

This page contains a dense grid of approximately 15 columns and 15 rows of small diagrams. Each diagram is a technical representation of memory, likely showing bit patterns, addresses, and data flow. The diagrams are arranged in a regular grid across the page, with some variations in layout and content between individual cells. The overall appearance is that of a comprehensive memory diagnostic or configuration manual.

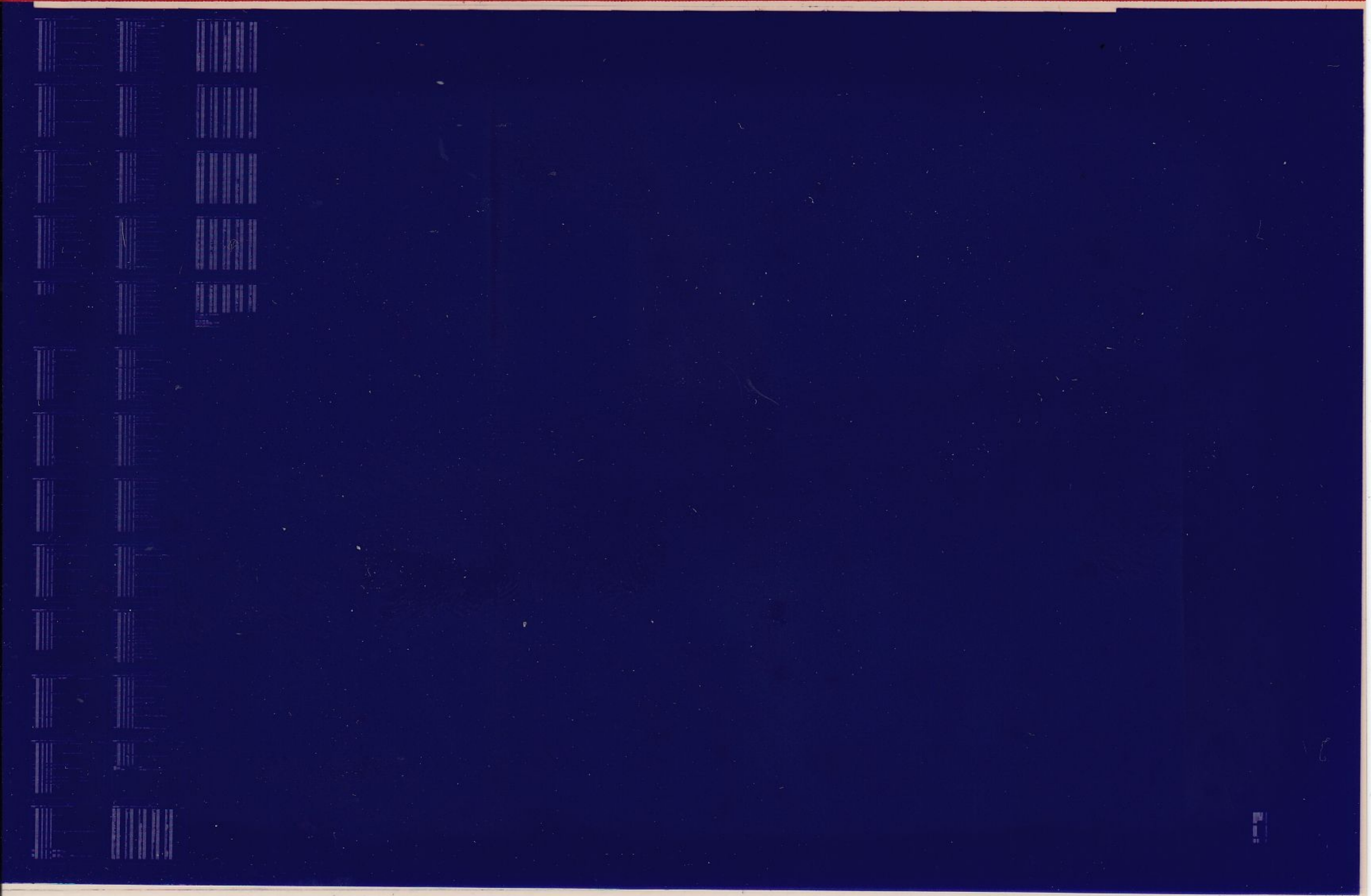
The image shows a dense grid of memory diagnostic data. The data is organized into columns and rows, with some sections highlighted in white. The text is small and difficult to read, but it appears to be a detailed memory dump or diagnostic report. The data is organized into columns and rows, with some sections highlighted in white. The text is small and difficult to read, but it appears to be a detailed memory dump or diagnostic report.

