAP
12742      :           BAD DATA IN R0 UNLESS ABORTED
12743      :           THEN BAD DATA IS POINTED TO BY -(R4)
12744      :           GOOD DATA IN R5
12745      :
12746      : PERR02    TRAP
12747      :           BAD DATA IN R1 UNLESS ABORTED
12748      :           THEN BAD DATA IS POINTED TO BY -(R4)
12749      :           GOOD DATA IN R2
12750      :
12751      : PERR03    TRAP
12752      :           BAD DATA IS POINTED TO BY -(R1)
12753      :           GOOD DATA IN R4
12754      :
12755      : PERR04    TRAP
12756      :           BAD DATA IN R4 UNLESS ABORTED
12757      :           THEN BAD DATA IS POINTED TO BY -2(R0)
12758      :           GOOD DATA IN R2
12759      :
12760      : PERR05    JSR    PC
12761      :           BAD DATA IS POINTED TO BY -(R0)
12762      :           GOOD DATA IN R2
12763      :           RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
12764      :
12765      : PERR06    JSR    PC
12766      :           BAD DATA IS POINTED TO BY -(R0)
12767      :           GOOD DATA IS ZERO
12768      :           RETURN AFTER SETTING UP GOOD,BAD,ADDRESS
12769      :
12770      : PERR07    TRAP
12771      :           BAD DATA IN R2 UNLESS ABORTED
12772      :           THEN BAD DATA IS POINTED TO BY (R1)
12773      :           GOOD DATA IN DATBUF
12774      :
12775      : PERR10    TRAP
12776      :           BAD DATA IN R2 UNLESS ABORTEC
12777      :           THEN BAD DATA IS POINTED TO BY 2(R1)
12778      :           GOOD DATA IN DATBUF+2
12779      :
12780      : PERR11    TRAP
12781      :           BYTE TEST
12782      :           BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
12783      :           THEN BAD DATA IS POINTED TO BY (R1)
12784      :           GOOD DATA IS A ZERO BYTE
12785      :
12786      : PERR12    TRAP
12787      :           BYTE TEST
12788      :           BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
12789      :           THEN BAD DATA IS POINTED TO BY (R1)
12790      :           GOOD DATA IS A BYTE OF ONES
12791      :
12792      : PERR13    TRAP
12793      :           BAD DATA IN R0 UNLESS ABORTED
  
```

```
12794 :                               THEN BAD DATA IS POINTED TO BY (R1)
12795 :                               GOOD DATA IS ZERO
12796 :
12797 : PERR14  TRAP
12798 :          BAD DATA IN R0 UNLESS ABORTED
12799 :          THEN BAD DATA IS POINTED TO BY (R1)
12800 :          GOOD DATA IS ONES
12801 :
12802 : PERR15  TRAP
12803 :          BAD DATA IN R0 UNLESS ABORTED
12804 :          THEN BAD DATA IS POINTED TO BY (R1)
12805 :          GOOD DATA IN TSTOAT
12806 :
12807 : PERR16  TRAP
12808 :          BAD DATA IN R0 UNLESS ABORTED
12809 :          THEN BAD DATA IS POINTED TO BY (R1)
12810 :          GOOD DATA IN TSTOAT+2
12811 :
12812 : PERR17  TRAP
12813 :          BAD DATA IN R0 UNLESS ABORTED
12814 :          THEN BAD DATA IS POINTED TO BY (R1)
12815 :          GOOD DATA IN R2
12816 :
12817 : PERR20  TRAP
12818 :          BAD DATA IN R0 UNLESS ABORTED
12819 :          THEN BAD DATA IS POINTED TO BY (R1)
12820 :          GOOD DATA IN R3
12821 :
12822 : PERR21  TRAP
12823 :          7 BIT BYTE TEST
12824 :          BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
12825 :          THEN BAD DATA IS POINTED TO BY (R1)
12826 :          GOOD DATA IS A 7 BIT BYTE ON ONES
12827 :
12828 : PERR22  TRAP
12829 :          BAD DATA IN R2 UNLESS ABORTED
12830 :          THEN BAD DATA IS POINTED TO BY (R1)
12831 :          GOOD DATA IN R0
12832 :
12833 : PERR23  TRAP
12834 :          BAD DATA IN R0 UNLESS ABORTED
12835 :          THEN BAD DATA IS POINTED TO BY (R1)
12836 :          GOOD DATA IN R4
12837 :
12838 : PERR24  TRAP
12839 :          BAD DATA IN R0 UNLESS ABORTED
12840 :          THEN BAD DATA IS POINTED TO BY (R2)
12841 :          GOOD DATA IN R3
12842 :
12843 : PERR25  TRAP
12844 :          BAD DATA POINTED TO BY -(R1)
12845 :          GOOD DATA IN R2 UNLESS LOC V177654 IS SET
12846 :          THEN GOOD DATA IS IN R3
12847 :
12848 : PERR26  TRAP
12849 :          BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
12850 :          GOOD DATA IS 000000..100000..100
```

```
12851      |
12852      | PERR27    TRAP
12853      |           BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
12854      |           GOOD DATA IS 000000,,000000,,077
12855      |
12856      | PERR30    TRAP
12857      |           BAD DATA IS POINTED TO BY -16(SP)
12858      |           GOOD DATA IS IN R1
12859      |
12860      | PERR31    TRAP
12861      |           SPECIAL ECC FAILURE HANDLER
12862      |
12863      | PERR32    TRAP
12864      |           SPECIAL ECC FAILURE HANDLER
12865      |
12866      | PERR33    TRAP
12867      |           SPECIAL ECC FAILURE HANDLER
12868      |
12869      | PERR34    TRAP
12870      |           SPECIAL ECC FAILURE HANDLER
12871      |
12872      | PERR35    TRAP
12873      |           SPECIAL BRANCH GOBBLE FAILURE HANDLER
12874      |
12875      |           CALLING SEQUENCE FOR TRAP TYPES
12876      |           BEQ      2#           ;NO - ERROR,BRANCH FOR CARD
12877      |           PERRXX   ;TRAP TO ERROR ROUTINE
12878      | 2# : NEXT INSTRUCTION ;CONTINUE TESTING
```

12881	047364	010437	002034	\$PER01: MOV	R4, ADDRESS	
12882	047370	162737	000002	002034 SUB	#2, ADDRESS	
12883	047376	010037	002052	MOV	R0, BAD	
12884	047402	010537	002044	MOV	R5, GOOD	
12885	047406	000137	050120	JMP	PERRAW	
12886						
12887	047412	010437	002034	\$PER02: MOV	R4, ADDRESS	
12888	047416	162737	000002	002034 SUB	#2, ADDRESS	
12889	047424	010137	002052	MOV	R1, BAD	
12890	047430	010237	002044	MOV	R2, GOOD	
12891	047434	000137	050120	JMP	PERRAW	
12892						
12893	047440	010137	002034	\$PER03: MOV	R1, ADDRESS	
12894	047444	162737	000002	002034 SUB	#2, ADDRESS	
12895	047452	010437	002044	MOV	R4, GOOD	
12896	047456	016137	177776	002052 MOV	-2(R1), BAD	
12897	047464	000137	050120	JMP	PERRAW	
12898						
12899	047470	010037	002034	\$PER04: MOV	R0, ADDRESS	
12900	047474	162737	000002	002034 SUB	#2, ADDRESS	
12901	047502	010437	002052	MOV	R4, BAD	
12902	047506	010237	002044	MOV	R2, GOOD	
12903	047512	000137	050120	JMP	PERRAW	
12904						
12905	047516	010237	002044	PERR05: MOV	R2, GOOD	
12906	047522	014037	002052	PERR05: MOV	-(R0), BAD	
12907	047526	010037	002034	MOV	R0, ADDRESS	
12908	047532	062700	000002	ADD	#2, R0	
12909	047536	004737	034040	CALL	BADSTACK	:RESTORE R0
12910	047542	000207		RETURN		
12911						
12912	047544	005037	002044	PERR06: CLR	GOOD	
12913	047550	000764		BR	PERR05	
12914						
12915	047552	010137	002034	\$PER07: MOV	R1, ADDRESS	
12916	047556	010237	002052	MOV	R2, BAD	
12917	047562	013737	002242	002044 MOV	DATBUF, GOOD	
12918	047570	000137	050120	JMP	PERRAW	
12919						
12920	047574			\$PER10: LET	ADDRESS := R1 + #2	
	047574	010137	002034			MOV R1, ADDRESS
	047600	062737	000002	002034		ADD #2, ADDRESS
12921	047606				LET BAD := R2	
	047606	010237	002052			MOV R2, BAD
12922	047612				LET GOOD := DATBUF+2	
	047612	013737	002244	002044		MOV DATBUF+2, GOOD
12923	047620	000137	050120	JMP	PERRAW	
12924						
12925	047624			\$PER11: LET	ADDRESS := R1	
	047624	010137	002034			MOV R1, ADDRESS
12926	047630				LET BAD := R0	
	047630	010037	002052			MOV R0, BAD
12927	047634				LET GOOD := #0	
	047634	005037	002044			CLR GOOD
12928	047640	000137	050172	JMP	PERRAB	
12929						
12930	047644			\$PER12: LET	ADDRESS := R1	

	047644	010137	002034	
12931	047650			
	047650	010037	002052	
12932	047654			
	047654	012737	000377	002044
12933	047662	000137	050172	

LET BAD := R0
LET GOOD := #377
JMP PERRAB

MOV R1,ADDRESS
MOV R0,BAD
MOV #377,GOOD

12936	047666			\$PER13: LET ADDRESS := R1	
	047666	010137	002034		MOV R1,ADDRESS
12937	047672			LET BAD := R0	
	047672	010037	002052		MOV R0,BAD
12938	047676			LET GOOD := #0	
	047676	005037	002044		CLR GOOD
12939	047702	000137	050120	JMP PERRAW	
12940					
12941	047706			\$PER14: LET ADDRESS := R1	
	047706	010137	002034		MOV R1,ADDRESS
12942	047712			LET BAD := R0	
	047712	010037	002052		MOV R0,BAD
12943	047716			LET GOOD := ONES	
	047716	013737	002614 002044		MOV ONES,GOOD
12944	047724	000137	050120	JMP PERRAW	
12945					
12946	047730			\$PER15: LET ADDRESS := R1	
	047730	010137	002034		MOV R1,ADDRESS
12947	047734			LET BAD := R0	
	047734	010037	002052		MOV R0,BAD
12948	047740			LET GOOD := TSTDAT	
	047740	013737	002246 002044		MOV TSTDAT,GOOD
12949	047746	000137	050120	JMP PERRAW	
12950					
12951	047752			\$PER16: LET ADDRESS := R1	
	047752	010137	002034		MOV R1,ADDRESS
12952	047756			LET BAD := R0	
	047756	010037	002052		MOV R0,BAD
12953	047762			LET GOOD := TSTDAT+2	
	047762	013737	002250 002044		MOV TSTDAT+2,GOOD
12954	047770	000453		BR PERRAW	
12955					
12956	047772			\$PER17: LET ADDRESS := R1	
	047772	010137	002034		MOV R1,ADDRESS
12957	047776			LET BAD := R0	
	047776	010037	002052		MOV R0,BAD
12958	050002			LET GOOD := R2	
	050002	010237	002044		MOV R2,GOOD
12959	050006	000444		BR PERRAW	
12960					
12961	050010			\$PER20: LET ADDRESS := R1	
	050010	010137	002034		MOV R1,ADDRESS
12962	050014			LET BAD := R0	
	050014	010037	002052		MOV R0,BAD
12963	050020			LET GOOD := R3	
	050020	010337	002044		MOV R3,GOOD
12964	050024	000435		BR PERRAW	
12965					
12966	050026			\$PER21: LET ADDRESS := R1	
	050026	010137	002034		MOV R1,ADDRESS
12967	050032			LET BAD := R0	
	050032	010037	002052		MOV R0,BAD
12968	050036			LET GOOD := #177	
	050036	012737	000177 002044		MOV #177,GOOD
12969	050044	000477		BR PERRAW	
12970					
12971	050046			\$PER22: LET ADDRESS := R1	

12972	050046	010137	002034		MOV R1,ADDRESS
	050052			LET BAD := R2	
12973	050052	010237	002052		MOV R2,BAD
	050056			LET GOOD := R0	
12974	050056	010037	002044		MOV R0,GOOD
	050062	000416		BR PERRAW	
12975					
12976	050064			%PER23: LET ADDRESS := R1	MOV R1,ADDRESS
	050064	010137	002034		
12977	050070			LET BAD := R0	MOV R0,BAD
	050070	010037	002052		
12978	050074			LET GOOD := R4	MOV R4,GOOD
	050074	010437	002044		
12979	050100	000407		BR PERRAW	
12980					
12981	050102			%PER24: LET ADDRESS := R2	MOV R2,ADDRESS
	050102	010237	002034		
12982	050106			LET BAD := R0	MOV R0,BAD
	050106	010037	002052		
12983	050112			LET GOOD := R3	MOV R3,GOOD
	050112	010337	002044		
12984	050116	000400		BR PERRAW	

12986 050120

PERRAW: SUBTST <<DATA WAS A WORD>>
;*****
;*SUBTEST DATA WAS A WORD
;*****

12987 050120 004737 050354

CALL PERBNK
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA

TST ABORTFLAG
BEQ L440
JSR PC.GETDATA

L440:;;;;;

12988 050124 005737 002144

050130 001402

050132 004737 045160

050136

12989 050136 005737 002022

IF BADPC EQ #0 THEN \$CALL BADSTACK

TST BADPC
BNE L441
JSR PC.BADSTACK

L441:;;;;;

050142 001002

050144 004737 034040

050150

12990 050150 004737 050330

CALL PERXOR
IF ABORTFLAG IS FALSE

TST ABORTFLAG
BNE L442

12991 050154 005737 002144

050160 001002

12992 050162 104011

ERROR +11
ELSE

BR L443

L442:;;;;;

12993 050164 000401

050166

12994 050166 104012

ERROR +12
END ;OF IF ABORTFLAG

L443:;;;;;

12995 050170

050170

12996 050170 000002

RTI

12997

12998 050172

PERRAB: SUBTST <<DATA WAS A BYTE>>
;*****
;*SUBTEST DATA WAS A BYTE
;*****

12999 050172 004737 050354

CALL PERBNK
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA

TST ABORTFLAG
BEQ L444
JSR PC.GETDATA

L444:;;;;;

13000 050176 005737 002144

050202 001402

050204 004737 045160

050210

13001 050210 005737 002022

IF BADPC EQ #0 THEN \$CALL BADSTACK

TST BADPC
BNE L445
JSR PC.BADSTACK

L445:;;;;;

050214 001002

050216 004737 034040

050222

13002 050222 004737 050330

CALL PERXOR
IF ABORTFLAG IS FALSE

TST ABORTFLAG
BNE L446

13003 050226 005737 002144

050232 001002

13004 050234 104014

ERROR +14
ELSE

BR L447

L446:;;;;;

13005 050236 000401

050240

13006 050240 104015

ERROR +15
END ;OF IF ABORTFLAG

L447:;;;;;

13007 050242

050242

13008 050242 000002

RTI

```
13011 050244 PERRA7: SUBTST <<DATA WAS A 7 BIT BYTE>>
;*****
; *SUBTEST DATA WAS A 7 BIT BYTE
;*****
13012 050244 005737 002022 IF BADPC EQ #0 THEN $CALL BADSTACK
050244 001002 TST BADPC
050250 004737 034040 BNE L450
050252 004737 JSR PC,BADSTACK
050256 L450:;;;;;
13013 050256 004737 050330 CALL PERXOR
13014 050262 004737 050354 CALL PERBNC
13015 050266 104022 ERROR +22
13016 050270 000002 RTI
13017
13018 050272 $PER26: LET GOOD2 := #100000 MOV #100000,GOOD2
050272 012737 100000 002046 LET GOOD3 := #100
13019 050300 012737 000100 002050 MOV #100,GOOD3
050300 000137 044746 JMP PERRA3
13020 050306
13021
13022 050312 $PER27: CLR GOOD2
13023 050316 012737 000077 002050 LET GOOD3 := #077 MOV #077,GOOD3
050316 000137 044746 JMP PERRA3
13024 050324
13025
13026 050330 PERXOR: SUBTST <<DETERMINE XOR OF GOOD & BAD>>
;*****
; *SUBTEST DETERMINE XOR OF GOOD & BAD
;*****
13027 050330 PUSH RO MOV RO,-(SP)
050330 010046
13028 050332 013700 002044 MOV GOOD,RO
13029 050336 013737 002052 002060 MOV BAD,BAD XOR
13030 050344 074037 002060 XOR RO,BAD XOR
13031 050350 POP RO MOV (SP),RO
050350 012600
13032 050352 000207 RETURN
```

```

13035 050354 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
13036
13037 050354 PUSH RO,R1
050354 010046 MOV RO,(SP)
050356 010146 MOV R1,(SP)
13038 050360 013701 002102 MOV BANK,R1
13039 050364 006301 ASL R1
13040 050366 006301 ASL R1
13041 050370 052761 000001 002664 BIS #BIT0,CONFIG(R1)
13042 050376 105261 002666 INCB CONFIG+2(R1) ;BUMP BANK COUNTER
13043 050402 001902 BNE 12# ;NO OVERFLOW SKIP
13044 050404 105361 002666 DECB CONFIG+2(R1) ;SET BACK TO 255.
13045 050410 126137 002666 002554 12# CMPB CONFIG+2(R1),ERRMAX ;IS IT PAST MAX?
13046 050416 101403 BLOS 11# ;NO - SKIP
13047 050420 SET TOOMANY ;YES
050420 012737 177777 002406 MOV #1,TOOMAN;
13048 050426 11# POP R1,R0
050426 012601 MOV (SP),R1
050430 012600 MOV (SP),R0
13049 050432 000207 RETURN
13050
13051 050434 010037 002052 PERECC: MOV RO,BAD
13052 050440 IF ADDRESS EQ TESTADD
050440 023737 002034 002412 CMP ADDRESS,TESTADD
050446 001004 BNE L451
13053 050450 013737 002246 002044 MOV TSTDAT,GOOD
13054 050456 ELSE
050456 000403 BR L452
050460 L451:;;;;;
13055 050460 013737 002250 002044 MOV TSTDAT+2,GOOD
13056 050466 END ;OF IF (R1)
050466 L452:;;;;;
13057 050466 004737 050330 CALL PERXOR
13058 050472 SET HEADER
050472 012737 177777 002612 MOV #1,HEADER
13059 050500 000207 RETURN
13060
13061 050502 $PER31: IF REALPAT EQ #41
050502 023727 002300 000041 CMP REALPAT,#41
050510 001001 BNE L453
13062 050512 104023 ERROR +23
13063 050514 END
050514 L453:;;;;;
13064 050514 IF BADPC EQ #0 THEN $CALL BADSTACK
050514 005737 002022 TST BADPC
050520 001002 BNE L454
050522 004737 034040 JSR PC,BADSTACK
050526 L454:;;;;;
13065 050526 004737 050434 CALL PERECC
13066 050532 IF REALPAT EQ #11
050532 023727 002300 000011 CMP REALPAT,#11
050540 001001 BNE L455
13067 050542 104037 ERROR +37
13068 050544 END ;OF IF REALPAT

```

13069	050544				
	050544	023727	002300	000015	
	050552	001001			
13070	050554	104043			
13071	050556				
	050556				
13072	050556				
	050556	023727	002300	000016	
	050564	001001			
13073	050566	104044			
13074	050570				
	050570				
13075	050570				
	050570	012737	177777	002612	
13076	050576	000002			

IF REALPAT EQ #15

ERROR +43
END ;OF IF REALPAT

IF REALPAT EQ #16

ERROR +44
END ;OF IF REALPAT

SET HEADER

RTI

L455:|||||

CMP REALPAT,#15
BNE L456

L456:|||||

CMP REALPAT,#16
BNE L457

L457:|||||

MOV #-1,HEADER

```

13079 050600          $PER32: IF BADPC EQ #0 THEN $CALL BADSTACK
      050600 005737 002022          TST BADPC
      050604 001002          BNE L460
      050606 004737 034040          JSR PC,BADSTACK
      050612          L460:;;;;;
13080 050612 010137 002034          MOV R1,ADDRESS
13081 050616 010037 002052          MOV RO,BAD
13082 050622 010237 002044          MOV R2,GOOD
13083 050626          SET HEADER
      050626 012737 177777 002612          MOV #-1,HEADER
13084 050634 104040          ERROR +40
13085 050636          SET HEADER
      050636 012737 177777 002612          MOV #-1,HEADER
13086 050644 000002          RTI
13087
13088 050646          $PER33: IF BADPC EQ #0 THEN $CALL BADSTACK
      050646 005737 002022          TST BADPC
      050652 001002          BNE L461
      050654 004737 034040          JSR PC,BADSTACK
      050660          L461:;;;;;
13089 050660 010137 002034          MOV R1,ADDRESS
13090 050664 010037 002052          MOV RO,BAD
13091 050670 105037 002053          CLRB BAD+1
13092 050674 012737 000377 002044          MOV #377,GOOD
13093 050702 004737 050330          CALL PERXOR
13094 050706          SET HEADER
      050706 012737 177777 002612          MOV #-1,HEADER
13095 050714 104041          ERROR +41
13096 050716          SET HEADER
      050716 012737 177777 002612          MOV #-1,HEADER
13097 050724 000002          RTI
13098
13099 050726          $PER34:IF BADPC EQ #0 THEN $CALL BADSTACK
      050726 005737 002022          TST BADPC
      050732 001002          BNE L462
      050734 004737 034040          JSR PC,BADSTACK
      050740          L462:;;;;;
13100 050740          IF #BIT15:BIT4 OFF.IN CSR
      050740 032737 100020 002150          BIT #BIT15:BIT4,CSR
      050746 001002          BNE L463
13101 050750 104016          ERROR +16          :NO SBE OR DBE
13102 050752          ELSE
      050752 000401          BR L464
      050754          L463:;;;;;
13103 050754 104001          ERROR +1          :EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
13104 050756          END ;OF IF #BIT15:BIT4
      050756          L464:;;;;;
13105 050756 000002          RTI
13106
13107          ;DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG
13108 050760 004737 050354          $PER35: CALL PERBNK
13109 050764 004737 034040          CALL BADSTACK
13110 050770 013737 002032 002052          MOV BADPSW,BAD
13111 050776 012737 000012 002044          MOV #12,GOOD
13112 051004 104047          ERROR +47
13113 051006 062706 000004          ADD #4,SP
13114 051012 000207          RETURN          ;FIX STACK FROM TRAP
          ;ABORTING TEST

```

13115									
13116	051014	010037	002044		\$PER36: MOV	R0.GOOD			
13117	051020	010137	002052		MOV	R1.BAD			
13118	051024				SET	HEADER			
	051024	012737	177777	002612				MOV	# 1.HEADER
13119	051032	104023			ERROR	+23			
13120	051034				SET	HEADER			
	051034	012737	177777	002612				MOV	#-1.HEADER
13121	051042	000002			RTI				
13122									
13123	051044	104053			\$PER37: ERROR	+53		:ILC;:REV B	
13124	051046	000002			RTI			:ILC;:REV B	
13125									
13126	051050	104054			\$PER40: ERROR	+54		:ILC;:REV B	
13127	051052	000002			RTI			:ILC;:REV B	


```

13130 .SBTTL ROUTINE SCOPE HANDLER
13131 ;*****
13132 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
13133 ;*AND LOAD THE DISPLAY DATA INTO THE DISPLAY REGISTER
13134 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
13135 ;*SW14=1 LOOP ON TEST
13136 ;*SW9=1 LOOP ON ERROR
13137 ;*CALL
13138 ;*
13139 051054 005237 056726 ;SCOPE: INC $DEVCT ;;SCOPE=IOT ;TELL APT WE ARE ALIVE
13140 051060 002004 056726 IF RESULT IS LT ; BGE L465
13141 051062 005037 056726 CLR $DEVCT
13142 051066 105237 056730 INCB $UNIT
13143 051072 004737 055222 END ;OF IF RESULT
13144 051072 104410 CKSWR ;;TEST FOR CHANGE IN SOFT-SWR L465:;;;;;
13145 051074 005737 006304 TST TRACE
13146 051100 001402 BEQ NOTRCE
13147 051102 C04737 055222 CALL CONTT ;TRACE
13148 051106 NOTRCE:
13149 051106 005737 052364 TST CPERRF ;IS THERE A CPU ERROR REGISTER? ;R-C
13150 051112 001410 BEQ SKJ ;BRANCH IF NOT ;R-C
13151 051114 013737 177766 052362 MOV @0177766,CPSAVE ;GET CONTENTS OF ERROR REGISTER ;R-C
13152 051122 032737 000001 052362 BIT @BIT0,CPSAVE ;IS THE POWER FAIL MONITOR BIT SET? ;R-C
13153 051130 001401 BEQ SKJ ;BRANCH IF NOT ;R-C
13154 051132 104177 ERROR +177 ;REPORT IF SO ;R-C
13163 051134 SKJ: IF STOPOK IS TRUE AND $SW8 SET.IN $SWR ;R-C
13164 051134 005737 002420 TST STOPOK
13165 051156 000137 040512 BEQ L466
13166 051162 001404 BIT $SW8,$SWR
13167 051162 005737 002440 BEQ L466
13168 051170 000002 IF NOSCOPE IS TRUE L466:;;;;;
13169 051172 001401 TST NOSCOPE
13170 051172 032777 040000 131436 BEQ L467
13171 051200 001051 ;*****START OF CODE FOR THE XOR TESTER*****
13172 051202 000425 $XTSTR: BR 2$ ;;IF RUNNING ON THE "XOR" TESTER CHANGE
13173 051204 013746 000004 MOV ERRVEC,-(SP) ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
13174 051210 012737 051230 000004 MOV @1$,ERRVEC ;;SAVE THE CONTENTS OF THE ERROR VECTOR
13175 051216 005737 177060 TST 177060 ;;SET FOR TIMEOUT
13176 051222 012637 000004 MOV (SP)+,ERRVEC ;;TIME OUT ON XOR?
13177 051226 000430 BR $SVLAD ;;RESTORE THE ERROR VECTOR
13178 051230 062706 000004 1$: ADD @4,SP ;;GO TO THE NEXT TEST
13179 051234 022737 000005 004064 CMP @5,PROTYP ;;FIX STACK FROM TRAP
13180 051242 001002 BNE 6$ ;IS THIS AN 11/83 ?
13181 051244 005037 177766 CLR CPUERR ;BRANCH IF NOT
;RESET CPU ERROR REGISTER

```