MSV11-J

MSV11-J MEMORY DIAG
CVMJAA0

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

digital
MADE IN USA



2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

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

IDENTIFICATION

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

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

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

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

COPYRIGHT (C) 1985 Digital Equipment Corporation

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

OPERATIONAL SWITCH SETTINGS
SWITCH REGISTER DEFINITIONS

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

TABLE OF CONTENTS

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

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

1.0 GENERAL PROGRAM INFORMATION

1.1 Program Purpose (Abstract)

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

1.2 System Requirements

1.2.1 Hardware Requirements -

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

***** NOTE *****

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

153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196

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)

198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249

1.5 Assumptions

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

2.0 OPERATING INSTRUCTIONS

2.1 Loading Starting Procedures

2.1.1 Quick Starting -

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

NOTE

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

2.1.2 Stopping -

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

2.1.3 Restarting (Preserve Configuration Table) -

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

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

2.1.4 Switch Register Options -

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

279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322

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

324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367

2.2.2 Test Protocol for MSV11-J ECC Memeory

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

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

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

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

2.3 Special Environments

2.3.1 XXDP+

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

2.3.2 ACT APT Automatic Mode -

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

2.3.2.1 APT Execution Times -

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

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

The first pass will be a quick verify pass

NOTE

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

422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478

2.3.2.2 APT Environment Table -

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

FIRST PASS RUN TIME:

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

NOTE

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

LONGEST TEST TIME:

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

ADDITIONAL RUN TIME:

Not Used By Program.

SOFTWARE ENVIRONMENT:

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

ENVIRONMENT MODE:

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

SWITCH 1:

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

SWITCH 2:

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

CPU OPTIONS:

Not Used By Program.

MEMORY TYPE n (n=1 to 4)

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

MAXIMUM ADDRESS n (n=1 to 4)

These four words are used in conjunction with the corresponding

479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527

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

NOTE

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

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

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

BASE ADDRESS:
Not Used By Program.

DEVICE MAP:
Not Used By Program.

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

DEVICE DESCRIPTOR CODES:

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

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

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

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

2.3.3 No SBE Free Banks -

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

2.3.4 Mixed Parity ECC Configurations -

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

2.4 Program Options

2.4.1 Switch Register Details -

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

550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602

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

657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709

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

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

For more details reference section 2.4.1.1, 2.4.1.2 and 2.4.1.3.

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

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

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

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

NOTE

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

2.4.1.1 Test Mode Example -

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

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

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

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

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

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

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

```

764
765           ;RELOCATE TEST PROGRAM SPACE (BANK 0 & 1)
766
767 TEST 1,    BANK 0
768 TEST 2,    BANK 0
769 TEST 3,    BANK 0
770 TEST 4,    BANK 0
771 TEST 5,    BANK 0
772 TEST 26,   BANK 0
773 TEST 1,    BANK 1
774 TEST 2,    BANK 1
775 TEST 3,    BANK 1
776 TEST 4,    BANK 1
777 TEST 5,    BANK 1
778 TEST 26,   BANK 1
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815

```

NOTE

This is an example not an actual sequence.

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

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

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

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

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

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

program space (from 0 to 167).

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

2.4.1.2 Test Mode Details -

MODE 0 = "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

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

NOTE

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

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

2.4.1.3 Test Mode Applications -

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

Advantages: Easy to understand.

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

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

Advantages: Seeks Bad Banks. Easy to understand.

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

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

J2

968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023

2.4.2 Software Switch register -

A software switch register exists in location 174.