```
13183 051250 012637 000004      6$:   MOV    (SP)+,ERRVEC    ;;RESTORE THE ERROR VECTOR
13184 051254 000407              BR     4$                ;;LOOP ON THE PRESENT TEST
13185 051256                    2$:;#####END OF CODE FOR THE XOR TESTER#####
13186 051256 105737 002014      3$:   TSTB   $ERFLG        ;;HAS AN ERROR OCCURRED?
13187 051262 001412              BEQ    $SVLAD           ;;BR IF NO
13188 051264 032777 001000 131344  BIT    $SW9,$SWR       ;;LOOP ON ERROR?
13189 051272 001404              BEQ    5$                ;;BR IF NO
13190 051274 013737 002624 002622  4$:   MOV    $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
13191 051302 000410              BR     $OVER
13192 051304 105037 002014      5$:   CLRB   $ERFLG        ;;ZERO THE ERROR FLAG
13193 051310 011637 002622  $SVLAD: MOV   (SP),$LPADR    ;;SAVE SCOPE LOOP ADDRESS
13194 051314 011637 002624      MOV   (SP),$LPERR    ;;SAVE ERROR LOOP ADDRESS
13195 051320 005037 002362      CLR   $ESCAPE        ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
13196 051324 004737 051336  $OVER: CALL  GETDIS
13197 051330 013716 002622      MOV   $LPADR,(SP)   ;;FUDGE RETURN ADDRESS
13198 051334 000002              RTI                    ;;FIXES PS
```

13200 051336

GETDIS: SUBTST <<SUBR DISPLAY>>
;*****
;*SUBTEST SUBR DISPLAY
;*****

13201 051336 113737 002102 002013
13202 051344 113737 002300 002012
13203 051352
051352 010046
13204 051354 005737 002126
13205 051360 001403
13206 051362 052737 100000 002012
13207 051370
13211 051370 013777 002012 131242
13212 051376 013737 002012 000174
13213 051404
051404 012600
13214 051406 000207

MOV BANK, #BANK
MOV REALPAT, #PATMAR
PUSH RO
TST RLFLAG ;ARE WE RELOCATED?
BEQ 1\$;NO - SKIP
BIS #BIT15, #PATMAR ;YES - SET MSB
1\$: MOV #PATMAR, @DISPLAY
MOV #PATMAR, DISPREG ;SOFTWARE DISPLAY REGISTER
POP RO
MOV (SP), .RO
RETURN

```

13217 .SBTTL ROUTINE ERROR HANDLER
13218 ;*****
13219 ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.
13220 ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
13221 ;*AND GO TO $ERRTYP ON ERROR
13222 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
13223 ;*SW15=1 HALT ON ERROR
13224 ;*SW13=1 INHIBIT ERROR TYPEOUTS
13225 ;*SW10=1 BELL ON ERROR
13226 ;*SW9=1 LOOP ON ERROR
13227 ;*CALL
13228 ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
13229
13230 .ENABL LSB
13236
13237 051410 005737 003762 $ERROR: TST UFDPLG ;;K ARE WE IN UFD MODE
13238 051414 001403 BEQ KAL ;;K IF NOT DO NOT TRY ABORT
13239 051416 005000 CLR RO ;;K ELSE CLEAR CNTRL Z AND C AND
13240 051420 004737 052220 JSR PC,ABORT ;;K CHECK FOR UFD ABORT
13241 051424 105037 052360 KAL: CLRIB IBSAVE ;R-C
13242 051430 IF NOERROR IS FALSE
13243 051430 005737 002430 TST NOERROR
13244 051434 001027 BNE L470
13243 051436 104410
13244 051440
13245 051440 105237 002014 BACK: CKSWR ;;TEST FOR CHANGE IN SOFT-SWR ;R-C
13246 051444 001775 1#: INCB $ERFLG ;;SET THE ERROR FLAG
13247 051446 004737 051336 BEQ 1# ;;DON'T LET THE FLAG GO TO ZERO
13248 051452 013737 002012 056722 CALL GETDIS ;;SETUP DISPLAY STUFF
13249 051460 032777 002000 131150 MOV $PATMAR,$TESTN ;FOR APT
13250 051466 001404 BIT $SW10,$SWR ;;BELL ON ERROR?
13251 051470 051470 104401 002653 BEQ 2# ;;NO - SKIP
13252 051474 104401 066130 TYPE $BELL ;;RING BELL
13253 051500 005237 002630 TYPEIT $BELL
13254 051504 100003 .DSABL CRF
13255 051506 012737 077777 002630 TYPE MSG014 ;CONTROL Z
13256 051514 051514 .DSABL CRF
13257 051514 051514 2#: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
13258 051514 011637 002020 IF RESULT IS MI BPL L471
13259 051520 162737 000002 002020 MOV $77777,$ERTTL
13260 051526 010637 002024 END ;OF IF RESULT
13261 051532 016637 000002 002030 END ;OF IF NOERROR L471:;;;;;
13262 051540 117737 130254 002015 MOV (SP),ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
13263 051546 122737 000177 002015 SUB #2,ERRPC
13264 051554 001431 MOV SP,ERRSP
13265 051556 105737 052360 MOV 2(SP),ERRPSW
13266 051562 001024 MOV BERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
13267 051564 005737 052364 CMPB #177,$ITEMB ;IS THIS THE POWER FAIL CALL? ;R-C
13268 051570 001423 BEQ 1001# ;BRANCH IF SO ;R-C
13269 051570 001423 TSTB IBSAVE ;2ND ERROR CALL? ;R-C
13269 051570 001423 BNE 1000# ;BRANCH IF SO ;R-C
13269 051570 001423 TST CPERRF ;IS THERE A CPU ERROR REGISTER? ;R-C
13269 051570 001423 BEQ 1001# ;BRANCH IF NOT ;R-C

```

```

13270 051572 013737 177766 052362      MOV    177766,CPSAVE    ;SAVE CONTENTS                ;R-C
13271 051600 032737 000001 052362      BIT    #BIT0,CPSAVE    ;POWER MONITOR BIT SET?      ;R-C
13272 051606 001414                BEQ    1001#           ;BRANCH IF NOT                ;R-C
13273 051610 042737 000001 177766      BIC    #BIT0,177766    ;CLEAR THE BIT                ;R-C
13274 051616 112737 002015 052360      MOVB   #ITEMB,IBSAVE   ;MAKE IBSAVE NON-ZERO FOR DUAL CALL ;R-C
13275 051624 112737 000177 002015      MOVB   #177,#ITEMB    ;SET #ITEMB TO POWER FAIL POINTER ;R-C
13276 051632 000402                BR     1001#           ;R-C
13277 051634 105037 052360      CLRB   IBSAVE         ;R-C
13278 051640                1000#:                ;R-C
13279 051640                1001#:                ;R-C
                                IF NOERROR IS FALSE
                                TST NOERROR
                                BNE L472
13280 051646                IF BADPC NE #0
                                TST BADPC
                                BEQ L473
                                051646 005737 002022
                                051652 001416
13281 051654 013737 002022 002020      MOV    BADPC,ERRPC    ;
13282 051662 162737 000002 002020      SUB    #2,ERRPC       ;
13283 051670 013737 002026 002024      MOV    BADSP,ERRSP    ;
13284 051676 013737 002032 002030      MOV    BADPSW,ERRPSW ;
13285 051704 C05037 002022      CLR    BADPC         ;
13286 051710                END ;IF
                                L473:;;;;;;
13287 051710 013737 002020 056720      MOV    ERRPC,#FATAL   ;FOR APT
13288 051716 004737 050354      CALL   PERBNK        ;:LOG ERROR ON BANK
13289 051722                IF #SW13 SET.IN @SWR
                                BIT #SW13,@SWR
                                BEQ L474
                                051722 032777 020000 130706
                                051730 001401
13290 051732 000420                BR     3#
13291 051734                END ;OF IF #SW13
                                L474:;;;;;;
13292 051734                IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE
                                BIT #SW5,@SWR
                                BEQ L475
                                051734 032777 000040 130674
                                051742 001404
                                051744 005737 002406
                                051750 001401
13293 051752                GOTO 3#
                                BR 3#
13294 051754                END ;OF IF #SW5
                                L475:;;;;;;
13295 051754                END ;OF IF NOERROR
                                L472:;;;;;;
13296 051754 004737 052366      CALL   #ERRTYP        ;:GO TO USER ERROR ROUTINE
13297 051760                IF MONFLG IS TRUE    ;:SHOULD WE RETURN TO XXDP MONITOR???
                                TST MONFLG
                                BEQ L476
                                051760 005737 002276
                                051764 001403
13298 051766 013706 002274      MOV    SAVMON,SP      ;:GET MONITOR ADDRESS
13299 051772 000207      RTS    PC             ;:GO TO MONITOR
13300 051774                END
                                ;
                                L476:;;;;;;

```

```

13302 051774 005737 002430      3#: IF NOERROR IS FALSE
                                TST NOERROR
                                BNE L477
13303 052000 001072 130630      TST BSWR          ;;HALT ON ERROR
13304 052006 100002 130630      BPL 7#           ;;SKIP IF CONTINUE
13305 052010 000000 130630      #HALT: HALT      ;;HALT ON ERROR!
13306 052012 104410 130630      CKSWR          ;;TEST FOR CHANGE IN SOFT SWR
13307 052014 005737 002440      7#: IF NOSCOPE IS FALSE AND #SW9 SET.IN BSWR
                                TST NOSCOPE
                                BNE L500
                                BIT #SW9,BSWR
                                BEQ L500
13308 052032 013716 002624      MOV #LPERR,(SP)  ;;FUDGE RETURN FOR LOOPING
13309 052036 013716 002624      END ;OF IF NOSCOPE
                                L500:;;;;;
13310 052036 005737 002362      TST #ESCAPE     ;;CHECK FOR AN ESCAPE ADDRESS
13311 052042 001402 002362      BEQ 9#         ;;BR IF NONE
13312 052044 013716 002362      MOV #ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
13313 052050 005737 002220      9#: IF DETFLAG IS FALSE
                                TST DETFLAG
                                BNE L501
13314 052056 022737 000005 004064      CMP #5,PROTYP  ;IS THIS AN 11/83 ?
13315 052064 001002 000005 004064      BNE 11#
13316 052066 005037 177766      CLR CPUERR
13317 052072 005737 002350      11#: IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL# IS TRUE
                                TST ACTFLAG
                                BNE L502
                                TST APTFLAG
                                BNE L502
                                TST FATAL#
                                BEQ L503
                                L502:;;;;;
13318 052114 012737 000001 056716      MOV #1,#MSGTY  ;FOR APT
13319 052122 000137 040512      JMP EXIT
13320 052126 000137 040512      END ;OF IF ACTFLAG
                                L503:;;;;;
13321 052126 005737 002354      IF XXDPCHAIN IS TRUE AND #ERTTL HI #20
                                TST XXDPCHAIN
                                BEQ L504
                                CMP #ERTTL,#20
                                BLOS L504
13322 052144 104401 070121      TYPE MSG066      ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN
                                TYPEIT ,MSG066
                                .DSABL CRF
                                MOV 42,R0
                                CLR 42
                                JMP #ZAP42
                                END ;OF IF XXDPCHAIN
                                L504:;;;;;
13323 052150 013700 000042      END ;OF IF DETFLAG
                                L501:;;;;;
13324 052154 005037 000042      ELSE
13325 052160 000137 013710      SET HEADER
                                L477:;;;;;
13326 052164 000403 000403      MOV #-1,HEADER
13327 052164 000403 000403
13328 052164 000403 000403
13329 052166 012737 177777 002612      SET HEADER
13330 052174 000403 000403      END ;OF IF NOERROR
  
```

```

                                L505:|||||
13331 052174                    104:  CLEAR  TOOMANY,NOERROR                CLR  TOOMANY
                                052174                    CLR  NOERROR
                                052200 005037 002406
13332 052204 105737 052360      TSTB   IBSAVE                    ;POWER FAIL ERROR CALL? ;R-C
13333 052210 001402              BEQ    2134                    ;R-C
13334 052212 000137 051440      JMP    BACK                    ;R-C
13335 052216 000002              RTI     ;RETURN
13336
13337 052220                    #ABORT
                                .SBTTL  ABORT ROUTINE FOR LCP/ORION UFD MODE

052220 005737 003762          ABORT:  TST   UFDFLG                ;TEST FOR USER FRIENDLY MODE
052224 001454                  BEQ    NOABRT                ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
052226 020027 000032          CMP    RO,#32                ;IS IT A 'Z' ?
052232 001443                  BEQ    ABORTZ                ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
052234 020027 000003          CMP    RO,#3                 ;IS IS A 'C' ?
052240 C01404                  BEQ    ABORTC                ;BR TO LOAD 'C ON XXDP+ STACK (NO ERROR)
052242 005737 003764          TST   UQUIET                ;TEST FOR USER-QUIET MODE
052246 001443                  BEQ    NOABRT                ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
                                ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
052250 000422                  BR     ABORTE                ;SET DRSERR THEN LEAVE

052252 013737 003756 000030  ABORTC:  MOV    SAV30,30          ;RESTORE EMT LOCATION (30)
052260 013737 003760 000032  MOV    SAV32,32          ;RESTORE EMT PRIORITY LOCATION (32)
052266 104043                  EMT    +43                ;GET XXDP STACK LOC. INTO RO FROM MONITOR
052270 105720                    1004:  TSTB   (RO)+                ;FIND END OF STACK
052272 001376                  BNE    1004
052274 112760 000057 177777  MOVB   #' /,-1(R0)        ;LOAD SLASH OVER ZERO
052302 112720 000136          MOVB   #' ,(R0)+          ;LOAD UPARROW
052306 112720 000103          MOVB   #' C,(R0)+        ;LOAD C
052312 105010                  CLRB   (R0)                ;MAKE NEW END TO STACK
052314 000412                  BR     ABORTZ                ;NOW LEAVE
052316 013737 003756 000030  ABORTE:  MOV    SAV30,30          ;RESTORE EMT LOCATION (30)
052324 013737 003760 000032  MOV    SAV32,32          ;RESTORE EMT PRIORITY LOCATION (32)
052332 104042                  EMT    +42                ;GET DCA LOCATION INTO RO FROM MONITOR
052334 012760 177777 000042  MOV    #-1,42(R0)        ;SET A -1 INTO LOCATION DRSERR IN MONITOR
052342 013700 000042          ABORTZ:  MOV    0042,RO      ;AND PUT THE MONITOR RETURN ADDRESS IN RO
052346 005037 000042          CLR    0042                ;CLEAR MONITOR RETURN FLAG
052352 000137 013730          JMP    $ENDAD              ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
052356 000207          NOABRT:  RTS    PC          ;IF NOTUFD RETURN TO MAINLINE

13338 052360 000000          IBSAVE: .WORD 0                ;R-C
13339 052362 000000          CPSAVE: .WORD 0                ;R-C
13340 052364 000000          CPERRF: .WORD 0                ;R-C
13341                                .DSABL  LSB

```

.SBTTL ROUTINE ERROR MESSAGE TYPEOUT

13344
13345
13346
13347
13348
13349
13350

; *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.

13351 052366 104415
13352 052370 104401 002660
052370

%ERPTYP: SAVREG
TYPE %CRLF ;;"CARRIAGE RETURN" & "LINE FEED"

13353 052374 005000
13354 052376 153700 002015
13355 052402 001004

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

13356 052404 013746 002020
052404

TYPOCT ERRPC,<ERROR ADDRESS>
MOV ERRPC,-(SP) ;;SAVE ERRPC FOR TYPEOUT
;;ERROR ADDRESS
;;GO TYPE--OCTAL ASCII(ALL DIGITS)

052410 104402

TYPOC
.DSABL CRF

13358 052412 000511

BR 11% ;;GET OUT

13359 052414 122700 000177

1%: CMPB %177,RO ;;POWER MONITOR CALL? ;R-C

13360 052420 001003

BNE 100% ;;BRANCH IF NOT ;R C

13361 052422 012700 052676

MOV %PFECWS,RO ;;MOV ADDRESS OF PFE BIT ERROR TO RO ;R C

13362 052426 000406

BR 110% ;;ADJUST THE INDEX SO THAT IT WILL ;R C

13363 052430 005300

100%: DEC RO ;;ADJUST THE INDEX SO THAT IT WILL ;R-C

13364 052432 006300

ASL RO ;; WORK FOR THE ERROR TABLE

13365 052434 006300

ASL RO

13366 052436 006300

ASL RO

13367 052440 062700 057344

ADD %ERRTB,RO ;;FORM TABLE POINTER

13368 052444 012037 052502

110%: MOV (RO)+,3% ;;PICKUP "ERROR MESSAGE" POINTER ;R C

13369 052450 001417

BEQ 4% ;;SKIP TYPEOUT IF NO POINTER

13370 052452 005737 002430

T,T NOERROR ;;IS THIS REALLY AN ERROR?

13371 052456 001003

BNE 12% ;;YES - SKIP

13372 052460 005737 002612

TST HEADER ;;TYPE HEADER?

13373 052464 100011

BPL 4% ;;NO - SKIP

13374 052466 005737 002064

12%: TST FATAL% ;;WAS IT A FATAL ERROR?

13375 052472 001402

BEQ 2% ;;NO SKIP

13376 052474 104401 070170

TYPE MSG067 ;;FATAL

052474

TYPEIT MSG067

13377 052500 104401

2%: .DSABL CRF ;;TYPE THE "ERROR MESSAGE"

052500

TYPE

13378 052502 000000

3%: .WORD 0 ;;"ERROR MESSAGE" POINTER GOES HERE

13379 052504 104401 002660

TYPE %CRLF ;;"CARRIAGE RETURN" & "LINE FEED"

052504

TYPEIT %CRLF

13380 052510 012037 052534

4%: .DSABL CRF ;;PICKUP "DATA HEADER" POINTER

13381 052514 001412

MOV (RO)+,5% ;;SKIP TYPEOUT IF 0

13382 052516 005737 002430

BEQ 6% ;;IS THIS REALLY AN ERROR?

13383 052522 001003

TST NOERROR ;;YES - SKIP

13384 052524 005737 002612

BNE 13% ;;TYPE HEADER?

13385 052530 100004

TST HEADER ;;TYPE HEADER?

13386 052532 104401

13%: BPL 6% ;;NO - SKIP

052532

TYPE ;;TYPE THE "DATA HEADER"

TYPEIT

.DSABL CRF

13387 052534 000000
13388 052536 104401 002660
052536
13389 052542 012001
13390 052544 001427
13391 052546 012002

```
58: .WORD 0 ;; "DATA HEADER" POINTER GOES HERE
      TYPE %CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
      TYPEIT %CRLF
      .DSABL CRF
68: MOV (R0),R1 ;; PICKUP "DATA TABLE" POINTER
      BEQ 108 ;; BR IF NO DATA TO BE TYPED
      MOV (R0),R2 ;; PICKUP "DATA FORMAT" POINTER
```

```

13394 052550 112203          7#:  MOVB  (R2),R3
13395 052552 006303          ASL   R3           ;MAKE IT A WORD ADDRESS
13396 052554 004773 052562    CALL  B8,(R3)
13397 052560 000412          BR    9#
13398 052562 053006          8#:  TAG70#
13399 052564 053016          TAG71#
13400 052566 053026          TAG72#
13401 052570 053076          TAG73#
13402 052572 053136          TAG74#
13403 052574 053150          TAG75#
13404 052576 053162          TAG76#
13405 052600 053226          TAG77#
13406 052602 053234          TAG78#
13407 052604 053314          TAG79#
13412 052606 062701 00000?    9#:  ADD   #2,R1       ;UPDATE DATA TABLE POINTER
13413 052612 005711          TST  (R1)         ;;IS THERE ANOTHER NUMBER?
13414 052614 001403          BEQ  10#         ;;BR IF NO
13415 052616          TYPE  MSG018     ;TYPE 2 SPACES
      052616 104401 066157          TYPEIT ,MSG018
      .DSABL CRF
      BR    7#           ;;LOOP
13416 052622 000752
13417
13418 052624 005737 002110    10#: TST  MUT         ;IS THERE A MEMORY UNDER TEST
13419 052630 001402          BEQ  11#         ;NO - SKIP
13420 052632 005237 002612    INC  HEADER      ;YES - BUMP HEADER FLAG
13421 052636 104416          11#: RESREG
13422 052640          IF #SW7 SET.IN @SWR AND DETFLAG IS FALSE AND NOERROR IS FALSE
      052640 032777 000200 127770    BIT  #SW7,@SWR
      052646 001410          BEQ  L506
      052650 005737 002220    TST  DETFLAG
      052654 001005          BNE  L506
      052656 005737 002430    TST  NOERROR
      052662 001002          BNE  L506
13423 052664 004737 053336          CALL  DETAIL
13424 052670          END ;OF IF #SW7
      052670          TYPE  MSG104     ;CONTROL Z
13425 052670 104401 070627          TYPEIT ,MSG104
      .DSABL CRF
      RETURN
      .EVEN
13426 052674 000207          .WORD  PFECM,PFECDH,PFECDT,PFECDF ;R C
13427
13428 052676 052706 052742 052772  PFECWS: .WORD
13429 052706 120 117 127  PFECM: .ASCIZ "POWER MONITOR BIT FOUND SET" ;R C
      052711 105 122 040
      052714 115 117 116
      052717 111 124 117
      052722 122 040 102
      052725 111 124 040
      052730 106 117 125
      052733 116 104 040
      052736 123 105 124
      052741 000
13430 052742 124 105 123  PFECDH: .ASCIZ "TESTNO ERR PC CPUERR" ;R C
      052745 124 116 117
      052750 040 040 105
      052753 122 122 040

```



```

13437 ;*****
13438 ;*** OCTAL ***
13439 ;*****
13440 053006 TAG70$: TYPOCT @R1) ;;TYPE AN OCTAL NUMBER
      053006 017146 000000 MOV @R1),-(SP) ;;SAVE @R1) FOR TYPEOUT
      053012 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
      RETURN

13441 053014 000207
13442
13443 ;*****
13444 ;*** DECIMAL ***
13445 ;*****
13446 053016 TAG71$: TYPDEC @R1) ;;TYPE A DECIMAL NUMBER
      053016 017146 000000 MOV @R1),-(SP) ;;SAVE @R1) FOR TYPEOUT
      053022 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
      RETURN

13447 053024 000207
13448
13449 ;*****
13450 ;*** INTERLEAVE ***
13451 ;*****
13452 053026 TAG72$: PUSH R1,R5
      053026 010146 MOV R1,-(SP)
      053030 010546 MOV R5,-(SP)
13453 053032 013701 002102 MOV BANK,R1
13454 053036 070127 000004 MUL #4,R1
13455 053042 012737 177777 002372 SET NOTAB ;INDICATE NO TABLE TO BE PRINTED - NOW
      MOV #-1,NOTAB
13456 053050 104401 066130 TYPE MSG014
      TYPEIT ,MSG014
      .DSABL CRF
      CALL TCFIG1
      CLR NOTAB
      POP R5,R1
13457 053054 004737 033272
13458 053060 005037 002372
13459 053064 012605 MOV (SP)+,R5
      053066 012601 MOV (SP)+,R1
13460 053070 104401 066130 TYPE MSG014 ;1 SPACE
      TYPEIT ,MSG014
      .DSABL CRF
      RETURN

13461 053074 000207
13462
13463 ;*****
13464 ;*** CSR ***
13465 ;*****
13466 053076 TAG73$: PUSH R1,R5
      053076 010146 MOV R1,(SP)
      053100 010546 MOV R5,-(SP)
13467 053102 013701 002102 MOV BANK,R1
13468 053106 070127 000004 MUL #4,R1
13469 053112 012737 177777 002372 SET NOTAB MOV #1,NOTAB
13470 053120 004737 033466 CALL TCFIG3
13471 053124 005037 002372 CLR NOTAB
13472 053130 012605 MOV (SP)+,R5
      053132 012601 MOV (SP)+,R1
13473 053134 000207 RETURN
  
```

13474
13475
13476
13477
13478 053136 013746 002300
053142 104403
053144 002
053145 001
13479 053146 000207
13480
13481
13482
13483
13484 053150 013746 002102
053154 104403
053156 003
053157 000
13485 053160 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
```

```
13487 ;*****
13488 ;*** MTYPE ***
13489 ;*****
13490 TAG76$: PUSH R1,R5
                                MOV R1,-(SP)
                                MOV R5,-(SP)
                                053162 010146
                                053164 010546
13491 053166 013701 002102      MOV BANK,R1
13492 053172 070127 000004      MUL #4,R1
13493 053176 012737 177777 002372 SET NOTAB
                                MOV #-1,NOTAB
13494 053204 104401 066162      TYPE MSG019
                                TYPEIT ,MSG019
                                .DSABL CRF
13495 053210 004737 033332      CALL TCFIG2
13496 053214 005037 002372      CLR NOTAB
13497 053220 012605
                                POP R5,R1
                                MOV (SP),R5
                                053220 012601
                                053222 012601
                                MOV (SP),R1
13498 053224 000207      RETURN
13499
13500 ;*****
13501 ;*** UNKNOWN DATA ***
13502 ;*****
13503 TAG77$: TYPE MSG061
                                TYPEIT ,MSG061
                                .DSABL CRF
13504 053226 104401 070037
                                RETURN
13505
13506 ;*****
13507 ;*** PHYSICAL ADDRESS ***
13508 ;*****
13509 TAG78$: MOV ADDRESS,PHYADD
13510 053234 013737 002034 002040 SUB #FIRST,PHYADD
13511 053242 162737 060000 002040 MOV BANK,PHYADD+2
13512 053250 013737 002102 002042 ASR PHYADD+2
13513 053256 006237 002042 BCC 1$
13514 053262 103003 BIS #BIT15,PHYADD
13515 053264 052737 100000 002040 1$: MOV #PHYADD,-(SP)
13516 053272 012746 002040 CALL #DB20 ;POINTER TO DOUBLE WORD ON STACK
13517 053276 004737 056576 ADD #2,SP ;CALL DOUBLE PRECISION CONVERSION ROUTINE
13518 053302 062706 000002 TYPE #OCT8 ;FIX STACK
13519 053306 104401 056704 TYPEIT ,#OCT8
                                .DSABL CRF
                                RETURN
13520
13521 ;*****
13522 ;*** OCTAL BYTE ***
13523 ;*****
13524 TAG79$: TYPE MSG018 ;2 SPACES
                                TYPEIT ,MSG018
                                .DSABL CRF
13525 053314 104401 066157 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
```

13526 053330
053330 104401 066130
13527 053334 000207

.DSABL CRF
TYPE MSG014 ;SPACE
TYPEIT ,MSG014
.DSABL CRF
RETURN

13571 053336

DETAIL: SUBTST <<SUBR DETAILED ERROR REPORT>>

 ;*SUBTEST SUBR DETAILED ERROR REPORT

13572	053336	005237	002220		INC	DETFLAG		
13573	053342	022737	000003	002220	CMP	#3,DETFLAG		
13574	053350	101473			BLOS	4#		
13575	053352	022737	000002	002220	CMP	#2,DETFLAG		
13576	053360	001435			BEQ	2#		
13577	053362				PUSH	HEADER,MUT		
	053362	013746	002612					MOV HEADER,-(SP)
	053366	013746	002110					MOV MUT,-(SP)
13578	053372				SET	HEADER		
	053372	012737	177777	002612				MOV #-1,HEADER
13579	053400	005037	002110		CLR	MUT		
13580	053404	010037	002200		MOV	RO,DETRO		
13581	053410	012700	002202		MOV	#DETR1,RO		
13582	053414	010120			MOV	R1,(RO)+		
13583	053416	010220			MOV	R2,(RO)+		
13584	053420	C10320			MOV	R3,(RO)+		
13585	053422	010420			MOV	R4,(RO)+		
13586	053424	010520			MOV	R5,(RO)+		
13587	053426	013720	002024		MOV	ERRSP,(RO)+		
13588	053432	013720	002030		MOV	ERRPSW,(RO)+		
13589	053436	013700	002200		MOV	DETRO,RO		
13590	053442				SET	NOERROR		
	053442	012737	177777	002430				MOV #1,NOERROR
13591	053450	104013			ERROR	+13		
13592	053452	000423			BR	1#		
13593	053454				2#:	PUSH	HEADER,MUT	
	053454	013746	002612					MOV HEADER,(SP)
	053460	013746	002110					MOV MUT,-(SP)
13594	053464				SET	HEADER		
	053464	012737	177777	002612				MOV #-1,HEADER
13595	053472	005037	002110		CLR	MUT		
13596	053476				SET	NOERROR		
	053476	012737	177777	002430				MOV #-1,NOERROR
13597	053504	104031			ERROR	+31		
13598	053506	022737	000005	004064	CMP	#5,PROTYP		:IS THIS AN 11/83 ?
13599	053514	001002			BNE	1#		
13600	053516	005037	177766		CLR	CPUERR		
13601	053522				1#:	POP	MUT,HEADER	
	053522	012637	002110					MOV (SP)+,MUT
	053526	012637	002612					MOV (SP)+,HEADER
13602								
13603	053532	004737	053336					
13604	053536	000207						

;WARNING RECURSIVE
 CALL DETAIL
 RETURN


```

13607 ;SIMULATE CONTROL "T"
13608 053540 004737 055222 44: CALL CONTT ;DISPLAY "DISPLAY" INFO
13609 ;TYPE CONTENTS OF ALL CSR'S
13610 PUSH CSR,CSRNO,R1
13611 053544 013746 002150 MOV CSR,-(SP)
053544 013746 002152 MOV CSRNO,-(SP)
053550 013746 002152 MOV R1,-(SP)
053554 010146
13612 053556 104401 070015 TYPE MSG058
TYPEIT ,MSG058
.DSABL CRF
13613 053562 104401 002660 TYPE $CRLF
TYPEIT $CRLF
.DSABL CRF
13614 053566 013701 002224 MOV TOTCSRS,R1
13615 053572 BEGIN DUMPCSRLOOP
053572 FOR CSRNO := #0 TO #36 BY #2 B102:;;;;;
13616 053572 005037 002152 CLR CSRNO
053576 B103:;;;;;
13617 053576 006301 ASL R1
13618 053600 ON.ERROR BCC L507
053600 103006
13619 053602 104426 READCSR
13620 053604 013746 002150 TYPOCT CSR ;SAVE CSR FOR TYPEOUT
MOV CSR,-(SP) ;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPOC CSR,-(SP)
.DSABL CRF
13621 053612 104401 066157 TYPE MSG018 ;2 SPACES
TYPEIT ,MSG018
.DSABL CRF
13622 053616 END ;OF ON.ERROR L507:;;;;;
053616 IF R1 EQ #0 THEN LEAVE DUMPCSRLOOP
13623 053616 005701 TST R1
053620 001407 BEQ E102
13624 053622 062737 000002 002152 ADD #2,CSRNO
053630 023727 002152 000036 CMP CSRNO,#36
053636 003757 BLE B103
13625 053640 END DUMPCSRLOOP E103:;;;;;
053640 POP R1,CSRNO,CSR E102:;;;;;
13626 053640 012601 MOV (SP)+,R1
053642 012637 002152 MOV (SP)+,CSRNO
053646 012637 002150 MOV (SP)+,CSR
13627 ;TYPE STACKS
13628 PUSH RO,R1
13629 053652 010046 MOV RO,-(SP)
053652 010146 MOV R1,-(SP)
053654 010146
13630 053656 104401 070406 TYPE MSG088 ;KERNEL STACK
TYPEIT ,MSG088
.DSABL CRF
13631 05366. 013701 002574 MOV KSTACK,R1
13632 053666 162701 000002 SUB #2,R1

```

```

13633 053672          FOR      RO := SP TO R1 BY #2
      053672 010600
      053674
13634 053674          TYPE      #CRLF
      053674 104401 002660  TYPEIT  #CRLF
      .DSABL CRF
13635 053700          TYP OCT      RO
      053700 010046  MOV      RO, -(SP)      ;;SAVE RO FOR TYPEOUT
      053702 104402  TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
13636 053704          TYPE      MSG018      ;2 SPACES
      053704 104401 066157  TYPEIT  ,MSG018
      .DSABL CRF
13637 053710          TYP OCT      (RO)
      053710 011046  MOV      (RO), -(SP)      ;;SAVE (RO) FOR TYPEOUT
      053712 104402  TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
13638 053714          END ;OF FOR RO
      053714 062700 000002
      053720 C20001
      053722 003764
      053724
13639
13640 053724 005737 002456      ;SET PREVIOUS MODE TO SUPERVISOR
13641 053730 001036      TST      NOSUPER
13642 053732 042737 030000 177776 BNE      DET1
13643 053740 052737 010000 177776 BIC      #BIT13:BIT12,PSW
13644 053746 006506      BIS      #BIT12,PSW
13645 053750          MFPI      SSP
      053750 012601          POP      R1,RO
      053752 012600
13646 053754          TYPE      MSG089      ;SUPERVISOR STACK
      053754 104401 070424  TYPEIT  ,MSG089
      .DSABL CRF
13647 053760          IF RO LT #SUPSTK
      053760 020027 000740
      053764 002016
13648 053766          FOR RO := RO TO #SUPSTK-2 BY #2
13649 053766          TYPE      #CRLF
      053766 104401 002660  TYPEIT  #CRLF
      .DSABL CRF
13650 053772          TYP OCT      RO
      053772 010046  MOV      RO, -(SP)      ;;SAVE RO FOR TYPEOUT
      053774 104402  TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
13651 053776          TYPE      MSG018      ;2 SPACES
      053776 104401 066157  TYPEIT  ,MSG018
      .DSABL CRF
13652 054002          TYP OCT      (RO)
      054002 011046  MOV      (RO), -(SP)      ;;SAVE (RO) FOR TYPEOUT
      054004 104402  TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL CRF
13653 054006          END ;OF FOR RO
      054006 062700 000002
      054012 020027 000736
      054016 003763

```

MOV SP,RO
B104:::~::~

ADD #2,RO
CMP RO,R1
BLE B104
E104:::~::~

MOV (SP)+,R1
MOV (SP)+,RO

CMP RO,#SUPSTK
BGE L510

B105:::~::~

ADD #2,RO
CMP RO,#SUPSTK-2
BLE B105

```

13654 054020 000402 ELSE
054020 054022 104401 070462 TYPE MSG091 ;IS EMPTY
054022 054022 104401 070462 TYPEIT ,MSG091
.DSABL CRF ;IS EMPTY
END ;OF IF RO
13655 054022 104401 070462
13656 054026 054026 ;SET PREVIOUS MODE TO USER
13657 054026 052737 030000 177776 DET1: BIS #BIT13:BIT12.PSW
13658 054026 052737 030000 177776 DET1: MFPI USP
13659 054034 006506 POP RO
13660 054036 012600 MOV (SP),.RO
054036 054040 104401 070446 TYPE MSG090 ;USER STACK
054040 054040 104401 070446 TYPEIT ,MSG090
.DSABL CRF
IF RO LT #USESTK
13662 054044 020027 000700 CMP RO,#USESTK
054044 020027 000700 BGE L512
054050 C02016
13663 054052 054052 FOR RO := RO TO #USESTK-2 BY #2
054052 054052 104401 002660 TYPE $CRLF B106:
TYPEIT , $CRLF
.DSABL CRF
TYPOCT RO
MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
13665 054056 010046 104402 TYPE MSG018 ;2 SPACES
054056 010046 104402 TYPEIT ,MSG018
.DSABL CRF
TYPOCT (RO)
MOV (RO),-(SP) ;;SAVE (RO) FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
13666 054062 104401 066157 END ;OF FOR RO
054062 104401 066157 ADD #2,RO
054072 062700 000002 CMP RO,#USESTK-2
054072 062700 000002 BLE B106
054076 020027 000676
054102 003763
054104
13669 054104 000402 ELSE
054104 000402
054106
13670 054106 104401 070462 TYPE MSG091 ;IS EMPTY
054106 104401 070462 TYPEIT ,MSG091
.DSABL CRF ;IS EMPTY
END ;OF IF RO
13671 054112
054112
13672 054112 104401 002660 TYPE $CRLF
054112 104401 002660 TYPEIT , $CRLF
.DSABL CRF
13673 054116 005037 002220 CLR DETFLAG
13674 054122 012600 POP RO
13675 054124 000207 RETURN MOV (SP),.RO

```

.SBTTL ROUTINE BINARY TO OCTAL (ASCII) AND TYPE

```

13713
13714
13715
13716
13717
13718
13719
13720
13721
13722
13723
13724
13725
13726
13727
13728
13729
13730
13731
13732
13733
13734
13735
13736
13737
13738 054126 017646 000000
13739 054132 116637 000001 054351
13740 054140 112637 054353
13741 054144 062716 000002
13742 054150 000406
13743 054152 112737 000001 054351
13744 054160 112737 000006 054353
13745 054166 112737 000005 054350
13746 054174 010346
13747 054176 010446
13748 054200 010546
13749 054202 113704 054353
13750 054206 005404
13751 054210 062704 000006
13752 054214 110437 054352
13753 054220 113704 054351
13754 054224 016605 000012
13755 054230 005003
13756 054232 006105
13757 054234 000404
13758 054236 006105
13759 054240 006105
13760 054242 006105
13761 054244 010503
13762 054246 006103
13763 054250 105337 054352
13764 054254 100016
13765 054256 042703 177770
13766 054262 001002
13767 054264 005704
13768 054266 001403
13769 054270 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     8(SP),-(SP)  ;;PICKUP THE MODE
;#         MOVB  1(SP),#OFILL ;;LOAD ZERO FILL SWITCH
;#         MOVB  (SP)+,#OMODE+1 ;;NUMBER OF DIGITS TO TYPE
;#         ADD   #2,(SP)      ;;ADJUST RETURN ADDRESS
;#         BR    #TYPON
;#TYPOC: MOVB  #1,#OFILL     ;;SET THE ZERO FILL SWITCH
;#         MOVB  #6,#OMODE+1 ;;SET FOR SIX(6) DIGITS
;#TYFJN: MOVB  #5,#OCNT     ;;SET THE ITERATION COUNT
;#         MOV   R3,-(SP)    ;;SAVE R3
;#         MOV   R4,-(SP)    ;;SAVE R4
;#         MOV   R5,-(SP)    ;;SAVE R5
;#         MOVB  #OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
;#         NEG   R4
;#         ADD   #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
;#         MOVB  R4,#OMODE  ;;SAVE IT FOR USE
;#         MOVB  #OFILL,R4  ;;GET THE ZERO FILL SWITCH
;#         MOV   12(SP),R5  ;;PICKUP THE INPUT NUMBER
;#         CLR   R3        ;;CLEAR THE OUTPUT WORD
;#         ROL  R5        ;;ROTATE MSB INTO "C"
;#         BR   3#        ;;GO DO MSB
;#         ROL  R5        ;;FORM THIS DIGIT
;#         ROL  R5
;#         ROL  R5
;#         MOV  R5,R3
;#         ROL  R3        ;;GET LSB OF THIS DIGIT
;#         DECB #OMODE    ;;TYPE THIS DIGIT?
;#         BPL  6#        ;;BR IF NO
;#         BIC  #177770,R3 ;;GET RID OF JUNK
;#         ONE  4#        ;;TEST FOR 0
;#         TST  R4        ;;SUPPRESS THIS 0?
;#         BEQ  5#        ;;BR IF YES
;#         INC  R4        ;;DON'T SUPPRESS ANYMORE 0'S
    
```

13770	054272	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
13771	054276	052703	000040	5#:	BIS	#' ,R3	::MAKE ASCII IF NOT ALREADY
13772	054302	110337	054346		MOVB	R3,R#	::SAVE FOR TYPING
13773	054306				TYPE	R#	::GO TYPE THIS DIGIT
	054306	104401	054346		TYPEIT	.R#	
					.DSABL	CRF	
13774	054312	105337	054350	6#:	DECB	\$OCNT	::COUNT BY 1
13775	054316	003347			BGT	2#	::BR IF MORE TO DO
13776	054320	002402			BLT	7#	::BR IF DONE
13777	054322	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
13778	054324	000744			BR	2#	::GO DO THE LAST DIGIT
13779	054326	012605		7#:	MOV	(SP)+,R5	::RESTORE R5
13780	054330	012604			MOV	(SP)+,R4	::RESTORE R4
13781	054332	012603			MOV	(SP)+,R3	::RESTORE R3
13782	054334	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
13783	054342	012616			MOV	(SP)+,(SP)	
13784	054344	000002			RTI		::RETURN
13785	054346	000		8#:	.BYTE	0	::STORAGE FOR ASCII DIGIT
13786	054347	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
13787	054350	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
13788	054351	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
13789	054352	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

```

13791          .SBTTL  ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
13792          ;*****
13793          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
13794          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
13795          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
13796          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
13797          ;*REPLACED WITH SPACES.
13798          ;*CALL:
13799          ;*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
13800          ;*      TYPDS          ;;GO TO THE ROUTINE
13801          ;TYPDS: PUSH      R0,R1,R2,R3,R5
13801          054354          010046          MOV      R0,-(SP)
13801          054354          010146          MOV      R1,-(SP)
13801          054356          010246          MOV      R2,-(SP)
13801          054360          010346          MOV      R3,(SP)
13801          054362          010546          MOV      R5,-(SP)
13802          054366          012746          020200      MOV      #20200,-(SP)          ;;SET BLANK SWITCH AND SIGN
13803          054372          016605          000020      MOV      20(SP),R5          ;;GET THE INPUT NUMBER
13804          054376          100004          BPL      1#          ;;BR IF INPUT IS POS.
13805          054400          C05405          NEG      R5          ;;MAKE THE BINARY NUMBER POS.
13806          054402          112766          000055      000001      MOVB     #'-,1(SP)          ;;MAKE THE ASCII NUMBER NEG.
13807          054410          005000          1#:        CLR      R0          ;;ZERO THE CONSTANTS INDEX
13808          054412          012703          054570      MOV      #DBLK,R3          ;;SETUP THE OUTPUT POINTER
13809          054416          112723          000040      MOVB     #' ,(R3)+          ;;SET THE FIRST CHARACTER TO A BLANK
13810          054422          005002          2#:        CLR      R2          ;;CLEAR THE BCD NUMBER
13811          054424          016001          054560      MOV      #DTBL(R0),R1          ;;GET THE CONSTANT
13812          054430          160105          3#:        SUB      R1,R5          ;;FORM THIS BCD DIGIT
13813          054432          002402          BLT      4#          ;;BR IF DONE
13814          054434          005202          INC      R2          ;;INCREASE THE BCD DIGIT BY 1
13815          054436          000774          BR       3#
13816          054440          060105          4#:        ADD      R1,R5          ;;ADD BACK THE CONSTANT
13817          054442          005702          TST      R2          ;;CHECK IF BCD DIGIT=0
13818          054444          001002          BNE     5#          ;;FALL THROUGH IF 0
13819          054446          105716          TSTB    (SP)          ;;STILL DOING LEADING 0'S?
13820          054450          100407          BMI     7#          ;;BR IF YES
13821          054452          106316          5#:        ASLB    (SP)          ;;MSD?
13822          054454          103003          BCC     6#          ;;BR IF NO
13823          054456          116663          000001      177777      MOVB     1(SP),-1(R3)          ;;YES--SET THE SIGN
13824          054464          052702          000060      6#:        BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
13825          054470          052702          000040      7#:        BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
13826          054474          110223          MOVB     R2,(R3)+          ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
13827          054476          005720          TST     (R0)+          ;;JUST INCREMENTING
13828          054500          020027          000010      CMP     R0,#10          ;;CHECK THE TABLE INDEX
13829          054504          002746          BLT     2#          ;;GO DO THE NEXT DIGIT
13830          054506          003002          BGT     8#          ;;GO TO EXIT
13831          054510          010502          MOV     R5,R2          ;;GET THE LSD
13832          054512          000764          BR      6#          ;;GO CHANGE TO ASCII
13833          054514          105726          8#:        TSTB    (SP)+          ;;WAS THE LSD THE FIRST NON-ZERO?
13834          054516          100003          BPL     9#          ;;BR IF NO
13835          054520          116663          177777      177776      MOVB     -1(SP),-2(R3)          ;;YES--SET THE SIGN FOR TYPING
13836          054526          105013          9#:        CLRB    (R3)          ;;SET THE TERMINATOR
13837          054530          POP      R5,R3,R2,R1,R0
13837          054530          012605          MOV     (SP),R5
13837          054532          012603          MOV     (SP),R3
13837          054534          012602          MOV     (SP),R2
13837          054536          012601          MOV     (SP),R1
13837          054540          012600          MOV     (SP),R0
    
```

Address	Hex	Dec	Hex	Hex	Type	DBLK	Comment
13838	054542		054570		TYPE	\$DBLK	::NOW TYPE THE NUMBER
	054542	104401			TYPEIT	.\$DBLK	
					.DSABL	CRF	
13839	054546	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
13840	054554	012616			MOV	(SP),.(SP)	
13841	054556	000002			RTI		::RETURN TO USER
13842	054560	023420			\$DTBL:	10000.	
13843	054562	001750				1000.	
13844	054564	000144				100.	
13845	054566	000012				10.	
13846	054570	000000	000000	000000	\$DBLK:	.WORD 0.0.0.0	
	054576	000000					

```

13848 .SBTTL ROUTINE TTY INPUT
13849 ;*****
13850 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
13851 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
13852 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
13853 ;*WHEN OPERATING IN TTY FLAG MODE.
13854 .ENABLE LSB
13855 #CKSWR:
13856 054600 005737 047154 TST XOCHAR ;; SOMETHING THERE?
13857 054604 001406 BEQ NOCH ;; GO ON IF NOT
13858 054606 013746 047154 MOV XOCHAR,-(SP) ;; USE IT
13859 054612 005037 047154 CLR XOCHAR
13860 054616 000137 054640 JMP CONTS1
13861 054622 105777 126014 NOCH: TSTB @TKS ;; CHAR THERE?
13862 054626 100136 BPL SWREND ;; IF NO, DON'T WAIT AROUND
13863 054630 117746 126010 MOVB @TKB,-(SP) ;; SAVE THE CHAR
13864 054634 042716 177600 BIC @C177,(SP) ;; STRIP-OFF THE ASCII
13865 054640 022716 000006 CONTS1: CMP @6,(SP) ;; IS IT CONTROL F?
13866 054644 001002 BNE 1# ;; NO SKIP
13867 054646 04737 040762 CALL FIELDSERVICE
13868 054652 022716 000024 1#: CMP @24,(SP) ;; IS IT CONTROL T?
13869 054656 001002 BNE 16# ;; NO - SKIP
13870 054660 004737 055222 CALL CONTT ;; YES - CALL CONTROL T ROUTINE
13871 054664 022716 000003 16#: CMP @3,(SP) ;; IS IT CONTROL C?
13872 054670 001454 BEQ 5# ;; YES EXIT *****NOTE***** STACK IS SCREWED UP!
13873 054672 022716 000023 2#: CMP @23,(SP) ;; IS IT CONTROL S?
13874 054676 001002 BNE 17# ;; NO - SKIP
13875 054700 004737 055276 CALL CONTS ;; YES - CALL CONTROL S ROUTINE
13876 054704 022716 000013 17#: CMP @13,(SP) ;; IS IT CONTROL K?
13877 054710 001005 BNE 6# ;; NO - SKIP
13878 054712 TYPE #CNTLK ;; TYPE A +K
054712 104401 055214 TYPEIT ,#CNTLK
.DSABL CRF
13879 054716 013706 002146 MOV CTLKVEC,SP ;; RESET KSP TO AFTER PATTERN EXEC ROUTINE
13880 054722 000207 RETURN ;; RETURN TO PATTERN EXEC ROUTINE
13881 054724 022737 000176 002636 6#: CMP #SWREG,SWR ;; IS THE SOFT-SWR SELECTED?
13882 054732 001075 BNE CKEND ;; BRANCH IF NO
13883 054734 022716 000007 CMP @7,(SP) ;; IS IT A CONTROL G?
13884 054740 001072 BNE CKEND ;; NO, RETURN TO USER
13885 054742 005737 002062 TST #AUTO ;; ARE WE RUNNING IN AUTO-MODE?
13886 054746 001067 BNE CKEND ;; BRANCH IF YES
13887 054750 TYPE #CNTLG ;; ECHO THE CONTROL-G (+G)
054750 104401 056020 TYPEIT ,#CNTLG
.DSABL CRF
13888 054754 #GTSWR: TYPE #MSWR ;; TYPE CURRENT CONTENTS
054754 104401 056025 TYPEIT ,#MSWR
.DSABL CRF
13889 054760 TYPOCT @SWR ;; OF THE SWR
054760 017746 125652 MOV @SWR,-(SP) ;; SAVE @SWR FOR TYPEOUT
054764 104402 TYPOC @SWR,-(SP) ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
13890 054766 TYPE #MNEW ;; PROMPT FOR NEW SWR
054766 104401 056036 TYPEIT ,#MNEW
.DSABL CRF
13891 054772 3#: CLR -(SP) ;; CLEAR COUNTER
13892 054774 005046 CLR -(SP) ;; THE NEW SWR
13893 054776 105777 125640 4#: TSTB @TKS ;; CHAR THERE?

```



```

13894 055002 100375          BPL      4#           ;;IF NOT TRY AGAIN
13895 055004 117746 125634  MOVB    @TKB,-(SP)   ;;PICK UP CHAR
13896 055010 042716 177600  BIC    @C177,(SP)   ;;MAKE IT 7-BIT ASCII
13897 055014 021627 000003  CMP     (SP),#3     ;;IS IT A CONTROL-C?
13898 055020 001006          BNE     7#           ;;BRANCH IF NOT
13899 055022          5#:  TYPE    $CNTLC    ;;YES, ECHO CONTROL-C (+C)
      055022 104401 056006  TYPEIT  , $CNTLC
      .DSABL CRF
13900 055026 062706 000006  ADD     #6,SP       ;;CLEAN UP STACK
13901 055032 000137 040442  JMP     QUIT        ;;CONTROL-C HALT
13902 055036 021627 000025  7#:  CMP     (SP),#25  ;;IS IT A CONTROL-U?
13903 055042 001005          BNE     9#           ;;BRANCH IF NOT
13904 055044          TYPE    $CNTLU    ;;YES, ECHO CONTROL-U (+U)
      055044 104401 056013  TYPEIT  , $CNTLU
      .DSABL CRF
13905 055050 062706 000006  8#:  ADD     #6,SP       ;;IGNORE PREVIOUS INPUT
13906 055054 000746          BR      3#           ;;LET'S TRY IT AGAIN
13907 055056 021627 000015  9#:  CMP     (SP),#15  ;;IS IT A <CR>?
13908 055062 001024          BNE    13#           ;;BRANCH IF NO
13909 055064 005766 000004  TST    4(SP)        ;;YES, IS IT THE FIRST CHAR?
13910 055070 001403          BEQ    10#          ;;BRANCH IF YES
13911 055072 016677 000002 125536 MOV     2(SP),@SWR   ;;SAVE NEW SWR
13912 055100          10#: IF SWRFLG IS TRUE
      055100 005737 002566          TST SWRFLG
      055104 001403          BEQ L514
13913 055106 062706 000006  ADD     #6,SP       ;;CLEAR UP STACK
13914 055112          ELSE
      055112 000402          BR L515
      055114          L514:;;;;;
13915 055114 062706 000010  ADD     #10,SP      ;;CLEAR UP STACK
13916 055120          END          L515:;;;;;
13917 055120          TYPE    $CRLF     ;;ECHO <CR> AND <LF>
      055120 104401 002660  TYPEIT  , $CRLF
      .DSABL CRF
13918 055124 000002  SWREND: RTI        ;;RETURN

```

13919	055126	062706	000002	CKEND:	ADD	#2,SP	::FIX STACK
13920	055132	000002			RTI		::RETURN
13921	055134	004737	047156	13#:	CALL	\$TYPEC	::ECHO CHAR
13922	055140	021627	000060		CMP	(SP),#60	::CHAR < 0?
13923	055144	002420			BLT	15#	::BRANCH IF YES
13924	055146	021627	000067		CMP	(SP),#67	::CHAR > ??
13925	055152	003015			BGT	15#	::BRANCH IF YES
13926	055154	042726	000060		BIC	#60,(SP)+	::STRIP-OFF ASCII
13927	055160	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
13928	055164	001403			BEQ	14#	::BRANCH IF YES
13929	055166	006316			ASL	(SP)	::NO, SHIFT PRESENT
13930	055170	006316			ASL	(SP)	:: CHAR OVER TO MAKE
13931	055172	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
13932	055174	005266	000002	14#:	INC	2(SP)	::KEEP COUNT OF CHAR
13933	055200	056616	177776		BIS	-2(SP),(SP)	::SET IN NEW CHAR
13934	055204	000674			BR	4#	::GET THE NEXT ONE
13935	055206			15#:	TYPE	\$QUES	::TYPE ?<CR><LF>
	055206	104401	002657		TYPEIT	,\$QUES	
					.DSABL	CRF	
13936	055212	C00716			BR	8#	::SIMULATE CONTROL-U
13937	055214	136	113	015	\$CNTLK:	.ASCIZ /+K/<15><12>	::CONTROL K ASCII STRING
	055217	012	000				
13938					.EVEN		
13939					.DSABL	LSB	

```
13942 055222          CONTT: SUBTST <<CONTROL T>>
;*****
;SUBTEST          CONTROL T
;*****
13943 055222          PUSH          RO
055222 010046          MOV RO,-(SP)
13944 055224          TYPE          $CRLF
055224 104401 002660  TYPEIT         $CRLF
                                .DSABL CRF
13954 055230          IF RLFLAG IS TRUE
055230 005737 002126          TST RLFLAG
055234 001402          BEQ L516
13955 055236          TYPE          MSG092          ;RELOCATED
055236 104401 070474  TYPEIT         ,MSG092
                                .DSABL CRF
13956 055242          END ;OF IF RLFLAG
055242          L516:::::
13957 055242          TYPE          MSG093          ;BANK=
055242 104401 070510  TYPEIT         ,MSG093
                                .DSABL CRF
13958 055246          TYPOCS          BANK,,3          ;TYPE 3 DIGITS
055246 013746 002102  MOV          BANK,-(SP)          ;;SAVE BANK FOR TYPEOUT
055252 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
055254 003          .BYTE          3          ;;TYPE 3 DIGIT(S)
055255 000          .BYTE          0          ;;SUPPRESS LEADING ZEROS
                                .DSABL CRF
13959 055256          TYPE          MSG095          ;PAT=
055256 104401 070516  TYPEIT         ,MSG095
                                .DSABL CRF
13960 055262          TYPOCS          REALPAT,,2          ;TYPE 2 DIGITS
055262 013746 002300  MOV          REALPAT,-(SP)          ;;SAVE REALPAT FOR TYPEOUT
055266 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
055270 002          .BYTE          2          ;;TYPE 2 DIGIT(S)
055271 000          .BYTE          0          ;;SUPPRESS LEADING ZEROS
                                .DSABL CRF
13964 055272          POP          RO
055272 012600          MOV (SP)+,RO
13965 055274 000207          RETURN
13966
```