2.4.3 Special Memory Locations -

2.4.3.1 CACHE Constant -

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

NOTE

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

2.4.3.2 Configuration Table

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

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

Configuration Words:

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

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

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

2.4.4 Terminal Commands -

2.4.4.1 Control "C"

This command will:

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

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

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

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

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

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

Example:

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

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

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

2.4.4.4 Control "S" (Stop)

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

2.4.4.5 Control "Q" (Continue)

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

2.4.4.6 Control "F" (Field Service mode)

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

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

NOTE

Typing just carriage return is a default command 0.

2.4.4.7.1 Field Service Command 0 (Exit)

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

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

2.4.4.7.2 Field Service Command 1 (Read CSR)

This command will typeout the contents of the CSR.

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

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

NOTE

CSR references are done in accordance with section 5.0.

2.4.4.7.3 Field Service Command 2 (Load CSR)

This command will enable you to load the CSR.

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

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

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

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

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

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

2.4.4.7.4 Field Service Command 3 (Examine Memory)

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

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

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

The contents of your physical address will be typed.

2.4.4.7.5 Field Service Command 4 (Modify Memory)

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

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

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

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

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

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

NOTE

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

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

2.4.4.7.6 Field Service Command 5 (Select Bank Test)

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

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

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

NOTE

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

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

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

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

2.4.4.7.7 Field Service Command 6 (Type Configuration Map)

This command types the configuration map.

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

For a detailed explanation of the map reference section 7.3.

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

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

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

2.4.4.7.9 Field Service Command 8 (Error Summary)

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

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

2.4.4.7.10 Field Service Command 9 (Refresh TEST)

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

2.4.4.7.11 Field Service Command 10 (Set Fill Count)

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

2.4.4.7.12 Field Service Command 11 (Enter Kamikaze Mode)

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

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

2.4.4.7.13 Field Service Command 12 (Exit Kamikaze Mode)

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

2.4.4.7.14 Field Service Command 13 (Turn Cache Off)

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

2.4.4.8.15 Field Service Command 14 (Turn Cache On)

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

2.4.4.7.16 Field Service Command 15 (Test Only Selected Banks)

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

2.4.4.7.17 Field Service Command 16 (Resume Testing All Banks)

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

2.4.4.7.18 Field Service Command 17 (Resume Testing All Banks)

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

2.4.4.7.19 Field Service Command 18 (Resume Testing All Banks)

Disable "Trace". (undo Command 17).

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

2.5 Execution Times

2.5.1 Typical (System) -

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

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

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

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

2.5.2 Calculations (System)

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

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

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

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

2.5.3 Typical (Tests)

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

1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532

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.

1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582

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.
MMRO	Memory Management Register #0.
MMR1	Memory Management Register #1.
MMR2	Memory Management Register #2.
MMR3	Memory Management Register #3.
MTYP	Memory Type (MSV11-J,MSV11-L, or MSV11-P).
PADD	Physical Address (asserted by the program after mapping).
PAT	Pattern number.
PC	Program Counter at the time the error occurred.
SBE	Single Bit Error (correctable error).
VADD	Virtual Address (asserted by the program before mapping).
WROTE1	The data that was written into the 1st half of a double word.
WROTE2	The data that was written into the 2nd half of a double word.
XOR	Exclusive OR of the good and bad data. Shows the bad bits.
AUT	Address under test

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

3.3 Error Halts

There are several Halts in the program.

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

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

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

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

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

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

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

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

4.0 PROGRESS REPORTS

Pass complete typeouts as follows:

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

NOTE

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

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

Example:

BANK= 2 TEST= 34

Reference Section 2.4.4.7.18 for more information on Tracing.