13967 055276

```
CONTS: SUBST <<CONTROL S & CONTROL Q>>  
;*****  
;*SUBTEST CONTROL S & CONTROL Q  
;*****
```

13968 055276 012600
13969 055300 105777 125336
13970 055304 100375
13971 055306 117716 125332
13972 055312 042716 177600
13973 055316 021627 000021
055322 001003
13974 055324 000137 054640
13975 055330 000401
055332
13976 055332 C00762
13977 055334
055334

```
POP R0 ;GET RID OF RETURN ADDRESS FROM STACK  
MOV (SP),R0  
CONTS2: TSTB B#TKS ;WAIT FOR CHARACTER  
BPL CONTS2  
MOVB B#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 L517  
JMP CONTS1  
ELSE  
BR CONTS2  
END ;OF IF (SP)  
L517:::~::~  
L520:::~::~
```

```

13979 ;*****
13980 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
13981 ;*CALL:
13982 ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
13983 ;*      RETURN HERE    ;;CHARACTER IS ON THE STACK
13984 ;*                    ;;WITH PARITY BIT STRIPPED OFF
13985 ;
13986
13987 055334 011646 $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC
13988 055336 016666      MOV      4(SP),2(SP)      ;;SAVE THE PS
13989 055344 105777 125272 1#:  TSTB    @TKS          ;;WAIT FOR
13990 055350 100375      BPL      1#          ;;A CHARACTER
13991 055352 117766 125266 000004      MOVB    @TKB,4(SP)      ;;READ THE TTY
13992 055360 042766 177600 000004      BIC     @+C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
13993 055366 026627 000004 000023      CMP     4(SP),@23     ;;IS IT A CONTROL-S?
13994 055374 001013      BNE     3#          ;;BRANCH IF NO
13995 055376 105777 125240 2#:  TSTB    @TKS          ;;WAIT FOR A CHARACTER
13996 055402 100375      BPL      2#          ;;LOOP UNTIL ITS THERE
13997 055404 117746 125234      MOVB    @KB,-(SP)     ;;GET CHARACTER
13998 055410 C42716 177600      BIC     @+C177,(SP)  ;;MAKE IT 7-BIT ASCII
13999 055414 022627 000021      CMP     (SP)+,@21    ;;IS IT A CONTROL-Q?
14000 055420 001366      BNE     2#          ;;IF NOT DISCARD IT
14001 055422 000750      BR      1#          ;;YES, RESUME
14002 055424 026627 000004 000021 3#:  CMP     4(SP),@21    ;;IS IT A RANDOM CONTROL-Q?      ;R-C
14003 055432 001744      BEQ     1#          ;;BRANCH BACK IF SO            ;R-C
14004 055434 026627 000004 000140      CMP     4(SP),@140  ;;IS IT UPPER CASE?
14005 055442 002407      BLT     4#          ;;BRANCH IF YES
14006 055444 026627 000004 000175      CMP     4(SP),@175  ;;IS IT A SPECIAL CHAR?
14007 055452 003003      BGT     4#          ;;BRANCH IF YES
14008 055454 042766 000040 000004      BIC     @40,4(SP)   ;;MAKE IT UPPER CASE
14009 055462 000002 4#:  RTI          ;;GO BACK TO USER
14010 ;*****
14011 ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
14012 ;*CALL:
14013 ;*      RDLIN          ;;INPUT A STRING FROM THE TTY
14014 ;*      RETURN HERE    ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
14015 ;*                    ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
14016 055464 010346 $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
14017 055466 005046      CLR     -(SP)       ;;CLEAR THE RUBOUT KEY
14018 055470 012703 055762 1#:  MOV     @TTYIN,R3     ;;GET ADDRESS
14019 055474 022703 056006 2#:  CMP     @TTYIN+20.,R3  ;;BUFFER FULL?
14020 055500 101477      BLOS    8#          ;;BR IF YES
14021 055502 104411      RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
14022 055504 112613      MOVB    (SP)+,(R3)   ;;GET CHARACTER
14023 055506 122713 000003      CMPB   @3,(R3)     ;;IS IT A CONTROL-C?
14024 055512 001016      BNE     3#          ;;BRANCH IF NO
14025 055514 055514 104401 056006      TYPE   %CNTLC      ;;TYPE A CONTROL C (+C)
14026 055520 005726      TYPEIT ,%CNTLC
14027 055522 012603      .DSABL CRF
14028 055524 032777 000400 125104      TST     (SP)+       ;;CLEAN RUBOUT KEY OFF OF THE STACK
14029 055532 001404      MOV     (SP)+,R3    ;;RESTORE R3
14030 055534 005037 002420      BIT     @BIT8,@SWR  ;;IS THERE A HALT FLAG SET IN THE SWR?
14031 055540 000137 040512      BEQ     11#        ;;BRANCH IF NOT TO HALT ROUTINE
14032 055544 000137 040442 11#:  CLR     STOPOK      ;;GET READY TO HALT PROGRAM
14033 055550 122713 000177 3#:  JMP     EXIT        ;;GO HALT PROGRAM
14033 055550 122713 000177 3#:  JMP     QUIT       ;;GOTO CONTROL-C HALT
14033 055550 122713 000177 3#:  CMPB   @177,(R3)   ;;IS IT A RUBOUT

```

14034	055554	001022			BNE	5#	::BR IF NO
14035	055556	005716			TST	(SP)	::IS THIS THE FIRST RUBOUT?
14036	055560	001007			BNE	4#	::BR IF NO
14037	055562	112737	000134	055760	MOVB	#'\,10#	::TYPE A BACK SLASH
14038	055570				TYPE	10#	
	055570	104401	055760		TYPEIT	,10#	
					.DSABL	CRF	
14039	055574	012716	177777		MOV	#-1,(SP)	::SET THE RUBOUT KEY
14040	055600	005303		4#:	DEC	R3	::BACKUP BY ONE
14041	055602	020327	055762		CMP	R3,#TTYIN	::STACK EMPTY?
14042	055606	103434			BLO	8#	::BR IF YES
14043	055610	111337	055760		MOVB	(R3),10#	::SETUP TO TYPEOUT THE DELETED CHAR.
14044	055614				TYPE	10#	::GO TYPE
	055614	104401	055760		TYPEIT	,10#	
					.DSABL	CRF	
14045	055620	000725			BR	2#	::GO READ ANOTHER CHAR.
14046	055622	005716		5#:	TST	(SP)	::RUBOUT KEY SET?
14047	055624	001406			BEQ	6#	::BR IF NO
14048	055626	112737	000134	055760	MOVB	#'\,10#	::TYPE A BACK SLASH
14049	055634				TYPE	10#	
	055634	104401	055760		TYPEIT	,10#	
					.DSABL	CRF	
14050	055640	005016			CLR	(SP)	::CLEAR THE RUBOUT KEY
14051	055642	122713	000025	6#:	CMPB	#25,(R3)	::IS CHARACTER A CTRL U?
14052	055646	001003			BNE	7#	::BR IF NO
14053	055650				TYPE	#CNTLU	::TYPE A CONTROL "U"
	055650	104401	056013		TYPEIT	,#CNTLU	
					.DSABL	CRF	
14054	055654	000705			BR	1#	::GO START OVER
14055	055656	122713	000022	7#:	CMPB	#22,(R3)	::IS CHARACTER A "R"?
14056	055662	001011			BNE	9#	::BRANCH IF NO
14057	055664	105013			CLRB	(R3)	::CLEAR THE CHARACTER
14058	055666				TYPE	#CRLF	::TYPE A "CR" & "LF"
	055666	104401	002660		TYPEIT	,#CRLF	
					.DSABL	CRF	
14059	055672				TYPE	#TTYIN	::TYPE THE INPUT STRING
	055672	104401	055762		TYPEIT	,#TTYIN	
					.DSABL	CRF	
14060	055676	000676			BR	2#	::GO PICKUP ANOTHER CHACTER
14061	055700			8#:	TYPE	#QUES	::TYPE A '?'
	055700	104401	002657		TYPEIT	,#QUES	
					.DSABL	CRF	
14062	055704	000671			BR	1#	::CLEAR THE BUFFER AND LOOP
14063	055706	111337	055760	9#:	MOVB	(R3),10#	::ECHO THE CHARACTER
14064	055712				TYPE	10#	
	055712	104401	055760		TYPEIT	,10#	
					.DSABL	CRF	
14065	055716	122723	000015		CMPB	#15,(R3)	::CHECK FOR RETURN
14066	055722	001264			BNE	2#	::LOOP IF NOT RETURN
14067	055724	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
14068	055730				TYPE	#LF	::TYPE A LINE FEED
	055730	104401	002661		TYPEIT	,#LF	
					.DSABL	CRF	
14069	055734	005726			TST	(SP)	::CLEAN RUBOUT KEY FROM THE STACK
14070	055736	012603			MOV	(SP)+,R3	::RESTORE R3
14071	055740	011646			MOV	(SP),-(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
14072	055742	016666	000004	000002	MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT

```
14073 055750 012766 055762 000004      MOV      @TTYIN,4(SP)
14074 055756 000002                RTI          ;;RETURN
14075 055760      000                104:      .BYTE 0          ;;STORAGE FOR ASCII CHAR. TO TYPE
14076 055761      000                .BYTE 0          ;;TERMINATOR
14077 055762 000024                $TTYIN: .REPT 20.  ;;RESERVE SIZE BYTES FOR TTY INPUT
14080 056006      136      103      015  $CNTLC: .ASCIZ /?C/<15><12>  ;;CONTROL "C"
14081 056011      012      000                .ASCIZ /?U/<15><12>  ;;CONTROL "U"
14082 056013      136      125      015  $CNTLU: .ASCIZ /?U/<15><12>
14082 056016      012      000                .ASCIZ /?G/<15><12>  ;;CONTROL "G"
14082 056020      136      107      015  $CNTLG: .ASCIZ /?G/<15><12>
14083 056023      012      000                .ASCIZ <15><12>/SWR = /
14083 056025      015      012      123  $MSWR: .ASCIZ <15><12>/SWR = /
14083 056030      127      122      040
14084 056033      075      040      000
14084 056036      040      040      116  $MNEW: .ASCIZ / NEW = /
14084 056041      105      127      040
14084 056044      075      040      000
14085                                .EVEN
```

```

14087 .SBTTL ROUTINE READ AN OCTAL NUMBER FROM THE TTY
14088 ;*****
14089 ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
14090 ;*CHANGE IT TO BINARY.
14091 ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
14092 ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
14093 ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
14094 ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
14095 ;*CALL:
14096 ;*      RDOCT          ;;READ AN OCTAL NUMBER
14097 ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
14098 ;*                  ;;HIGH ORDER BITS ARE IN #HIOCT
14099 056050 011646      #RDOCT: MOV      (SP),-(SP)      ;;PROVIDE SPACE FOR THE
14100 056052 016666 000004 000002      MOV      4(SP),2(SP)      ;;INPUT NUMBER
14101 056060          PUSH      R0,R1,R2
14102 056066 010046          ;;READ AN ASCII LINE
14103 056070 010146          MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
14104 056072 010246          MOV      R0,5#      ;;AND SAVE IT
14105 056076 010346          CLR      R1      ;;CLEAR DATA WORD
14106 056100 010446          CLR      R2
14107 056102 010546          1#:  ROLIN      (R0)+,-(SP)      ;;PICKUP THIS CHARACTER
14108 056104 010646          BEQ      3#      ;;IF ZERO GET OUT
14109 056106 010746          CMPB     #'0,(SP)      ;;MAKE SURE THIS CHARACTER
14110 056112 010846          BGT      4#      ;;IS AN OCTAL DIGIT
14111 056114 010946          CMPB     #'7,(SP)
14112 056120 011046          BLT      4#
14113 056122 011146          ASL      R1      ;;*2
14114 056124 011246          ROL      R2      ;;*4
14115 056126 011346          ASL      R1      ;;*8
14116 056130 011446          ROL      R2
14117 056132 011546          ASL      R1
14118 056134 011646          ROL      R2
14119 056136 011746          BIC      #'C7,(SP)      ;;STRIP THE ASCII JUNK
14120 056142 011846          ADD      (SP)+,R1      ;;ADD IN THIS DIGIT
14121 056144 011946          BR       2#      ;;LOOP
14122 056146 012046          3#:  TST      (SP)+      ;;CLEAN TERMINATOR FROM STACK
14123 056150 012146          MOV      R1,12(SP)      ;;SAVE THE RESULT
14124 056154 012246          MOV      R2,#HIOCT
14125 056160 012346          POP      R2,R1,R0
14126 056166 012446          MOV      (SP)+,R2
14127 056170 012546          MOV      (SP)+,R1
14128 056172 012646          MOV      (SP)+,R0
14129 056174 012746          RTI          ;;RETURN
14130 056176 012846          4#:  TST      (SP)+      ;;CLEAN PARTIAL FROM STACK
14131 056200 012946          CLRB     (R0)      ;;SET A TERMINATOR
14132 056204 013046          TYPE     .DSABL CRF      ;;TYPE UP THRU THE BAD CHAR.
14133 056208 013146          .WORD     0
14134 056212 013246          TYPE     MSG062      ;INPUT MUST BE A
14135 056216 013346          TYPE     .MSG062
14136 056220 013446          .DSABL     CRF
14137 056224 013546          TYPE     MSG063      ;N OCTAL
14138 056228 013646          TYPE     .MSG063

```



```

14133 056210      .DSABL CRF
          056210 104401 070077      TYPE MSG064 ;NUMBER
          .DSABL CRF
          BR 18 ;TRY AGAIN
14134 056214 000724 #HIOCT: .WORD 0 ;HIGH ORDER BITS GO HERE
14135 056216 000000      .SBTTL ROUTINE READ A DECIMAL NUMBER FROM THE TTY
14136
14137
14138 ;*****
14139 ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
14140 ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
14141 ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
14142 ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
14143 ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
14144 ;*POSITIVE 32767 TO NEGATIVE 32768.
14145 ;*CALL:
14146 ;* RDDEC ;:READ A DECIMAL NUMBER
14147 ;* RETURN HERE ;:NUMBER IS ON TOP OF THE STACK
14148 ;
14149
14150 056220 011646 #RDDEC: MOV (SP),-(SP) ;:PROVIDE SPACE FOR
14151 056222 016666 000004 000002 MOV 4(SP),2(SP) ;:THE INPUT NUMBER
14152 056230      PUSH R0,R1,R2
          056230 010046 ;:MOV R0,-(SP)
          056232 010146 ;:MOV R1,-(SP)
          056234 010246 ;:MOV R2,-(SP)
14153 056236 104412 18: RDLIN ;:READ AN ASCII LINE
14154 056240 012600 MOV (SP),R0 ;:ADDRESS OF 1ST CHAR.
14155 056242 010037 056366 MOV R0,6 ;:SAVE INCASE OF BAD INPUT
14156 056244 005046 CLR -(SP) ;:CLEAR DATA WORD
14157 056250 005002 CLR R2 ;:SIGN SET POSITIVE
14158 056252 122710 000055 CMPB @'-(R0) ;:SEE IF A MINUS SIGN WAS TYPED
14159 056254 001001 BNE 2 ;:BR IF NO MINUS SIGN
14160 056260 112002 MOVB (R0),R2 ;:SAVE FOR LATER USE
14161 056262 112001 21: MOVB (R0),R1 ;:PICKUP THIS CHARACTER
14162 056264 001424 BEQ 3 ;:GET OUT IF ZERO
14163 056266 122701 000060 CMPB @'0,R1 ;:MAKE SURE THIS CHARACTER
14164 056272 003032 BGT 5 ;:IS A DIGIT BETWEEN 0 & 9
14165 056274 122701 000071 CMPB @'9,R1
14166 056300 002427 BLT 5
14167 056302 032716 170000 BIT @'C7777,(SP) ;:DON'T LET NUMBER GET TO BIG
14168 056306 001024 BNE 5 ;:BR IF NUMBER WOULD OVERFLOW
14169 056310 006316 ASL (SP) ;:*2
14170 056312 011646 MOV (SP),-(SP) ;:SAVE FOR LATER
14171 056314 006316 ASL (SP) ;:*4
14172 056316 006316 ASL (SP) ;:*8
14173 056320 062616 ADD (SP),-(SP) ;:*10
14174 056322 102416 BVS 5 ;:OVERFLOW ISN'T ALLOWED
14175 056324 162701 000060 SUB @'0,R1 ;:STRIP AWAY THE ASCII JUNK
14176 056330 060116 ADD R1,(SP) ;:ADD IN THIS DIGIT
14177 056332 102412 BVS 5 ;:OVERFLOW ISN'T ALLOWED
14178 056334 000752 BR 2 ;:LOOP
14179 056336 005702 38: TST R2 ;:CHECK IF NUMBER IS NEG
14180 056340 001401 BEQ 4 ;:BR IF NO
14181 056342 005416 NEG (SP) ;:YES--NEGATE THE NUMBER
14182 056344 012666 000012 48: MOV (SP),12(SP) ;:SAVE THE RESULT
14183 056350      POP R2,R1,R0
    
```

```
                                MOV (SP)+,R2
                                MOV (SP)+,R1
                                MOV (SP)+,R0
14184 056350 012602
14185 056352 012601
14186 056354 012600
14187 056356 000002
14188 056356 000002
14189 056360 005726
14187 056362 105010
14188 056364 104401
14189 056366 000000
14190 056370 104401 070046
14191 056374 104401 070107
14192 056400 104401 070077
14193 056404 000714

RTI                                ;;RETURN

5$: TST (SP)+                       ;;CLEAN PARTIAL NUMBER FROM STACK
    CLRB (R0)                       ;;SET A TERMINATOR
    TYPE                               ;;TYPE THE INPUT UP TO BAD CHAR.
    TYPEIT
    .DSABL CRF
    .WORD 0
    TYPE MSG062                       ;;POINTER GOES HERE
    TYPEIT .MSG062                   ;INPUT MUSST BE A
    .DSABL CRF
    TYPE MSG065                       ;DECIMAL
    TYPEIT .MSG065
    .DSABL CRF
    TYPE MSG064                       ;NUMBER
    TYPEIT .MSG064
    .DSABL CRF
    BR 1$                             ;;TRY AGAIN
```

.SBTTL ROUTINE SAVE AND RESTORE RO-R5

```

14195
14196
14197
14198
14199
14200
14201
14202
14203
14204
14205
14206
14207
14208
14209
14210
14211
14212 056406
14213 056406
      056406 C10046
      056410 010146
      056412 010246
      056414 010346
      056416 010446
      056420 010546
14214 056422 016646 000022
14215 056426 016646 000022
14216 056432 016646 000022
14217 056436 016646 000022
14218 056442 000002
14219
14220
14221
14222
14223 056444
14224 056444 012666 000022
14225 056450 012666 000022
14226 056454 012666 000022
14227 056460 012666 000022
14228 056464
      056464 012605
      056466 012604
      056470 012603
      056472 012602
      056474 012601
      056476 012600
14229 056500 000002

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

$SAVREG:
    PUSH    RO,R1,R2,R3,R4,R5

    MOV     R0,-(SP)
    MOV     R1,-(SP)
    MOV     R2,-(SP)
    MOV     R3,-(SP)
    MOV     R4,-(SP)
    MOV     R5,-(SP)

    MOV     22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
    MOV     22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
    MOV     22(SP),-(SP)    ;;SAVE PS OF CALL
    MOV     22(SP),-(SP)    ;;SAVE PC OF CALL
    RTI

;*RESTORE RO-R5
;*CALL:
;* RESREG
;$RESREG:
    MOV     (SP)+,22(SP)    ;;RESTORE PC OF CALL
    MOV     (SP)+,22(SP)    ;;RESTORE PS OF CALL
    MOV     (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
    MOV     (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
    POP     R5,R4,R3,R2,R1,R0

    MOV     (SP)+,R5
    MOV     (SP)+,R4
    MOV     (SP)+,R3
    MOV     (SP)+,R2
    MOV     (SP)+,R1
    MOV     (SP)+,R0

    RTI
    
```

```

14231                                     .SBTTL  ROUTINE RANDOM NUMBER GENERATOR
14232
14233                                     ;*****
14234                                     ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
14235                                     ;*WITH A RANGE OF 0 TO 2**(+33)-1.
14236                                     ;*CALL:
14237                                     ;*   CALL      $RAND          ;;CALL THE ROUTINE
14238                                     ;*   RETURN    ;;RETURN HERE THE RANDOM
14239                                     ;*                                     ;;NUMBER WILL BE IN
14240                                     ;*                                     ;;$MINUM,$LONUM
14241
14242 $RAND:  PUSH      R0,R1,R2
14243                                     MOV      SEEDLO,R0          ;;SET R0 WITH LOW
14244                                     MOV      SEEDHI,R1          ;;SET R1 WITH HIGH
14245                                     MOV      #7,R2           ;;SET SHIFT COUNT
14246 1$:   ASL      R0           ;;SHIFT R0 LEFT AND
14247       ROL      R1           ;;ROTATE CARRY INTO R1 AND
14248       SOB      R2,1$
14249       ADD      SEEDLO,R0          ;;ADD NUMBER TO MAKE X 129
14250       ADC      R1           ;;PROPOGATE CARRY
14251       ADD      SEEDHI,R1          ;;ADD NUMBER TO MAKE X 129
14252       ADD      #1057,R0         ;;ADD LOW CONSTANT
14253       ADC      R1           ;;PROPOGATE CARRY
14254       ADD      #47401,R1         ;;ADD HIGH CONSTANT
14255       MOV      R0,SEEDLO        ;;SAVE R0
14256       MOV      R1,SEEDHI        ;;SAVE R1
14257       POP      R2,R1,R0
14258                                     MOV (SP)+,R2
14259                                     MOV (SP)+,R1
14260                                     MOV (SP)+,R0
14261                                     RETURN
14262
14263 056502 010046
14264 056504 010146
14265 056506 010246
14266 056510 013700 002604
14267 056514 013701 002602
14268 056520 012702 000007
14269 056524 006300
14270 056526 C06101
14271 056530 077203
14272 056532 063700 002604
14273 056536 005501
14274 056540 063701 002602
14275 056544 062700 001057
14276 056550 005501
14277 056552 062701 047401
14278 056556 010037 002604
14279 056562 010137 002602
14280 056566 012602
14281 056570 012601
14282 056572 012600
14283 056574 000207
  
```

```

14261          .SBTTL ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
14262          ;*****
14263          ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
14264          ;*UNSIGNED OCTAL ASCII NUMBER.
14265          ;*CALL
14266          ;*      MOV      #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
14267          ;*      CALL    #DB20           ;; CALL THE ROUTINE
14268          ;*      RETURN                    ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
14269
14270
14271 056576 104415          $DB20: SAVREG                ;; SAVE ALL REGISTERS
14272 056600 016601 000002  MOV      2(SP),R1          ;; PICKUP THE POINTER TO LOW WORD
14273 056604 012705 056715  MOV      #OCTVL+13.,R5    ;; POINTER TO DATA TABLE
14274 056610 012704 000014  MOV      #12.,R4         ;; DO ELEVEN CHARACTERS
14275 056614 012703 177770  MOV      #C7,R3         ;; MASK
14276 056620 012100  MOV      (R1)+,R0        ;; LOWER WORD
14277 056622 012101  MOV      (R1)+,R1        ;; HIGH WORD
14278 056624 005002  CLR      R2             ;; TERMINATOR
14279 056626 110245  1#: MOVB    R2,-(R5)      ;; PUT CHARACTER IN DATA TABLE
14280 056630 C10002  MOV      R0,R2         ;; GET THIS DIGIT
14281 056632 005304  DEC      R4            ;; COUNT THIS CHARACTER
14282 056634 003007  BGT     3#            ;; BR IF NOT THE LAST DIGIT
14283 056636 001405  BEQ     2#            ;; BR IF IT IS THE LAST DIGIT
14284 056640 005205  INC     R5            ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
14285 056642 010566 000002  MOV     R5,2(SP)      ;; ASCII CHAR. & PUT IT ON THE STACK
14286 056646 104416  RESREG                    ;; RESTORE ALL REGISTERS
14287 056650 000207  RETURN                    ;; RETURN TO USER
14288 056652 006203  2#: ASR     R3         ;; POSITION THE MASK FOR THE LAST DIGIT
14289 056654 006001  3#: ROR     R1         ;; POSITION THE BINARY NUMBER FOR
14290 056656 006000  ROR     R0            ;; THE NEXT OCTAL DIGIT
14291 056660 006001  ROR     R1
14292 056662 006000  ROR     R0
14293 056664 006001  ROR     R1
14294 056666 006000  ROR     R0
14295 056670 040302  BIC     R3,R2         ;; MASK OUT ALL JUNK
14296 056672 062702 000060  ADD     #0,R2        ;; MAKE THIS CHAR. ASCII
14297 056676 000753  BR      1#          ;; GO PUT IT IN THE DATA TABLE
14298 056700 000016  $OCTVL: .REPT 1#    ;; RESERVE DATA TABLE
14301          056704  $OCT8=#OCTVL*4     ;; POINTER TO 11 DIGIT NUMBER
    
```

```

14303          .SBTTL TABLES
14304
14305          .SBTTL APT MAILBOX-ETABLE
14306 056716    $MAIL:
14307 056716 000000 $MSGTY: .WORD 0      ;;MESSAGE TYPE CODE
14308 056720 000000 $FATAL: .WORD 0      ;;FATAL ERROR NUMBER (ERROR PC)
14309 056722 000000 $TESTN: .WORD 0      ;;TEST PATTERN NUMBER
14310 056724 000000 $PASS: .WORD 0       ;;PASS COUNT
14311 056726 000000 $DEVCT: .WORD 0      ;;DEVICE COUNT
14312 056730 000000 $UNIT: .WORD 0       ;;I/O UNIT NUMBER
14313 056732 000000 $MSGAD: .WORD 0      ;;MESSAGE ADDRESS
14314 056734 000000 $MSGLG: .WORD 0      ;;MESSAGE LENGTH
14315 056736    $ETABLE:
14316 056736      000  $ENV: .BYTE 0      ;;APT ENVIRONMENT TABLE
14317          ;NOTE: IF BIT #7 IS SET IN $ENVM THE TABLE BELOW (BEGINNING AT $MAMS1 AND
14318          ;      ENDING AT $MADR4) MUST BE FILLED IN TO INDICATE THE PROPER AMOUNT OF
14319          ;      EACH TYPE OF MEMORY.
14320 056737      000  $ENVM: .BYTE 0      ;;ENVIRONMENT BYTE ;SET TO A 1 FOR APT AUTO MODE
14321          ;ENVIRONMENT MODE
14322 056740 C00101 $SWREG: .WORD 101    ;;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
14323 056742 000000 $USWR: .WORD 0      ;;APT SWITCH REGISTER
14324 056744 000000 $CPUOP: .WORD 0      ;;USED TO LIMIT THE NUMBER OF PASSES
14325          ;CPU TYPE,OPTIONS
14326          ;BITS 15-11-CPU TYPE
14327          ;      11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
14328          ;      11/70=06,PDQ=07,Q=10
14329          ;BIT 10-REAL TIME CLOCK
14330          ;BIT 9-FLOATING POINT PROCESSOR
14331 056746      001  $MAMS1: .BYTE 1      ;;HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
14332 056747      004  $MTYP1: .BYTE 4      ;;MEM. TYPE,BLK#1
14333          ;MEM. TYPE BYTE -- (HIGH BYTE)
14334          ;      900 NSEC CORE=001
14335          ;      300 NSEC BIPOLAR=002
14336          ;      PARITY MOS=003
14337          ;      ERROR CORRECTING MOS=004
14338 056750 177776 $MADR1: .WORD 177776 ;;HIGH ADDRESS,BLK#1
14339          ;MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
14340 056752      000  $MAMS2: .BYTE 0      ;;HIGH ADDRESS,M.S. BYTE
14341 056753      000  $MTYP2: .BYTE 0      ;;MEM. TYPE,BLK#2
14342 056754 000000 $MADR2: .WORD 0      ;;MEM.LAST ADDRESS,BLK#2
14343 056756      000  $MAMS3: .BYTE 0      ;;HIGH ADDRESS,M.S.BYTE
14344 056757      000  $MTYP3: .BYTE 0      ;;MEM. TYPE,BLK#3
14345 056760 000000 $MADR3: .WORD 0      ;;MEM.LAST ADDRESS,BLK#3
14346 056762      000  $MAMS4: .BYTE 0      ;;HIGH ADDRESS,M.S.BYTE
14347 056763      000  $MTYP4: .BYTE 0      ;;MEM. TYPE,BLK#4
14348 056764 000000 $MADR4: .WORD 0      ;;MEM.LAST ADDRESS,BLK#4
14349 056766 000000 $VECT1: .WORD 0      ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
14350 056770 000000 $VECT2: .WORD 0      ;;INTERRUPT VECTOR#2BUS PRIORITY#2
14351 056772 000000 $BASE: .WORD 0       ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
14352 056774 000000 $DEVM: .WORD 0       ;;DEVICE MAP
14353
14354 056776 000000 $CDW1: .WORD 0
14355 057000 000000 $CDW2: .WORD 0
14356 057002 000000 $DDW7: .WORD 0      ;;UFD MODE FLAG 1-UFD MODE