5.0 CSR INFORMATION TABLES

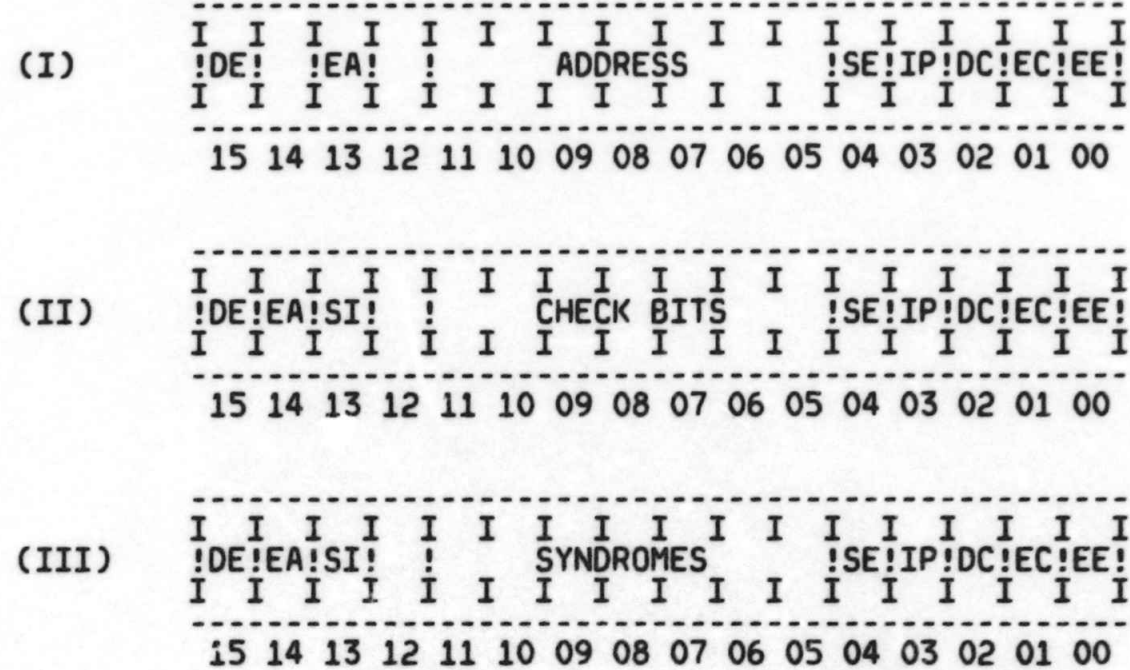
The following is a picture view of the current control status registers which can be tested by this program. It shows bit assignments and definitions to provide a handy reference, and shows the similarities and differences between each one:

NOTE

All unused bits in each CSR are equal to zero.

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

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.

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

BIT13 SET INHIBIT MODE

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

BITS05-10 CHECK BIT STORAGE (CSR II)

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

BITS05-11 QBUS ADDRESS STORAGE (CSR I)

(Diag Ck Bits 2 = 0, ECC Disable Bit 1 = 0)

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

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

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

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

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

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

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

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

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

BIT05-10 SYNDROME STORAGE (CSR III)

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

BIT04 SINGLE ERROR

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

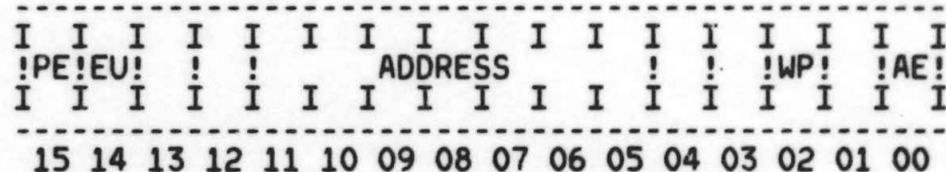
have been generated on the data written. This means that if a single or double error existed in the location accessed, it would be cleared (unless the errors were hard).

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

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

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

5.2 MSV11-L/P CSR



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.

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

6.0 SUB-TEST SUMMARIES

6.1 Tests

- TEST 1
BIT TEST OF ALL CSR'S/MATCH ALL CSR'S WITH MEMORY
(CSR Access may cause wrong Type of Traps)
- TEST 2
TEST BANK 0 ACCESSES
Failures are fatal.
- TEST 3
TEST BANKS 1-177 (OCTAL) FOR ZEROS AND ONES
Errors are not typed here - only logged in
the configuration table
- TEST 4
ECC INHIBIT MODE POINTER TEST
- TEST 5
DIAGNOSTIC MODE DISPATCH ROUTINE
This test runs all the patterns in the
mode selected.
- TEST 6
UNIQUE BANK TEST
Pattern 27 is run

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

6.2 Tests

6.2.1 General Test Information

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

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

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

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

Example E-table Segment:

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

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

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

6.2.2 Specific Tests

6.2.2.1 Test 0 Basic Data Test

Writes Reads R2 into a 16K Bank.