```

```

14358 ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
14359 ;ARE TO BE RUN FOR PARTICULAR MEMORIES
14360 ;
14361 ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
14362 ;BIT0 SET WILL RUN THE FIRST ENTRY IN THE TABLE, BIT0 SET
14363 ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
14364 ;
14365 ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
14366 ;
14367 057004 177777 ;FIELD SERVICE VALUE
14368 057006 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
14369 057010 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
14370 057012 177777 ;ECC PATTERNS 103777 TABLE = MKPAT:
14371 057014 177777 ;ECC PATTERNS 177777 TABLE = MKPAT:
14372 057016 177777 ;PARITY PATTERNS 003777 TABLE = MJPAT:
14376 057020 ;PARITY PATTERNS 177774 TABLE = MJPAT:
14377 ;ETEND:
14378 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
14379 ;INTERFACE SPEC.
14380 057020 $APTHD:
14381 057020 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
14382 057022 056716 $MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
14383 057024 000043 $TSTM: .WORD 35. ;;RUN TIM OF LONGEST TEST
14384 057026 001274 $PASTM: .WORD 700. ;;RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
14385 057030 000000 $UNITM: .WORD 0. ;;EXTRA RUN TIME OF A PASS FOR EACH ADDITIONAL 128K (QV)
14386 057032 000041 .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

.SBTTL ROUTINE TRAP DECODER

14388
 14389
 14390
 14391
 14392
 14393
 14394
 14395
 14396 057034 010046
 14397 057036 016600 000002
 14398 057042 005740
 14399 057044 111000
 14400 057046 006300
 14401 057050 016000 057076
 14402 057054 000200
 14403
 14404
 14405
 14406
 14407 057056 C11646
 14408 057060 016666 000004 000002
 14409 057066 000002
 14410
 14411 057070
 057070 104401 065557
 14412 057074 000000

```

;*****
;*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.
$TRAP:  MOV    RO,-(SP)           ;;SAVE RO
        MOV    2(SP),RO         ;;GET TRAP ADDRESS
        TST    -(RO)           ;;BACKUP BY 2
        MOVB   (RO),RO         ;;GET RIGHT BYTE OF TRAP
        ASL    RO              ;;POSITION FOR INDEXING
        MOV    $TR AD(RO),RO    ;;INDEX TO TABLE
        RTS    RO              ;;GO TO ROUTINE

;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
$TRAP2: MOV    (SP),-(SP)      ;;MOVE THE PC DOWN
        MOV    4(SP),2(SP)    ;;MOVE THE PSW DOWN
        RTI                    ;;RESTORE THE PSW

$NOTRAP:TYPE  MSG006          ;UNDEFINED TRAP INSTRUCTION
        TYPEIT ,MSG006
        .DSABL CRF

$HALT2: HALT
  
```


14415
14416
14417
14418
14419
14420
14421
14422 057076 057056
14423 057100 047030
14424 057102 054152
14425 057104 054126
14426 057106 057070
14427 057110 054354
14428 057112 057070
14429
14430 057114 054754
14431 057116 054600
14432
14433 057120 055334
14434 057122 055464
14435 057124 056050
14436 057126 056220
14437
14438 057130 056406
14439 057132 056444
14440
14441 057134 034070
14442 057136 034100
14443 057140 034110
14444
14445 057142 036202
14446
14447 057144 034120
14448 057146 034144
14449
14450 057150 034162
14451 057152 034256
14452
14453 057154 047364
14454 057156 047412
14455 057160 047440
14456 057162 047470
14457 057164 047552
14458 057166 047574
14459 057170 047624
14460 057172 047644
14461 057174 047666
14462 057176 047706
14463 057200 047730
14464 057202 047752
14465 057204 047772
14466 057206 050010
14467 057210 050026
14468 057212 050046
14469 057214 050064
14470 057216 050102
14471 057220 044664

.SBTTL TRAP TABLE
; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
; *BY THE "TRAP" INSTRUCTION.
;
; ROUTINE
; -----
;TRPAD: .WORD \$TRAP2
;TYPE ;CALL=TYPEIT TRAP+1(104401) TTY TYPEOUT ROUTINE
;TYPOC ;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
;TYPOS ;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
;NOTRAP; \$TYPON ;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
;TYPDS ;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
;NOTRAP; \$TYPBN ;CALL=TYPBN TRAP+6(104406) TYPE BINARY (ASCII) NUMBER

;GTSWR ;CALL=GTSWR TRAP+7(104407) GET SOFT-SWR SETTING
;CKSWR ;CALL=CKSWR TRAP+10(104410) TEST FOR CHANGE IN SOFT-SWR

;RDCHR ;CALL=RDCHR TRAP+11(104411) TTY TYPEIN CHARACTER ROUTINE
;RDLIN ;CALL=RDLIN TRAP+12(104412) TTY TYPEIN STRING ROUTINE
;RDOCT ;CALL=RDOCT TRAP+13(104413) READ AN OCTAL NUMBER FROM TTY
;RDDEC ;CALL=RDDEC TRAP+14(104414) READ A DECIMAL NUMBER FROM TTY

;SAVREG ;CALL=SAVREG TRAP+15(104415) SAVE R0-R5 ROUTINE
;RESREG ;CALL=RESREG TRAP+16(104406) RESTORE R0-R5 ROUTINE

;KERNEL ;CALL=KERNEL TRAP+17(104417) ENTER KERNEL MODE
;ENERGIZE;CALL=ENERGIZETRAP+20(104420) TURN ON MEMORY MANAGEMENT & TRAPS
;DEENERGI;CALL=DEENERGITRAP+21(104421) TURN OFF MEMORY MANAGEMENT & TRAPS

;KMAP ;CALL=KMAP TRAP+22(104422) MAP KERNEL 1 TO 1

;CACHN ;CALL=CACHON TRAP+23(104423) TURN CACHE ON
;CACHF ;CALL=CACHOFF TRAP+24(104424) TURN CACHE OFF

;LOADC ;CALL=LOADCSR TRAP+25(104425) LOAD CORRECT CSR
;READC ;CALL=READCSR TRAP+26(104426) READ CORRECT CSR

;PER01 ;CALL=PERR01 TRAP+27(104427) PROGRAM DETECTED ERROR
;PER02 ;CALL=PERR02 TRAP+30(104430) PROGRAM DETECTED ERROR
;PER03 ;CALL=PERR03 TRAP+31(104431) PROGRAM DETECTED ERROR
;PER04 ;CALL=PERR04 TRAP+32(104432) PROGRAM DETECTED ERROR
;PER07 ;CALL=PERR07 TRAP+33(104433) PROGRAM DETECTED ERROR
;PER10 ;CALL=PERR10 TRAP+34(104434) PROGRAM DETECTED ERROR
;PER11 ;CALL=PERR11 TRAP+35(104435) PROGRAM DETECTED ERROR
;PER12 ;CALL=PERR12 TRAP+36(104436) PROGRAM DETECTED ERROR
;PER13 ;CALL=PERR13 TRAP+37(104437) PROGRAM DETECTED ERROR
;PER14 ;CALL=PERR14 TRAP+40(104440) PROGRAM DETECTED ERROR
;PER15 ;CALL=PERR15 TRAP+41(104441) PROGRAM DETECTED ERROR
;PER16 ;CALL=PERR16 TRAP+42(104442) PROGRAM DETECTED ERROR
;PER17 ;CALL=PERR17 TRAP+43(104443) PROGRAM DETECTED ERROR
;PER20 ;CALL=PERR20 TRAP+44(104444) PROGRAM DETECTED ERROR
;PER21 ;CALL=PERR21 TRAP+45(104445) PROGRAM DETECTED ERROR
;PER22 ;CALL=PERR22 TRAP+46(104446) PROGRAM DETECTED ERROR
;PER23 ;CALL=PERR23 TRAP+47(104447) PROGRAM DETECTED ERROR
;PER24 ;CALL=PERR24 TRAP+50(104450) PROGRAM DETECTED ERROR
;PER25 ;CALL=PERR25 TRAP+51(104451) PROGRAM DETECTED ERROR

14472	057222	050272	#PER26	:CALL=PERR26	TRAP+52(104452)	PROGRAM DETECTED ERROR
14473	057224	050312	#PER27	:CALL=PERR27	TRAP+53(104453)	PROGRAM DETECTED ERROR
14474	057226	045112	#PER30	:CALL=PERR30	TRAP+54(104454)	PROGRAM DETECTED ERROR
14475	057230	050502	#PER31	:CALL=PERR31	TRAP+55(104455)	PROGRAM DETECTED ERROR
14476	057232	050600	#PER32	:CALL=PERR32	TRAP+56(104456)	PROGRAM DETECTED ERROR
14477	057234	050646	#PER33	:CALL=PERR33	TRAP+57(104457)	PROGRAM DETECTED ERROR
14478	057236	050726	#PER34	:CALL=PERR34	TRAP+60(104460)	PROGRAM DETECTED ERROR
14479	057240	050760	#PER35	:CALL=PERR35	TRAP+61(104461)	PROGRAM DETECTED ERROR
14480	057242	051014	#PER36	:CALL=PERR36	TRAP+62(104462)	PROGRAM DETECTED ERROR
14481	057244	051044	#PER37	:CALL=PERR37	TRAP+63(104463)	PROGRAM DETECTED ERROR ;:ILC:REV B
14482	057246	051050	#PER40	:CALL=PERR40	TRAP+64(104464)	PROGRAM DETECTED ERROR ;:ILC:REV B
14483	057250	057070	#NOTRAP	:CALL=PERR41	TRAP+65(104465)	PROGRAM DETECTED ERROR
14484	057252	057070	#NOTRAP	:CALL=PERR42	TRAP+66(104466)	PROGRAM DETECTED ERROR
14485	057254	057070	#NOTRAP	:CALL=PERR43	TRAP+67(104467)	PROGRAM DETECTED ERROR
14486						
14487	057256	034456	#ECCDIS	:CALL=ECCDIS	TRAP+70(104470)	DISABLE ECC ON ALL CSR'S
14488	057260	034472	#ECC1DIS	:CALL=ECC1DIS	TRAP+71(104471)	DISABLE ECC ON 1 SELECTED CSR
14489	057262	034504	#ECCINIT	:CALL=ECCINIT	TRAP+72(104472)	INITIALIZE ALL MK11 CSR'S
14490	057264	034520	#ECC1INIT	:CALL=ECC1INIT	TRAP+73(104473)	INITIALIZE 1 SELECTED MK11 CSR
14491	057266	034560	#CBCSR	:CALL=CBCSR	TRAP+74(104474)	WRITE GENERATED CHECKBITS IN ALL CSR'S
14492	057270	034602	#CB1CSR	:CALL=CB1CSR	TRAP+75(104475)	WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
14493	057272	034622	#WASSBE	:CALL=WASSBE	TRAP+76(104476)	WAS THERE A SBE ON ANY CSR?
14494	057274	034736	#WAS1SBE	:CALL=WAS1SBE	TRAP+77(104477)	WAS THERE A SBE ON 1 SELECTED CSR?
14495	057276	034766	#WASDBE	:CALL=WASDBE	TRAP+100(104500)	WAS THERE A DBE ON ANY CSR?
14496	057300	035102	#WAS1DBE	:CALL=WAS1DBE	TRAP+101(104501)	WAS THERE A DBE ON 1 SELECTED CSR?
14497	057302	035132	#CLRCSR	:CALL=CLRCSR	TRAP+102(104502)	CLEAR ALL CSR'S
14498	057304	035144	#CLR1CSR	:CALL=CLR1CSR	TRAP+103(104503)	CLEAR 1 SELECTED CSR
14499	057306	035154	#CHKDIS	:CALL=CHKDIS	TRAP+104(104504)	DISABLE ECC & WRITE CKBITS FROM ALL CSR'S
14500	057310	035170	#CHK1DIS	:CALL=CHK1DIS	TRAP+105(104505)	DISABLE ECC & WRITE CKBITS FROM 1 CSR
14501	057312	034532	#ENASBE	:CALL=ENASBE	TRAP+106(104506)	ENABLE TRAPS ON SBE'S FROM ALL CSR'S
14502	057314	034546	#ENA1SBE	:CALL=ENA1SBE	TRAP+107(104507)	ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
14503	057316	034276	#TSTRD	:CALL=TSTREAD	TRAP+110(104510)	TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
14504	057320	035250	#INVALID	:CALL=INVALID	TRAP+111(104511)	INVALIDATE BACKGROUND PATTERN ON BANK
14505	057322	035300	#ERRGEN	:CALL=ERRGEN	TRAP+112(104512)	TEST ERROR ADDRESS
14506	057324	035550	#CBREG	:CALL=CBREG	TRAP+113(104513)	ENABLE CHECK BIT REGISTER
14507	057326	035566	#SYNREG	:CALL=SYNREG	TRAP+114(104514)	ENABLE SYNDROME BIT REGISTER
14508	057330	057070	#NOTRAP			
14509	057332	057070	#NOTRAP			
14510	057334	057070	#NOTRAP			
14511	057336	057070	#NOTRAP			
14512	057340	057070	#NOTRAP			
14513	057342	057070	#NOTRAP			
14516	177776		ST	=	177776	:STATUS REGISTER

14519
14520
14521
14522
14523
14524
14525
14526
14527
14528
14529
14530
14531
14532
14533 057344
14534 057344 062032
14535 057346 064242
14536 057350 060370
14537 057352 060751
14538
14539 057354 061017
14540 057356 063551
14541 057360 060214
14542 057362 060627
14543
14544 057364 061055
14545 057366 063631
14546 057370 060232
14547 057372 060744
14548
14549 057374 061107
14550 057376 063631
14551 057400 060242
14552 057402 060744
14553
14554 057404 061155
14555 057406 063665
14556 057410 060252
14557 057412 060627
14558
14559 057414 061232
14560 057416 063665
14561 057420 060252
14562 057422 060627
14563
14564 057424 061257
14565 057426 063665
14566 057430 060252
14567 057432 060627
14568
14569 057434 063243
14570 057436 064764
14571 057440 060546
14572 057442 060627

.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

14575			;ERROR	11
14576	057444	061317	EM11	
14577	057446	064011	DH7	
14578	057450	060304	DT7	
14579	057452	060653	DF3	
14580			;ERROR	12
14581	057454	061317	EM11	
14582	057456	064011	DH7	
14583	057460	060304	DT7	
14584	057462	060666	DF4	
14585			;ERROR	13
14586	057464	061341	EM12	
14587	057466	064121	DH10	
14588	057470	060334	DT10	
14589	057472	060627	DF2	
14590			;ERROR	14
14591	057474	061317	EM11	
14592	057476	064011	DH7	
14593	057500	060304	DT7	
14594	057502	060701	DF5	
14595			;ERROR	15
14596	057504	061317	EM11	
14597	057506	064011	DH7	
14598	057510	060304	DT7	
14599	057512	060714	DF6	
14600			;ERROR	16
14601	057514	061365	EM13	
14602	057516	064242	DH13	
14603	057520	060370	DT13	
14604	057522	060751	DF11	
14605			;ERROR	17
14606	057524	061417	EM14	
14607	057526	064242	DH13	
14608	057530	060370	DT13	
14609	057532	060751	DF11	
14610			;ERROR	20
14611	057534	061463	EM15	
14612	057536	064242	DH13	
14613	057540	060370	DT13	
14614	057542	060751	DF11	
14615			;ERROR	21
14616	057544	063272	EM55	
14617	057546	065022	DH26	
14618	057550	060560	DT26	
14619	057552	060627	DF2	
14620			;ERROR	22
14621	057554	061531	EM17	
14622	057556	064011	DH7	
14623	057560	060304	DT7	
14624	057562	060701	DF5	
14625			;ERROR	23
14626	057564	063112	EM50	
14627	057566	064636	DH23	
14628	057570	060504	DT23	
14629	057572	060762	DF13	

14632			;ERROR	24	
14633	057574	061571	EM19		
14634	057576	064242	DH13		
14635	057600	060370	DT13		
14636	057602	060751	DF11		
14637			;ERROR	25	
14638	057604	061643	EM20		
14639	057606	064242	DH13		
14640	057610	060370	DT13		
14641	057612	060751	DF11		
14642			;ERROR	26	
14643	057614	000000	0		;NO MESSAGE
14644	057616	064235	DH12		
14645	057620	060364	DT12		
14646	057622	060627	DF2		
14647			;ERROR	27	
14648	057624	061722	EM21		
14649	057626	064217	DH11		
14650	057630	060356	DT11		
14651	057632	C60627	DF2		
14652			;ERROR	30	
14653	057634	061756	EM22		
14654	057636	064242	DH13		
14655	057640	060370	DT13		
14656	057642	060751	DF11		
14657			;ERROR	31	
14658	057644	000000	0		;NO MESSAGE
14659	057646	064337	DH14		
14660	057650	060412	DT14		
14661	057652	060627	DF2		
14662			;ERROR	32	
14663	057654	062003	EM23		
14664	057656	063665	DH5		
14665	057660	060252	DT5		
14666	057662	060627	DF2		
14667			;ERROR	33	
14668	057664	062111	EM25		
14669	057666	064416	DH15		
14670	057670	060430	DT16		
14671	057672	060727	DF7		
14672			;ERROR	34	
14673	057674	062136	EM26		
14674	057676	064535	DH16		
14675	057700	060460	DT17		
14676	057702	060653	DF3		

14679			;ERROR	35
14680	057704	063216	EM52	
14681	057706	064764	DM25	
14682	057710	060546	DT25	
14683	057712	060627	DF2	
14684			;ERROR	36
14685	057714	062207	EM27	
14686	057716	064535	DM16	
14687	057720	060460	DT17	
14688	057722	060742	DF8	
14689			;ERROR	37
14690	057724	062704	EM35	
14691	057726	064011	DM7	
14692	057730	060304	DT7	
14693	057732	060653	DF3	
14694			;ERROR	40
14695	057734	062277	EM29	
14696	057736	064011	DM7	
14697	057740	060304	DT7	
14698	057742	060653	DF3	
14699			;ERROR	41
14700	057744	0 2361	EM30	
14701	057746	0 1011	DM7	
14702	057750	01 J304	DT7	
14703	057752	060701	DF5	
14704			;ERROR	42
14705	057754	063413	EM60	
14706	057756	064557	DM20	
14707	057760	060504	DT23	
14708	057762	060762	DF13	
14709			;ERROR	43
14710	057764	062471	EM32	
14711	057766	064011	DM7	
14712	057770	060304	DT7	
14713	057772	060653	DF3	
14714			;ERROR	44
14715	057774	062576	EM33	
14716	057776	064011	DM7	
14717	060000	060304	DT7	
14718	060002	060653	DF3	
14719			;ERROR	45
14720	060004	063146	EM51	
14721	060006	064715	DM24	
14722	060010	060526	DT24	
14723	060012	060772	DF14	
14724			;ERROR	46
14725	060014	062771	EM36	
14726	060016	063744	DM6	
14727	060020	060270	DT6	
14728	060022	060627	DF2	

14731			;ERROR	47
14732	060024	063040	EM40	
14733	060026	063606	DH2	
14734	060030	060466	DT20	
14735	060032	060627	DF2	
14736			;ERROR	50
14737	060034	063313	EM56	
14738	060036	065040	DH27	
14739	060040	060566	DT27	
14740	060042	060626	DF1	
14741			;ERROR	51
14742	060044	063455	EM61	
14743	060046	064715	DH24	
14744	060050	060526	DT24	
14745	060052	060772	DF14	
14746			;ERROR	52
14747	060054	062277	EM29	
14748	060056	064242	DH13	
14749	060060	060370	DT13	
14750	060062	060751	DF11	
14751			;ERROR	53
14752	060064	062111	EM25	
14753	060066	065114	DH30	
14754	060070	060606	DT30	
14755	060072	061010	DF16	
14756			;ERROR	54
14757	060074	063345	EM57	
14758	060076	065114	DH30	
14759	060100	060606	DT30	
14760	060102	061010	DF16	
14761			;ERROR	55
14762	060104	061643	EM20	
14763	060106	064715	DH24	
14764	060110	060526	DT24	
14765	060112	060772	DF14	
14766			;ERROR	56
14767	060114	061643	EM20	
14768	060116	065114	DH30	
14769	060120	060606	DT30	
14770	060122	061010	DF16	
14771			;ERROR	57
14772	060124	061571	EM19	
14773	060126	064715	DH24	
14774	060130	060526	DT24	
14775	060132	060772	DF14	
14776			;ERROR	60
14777	060134	061365	EM13	
14778	060136	064557	DH20	
14779	060140	060504	DT23	
14780	060142	060762	DF13	
14781			;ERROR	61
14782	060144	062361	EM30	
14783	060146	064557	DH20	
14784	060150	060504	DT23	
14785	060152	060762	DF13	
14786			;ERROR	62
14787	060154	061643	EM20	

14788	060156	064715	DH24	
14789	060160	060526	DT24	
14790	060162	060772	DF14	
14791			;ERROR	63
14792	060164	061756	EM22	
14793	060166	064715	DH24	
14794	060170	060526	DT24	
14795	060172	060772	DF14	
14796			;ERROR	64
14797	060174	063476	EM62	
14798	060176	063665	DH5	
14799	060200	060252	DT5	
14800	060202	060627	DF2	
14801			;ERROR	65
14802	060204	061756	EM22	
14803	060206	064557	DH20	
14804	060210	060504	DT23	
14805	060212	060762	DF13	

14807						.SBTTL	ERROR DATA TAGS (DT)
14808	060214	002020	002034	002044	DT1:	.WORD	ERRPC, ADDRESS, GOOD, BAD, 0
	060222	002052	000000				
14809	060226	002020	000000		DT2:	.WORD	ERRPC, 0
14810	060232	002020	002036	002072	DT3:	.WORD	ERRPC, PADDRESS, PARCNT, 0
	060240	000000					
14811	060242	002020	002034	002070	DT4:	.WORD	ERRPC, ADDRESS, NEMCNT, 0
	060250	000000					
14812	060252	002020	177572	177574	DT5:	.WORD	ERRPC, MMRO, MMR1, MMR2, MMR3, CPUERR, 0
	060260	177576	172516	177766			
	060266	000000					
14813	060270	002020	002422	002400	DT6:	.WORD	ERRPC, APTPAR, LSIZE, APTECC, MSIZE, 0
	060276	002424	002402	000000			
14814	060304	002020	002176	002034	DT7:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, GOOD, BAD, BADXOR
	060312	002176	002044	002052			
	060320	002060					
14815	060322	002176	002176	002176		.WORD	DUMMY, DUMMY, DUMMY, DUMMY, 0
	060330	002176	000000				
14816	060334	002200	002202	002204	DT10:	.WORD	DETRO, DETR1, DETR2, DETR3, DETR4, DETR5, DETSP, DETPSW, 0
	060342	002206	002210	002212			
	060350	002214	002216	000000			
14817	060356	002020	002150	000000	DT11:	.WORD	ERRPC, CSR, 0
14818	060364	002150	000000		DT12:	.WORD	CSR, 0
14819	060370	002020	002176	002034	DT13:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, TSTDAT, TSTDAT+2, CHECK, CSR, 0
	060376	002176	002246	002250			
	060404	002314	002150	000000			
14820	060412	177746	177572	177574	DT14:	.WORD	CONTRL, MMRO, MMR1, MMR2, MMR3, CPUERR, 0
	060420	177576	172516	177766			
	060426	000000					
14821	060430	002020	002176	002176	DT16:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, GOOD2, GOOD3
	060436	002044	002046	002050			
14822	060444	002052	002054	002056		.WORD	BAD, BAD2, BAD3, DUMMY, DUMMY, 0
	060452	002176	002176	000000			
14823	060460	002020	002176	000000	DT17:	.WORD	ERRPC, DUMMY, 0
14824	060466	002020	002044	002052	DT20:	.WORD	ERRPC, GOOD, BAD, 0
	060474	000000					
14825	060476	002020	002176	000000	DT22:	.WORD	ERRPC, DUMMY, 0
14826	060504	002020	002176	002044	DT23:	.WORD	ERRPC, DUMMY, GOOD, BAD, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060512	002052	002176	002176			
	060520	002176	002176	000000			
14827	060526	002020	002176	002150	DT24:	.WORD	ERRPC, DUMMY, CSR, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060534	002176	002176	002176			
	060542	002176	000000				
14828	060546	002020	002044	002150	DT25:	.WORD	ERRPC, GOOD, CSR, CSRNO, 0
	060554	002152	000000				
14829	060560	002020	002052	000000	DT26:	.WORD	ERRPC, BAD, 0
14830	060566	002020	002176	002034	DT27:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060574	002176	002176	002176			
	060602	002176	000000				
14831	060606	002020	002176	002176	DT30:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, BAD, CSR, DUMMY, 0
	060614	002044	002052	002150			
	060622	002176	000000				

Address	Hex	Hex	Hex	Hex	DF	Format	Value
14834						.SBTTL	ERROR DATA FORMATS (DF)
14835	060626	000			DF1:	.BYTE	0
14836	060627	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
	060632	000	000	000			
	060635	000	000	000			
	060640	000	000	000			
	060643	000	000	000			
	060646	000	000	000			
	060651	000	000	000			
14837	060653	000	005	000	DF3:	.BYTE	0.5.0.8..0.0.0.3.6.2.4
	06065C	010	000	000			
	060661	000	003	006			
	060664	002	004				
14838	060666	000	005	000	DF4:	.BYTE	0.5.0.8..0.8..8..3.6.2.4
	060671	010	000	010			
	060674	010	003	006			
	060677	002	004				
14839	060701	000	005	000	DF5:	.BYTE	0.5.0.8..9..9..9..3.6.2.4
	060704	010	011	011			
	060707	011	003	006			
	060712	002	004				
14840	060714	000	005	000	DF6:	.BYTE	0.5.0.8..9..8..8..3.6.2.4
	060717	010	011	010			
	060722	010	003	006			
	060725	002	004				
14841	060727	000	005	010	DF7:	.BYTE	0.5.8..0.0.9..0.0.9..2.4
	060732	000	000	011			
	060735	000	000	011			
	060740	002	004				
14842	060742	000	005		DF8:	.BYTE	0.5
14843	060744	000	001	001	DF9:	.BYTE	0.1.1.1.1
	060747	001	001				
14844	060751	000	005	000	DF11:	.BYTE	0.5.0.8..0.0.0.0.0
	060754	010	000	000			
	060757	000	000	000			
14845	060762	000	005	000	DF13:	.BYTE	0.5.0.0.3.6.2.4
	060765	000	003	006			
	060770	002	004				
14846	060772	000	005	000	DF14:	.BYTE	0.5.0.3.6.2.4
	060775	003	006	002			
	061000	004					
14847	061001	000	005	000	DF15:	.BYTE	0.5.0.8..3.6.4
	061004	010	003	006			
	061007	004					
14848	061010	000	005	010	DF16:	.BYTE	0.5.8..0.0.3.4
	061013	000	000	003			
	061016	004					

14851						.SBTTL	ERROR MESSAGES (EM)
14857	061017	103	101	116	EM2:	.ASCIZ	/CAN'T SET 22 BIT MODE IN MMR3/
	061022	047	124	040			
	061025	123	105	124			
	061030	040	062	062			
	061033	040	102	111			
	061036	124	040	115			
	061041	117	104	105			
	061044	040	111	116			
	061047	040	115	115			
	061052	122	063	000			
14858	061055	120	101	122	EM3:	.ASCIZ	/PARITY ERROR(S) IN BANK 0/
	061060	111	124	131			
	061063	040	105	122			
	061066	122	117	122			
	061071	050	123	051			
	061074	040	111	116			
	061077	040	102	101			
	061102	116	113	040			
	061105	060	000				
14859	061107	116	117	116	EM4:	.ASCIZ	/NON-EXISTANT MEMORY (HOLES) IN BANK 0/
	061112	055	105	130			
	061115	111	123	124			
	061120	101	116	124			
	061123	040	115	105			
	061126	115	117	122			
	061131	131	040	050			
	061134	110	117	114			
	061137	105	123	051			
	061142	040	111	116			
	061145	040	102	101			
	061150	116	113	040			
	061153	060	000				
14860	061155	111	114	114	EM5:	.ASCIZ	/ILLEGAL OR RESERVED INSTRUCTION (TRAP TO 10)/
	061160	105	107	101			
	061163	114	040	117			
	061166	122	040	122			
	061171	105	123	105			
	061174	122	126	105			
	061177	104	040	111			
	061202	116	123	124			
	061205	122	125	103			
	061210	124	111	117			
	061213	116	040	050			
	061216	124	122	101			
	061221	120	040	124			
	061224	117	040	061			
	061227	060	051	000			
14861	061232	125	116	105	EM6:	.ASCIZ	/UNEXPECTED TRAP TO 4/
	061235	130	120	105			
	061240	103	124	105			
	061243	104	040	124			
	061246	122	101	120			
	061251	040	124	117			
	061254	040	064	000			
14862	061257	115	105	115	EM7:	.ASCIZ	/MEMORY MANAGEMENT (TRAP TO 250)/
	061262	117	122	131			

	061265	040	115	101	
	061270	116	101	107	
	061273	105	115	105	
	061276	116	124	040	
	061301	050	124	122	
	061304	101	120	040	
	061307	124	117	040	
	061312	062	065	060	
14863	061315	051	000		
	061317	115	105	115	EM11: .ASCIZ /MEMORY DATA ERROR/
	061322	117	122	131	
	061325	040	104	101	
	061330	124	101	040	
	061333	105	122	122	
14864	061336	117	122	000	
	061341	104	105	124	EM12: .ASCIZ /DETAILED ERROR DUMP/
	061344	101	111	114	
	061347	105	104	040	
	061352	105	122	122	
	061355	117	122	040	
	061360	104	125	115	
14865	061363	120	000		
	061365	115	111	123	EM13: .ASCIZ /MISSING EXPECTED SBE FLAG/
	061370	123	111	116	
	061373	107	040	105	
	061376	130	120	105	
	061401	103	124	105	
	061404	104	040	123	
	061407	102	105	040	
	061412	106	114	101	
14866	061415	107	000		
	061417	127	122	111	EM14: .ASCIZ /WRITE BYTE FAILED TO CLEAR SBE FLAG/
	061422	124	105	040	
	061425	102	131	124	
	061430	105	040	106	
	061433	101	111	114	
	061436	105	104	040	
	061441	124	117	040	
	061444	103	114	105	
	061447	101	122	040	
	061452	123	102	105	
	061455	040	106	114	
14867	061460	101	107	000	
	061463	106	101	111	EM15: .ASCIZ /FAILED TO GET INTERRUPT WITH DBE FLAG/
	061466	114	105	104	
	061471	040	124	117	
	061474	040	107	105	
	061477	124	040	111	
	061502	116	124	105	
	061505	122	122	125	
	061510	120	124	040	
	061513	127	111	124	
	061516	110	040	104	
	061521	102	105	040	
	061524	106	114	101	
14868	061527	107	000		
	061531	115	105	115	EM17: .ASCIZ /MEMORY DATA ERROR IN CHECK BITS/

	061534	117	122	131	
	061537	040	104	101	
	061542	124	101	040	
	061545	105	122	122	
	061550	117	122	040	
	061553	111	116	040	
	061556	103	110	105	
	061561	103	113	040	
	061564	102	111	124	
14869	061567	123	000		
	061571	123	102	105	EM19: .ASCIZ /SBE-DBE CAUSED PARITY TRAP WHEN INHIBITED/
	061574	055	104	102	
	061577	105	040	103	
	061602	101	125	123	
	061605	105	104	040	
	061610	120	101	122	
	061613	111	124	131	
	061616	040	124	122	
	061621	101	120	040	
	061624	127	110	105	
	061627	116	040	111	
	061632	116	110	111	
	061635	102	111	124	
14870	061640	105	104	000	
	061643	123	102	105	EM20: .ASCIZ /SBE-DBE DID NOT CAUSE PARITY TRAP WHEN ENABLED/
	061646	055	104	102	
	061651	105	040	104	
	061654	111	104	040	
	061657	116	117	124	
	061662	040	103	101	
	061665	125	123	105	
	061670	040	120	101	
	061673	122	111	124	
	061676	131	040	124	
	061701	122	101	120	
	061704	040	127	110	
	061707	105	116	040	
	061712	105	116	101	
	061715	102	114	105	
14871	061720	104	000		
	061722	123	102	105	EM21: .ASCIZ /SBE-DBE ON MASTER TEST WORD/
	061725	055	104	102	
	061730	105	040	117	
	061733	116	040	115	
	061736	101	123	124	
	061741	105	122	040	
	061744	124	105	123	
	061747	124	040	127	
	061752	117	122	104	
14872	061755	000			
	061756	115	111	123	EM22: .ASCIZ /MISSING EXPECTED DBE/
	061761	123	111	116	
	061764	107	040	105	
	061767	130	120	105	
	061772	103	124	105	
	061775	104	040	104	
	062000	102	105	000	

14873	062003	125	116	105	EM23:	.ASCIZ /UNEXPECTED PARITY TRAP/
	062006	130	120	105		
	062011	103	124	105		
	062014	104	040	120		
	062017	101	122	111		
	062022	124	131	040		
	062025	124	122	101		
	062030	120	000			
14874	062032	122	105	103	EM24:	.ASCIZ /RECEIVED DBE FLAG WHEN EXPECTING ONLY SBE FLAG/
	062035	105	111	126		
	062040	105	104	040		
	062043	104	102	105		
	062046	040	106	114		
	062051	101	107	040		
	062054	127	110	105		
	062057	116	040	105		
	062062	130	120	105		
	062065	103	124	111		
	062070	116	107	040		
	062073	117	116	114		
	062076	131	040	123		
	062101	102	105	040		
	062104	106	114	101		
	062107	107	000			
14875	062111	103	110	105	EM25:	.ASCIZ /CHECK BIT DATA ERROR/
	062114	103	113	040		
	062117	102	111	124		
	062122	040	104	101		
	062125	124	101	040		
	062130	105	122	122		
	062133	117	122	000		
14876	062136	101	104	104	EM26:	.ASCIZ /ADDRESS PARITY ERROR DID NOT CAUSE ABORT/
	062141	122	105	123		
	062144	123	040	120		
	062147	101	122	111		
	062152	124	131	040		
	062155	105	122	122		
	062160	117	122	040		
	062163	104	111	104		
	062166	040	116	117		
	062171	124	040	103		
	062174	101	125	123		
	062177	105	040	101		
	062202	102	117	122		
	062205	124	000			
14877	062207	105	103	103	EM27:	.ASCIZ /ECC INHIBIT MODE POINTER FAILURE - DID NOT PROTECT BANK/
	062212	040	111	116		
	062215	110	111	102		
	062220	111	124	040		
	062223	115	117	104		
	062226	105	040	120		
	062231	117	111	116		
	062234	124	105	122		
	062237	040	106	101		
	062242	111	114	125		
	062245	122	105	040		
	062250	055	040	104		

	062253	111	104	040	
	062256	116	117	124	
	062261	040	120	122	
	062264	117	124	105	
	062267	103	124	040	
	062272	102	101	116	
	062275	113	000		
14878	062277	103	117	122	EM29: .ASCIZ /CORRECTION FAILURE WITH ECC ENABLED ON FORCED SBE/
	062302	122	105	103	
	062305	124	111	117	
	062310	116	040	106	
	062313	101	111	114	
	062316	125	122	105	
	062321	040	127	111	
	062324	124	110	040	
	062327	105	103	103	
	062332	040	105	116	
	062335	101	102	114	
	062340	105	104	040	
	062343	117	116	040	
	062346	106	117	122	
	062351	103	105	104	
	062354	040	123	102	
	062357	105	000		
14879	062361	127	122	111	EM30: .ASCII /WRITE BYTE WITH ECC ENABLED FAILED TO CLEAR DATA AT/<CRLF>
	062364	124	105	040	
	062367	102	131	124	
	062372	105	040	127	
	062375	111	124	110	
	062400	040	105	103	
	062403	103	040	105	
	062406	116	101	102	
	062411	114	105	104	
	062414	040	106	101	
	062417	111	114	105	
	062422	104	040	124	
	062425	117	040	103	
	062430	114	105	101	
	062433	122	040	104	
	062436	101	124	101	
	062441	040	101	124	
	062444	200			
14880	062445	106	117	122	.ASCIZ /FORCED SBE LOCATION/
	062450	103	105	104	
	062453	040	123	102	
	062456	105	040	114	
	062461	117	103	101	
	062464	124	111	117	
	062467	116	000		
14881	062471	115	117	126	EM32: .ASCIZ /MOVB #360.(R2)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	062474	102	040	043	
	062477	063	066	060	
	062502	054	050	122	
	062505	062	051	053	
	062510	040	127	111	
	062513	124	110	040	
	062516	105	103	103	

	062521	040	105	116	
	062524	101	102	114	
	062527	105	104	040	
	062532	103	110	101	
	062535	116	107	105	
	062540	104	040	104	
	062543	101	124	101	
	062546	040	101	124	
	062551	040	106	117	
	062554	122	103	105	
	062557	104	040	104	
	062562	102	105	040	
	062565	114	117	103	
	062570	101	124	111	
	062573	117	116	000	
14882	062576	115	117	126	EM33: .ASCIZ /MOV #177400,(R1) WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	062601	040	043	061	
	062604	067	067	064	
	062607	060	060	054	
	062612	050	122	061	
	062615	051	040	127	
	062620	111	124	110	
	062623	040	105	103	
	062626	103	040	105	
	062631	116	101	102	
	062634	114	105	104	
	062637	040	103	110	
	062642	101	116	107	
	062645	105	104	040	
	062650	104	101	124	
	062653	101	040	101	
	062656	124	040	106	
	062661	117	122	103	
	062664	105	104	040	
	062667	104	102	105	
	062672	040	114	117	
	062675	103	101	124	
	062700	111	117	116	
	062703	000			
14883	062704	125	116	105	EM35: .ASCIZ /UNEXPECTED CORRECTION WITH ECC DISABLE ON FORCED SBE/
	062707	130	120	105	
	062712	103	124	105	
	062715	104	040	103	
	062720	117	122	122	
	062723	105	103	124	
	062726	111	117	116	
	062731	040	127	111	
	062734	124	110	040	
	062737	105	103	103	
	062742	040	104	111	
	062745	123	101	102	
	062750	114	105	040	
	062753	117	116	040	
	062756	106	117	122	
	062761	103	105	104	
	062764	040	123	102	
	062767	105	000		

14884	062771	101	120	124	EM36:	.ASCIZ /APT SIZE DISAGREES WITH PROGRAM SIZING/
	062774	040	123	111		
	062777	132	105	040		
	063002	104	111	123		
	063005	101	107	122		
	063010	105	105	123		
	063013	040	127	111		
	063016	124	110	040		
	063021	120	122	117		
	063024	107	122	101		
	063027	115	040	123		
	063032	111	132	111		
	063035	116	107	000		
14885	063040	102	122	101	EM40:	.ASCIZ /BRANCH GOBBLE FAILED CONDITION CODES TEST/
	063043	116	103	110		
	063046	040	107	117		
	063051	102	102	114		
	063054	105	040	106		
	063057	101	111	114		
	063062	105	104	040		
	063065	103	117	116		
	063070	104	111	124		
	063073	111	117	116		
	063076	040	103	117		
	063101	104	105	123		
	063104	040	124	105		
	063107	123	124	000		
14886	063112	102	101	104	EM50:	.ASCIZ /BAD ERROR ADDRESS GENERATED/
	063115	040	105	122		
	063120	122	117	122		
	063123	040	101	104		
	063126	104	122	105		
	063131	123	123	040		
	063134	107	105	116		
	063137	105	122	101		
	063142	124	105	104		
	063145	000				
14887	063146	106	114	101	EM51:	.ASCIZ /FLAGS NOT SET ON FORCED UNCORRECTED SBE/
	063151	107	123	040		
	063154	116	117	124		
	063157	040	123	105		
	063162	124	040	117		
	063165	116	040	106		
	063170	117	122	103		
	063173	105	104	040		
	063176	125	116	103		
	063201	117	122	122		
	063204	105	103	124		
	063207	105	104	040		
	063212	123	102	105		
	063215	000				
14888	063216	102	111	124	EM52:	.ASCIZ /BIT SET ERROR IN CSR/
	063221	040	123	105		
	063224	124	040	105		
	063227	122	122	117		
	063232	122	040	111		
	063235	116	040	103		

	063240	123	122	000	
14889	063243	102	111	124	EM53: .ASCIZ /BIT CLEAR ERROR IN CSR/
	063246	040	103	114	
	063251	105	101	122	
	063254	040	105	122	
	063257	122	117	122	
	063262	040	111	116	
	063265	040	103	123	
	063270	122	000		
14890	063272	111	114	114	EM55: .ASCIZ /ILLEGAL CSR TYPE/
	063275	105	107	101	
	063300	114	040	103	
	063303	123	122	040	
	063306	124	131	120	
	063311	105	000		
14891	063313	102	101	104	EM56: .ASCIZ /BAD PARITY TRAP GENERATED/
	063316	040	120	101	
	063321	122	111	124	
	063324	131	040	124	
	063327	122	101	120	
	063332	040	107	105	
	063335	116	105	122	
	063340	101	124	105	
	063343	104	000		
14892	063345	127	122	117	EM57: .ASCIZ /WRONG CHECK BIT READ BACK FROM MEMORY/
	063350	116	107	040	
	063353	103	110	105	
	063356	103	113	040	
	063361	102	111	124	
	063364	040	122	105	
	063367	101	104	040	
	063372	102	101	103	
	063375	113	040	106	
	063400	122	117	115	
	063403	040	115	105	
	063406	115	117	122	
	063411	131	000		
14893	063413	127	122	117	EM60: .ASCIZ /WRONG SYNDROME BITS READ INTO CSR/
	063416	116	107	040	
	063421	123	131	116	
	063424	104	122	117	
	063427	115	105	040	
	063432	102	111	124	
	063435	123	040	122	
	063440	105	101	104	
	063443	040	111	116	
	063446	124	117	040	
	063451	103	123	122	
	063454	000			
14894	063455	103	123	122	EM61: .ASCIZ /CSR UPDATE ERROR/
	063460	040	125	120	
	063463	104	101	124	
	063466	105	040	105	
	063471	122	122	117	
	063474	122	000		
14895	063476	120	122	117	EM62: .ASCIZ /PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC/
	063501	103	105	123	

063504	123	117	122
063507	040	116	117
063512	124	040	123
063515	125	120	120
063520	117	122	124
063523	105	104	040
063526	102	131	040
063531	124	110	111
063534	123	040	104
063537	111	101	107
063542	116	117	123
063545	124	111	103
063550	000		

Address	Hex 1	Hex 2	Hex 3	Header
14898				.SBTTL ERROR DATA HEADERS (DH)
14899	063551	040	040	.ASCIZ / PC DEV ADD GOOD BAD/
	063554	103	040	
	063557	040	104	
	063562	126	040	
	063565	104	104	
	063570	040	040	
	063573	117	117	
	063576	040	040	
	063601	040	102	
	063604	104	000	
14900	063606	040	040	DH2: .ASCIZ / PC GD-CC BD-CC/
	063611	103	040	
	063614	040	107	
	063617	055	103	
	063622	040	102	
	063625	055	103	
	063630	000		
14901	063631	040	040	DH3: .ASCIZ / PC 1ST ADD # OF ERRORS/
	063634	103	040	
	063637	040	061	
	063642	124	040	
	063645	104	104	
	063650	040	043	
	063653	117	106	
	063656	105	122	
	063661	117	122	
	063664	000		
14902	063665	040	040	DH5: .ASCIZ / PC MMRO MMR1 MMR2 MMR3 CPUERR/
	063670	103	040	
	063673	040	040	
	063676	115	115	
	063701	060	040	
	063704	040	040	
	063707	115	122	
	063712	040	040	
	063715	040	115	
	063720	122	062	
	063723	040	040	
	063726	115	115	
	063731	063	040	
	063734	040	103	
	063737	125	105	
	063742	122	000	
14903	063744	040	040	DH6: .ASCIZ / PC APTPAR LSIZE APTECC MSIZE/
	063747	103	040	
	063752	040	101	
	063755	124	120	
	063760	122	040	
	063763	040	114	
	063766	111	132	
	063771	040	040	
	063774	120	124	
	063777	103	103	
	064002	040	115	
	064005	111	132	
	064010	000		

Address	Value	Value	Value	Label	Value	Value	Value	Value
14925				.SBTTL	MESSAGES			
14926	065173	200	040	MSG001: .ASCIZ	<CRLF>/			MEMORY CONFIGURATION MAP/
	065176	040	040					
	065201	040	040					
	065204	040	040					
	065207	040	040					
	065212	040	040					
	065215	040	040					
	065220	040	040					
	065223	040	115					
	065226	115	117					
	065231	131	040					
	065234	117	116					
	065237	111	107					
	065242	122	101					
	065245	111	117					
	065250	040	115					
	065253	120	000					
14927	065255	200	040	MSG002: .ASCIZ	<CRLF>/			16K WORD BANKS/
	065260	040	040					
	065263	040	040					
	065266	040	040					
	065271	040	040					
	065274	040	040					
	065277	040	040					
	065302	040	040					
	065305	040	040					
	065310	040	040					
	065313	061	066					
	065316	040	127					
	065321	122	104					
	065324	102	101					
	065327	113	123					
14928	065332	200	040	MSG003: .ASCII	<CRLF>/	1	2	3/
	065335	040	040					
	065340	040	040					
	065343	040	040					
	065346	040	040					
	065351	040	040					
	065354	040	040					
	065357	040	040					
	065362	040	062					
	065365	040	040					
	065370	040	040					
	065373	063						
14929	065374	040	040	.ASCIZ /	4	5	6	7 /
	065377	040	040					
	065402	040	064					
	065405	040	040					
	065410	040	040					
	065413	065	040					
	065416	040	040					
	065421	040	040					
	065424	040	040					
	065427	040	040					
	065432	040	067					
	065435	040	000					

14930	065437	200	040	040	MSG004: .ASCII <CRLF>/	012345670123456701234567/
	065442	040	040	040		
	065445	040	040	040		
	065450	060	061	062		
	065453	063	064	065		
	065456	066	067	060		
	065461	061	062	063		
	065464	064	065	066		
	065467	067	060	061		
	065472	062	063	064		
	065475	065	066	067		
14931	065500	060	061	062	.ASCIZ /012345670123456701234567012345670123/	
	065503	063	064	065		
	065506	066	067	060		
	065511	061	062	063		
	065514	064	065	066		
	065517	067	060	061		
	065522	062	063	064		
	065525	065	066	067		
	065530	060	061	062		
	065533	063	064	065		
	065536	066	067	060		
	065541	061	062	063		
	065544	000				
14932	065545	200	105	122	MSG005: .ASCIZ <CRLF>/ERRORS /	
	065550	122	117	122		
	065553	123	040	040		
	065556	000				
14933	065557	200	125	116	MSG006: .ASCIZ <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>	
	065562	104	105	106		
	065565	111	116	105		
	065570	104	040	124		
	065573	122	101	120		
	065576	040	111	116		
	065601	123	124	122		
	065604	125	103	124		
	065607	111	117	116		
	065612	032	000			
14934	065614	200	115	105	MSG009: .ASCIZ <CRLF>/MEMTYPE /	:MEMORY TYPE
	065617	115	124	131		
	065622	120	105	040		
	065625	000				
14935	065626	200	120	122	MSG010: .ASCIZ <CRLF>/PROTECT /	:MEMORY PROTECTED
	065631	117	124	105		
	065634	103	124	040		
	065637	000				
14936	065640	040	040	040	MSG11A: .ASCIZ /	1 1 1 1 1 1 1/
	065643	040	061	040		
	065646	040	040	040		
	065651	040	040	040		
	065654	061	040	040		
	065657	040	040	040		
	065662	040	040	061		
	065665	040	040	040		
	065670	040	040	040		
	065673	040	061	040		
	065676	040	040	040		

	066145	000				
14946	066146	040	040	040	MSG017: .ASCIZ / /	:8 SPACES
	066151	040	040	040		
	066154	040	040	000		
14947	066157	040	040	000	MSG018: .ASCIZ / /	:2 SPACES
14948	066162	040	040	040	MSG019: .ASCIZ / /	:3 SPACES
	066165	000				
14949	066166	200	106	123	MSG020: .ASCIZ <CRLF>/FS COMMAND MODE/	
	066171	040	103	117		
	066174	115	115	101		
	066177	116	104	040		
	066202	115	117	104		
	066205	105	000			
14950	066207	200	103	117	MSG021: .ASCII <CRLF>/COMMANDS AVAILABLE:/	
	066212	115	115	101		
	066215	116	104	123		
	066220	040	101	126		
	066223	101	111	114		
	066226	101	102	114		
	066231	105	072			
14951	066233	200	060	040	.ASCII <CRLF>/0 = EXIT/	
	066236	075	040	105		
	066241	130	111	124		
14952	066244	200	061	040	.ASCII <CRLF>/1 = READ CSR/	
	066247	075	040	122		
	066252	105	101	104		
	066255	040	103	123		
	066260	122				
14953	066261	200	062	040	.ASCII <CRLF>/2 = LOAD CSR/	
	066264	075	040	114		
	066267	117	101	104		
	066272	040	103	123		
	066275	122				
14954	066276	200	063	040	.ASCII <CRLF>/3 = EXAMINE MEMORY/	
	066301	075	040	105		
	066304	130	101	115		
	066307	111	116	105		
	066312	040	115	105		
	066315	115	117	122		
	066320	131				
14955	066321	200	064	040	.ASCII <CRLF>/4 = MODIFY MEMORY/	
	066324	075	040	115		
	066327	117	104	111		
	066332	106	131	040		
	066335	115	105	115		
	066340	117	122	131		
14956	066343	200	065	040	.ASCII <CRLF>/5 = SELECT BANK & TEST/	
	066346	075	040	123		
	066351	105	114	105		
	066354	103	124	040		
	066357	102	101	116		
	066362	113	040	046		
	066365	040	124	105		
	066370	123	124			
14957	066372	200	066	040	.ASCII <CRLF>/6 = TYPE CONFIG MAP/	
	066375	075	040	124		
	066400	131	120	105		

	066403	040	103	117	
	066406	116	106	111	
	066411	107	040	115	
	066414	101	120		
14958	066416	200	067	040	.ASCII <CRLF>/7 = SOB-A-LONG TEST/
	066421	075	040	123	
	066424	117	102	055	
	066427	101	055	114	
	066432	117	116	107	
	066435	040	124	105	
	066440	123	124		
14959	066442	200	070	040	.ASCII <CRLF>/8 = ERROR SUMMARY/
	066445	075	040	105	
	066450	122	122	117	
	066453	122	040	123	
	066456	125	115	115	
14960	066461	101	122	131	
	066464	200	071	075	.ASCII <CRLF>/9 = REFRESH TEST/
	066467	040	040	122	
	066472	105	106	122	
	066475	105	123	110	
	066500	040	124	105	
	066503	123	124		
14961	066505	200	061	060	.ASCII <CRLF>/10 = SET FILL COUNT/
	066510	075	040	123	
	066513	105	124	040	
	066516	106	111	114	
	066521	114	040	103	
	066524	117	125	116	
	066527	124			
14962	066530	200	061	061	.ASCII <CRLF>/11 = ENTER KAMIKAZE MODE/
	066533	075	040	105	
	066536	116	124	105	
	066541	122	040	113	
	066544	101	115	111	
	066547	113	101	132	
	066552	105	040	115	
	066555	117	104	105	
14963	066560	200	061	062	.ASCII <CRLF>/12 = EXIT KAMIKAZE MODE/
	066563	075	040	105	
	066566	130	111	124	
	066571	040	113	101	
	066574	115	111	113	
	066577	101	132	105	
	066602	040	115	117	
	066605	104	105		
14964	066607	200	061	063	.ASCII <CRLF>/13 = TURN CACHE OFF/
	066612	075	040	124	
	066615	125	122	116	
	066620	040	103	101	
	066623	103	110	105	
	066626	040	117	106	
	066631	106			
14965	066632	200	061	064	.ASCII <CRLF>/14 = TURN CACHE ON/
	066635	075	040	124	
	066640	125	122	116	
	066643	040	103	101	

	066646	103	110	105	
	066651	040	117	116	
14966	066654	200	061	065	.ASCII <CRLF>/15= TEST SELECTED BANKS/
	066657	075	040	124	
	066662	105	123	124	
	066665	040	123	105	
	066670	114	105	103	
	066673	124	105	104	
	066676	040	102	101	
	066701	116	113	123	
14967	066704	200	061	066	.ASCII <CRLF>/16= TEST ALL BANKS/
	066707	075	040	124	
	066712	105	123	124	
	066715	040	101	114	
	066720	114	040	102	
	066723	101	116	113	
	066726	123			
14968	066727	200	061	067	.ASCII <CRLF>/17= ENABLE TRACE/
	066732	075	040	105	
	066735	116	101	102	
	066740	114	105	040	
	066743	124	122	101	
	066746	103	105		
14969	066750	200	061	070	.ASCII <CRLF>/18= DISABLE TRACE/
	066753	075	040	104	
	066756	111	123	101	
	066761	102	114	105	
	066764	040	124	122	
	066767	101	103	105	
14970	066772	015	012	000	.BYTE 15,12,0
14971	066775	200	127	110	MSG022: .ASCIZ <CRLF>/WHICH CSR(O-F)? /
	067000	111	103	110	
	067003	040	103	123	
	067006	122	050	060	
	067011	055	106	051	
	067014	077	040	000	
14972	067017	200	103	123	MSG023: .ASCIZ <CRLF>/CSR WORD? /
	067022	122	040	127	
	067025	117	122	104	
	067030	077	040	000	
14973	067033	200	103	123	MSG025: .ASCIZ <CRLF>/CSR DOES NOT EXIST/
	067036	122	040	104	
	067041	117	105	123	
	067044	040	116	117	
	067047	124	040	105	
	067052	130	111	123	
	067055	124	000		
14974	067057	200	103	117	MSG026: .ASCIZ <CRLF>/COMMAND:/
	067062	115	115	101	
	067065	116	104	072	
	067070	000			
14975	067071	200	117	114	MSG027: .ASCIZ <CRLF>/OLD CSR WAS/
	067074	104	040	103	
	067077	123	122	040	
	067102	127	101	123	
	067105	000			
14976	067106	200	103	123	MSG028: .ASCIZ <CRLF>/CSR IS NOW/

	067111	122	040	111	
	067114	123	040	116	
	067117	117	127	000	
14977	067122	200	105	130	MSG029: .ASCIZ <CRLF>/EXAMINE MEMORY/
	067125	101	115	111	
	067130	116	105	040	
	067133	115	105	115	
	067136	117	122	131	
	067141	000			
14978	067142	200	102	101	MSG030: .ASCIZ <CRLF>/BANK(0-177)? /
	067145	116	113	050	
	067150	060	055	061	
	067153	067	067	051	
	067156	077	040	000	
14979	067161	200	120	110	MSG031: .ASCIZ <CRLF>/PHYSICAL ADDRESS(0-17757776)? /
	067164	131	123	111	
	067167	103	101	114	
	067172	040	101	104	
	067175	104	122	105	
	067200	123	123	050	
	067203	060	055	061	
	067206	067	067	065	
	067211	067	067	067	
	067214	066	051	077	
	067217	040	000		
14980	067221	200	120	101	MSG032: .ASCIZ <CRLF>/PARITY ABORT/<32>
	067224	122	111	124	
	067227	131	040	101	
	067232	102	117	122	
	067235	124	032	000	
14981	067240	200	124	111	MSG033: .ASCIZ <CRLF>/TIMEOUT TRAP/<32>
	067243	115	105	117	
	067246	125	124	040	
	067251	124	122	101	
	067254	120	032	000	
14982	067257	200	102	131	MSG034: .ASCIZ <CRLF>/BYPASSING ECC TESTS ON BANK /
	067262	120	101	123	
	067265	123	111	116	
	067270	107	040	105	
	067273	103	103	040	
	067276	124	105	123	
	067301	124	123	040	
	067304	117	116	040	
	067307	102	101	116	
	067312	113	040	000	
14983	067315	040	104	125	MSG034: .ASCIZ / DUE TO SBE LOCATIONS/
	067320	105	040	124	
	067323	117	040	123	
	067326	102	105	040	
	067331	114	117	103	
	067334	101	124	111	
	067337	117	116	123	
	067342	000			
14984	067343	121	126	000	MSG035: .ASCIZ /QV/
14985	067346	200	115	117	MSG036: .ASCIZ <CRLF>/MODIFY MEMORY/
	067351	104	111	106	
	067354	131	040	115	

	067357	105	115	117	
	067362	122	131	000	
14986	067365	200	117	114	MSG037: .ASCIZ <CRLF>/OLD DATA WAS /
	067370	104	040	104	
	067373	101	124	101	
	067376	040	127	101	
	067401	123	040	000	
14987	067404	200	104	101	MSG038: .ASCIZ <CRLF>/DATA IS NOW /
	067407	124	101	040	
	067412	111	123	040	
	067415	116	117	127	
	067420	040	000		
14988	067422	200	111	116	MSG039: .ASCIZ <CRLF>/INPUT NEW DATA? /
	067425	120	125	124	
	067430	040	116	105	
	067433	127	040	104	
	067436	101	124	101	
	067441	077	040	000	
14989	067444	200	123	105	MSG040: .ASCIZ <CRLF>/SELECT BANK & TEST/
	067447	114	105	103	
	067452	124	040	102	
	067455	101	116	113	
	067460	040	046	040	
	067463	124	105	123	
	067466	124	000		
14990	067470	200	102	101	MSG041: .ASCIZ <CRLF>/BANK NOT ACCESSABLE/
	067473	116	113	040	
	067476	116	117	124	
	067501	040	101	103	
	067504	103	105	123	
	067507	123	101	102	
	067512	114	105	000	
14991	067515	200	124	105	MSG042: .ASCIZ <CRLF>/TEST(0-47)? /
	067520	123	124	050	
	067523	060	055	064	
	067526	067	051	077	
	067531	040	000		
14992	067533	200	124	105	MSG043: .ASCIZ <CRLF>/TEST 0 DATA IS? /
	067536	123	124	040	
	067541	060	040	104	
	067544	101	124	101	
	067547	040	111	123	
	067552	077	040	000	
14993	067555	200	124	117	MSG046: .ASCIZ <CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
	067560	040	105	123	
	067563	103	101	120	
	067566	105	040	124	
	067571	131	120	105	
	067574	040	101	116	
	067577	131	040	113	
	067602	105	131	200	
	067605	012	012	000	
14994	067610	200	124	105	MSG047: .ASCIZ <CRLF>/TEST COMPLETE/
	067613	123	124	040	
	067616	103	117	115	
	067621	120	114	105	
	067624	124	105	000	

14995	067627	040	116	117	MSG048: .ASCIZ / NOT AVAILABLE NOW TRY LATER! /
	067632	124	040	101	
	067635	126	101	111	
	067640	114	101	102	
	067643	114	105	040	
	067646	116	117	127	
	067651	040	055	040	
	067654	124	122	131	
	067657	040	114	101	
	067662	124	105	122	
	067665	041	000		
14996	067667	200	102	101	MSG049: .ASCIZ <CRLF>/BANK REQUIRES RELOCATION/
	067672	116	113	040	
	067675	122	105	121	
	067700	125	111	122	
	067703	105	123	040	
	067706	122	105	114	
	067711	117	103	101	
	067714	124	111	117	
	067717	116	000		
14997					.EVEN
14998	067722	120	117	127	MSG051: .ASCIZ /POWER RECOVERY/
	067725	105	122	040	
	067730	122	105	103	
	067733	117	126	105	
	067736	122	131	000	
14999	067741	200	123	117	MSG055: .ASCIZ <CRLF>/SOB A-LONG TEST/
	067744	102	055	101	
	067747	055	114	117	
	067752	116	107	040	
	067755	124	105	123	
	067760	124	000		
15000	067762	200	102	105	MSG056: .ASCIZ <CRLF>/BELL = EACH PASS COMPLETE/
	067765	114	114	040	
	067770	075	040	105	
	067773	101	103	110	
	067776	040	120	101	
	070001	123	123	040	
	070004	103	117	115	
	070007	120	114	105	
	070012	124	105	000	
15001	070015	200	040	040	MSG058: .ASCIZ <CRLF>/ CSR CSR .../
	070020	103	123	122	
	070023	040	040	040	
	070026	040	103	123	
	070031	122	040	056	
	070034	056	056	000	
15002	070037	077	077	077	MSG061: .ASCIZ /??????/
	070042	077	077	077	
	070045	000			
15003	070046	111	116	120	MSG062: .ASCIZ /INPUT MUST BE A/
	070051	125	124	040	
	070054	115	125	123	
	070057	124	040	102	
	070062	105	040	101	
	070065	000			
15004	070066	116	040	117	MSG063: .ASCIZ /N OCTAL /

	070071	103	124	101	
	070074	114	040	000	
15005	070077	116	125	115	MSG064: .ASCIZ /NUMBER/<CRLF>
	070102	102	105	122	
	070105	200	000		
15006	070107	040	104	105	MSG065: .ASCIZ / DECIMAL /
	070112	103	111	115	
	070115	101	114	040	
	070120	000			
15007	070121	200	105	122	MSG066: .ASCIZ <CRLF>/ERRORS > 20 - ABORTING FOR XXOP CHAIN/
	070124	122	117	122	
	070127	123	040	076	
	070132	040	062	060	
	070135	040	055	040	
	070140	101	102	117	
	070143	122	124	111	
	070146	116	107	040	
	070151	106	117	122	
	070154	040	130	130	
	070157	104	120	040	
	070162	103	110	101	
	070165	111	116	000	
15008	070170	106	101	124	MSG067: .ASCIZ /FATAL /
	070173	101	114	040	
	070176	000			
15009	070177	113	040	127	MSG070: .ASCIZ /K WORDS OF MEMORY TOTAL/<CRLF>
	070202	117	122	104	
	070205	123	040	117	
	070210	106	040	115	
	070213	105	115	117	
	070216	122	131	040	
	070221	124	117	124	
	070224	101	114	200	
	070227	000			
15010	070230	200	122	105	MSG073: .ASCIZ <CRLF>/REFRESH TEST/
	070233	106	122	105	
	070236	123	110	040	
	070241	124	105	123	
	070244	124	000		
15011	070246	200	122	105	MSG075: .ASCIZ <CRLF>/RELOCATION NOT POSSIBLE/<32>
	070251	114	117	103	
	070254	101	124	111	
	070257	117	116	040	
	070262	116	117	124	
	070265	040	120	117	
	070270	123	123	111	
	070273	102	114	105	
	070276	032	000		
15012	070300	200	040	040	MSG076: .ASCIZ <CRLF>/ BANK ERRORS/<CRLF>
	070303	102	101	116	
	070306	113	040	040	
	070311	105	122	122	
	070314	117	122	123	
	070317	200	000		
15013	070321	200	105	116	MSG077: .ASCIZ <CRLF>/END PASS #/
	070324	104	040	120	
	070327	101	123	123	

	070332	040	043	000	
15014	070335	040	105	122	MSG079: .ASCIZ / ERROR(S) DETECTED/<CRLF>
	070340	122	117	122	
	070343	050	123	051	
	070346	040	104	105	
	070351	124	105	103	
	070354	124	105	104	
	070357	200	000		
15015	070361	200	106	111	MSG085: .ASCIZ <CRLF>/FILL COUNT(OCTAL)? /
	070364	114	114	040	
	070367	103	117	125	
	070372	116	124	050	
	070375	117	103	124	
	070400	101	114	051	
	070403	077	040	000	
15016	070406	200	113	105	MSG088: .ASCIZ <CRLF>/KERNEL STACK/
	070411	122	116	105	
	070414	114	040	123	
	070417	124	101	103	
	070422	113	000		
15017	070424	200	123	125	MSG089: .ASCIZ <CRLF>/SUPERVISOR STACK/
	070427	120	105	122	
	070432	126	111	123	
	070435	117	122	040	
	070440	123	124	101	
	070443	103	113	000	
15018	070446	200	125	123	MSG090: .ASCIZ <CRLF>/USER STACK/
	070451	105	122	040	
	070454	123	124	101	
	070457	103	113	000	
15019	070462	040	111	123	MSG091: .ASCIZ / IS EMPTY/
	070465	040	105	115	
	070470	120	124	131	
	070473	000			
15020	070474	122	105	114	MSG092: .ASCIZ /RELOCATED /
	070477	117	103	101	
	070502	124	105	104	
	070505	040	040	000	
15021	070510	102	101	116	MSG093: .ASCIZ /BANK= /
	070513	113	075	000	
15022	070516	040	040	124	MSG095: .ASCIZ / TEST= /
	070521	105	123	124	
	070524	075	000		
15023	070526	200	105	116	MSG101: .ASCIZ <CRLF>/ENTERING KAMIKAZE MODE/
	070531	124	105	122	
	070534	111	116	107	
	070537	040	113	101	
	070542	115	111	113	
	070545	101	132	105	
	070550	040	115	117	
	070553	104	105	000	
15024	070556	200	114	105	MSG102: .ASCIZ <CRLF>/LEAVING KAMIKAZE MODE/
	070561	101	126	111	
	070564	116	107	040	
	070567	113	101	115	
	070570	111	113	101	
	070575	132	105	040	

	070600	115	117	104	
	070603	105	000		
15025	070605	200	114	105	MSG103: .ASCIZ <CRLF>/LEAVING FS MODE/<CRLF>
	070610	101	126	111	
	070613	116	107	040	
	070616	106	123	040	
	070621	115	117	104	
	070624	105	200	000	
15026	070627	032	000		MSG104: .BYTE 32,0 ;CONTROL Z
15027	070631	200	105	116	MSG105: .ASCIZ <CRLF>/ENTER BANKS USE NUMBER 200 TO TERMINATE/
	070634	124	105	122	
	070637	040	102	101	
	070642	116	113	123	
	070645	040	055	040	
	070650	125	123	105	
	070653	040	116	125	
	070656	115	102	105	
	070661	122	040	062	
	070664	060	060	040	
	070667	124	117	040	
	070672	124	105	122	
	070675	115	111	116	
	070700	101	124	105	
	070703	000			
15028	070704	200	103	101	MSG106: .ASCIZ <CRLF>/CACHE IS OFF/
	070707	103	110	105	
	070712	040	111	123	
	070715	040	117	106	
	070720	106	000		
15029	070722	200	103	101	MSG107: .ASCIZ <CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
	070725	103	110	105	
	070730	040	111	123	
	070733	040	117	116	
	070736	040	050	105	
	070741	130	103	105	
	070744	120	124	040	
	070747	104	125	122	
	070752	111	116	107	
	070755	040	101	103	
	070760	124	125	101	
	070763	114	040	120	
	070766	101	124	124	
	070771	105	122	116	
	070774	123	051	000	
15030	070777	200	117	116	MSG110: .ASCIZ <CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
	071002	114	131	040	
	071005	123	105	114	
	071010	105	103	124	
	071013	105	104	040	
	071016	102	101	116	
	071021	113	123	040	
	071024	127	111	114	
	071027	114	040	102	
	071032	105	040	124	
	071035	105	123	124	
	071040	105	104	000	
15031	071043	200	101	114	MSG111: .ASCIZ <CRLF>/ALL BANKS WILL BE TESTED/

	071046	114	040	102	
	071051	101	116	113	
	071054	123	040	127	
	071057	111	114	114	
	071062	040	102	105	
	071065	040	124	105	
	071070	123	124	105	
	071073	104	000		
15032	071075	113	040	117	MSG112: .ASCIZ /K OF Q-BUS PARITY MEMORY/<CRLF>
	071100	106	040	121	
	071103	055	102	125	
	071106	123	040	120	
	071111	101	122	111	
	071114	124	131	040	
	071117	115	105	115	
	071122	117	122	131	
	071125	200	000		
15033	071127	113	040	117	MSG113: .ASCIZ /K OF ECC MEMORY/<CRLF>
	071132	106	040	105	
	071135	103	103	040	
	071140	115	105	115	
	071143	117	122	131	
	071146	200	000		
15034	071150	200	040	040	M1184: .ASCIZ <CRLF>" 11/84"
	071153	040	061	061	
	071156	057	070	064	
	071161	000			
15035	071162	200	040	040	MSG117: .ASCIZ <CRLF>" 11/83"
	071165	040	061	061	
	071170	057	070	063	
	071173	000			
15036	071174	200	040	040	MSG119: .ASCIZ <CRLF>/ NO/
	071177	040	116	117	
	071202	000			
15037	071203	040	103	101	MSG120: .ASCIZ / CACHE AVAILABLE/
	071206	103	110	105	
	071211	040	101	126	
	071214	101	111	114	
	071217	101	102	114	
	071222	105	000		
15038	071224	040	103	101	MSG121: .ASCIZ / CACHE BYPASSED/
	071227	103	110	105	
	071232	040	102	131	
	071235	120	101	123	
	071240	123	105	104	
	071243	000			
15039	071244	200	103	123	MSG122: .ASCII <CRLF>/CSR NUMBER /
	071247	122	040	116	
	071252	125	115	102	
	071255	105	122	040	
15040	071260	000			MSG122: .BYTE 0
15041	071261	040	103	117	.ASCIZ / CONTROLS TOO MANY BANKS/
	071264	116	124	122	
	071267	117	114	123	
	071272	040	124	117	
	071275	117	040	115	
	071300	101	116	131	

	071303	040	102	101	
	071306	116	113	123	
	071311	000			
15042	071312	040	120	101	MSG125: .ASCIZ / PASSES COMPLETED/
	071315	123	123	105	
	071320	123	040	103	
	071323	117	115	120	
	071326	114	105	124	
	071331	105	104	000	
15043	071334	200	120	122	MSG126: .ASCIZ <CRLF>/PROGRAM CSR COULD NOT BE DETERMINED/
	071337	117	107	122	
	071342	101	115	040	
	071345	103	123	122	
	071350	040	103	117	
	071353	125	114	104	
	071356	040	116	117	
	071361	124	040	102	
	071364	105	040	104	
	071367	105	124	105	
	071372	122	115	111	
	071375	116	105	104	
	071400	000			
15044	071401	200	124	122	MSG127: .ASCIZ <CRLF>/TRACE ENABLED/
	071404	101	103	105	
	071407	040	105	116	
	071412	101	102	114	
	071415	105	104	000	
15045	071420	200	124	122	MSG128: .ASCIZ <CRLF>/TRACE DISABLED/
	071423	101	103	105	
	071426	040	104	111	
	071431	123	101	102	
	071434	114	105	104	
	071437	000			
15046	071440	200	200	040	MSG008: .ASCIZ <CRLF><CRLF>/
	071443	040	040	040	CSR MAP/<CRLF>
	071446	040	040	040	
	071451	040	040	040	
	071454	040	040	040	
	071457	040	040	103	
	071462	123	122	040	
	071465	115	101	120	
	071470	200	000		
15047	071472	200	040	103	MSG000: .ASCIZ <CRLF>" CVMJABO ECC/PARITY MEMORY DIAGNOSTIC"
	071475	126	115	112	
	071500	101	102	060	
	071503	040	040	105	
	071506	103	103	057	
	071511	120	101	122	
	071514	111	124	131	
	071517	040	115	105	
	071522	115	117	122	
	071525	131	040	104	
	071530	111	101	107	
	071533	116	117	123	
	071536	124	111	103	
	071541	000			
15048	071542	200	200	000	MSG129: .ASCIZ <CRLF><CRLF>

15049	071545	120	122	117	MSG130: .ASCIZ /PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC/
	071550	103	105	123	
	071553	123	117	122	
	071556	040	116	117	
	071561	124	040	123	
	071564	125	120	120	
	071567	117	122	124	
	071572	105	104	040	
	071575	102	131	040	
	071600	124	110	111	
	071603	123	040	104	
	071606	111	101	107	
	071611	116	117	123	
	071614	124	111	103	
	071617	000			

15050	071620	200	125	116	NOUBMT: .ASCIZ <CRLF>/UNIBUS MEMCRY WILL NOT BE TESTED/<CRLF>
	071623	111	102	125	
	071626	123	040	115	
	071631	105	115	117	
	071634	122	131	040	
	071637	127	111	114	
	071642	114	040	116	
	071645	117	124	040	
	071650	102	105	040	
	071653	124	105	123	
	071656	124	105	104	
	071661	200	000		

15051					
15057	071664				
15058	071664				
15059	071664	010414			
15063		000200			

```
      .EVEN  
      #END  
END:  .PRINT 60000-SUPLIMIT ;SUPERVISOR ADDRESSES LEFT  
      .END START3
```

ABORT	052220	BRG08B	027500	B61	035004	CPUERR=	177766	DH13	064242
ABORTC	052252	BSIZE	002374	B62	035210	CR	= 000015	DH14	064337
ABORTE	052316	B0	004740	B63	035214	CRLF	= 000200	DH15	064416
ABORTF	002144	B1	005744	B64	036352	CSR	002150	DH16	064535
ABORTZ	052342	B10	012350	B65	036360	CSRADD=	172100	DH19	064552
ACFLAG	002116	B100	045746	B66	036510	CSRBMP	006112	DH2	063606
ACTFLA	002350	B101	045754	B67	036524	CSRCAS	016672	DH20	064557
ADDRES	002034	B102	053572	B7	012306	CSRFB	002232	DH23	064636
ANA2	007460	B103	053576	B70	043416	CSRFR	002232	DH24	064715
APTDOW	040626	B104	053674	B71	043654	CSRHOL	002526	DH25	064764
APTECC	002424	B105	053716	B72	044136	CSRINC	002330	DH26	065022
APTFLA	002352	B106	054052	B73	044432	CSRINF	002462	DH27	065040
APTHAN	014014	B11	013076	B74	044432	CSrint	002236	DH3	063631
APTHLT	040674	B12	013146	B75	044530	CSRLAS	002230	DH30	065114
APTPAR	002422	B13	013210	B76	045470	CSRLBA	002234	DH5	063665
APTSIZ	002444	B14	014060	B77	045474	CSRL00	002332	DH6	063744
BACK	051440	B15	015476	CACHKF	002550	CSRMAP	006122	DH7	064011
BACKGN	032516	B16	015746	CACHKN	002544	CSRNO	002152	DIAGFL	002008
BAD	002052	B17	015754	CACHOF=	104424	CSROUT	035202	DISPLA	002640
BADPC	002022	B2	006136	CACHON=	104423	CSRSTU=	000020	DISPRE=	000174
BADPSW	002032	B20	015772	CACHVE=	000114	CSR15	002322	DISPTB	013466
BADSP	002026	B21	016034	CBCSR	= 104474	CTEST	007002	DOBACK	014050
BADSTA	034040	B22	016106	CBITS	002316	CTLKVE	002146	DONE	006766
BADXOR	002060	B23	021530	CBREG	= 104513	DATARG=	177754	DSWR	= 177570
BAD2	002054	B24	021534	CB1CSR=	104475	DATBUF	002242	DT1	060214
BAD3	002056	B25	021714	CHECK	002314	DBEMSK	002256	DT10	060334
BAFPAF	014150	B26	021722	CHKDIS=	104504	DDISP	= 177570	DT11	060356
BAFPAR	014256	B27	022152	CHKTRP	032106	DEENER=	104421	DT12	060364
BAKPAT	002632	B3	006206	CHK1DI=	104505	DETAIL	053336	DT13	060370
BANK	002102	B30	025636	CKEND	055126	DEYFLA	002220	DT14	060412
BANKIN	002104	B31	026026	CKSMR	= 104410	DETPSW	002216	DT16	060430
BANKMO	037350	B32	026064	CLRCSR=	104502	DETRO	002200	DT17	060460
BANKOK	040364	B33	026506	CLREX	007430	DETR1	002202	DT2	060226
BAMPAF	014364	B34	026544	CLRMEM	007320	DETR2	002204	DT20	060466
BAMPAR	014514	B35	026560	CLR1CS=	104503	DETR3	002206	DT22	060476
BGTEST	027476	B36	026700	CMD16A	044516	DETR4	002210	DT23	060504
BITNO	002324	B37	027060	CMD16L=	000073	DETR5	002212	DT24	060526
BIT0	= 000001	B4	006466	CMD5B	042660	DETSF	002214	DT25	060546
BIT1	= 000002	B40	027102	CMD5C	043154	DET1	054026	DT26	060560
BIT10	= 002000	B41	027630	CMD7B	043412	DF1	060626	DT27	060566
BIT11	= 004000	B42	027770	CMD7C	043466	DF11	060751	DT3	060232
BIT12	= 010000	B43	030026	CMD9B	044132	DF13	060762	DT30	060606
BIT13	= 020000	B44	030320	CMD9C	044206	DF14	060772	DT4	060242
BIT14	= 040000	B45	030534	CONFGE	002450	DF15	061001	DT5	060252
BIT15	= 100000	B46	030736	CONFIE	003670	DF16	061010	DT6	060270
BIT2	= 000004	B47	031112	CONF IG	002664	DF2	060627	DT7	060304
BIT3	= 000010	B5	006522	CONTFI	002222	DF3	060653	DUPPY	002176
BIT4	= 000020	B50	031132	CONTRL=	177746	DF4	060666	DUMPCS=	000102
BIT5	= 000040	B51	031142	CONTS	055276	DF5	060701	ECCDIS=	104470
BIT6	= 000100	B52	031460	CONTS1	054640	DF6	060714	ECCINI=	104472
BIT7	= 000200	B53	031630	CONTS2	055300	DF7	060727	ECC1DI=	104471
BIT8	= 000400	B54	031644	CONTS3	047246	DF8	060742	ECC1IN=	104473
BIT9	= 001000	B55	031710	CONTT	055222	DF9	060744	EMTSAV	003670
BLOCK1	040676	B56	034634	COUNT	002370	DH1	063551	EMTVEC=	000030
BLOCK2	040716	B57	034640	CPERRF	052364	DH10	064121	EM11	061317
BLOCK3	040732	B6	011632	CPSAVE	052362	DH11	064217	EM12	061341
BMFLAG	002130	B60	035000	CPUBIT	002106	DH12	064235	EM13	061365

EM14	061417	E0	004766	E63	035244	HEADER	002612	LOADBA	002432
EM15	061463	E1	006064	E64	036510	HIPAT	040416	LOADCS-	104425
EM17	061531	E10	012434	E65	036470	HOLDLO-	000015	LOADER-	000064
EM19	061571	F100	046010	E66	036724	HT	000011	LOADHO	002576
EM2	061017	E101	046010	E67	036636	I	002452	LOOP	013440
EM20	061643	E102	053640	E7	012450	IBSAVE	052360	LSIZE	002400
EM21	061722	E103	053640	E70	043460	IIII	177777	LST##	000000
EM22	061756	E104	053724	E71	043756	ILLCSR	013012	LWDBE	000060
EM23	062003	E105	054020	E72	044200	IMPTES	012036	LWSBE	000056
EM24	062032	E106	054104	E73	044472	INCBNK	040426	LO	004750
EM25	062111	E11	013140	E74	044472	INCPAT	040402	L1	005012
EM26	062136	E12	013206	E75	044546	INCRPT	040402	L10	005310
EM27	062207	E13	013236	E76	045530	INMBAN	002540	L100	012422
EM29	062277	E14	014146	E77	045530	INNECC	002536	L101	012426
EM3	061055	E15	015516	FASTCI-	177640	INTFLA	002136	L102	013130
EM30	062361	E16	016302	FATAL#	002064	INT64K	002140	L103	013124
EM32	062471	E17	016262	FCMD10	044272	INVALI-	104511	L104	013130
EM33	062576	E2	006202	FCMD11	044320	IOTVEC-	000020	L105	013224
EM35	062704	E20	016212	FCMD12	044342	JMPRL1	037004	L106	013274
EM36	062771	E21	016212	FCMD13	044362	KAL	051424	L107	013440
EM4	061107	E22	016162	FCMD14	044404	KAMIKA	002006	L11	005316
EM40	063040	E23	021672	FCMD15	044422	KAMITE	024122	L110	013554
EM5	061155	E24	021656	FCMD16	044506	KDIAG	000010	L111	013664
EM50	063112	E25	022060	FCMD17	044550	KDPAR0-	172360	L112	013674
EM51	063146	E26	022044	FCMD18	044564	KDPAR6-	172374	L113	014044
EM52	063216	E27	022230	FIELDS	040762	KDPAR7-	172376	L114	014044
EM53	063243	E3	006276	FINDBA-	000066	KERNEL-	104417	L115	014132
EM55	063272	E30	025722	FINT#	007302	KERSTK-	002000	L116	014244
EM56	063313	E31	026302	FIRST	060000	KFLAG	002530	L117	014352
EM57	063345	E32	026270	FLIPL0	002616	KIPAR0-	172340	L12	005366
EM6	061232	E33	027052	FLIPWA	032366	KIPAR4-	172350	L120	014502
EM60	063413	E34	027036	FLUSH	013560	KIPAR5-	172352	L121	014632
EM61	063455	E35	026652	FSCMD0	041160	KIPAR6-	172354	L122	014762
EM62	063476	E36	027022	FSCMD1	041262	KIPDR0-	172300	L123	015134
EM7	061257	E37	027076	FSCMD2	041372	KMAP	104422	L124	015264
ENASBE-	104506	E4	006710	FSCMD3	041540	KPFLAG	002114	L125	015436
ENASB-	104507	E40	027122	FSCMD4	042014	KSIZE	002376	L126	015516
END	071664	E41	027752	FSCMD5	042334	KSTACK	002574	L127	015554
ENDADD	002562	E42	030156	FSCMD6	043252	LAST	157776	L13	005362
ENDFLG	002564	E43	030144	FSCMD7	043260	LASTBA	002526	L130	015564
ENERGI-	104420	E44	030424	FSCMD8	043552	LASTBL	002560	L131	016212
ENEXBK	040354	E45	030710	FSCMD9	044000	LASTER	002016	L132	016174
EQFLAG	002132	E46	031106	FSINFL	002442	LBLS0	000520	L133	016244
ERRADD	002460	E47	031430	FSPAT	043034	LBLS1	000106	L134	016514
ERRGEN-	104512	E5	006700	FSSTAC	002306	LBLS2	000510	L135	016624
ERRMAX	002554	E50	031412	FS1	041046	LBLS3	000503	L136	016762
ERROR	104000	E51	031376	FS7FLA	002446	LBLS4	000356	L137	017010
ERRPC	002020	E52	031614	FULLRE	002542	LBLS5	000360	L14	005366
ERRPSW	002030	E53	032100	GBLENG-	000076	LBLS6	000022	L140	017014
ERRSP	002024	E54	032070	GETCSR	032562	LCSROU-	000062	L141	017016
ERRVEC-	000004	E55	032056	GETDAT	045160	LCSRRE-	000100	L142	017062
EVEN	002364	E56	034706	GETDA1	045256	LCSRSA-	000076	L143	017116
EXBANK	037760	E57	034706	GETDIS	051336	LEGALC-	000010	L144	017122
EXCMD3	041746	E6	011764	GOOD	002044	LF	000012	L145	017124
EXCMD4	042266	E60	035052	GOOD2	002046	LINK1	002522	L146	017272
EXIT	040512	E61	035052	GOOD3	002050	LINK2	002524	L147	017652
EXIT2	040516	E62	035244	GTSWR	104407	LKS	177546	L15	005676

L150	020256	L237	026734	L334	034426	L416	043712	L501	052164
L151	020266	L24	006244	L335	034664	L417	044050	L502	052114
L152	020266	L240	026766	L336	034664	L42	006640	L503	052126
L153	020360	L241	026752	L337	034726	L420	044052	L504	052164
L154	020370	L242	026764	L34	006462	L421	044164	L505	052174
L155	020370	L243	027010	L340	034734	L422	044454	L506	052670
L156	020724	L244	026776	L341	035030	L423	044456	L507	053616
L157	020756	L245	027010	L342	035030	L424	044722	L51	006766
L16	005676	L246	027122	L343	035072	L425	044736	L510	054022
L160	021022	L247	030022	L344	035100	L426	044742	L511	054026
L161	021642	L25	006234	L345	035222	L427	044760	L512	054106
L162	021620	L250	030026	L346	036324	L43	006642	L513	054112
L163	021642	L251	030112	L347	036340	L430	045106	L514	055114
L164	021706	L252	030130	L35	006516	L431	045110	L515	055120
L165	022030	L253	030134	L350	036352	L432	045154	L516	055242
L166	022006	L256	030252	L351	036352	L433	045400	L517	055332
L167	022030	L257	030410	L352	036454	L434	045506	L52	010724
L17	006012	L26	006242	L353	036504	L435	045766	L520	055334
L170	022214	L261	030456	L354	036626	L436	047304	L53	010730
L171	022214	L262	030464	L355	036626	L437	047306	L54	010732
L172	022322	L263	030610	L356	036626	L44	006664	L55	011462
L173	022274	L264	030660	L357	036606	L440	050136	L56	011476
L174	022352	L266	031020	L36	006522	L441	050150	L57	011600
L175	022542	L267	031070	L360	036626	L442	050166	L6	005316
L176	023056	L27	006252	L361	036670	L443	050170	L60	011604
L177	023312	L271	031242	L362	036670	L444	050210	L61	011750
L2	005120	L272	031316	L363	036720	L445	050222	L62	011750
L20	006040	L273	031334	L364	040112	L446	050240	L63	011750
L200	023332	L274	031362	L365	040302	L447	050242	L64	011750
L201	023432	L3	005142	L366	040464	L45	006670	L65	011744
L202	023612	L300	031524	L367	040532	L450	050256	L66	012022
L203	023620	L301	031566	L37	006552	L451	050460	L67	012026
L204	023756	L303	031704	L370	040536	L452	050466	L7	005316
L205	023764	L304	031710	L371	040544	L453	050514	L70	012030
L206	024144	L305	031762	L372	040560	L454	050526	L71	012212
L207	024152	L306	031764	L373	040572	L455	050544	L72	012212
L21	006056	L307	032022	L374	041004	L456	050556	L73	012212
L210	024156	L31	006350	L375	041014	L457	050570	L74	012216
L211	025572	L310	032042	L376	041150	L460	050612	L75	012434
L212	025630	L311	032046	L377	041210	L461	050660	L76	012434
L213	025710	L315	032140	L4	005160	L462	050740	L77	012424
L215	026060	L316	032136	L40	006576	L463	050754	MAINT	177750
L216	026064	L317	032160	L400	041214	L464	050756	MAPHO	170202
L217	026160	L32	006354	L401	041244	L465	051072	MAPKER	037646
L220	026236	L320	032160	L402	041256	L466	051162	MAPLO	170200
L221	026254	L321	032276	L403	042444	L467	051172	MAPL1	170204
L222	026260	L322	032344	L404	042504	L470	051514	MAPPER	035604
L225	026530	L323	032722	L405	042544	L471	051514	MASK	002320
L226	026540	L324	033322	L406	042752	L472	051754	MBERR	012670
L227	026652	L325	033416	L407	042766	L473	051710	MEMDON	013506
L230	026602	L326	033406	L41	006650	L474	051734	MFPT	000007
L231	026614	L327	033414	L410	043000	L475	051754	MJPAT	017316
L232	026632	L33	006420	L411	043330	L476	051774	MJTEST	017212
L233	026640	L330	033424	L412	043332	L477	052166	MKCNT	016364
L234	026664	L331	033450	L413	043444	L5	005236	MKCONT	015544
L235	027022	L332	033552	L414	043756	L50	006724	MKCSRT	016702
L236	026722	L333	034420	L415	043742	L500	052036	MKFLAG	002120

MKLOOP	015726	MSG047	067610	MTPA03	024636	MT0022	020714	PARITY	033632
MKPAT	017132	MSG048	067627	MTPA04	024774	MT0023	020746	PARTHE	002304
MKTEST	016772	MSG049	067667	MTPA21	026330	MT0024	021012	PARVEC	000114
MNR0	177572	MSG051	067722	MTPA24	027154	MT0026	021220	PASCNT	002570
MNR1	177574	MSG055	067741	MTPA26	027246	MT0027	021476	PASFLG	002264
MNR2	177576	MSG056	067762	MTPB03	024676	MT0030	022066	PASSNO	002266
MNR3	172516	MSG058	070015	MTPB04	025030	MT0031	022342	PATERR	002074
MNTRAP	034014	MSG061	070037	MTPB21	026360	MT0032	022532	PATPLU	004732
MNVEC	000250	MSG062	070046	MTPB24	027214	MT0033	023046	PATTER	002112
MONFLG	002276	MSG063	070066	MTPB26	027262	MT0034	023234	PCBUMP	002326
MSEEDM	002606	MSG064	070077	MTPC03	024736	MT0035	023334	PCONFI	032610
MSEEDL	002610	MSG065	070107	MTPC21	026414	MT0036	023446	PCONFS	033222
MSG012	071260	MSG066	070121	MTPC24	027230	MT0037	023510	PCONF1	033170
MSG034	067257	MSG067	070170	MTPC26	027316	MT0041	023546	PCONF2	033170
MSG034	067315	MSG070	070177	MTPD03	024754	MT0042	023626	PDP110	034025
MSG000	071472	MSG073	070230	MTPD21	026450	MT0043	023662	PD1	045400
MSG001	065173	MSG075	070246	MTPD26	027336	MT0044	023712	PORA05	047522
MSG002	065255	MSG076	070300	MTP000	024526	MT0045	023772	PERBNK	050354
MSG003	065332	MSG077	070321	MTP001	024552	MT0046	024022	PERECC	050434
MSG004	065437	MSG079	070335	MTP002	024604	MT0047	024052	PERRAB	050172
MSG005	065545	MSG085	070361	MTP005	025050	MT0999	024106	PERRAW	050120
MSG006	065557	MSG088	070406	MTP006	025104	MT1	015524	PERRA3	044746
MSG008	071440	MSG089	070424	MTP007	025304	MT2	015530	PERRA7	050244
MSG009	065614	MSG090	070446	MTP010	025404	MUT	002110	PERR01	104427
MSG010	065626	MSG091	070462	MTP014	025512	M1184	071150	PERR02	104430
MSG011	065726	MSG092	070474	MTP017	025736	NC	047306	PERR03	104431
MSG012	066014	MSG093	070510	MTP020	026014	NEMCNT	002070	PERR04	104432
MSG013	066126	MSG095	070516	MTP022	026500	NEWMAN	002310	PERR05	047516
MSG014	066130	MSG101	070526	MTP030	027354	NEWKER	037600	PERR06	047544
MSG015	066132	MSG102	070556	MTP031	027364	NEWLOA	037702	PERR07	104433
MSG016	066134	MSG103	070605	MTP032	027442	NOABRT	052356	PERR10	104434
MSG017	066146	MSG104	070627	MTP033	027474	NOCH	054622	PERR11	104435
MSG018	066157	MSG105	070631	MTP034	027572	NOERRO	002430	PERR12	104436
MSG019	066162	MSG106	070704	MTP035	027616	NOFSMO	002426	PERR13	104437
MSG020	066166	MSG107	070722	MTP036	027760	NONEH	002100	PERR14	104440
MSG021	066207	MSG11A	065640	MTP037	030204	NONEXI	033736	PERR15	104441
MSG022	066775	MSG11B	066111	MTP041	030256	NOOJ	032722	PERR16	104442
MSG023	067017	MSG11C	066113	MTP042	030430	NOOJ1	033032	PERR17	104443
MSG025	067033	MSG11D	066115	MTP043	030716	NOPAR	002076	PERR20	104444
MSG026	067057	MSG110	070777	MTP044	031112	NORES	003776	PERR21	104445
MSG027	067071	MSG111	071043	MTP045	031434	NOSCOP	002440	PERR22	104446
MSG028	067106	MSG112	071075	MTP046	031622	NOSUPE	002456	PERR23	104447
MSG029	067122	MSG113	071127	MTP047	032162	NOTAB	002372	PERR24	104450
MSG030	067142	MSG117	071162	MTST3	011502	NOTRCE	051106	PERR25	104451
MSG031	067161	MSG119	071174	MT0000	017376	NOUBMT	071620	PERR26	104452
MSG032	067221	MSG120	071203	MT0001	017432	NO22BI	002454	PERR27	104453
MSG033	067240	MSG121	071224	MT0002	017526	NULLFL	002344	PERR30	104454
MSG035	067343	MSG122	071244	MT0003	017642	HXTCSR	006040	PERR31	104455
MSG036	067346	MSG125	071312	MT0004	017756	OLDCAC	002302	PERR32	104456
MSG037	067365	MSG126	071334	MT0005	020026	OLDCSR	002156	PERR33	104457
MSG038	067404	MSG127	071401	MT0006	020110	ONES	002614	PERR34	104460
MSG039	067422	MSG128	071420	MT0007	020144	PADDRE	002036	PERR35	104461
MSG040	067444	MSG129	071542	MT0010	020206	PAFBAF	014644	PERR36	104462
MSG041	067470	MSG130	071545	MT0014	020242	PAFBAW	014774	PERR37	104463
MSG042	067515	MSIZE	002402	MT0017	020322	PARBAF	015146	PERR40	104464
MSG043	067533	MTA030	022100	MT0020	020344	PARBAW	015276	PERR41	104465
MSG046	067555	MTEST	015450	MT0021	020424	PARCNT	002072	PERR42	104466

PFRR43- 104467	SDPAR0- 172260	SUPSTK- 000740	TSTRD1 034430	ZEROS 002336
PERXOR 050330	SDPAR5- 172272	SVKPAR 036306	TSTREA- 104510	#APTHD 057020
PFECDF 053002	SDPAR6- 172274	SWAPAT 002634	TST1 005716	#AUTO 002062
PFECDM 052742	SDPAR7- 172276	SWR 002636	TST2 010546	#BANK 002013
PFECDT 052772	SEEDMI 002602	SWREG - 000176	TST3 010732	#BASE 056772
PFECEN 052706	SEEDLO 002604	SWREND 055124	TST4 011614	#BELL 002653
PFECWS 052676	SELONL 002002	SWRFLG 002566	TST5 013440	#CACHF 034144
PFLAG 002122	SETPAT 040416	SWO = 000001	TST6 013512	#CACHN 034120
PGMCSR 002532	SHADL1 011532	SW1 = 000002	TYPDS = 104405	#CBCSR 034560
PHEBE 012672	SHUTUP 040544	SW10 = 002000	TYPEIT- 104401	#CBREG 035550
PHYADD 002040	SIPAR0- 172240	SW11 = 004000	TYPOC = 104402	#CB1CS 034602
PMEMFL 002142	SIPAR3- 172246	SW12 = 010000	TYPDS = 104403	#CDW1 056776
PROTYP 004064	SIPAR5- 172252	SW13 = 020000	TYPS0 = 000000	#CDW2 057000
PSIZE 002404	SIPAR6- 172254	SW14 = 040000	TYPS1 = 000002	#CHARC 047342
PSW = 177776	SIPDR0- 172200	SW15 = 100000	TYPS2 = 000000	#CHKDI 035154
PTABLE 030164	SIZE = 040000	SW2 = 000004	TYPS3 = 000000	#CHKID 035170
PWRVEC- 000024	SKIPKA 002010	SW3 = 000010	TYPS4 = 000000	#CKSMR 054600
QUICK 002436	SKIPMK 002342	SW4 = 000020	TYPS5 = 000000	#CLRCS 035132
QUIT 040442	SKJ 051134	SW5 = 000040	TYPS6 = 000002	#CLR1C 035144
QUIT1 040506	SKPERR 002066	SW6 = 000100	UDPAR0- 177660	#CHTAG 002000
QVFLAG 002346	SKUB 036720	SW7 = 000200	UDPAR7- 177676	#CHTGE 002544
RANODD 027276	SKUJ 012674	SW8 = 000400	UFDLFG 003762	#CNTLC 056706
RDCHR = 104411	SOBK 002572	SW9 = 001000	UFDSET 002000	#CNTLG 056020
RDDEC = 104414	SOBLEN- 000056	SYNREG- 104514	UIPAR0- 177640	#CNTLK 055214
RDLIN = 104412	SOFTPA 002620	SYSSIZ 004066	UIPAR1- 177642	#CNTLU 056013
RDOCT = 104413	SOURCE 002312	TAG2# 011134	UIPAR2- 177644	#CPUOP 056744
REAACS- 104426	SPLTCS 002240	TAG3# 011170	UIPAR3- 177646	#CRLF 002660
READON 002410	SSP =#000006	TAG4# 024252	UIPAR4- 177650	#DBLK 054570
REALPA 002300	ST = 177776	TAG70# 053006	UIPAR5- 177652	#DB20 056576
REFRES 027054	STACK = 002000	TAG71# 053016	UIPAR6- 177654	#DDW0 057004
REFSUB 027124	START 003670	TAG72# 053026	UIPDRO- 177600	#DDW1 057006
REGCOP 032356	START1 000300	TAG73# 053076	UNITOP 002416	#DDW2 057010
RELENT 036730	START2 000310	TAG74# 053136	UNMAP 003734	#DDW3 057012
RELOCA 036310	START3 000200	TAG75# 053150	UNRELO 037122	#DDW4 057014
RELOC1 036744	STAR27 021522	TAG76# 053162	UPPFLG 002265	#DDW5 057016
RESREG- 104416	STOPOK 002420	TAG77# 053226	UQUIET 003764	#DDW7 057002
RESTAR 002626	STRIPE 002366	TAG78# 053234	USERMA 037516	#DEENE 034110
RESVEC- 000010	SUBAAA 004770	TAG79# 053314	USESTK- 000700	#DEVCT 056726
RESO 041256	SUBAAB 005120	TAG9# 010776	USP =#000006	#DEVM 056774
RES1 041336	SUBAAI 011526	TBG4# 024430	VPKOR 003766	#DIDDO- 000000
RES2 041504	SUBAAP 013054	TCFIG1 033272	WARN1 011064	#DOAGA 014044
RLFLAG 002126	SUBAAR 012260	TCFIG2 033332	WARN2 024650	#DOAGN 013740
RRFLAG 002124	SUBAAS 010542	TCFIG3 033466	WARN3 024664	#DOWN 045630
RTNVAL-#000000	SUCCESS 002334	TCONFI 033224	WARN4 024710	#DTBL 054560
RWCSR 006306	SUPDOA 002262	TEMP 002434	WARN5 024724	#ECCDI 034456
SAVCSR 002154	SUPD01 024156	TESTAD 002412	WASDBE- 104500	#ECCIN 034504
SAVMON 002274	SUPD02 024172	TESTMO 002552	WASSBE- 104476	#ECC1D 034472
SAVPAR 002272	SUPD03 024334	TIME 002340	WAS1DB- 104501	#ECC1I 034520
SAVREG- 104415	SUPD04 024350	TIMEOU 034002	WAS1SB- 104477	#ENASB 034532
SAV30 003756	SUPDR0 002160	TKVEC = 000060	WHICHC 044576	#ENAS1 034546
SAV32 003760	SUPDR1 002162	TMFLAG 002134	WOOPEN 046562	#ENDAD 013730
SAV4 002270	SUPDR2 002164	TOOMAN 002406	WOOPS 046214	#ENERG 034100
SBEMSK 002252	SUPDR3 002166	TOTCSR 002224	WOOPSA 046612	#ENV 056736
SBENT 016644	SUPDR4 002170	TRACE 006304	WOOPUP 046400	#ENV# 056737
SBESYN 026310	SUPDR5 002172	TRAPVE- 000034	WORST 002600	#EOP 013564
SBETES 016366	SUPDR6 002174	TSTBAN 011370	XOCHAR 047154	#ERFLG 002014
SCOPE = 000004	SUPLIM 047364	TSTDAT 002246	XXDPCH 002354	#ERRGE 035300

Symbol table

\$ERROR	051410	\$L\$	= 000000	\$PATMA	002012	\$PWRDN	045260	\$TRPAD	057076
\$ERRTB	057344	\$MADR1	056750	\$PER01	047364	\$PWRUP	045634	\$TSTM	057024
\$ERRTY	052366	\$MADR2	056754	\$PER02	047412	\$QUES	002657	\$TSTRD	034276
\$ERTTL	002630	\$MADR3	056760	\$PER03	047440	\$R	= 177777	\$TTYIN	055762
\$ESCAP	002362	\$MADR4	056764	\$PER04	047470	\$RAND	056502	\$TYPDS	054354
\$ETABL	056736	\$MAIL	056716	\$PER07	047552	\$RDCHR	055334	\$TYPE	047030
\$ETEND	057020	\$MAMS1	056746	\$PER10	047574	\$PDDEC	056220	\$TYPEC	047156
\$EXMAL	040536	\$MAMS2	056752	\$PER11	047624	\$RDLIN	055464	\$TYPEX	047344
\$E\$	= 000001	\$MAMS3	056756	\$PER12	047644	\$RDOCT	056050	\$TYPOC	054152
\$FATAL	056720	\$MAMS4	056762	\$PER13	047666	\$READC	034256	\$TYPON	054166
\$FILLC	002652	\$MBADR	057022	\$PER14	047706	\$RESRE	056444	\$TYPOS	054126
\$FILLS	002357	\$MNEW	056036	\$PER15	047730	\$SAVRE	056406	\$T1	= 000000
\$F\$	= 000000	\$MSGAD	056732	\$PER16	047752	\$SAVR\$	046212	\$T2	= 000520
\$GTSWR	054754	\$MSGLG	056734	\$PER17	047772	\$SCOPE	051054	\$UNIT	056730
\$HALT	052010	\$MSGTY	056716	\$PER20	050010	\$STN	= 000001	\$UNITH	057030
\$HALT2	057074	\$MSWR	056025	\$PER21	050026	\$SVLAD	051310	\$USWR	056742
\$HIBTS	057020	\$MTYP1	056747	\$PER22	050046	\$SV\$	= 000000	\$VECT1	056766
\$HIOCT	056216	\$MTYP2	056753	\$PER23	050064	\$SWR	= 163000	\$VECT2	056770
\$ILLUP	046206	\$MTYP3	056757	\$PER24	050102	\$SWREG	056740	\$WASDB	034766
\$INVAL	035250	\$MTYP4	056763	\$PER25	044664	\$SYNRE	035566	\$WASSB	034622
\$ITEMB	002015	\$NOTRA	057070	\$PER26	050272	\$T	= 000521	\$WAS1D	035102
\$I\$	= 000001	\$NULL	002356	\$PER27	050312	\$TESTN	056722	\$WAS1S	034736
\$KERNE	034070	\$NWTST	= 000001	\$PER30	045112	\$TKB	002644	\$XTSTR	051202
\$KMAP	036202	\$OCNT	054350	\$PER31	050502	\$TKS	002642	\$Y\$	= 000000
\$K\$	= 000102	\$OCTVL	056700	\$PER32	050600	\$TN	= 000007	\$ZAP42	013710
\$L	= 000107	\$OCT8	= 056704	\$PER33	050646	\$TPB	002650	\$Z\$	= 000000
\$LF	002661	\$OMODE	054352	\$PER34	050726	\$TPFLG	002360	\$S	= 000000
\$LL	= 000105	\$OVER	051324	\$PER35	050760	\$TPS	002646	\$T	= 000502
\$LOADC	034162	\$O\$	= 000000	\$PER36	051014	\$TRAP	057034	\$TT	= 000510
\$LPADR	002622	\$PASS	056724	\$PER37	051044	\$TRAP2	057056	\$OFILL	054351
\$LPERR	002624	\$PASTH	057026	\$PER40	051050				

. ABS. 071664 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 1377
 Work file writes: 1005
 Size of work file: 27640 Words (108 Pages)
 Size of core pool: 17990 Words (69 Pages)
 Operating system: RSX-11M/PLUS

Elapsed time: 00:12:53.05
 CVMJAB.BIC,CVMJAB.LST/-SP/NL:TOC=CVMJAB/ML,CVMJAB.MAC