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

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

NOTE

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

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

6.2.2.2 Test 1 Address Test

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

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

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

6.2.2.3 Test 2 Complement Address Test

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

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

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

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

6.2.2.4 Test 3 3 XOR 9

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

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

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

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

6.2.2.5 Test 4 Rotating Zeros Test

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

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

6.2.2.6 Test 5 Rotating Ones Test

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

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

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

NOTE

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

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

6.2.2.7 Test 6 Initial Data Test

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

This is a very quick check of the data paths.

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

6.2.2.8 Test 7 Address Bit Test

Writes a background of all Zeros.

Read Address 1 for a 0 Byte.

Complement Address 1.

Read Address 1 for a non 0 Byte.

For each Address Bit position from Bit 1:

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

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

Read Address for a 0 word.

Complement Address contents.

Read Address for a non-zero word.

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

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

6.2.2.9 Test 10 Byte Addressing Test

With ECC Disabled.

Writes all ones to a double word.

For each of the 4 Bytes in the Double Word.

Clears one byte.

Reads all 4 bytes from double word.

Checks for only proper byte clear.

All other bytes set to all ones.

This is only done on one double word address.

NOTE

This is run for ECC memory only

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

6.2.2.10 Test 14 Basic Double Bit Error Test

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

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

NOTE

This test is only run for the MSV11-J

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

6.2.2.11 Test 17 Holding 1's 0's Test

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

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

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

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

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

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

NOTE

This test is only run for the MSV11-J

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

6.2.2.13 Test 21 Marching 0's 1's Test

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

This checks the integrity of the 32 Bit Double Words.

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

NOTE

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

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

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).

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

6.2.2.15 Test 23 Shifting Diagonal Test

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

NOTE

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

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

6.2.2.16 Test 24 Fast Galloping Pattern Test

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

NOTE

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

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

6.2.2.17 Test 26 Random Data Test

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

Read check Random Data.

This routine regenerates the same random numbers by using the same

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

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

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

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

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

6.2.2.18 Test 27 Unique Bank Test

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

It does not test Banks that require relocation to test.

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

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

6.2.2.19 Test 30 Flush Out DBE's Test

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

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

2625
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

6.2.2.20 Test 31 SOB-A-LONG Test

Rationalization

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

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

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

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

Simplified SOB Example:

```

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

```

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

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

NOTE

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

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

6.2.2.21 Test 32 Write Recovery Test

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

Write Recovery Test

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

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

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

If you understand this so far the rest is easy.

The test execution is as follows:

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

NOTE

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

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

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$:      BVS           ERROR              ;ERROR TRAP
          ERROR        ;CLEAR V-BIT
2$:      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).

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

6.2.2.23 Test 34 Soft Error Test

Rationalization

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

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

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

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

6.2.2.24 Test 35 Worst Case Parity Test

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

NOTE

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

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

6.2.2.25 Test 36 Correction Code Test

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

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

NOTE

This test is only run for the MSV11-J

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

6.2.2.26 Test 37 Check ECC Disable Test

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

NOTE

This test is run on the MSV11-J only.

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

6.2.2.27 Test 41 Address to CSR on DBE Test

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

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

NOTE

This test is run on a MSV11-J only.

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

6.2.2.28 Test 42 Extended Address to CSR on Error Test

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

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

NOTE

This test is only run for the MSV11-J

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

6.2.2.29 Test 43 Write Byte Test

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

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

NOTE

This test is only run for the MSV11-J

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

6.2.2.30 Test 44 Shifting Checkbits through the CSR Test

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

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

NOTE

This test is only run for the MSV11-J

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

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

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

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

NOTE

This test is only run for the MSV11-J

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

6.2.2.32 Test 46 Check Single Bit Errors with ECC Disabled Test

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

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

NOTE

This test is only run for the MSV11-J

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

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

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

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

NOTE

This test is only run for the MSV11-J

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

6.2.2.34 Test 999 Null Test

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

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

3165
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

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

3220
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

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

7.1.3 CACHE

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

7.2 Bank Zero Testing

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

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

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

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

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

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

NOTE

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

7.3 Memory Configuration Map

3275
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

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							
		0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7
ERRORS																																	
MEMTYPE		E	E	E	E	E	E	E	E	P	P	P	P	P	P	P	P	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E	E
CSR		0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
PROTECT		P								I								P								I							
			1								1								1								1						
		0								1								2								3							
		4								5								6								7							
ERRORS																																	
MEMTYPE		E	E	E	E																												
CSR		4	4	4	4																												
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:

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

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

PROTECT:

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

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

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

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

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

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

```

3368
3369
3370      7.4.1 Sample Source File -
3371      .ENABL ABS
3372      .ENABL AMA
3373      .MCALL .SUPER
3374      .SUPER
3375      ;LST$$=0
3376      BIT5=40
3377      A:      0
3378      B:      0
3379      C:      0
3380      D:      0
3381      E:      0
3382      F:      0
3383      G:      0
3384      H:      0
3385      I:      0
3386      J:      0
3387      .PAGE
3388      ;LET EXAMPLES
3389      LET RO := A
3390      LET B := C + D
3391      LET E := F + 1
3392      LET G := H + 2
3393      LET J := J + 01
3394      LET A :B= B
3395      ;IF EXAMPLES
3396      IF A IS TRUE
3397      MOV 23,D
3398      END ;OF IF A
3399      IF B IS FALSE
3400      MOV 34,E
3401      END ;OF IF B
3402      IF A EQ B THEN LET C := D
3403      IF A LT B
3404      MOV C,D
3405      ELSE
3406      MOV E,D
3407      END ;OF IF A
3408      IF A EQ B AND C NE D
3409      MOV F,G
3410      END ;OF IF A
3411      IF A EQ B OR C NE D
3412      MOV F,G
3413      END ;OF IF A
3414      IFB A EQ B AND C EQ 1
3415      MOV H,J
3416      ELSE
3417      MOV E,J
3418      END ;OF IFB A
3419      IFB A EQ B ANDB C EQ 1
3420      MOV H,J
3421      ELSE
3422      MOV E,J

```

```

3423          END ;OF IFB A
3424          IF RESULT IS EQ
3425             MOV  A,B
3426          END ;OF IF RESULT
3427          IF BITS SET.IN A
3428             MOV  B,C
3429          END ;OF IF BITS
3430          IF BITS OFF.IN A
3431             MOV  C,D
3432          END ;OF IF BITS
3433          ;ON.ERROR IS LIKE AN IF STATEMENT ON THE C-BIT
3434          ;ON.ERROR EXAMPLES
3435          ON.ERROR
3436             MOV  A,B
3437          ELSE
3438             MOV  C,B
3439          END ;OF ON.ERROR
3440          ON.NOERROR
3441
3442
3443             MOV  C,B
3444          ELSE
3445             MOV  A,B
3446          END ;OF ON.NOERROR
3447          ON.ERROR THEN LET A :B= B
3448          ;FOR EXAMPLES
3449          FOR I := -5 TO 23
3450             INC  A
3451          END ;OF FOR I
3452          FOR RO := 0 TO 140 BY 4
3453             DEC  A(RO)
3454          END ;OF FOR RO
3455          FOR I := 133 DOWNTO 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

```


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

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

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

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

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

18				;LET EXAMPLES
19	000024			LET R0 := 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	MOV 23,D
28	000122		000006	END ;OF IF A
29	000122			IF B IS FALSE
30	000130	012737	000034	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	MOV C,D
35	000172			ELSE

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

```

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

```

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

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

```

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

```

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

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

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

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

18
19 000024 013700 000000
20 000030 013737 000004 000002
000036 063737 000006 000002
21 000044 013737 000012 000010
000052 005237 000010
22 000056 013737 000016 000014
000064 062737 000002 000014
23 000072 062737 000001 000022
24 000100 113737 000002 000000
000100
25
26 000106 005737 000000
000112 001403
27 000114 012737 000023 000006
28 000122
000122
29 000122 005737 000002
000126 001003
30 000130 012737 000034 000010
31 000136
000136
32 000136 023737 000000 000002
000144 001003
000146 013737 000006 000004
000154
33 000154 023737 000000 000002
000162 002004
34 000164 013737 000004 000006
35 000172 000403
000174
36 000174 013737 000010 000006
37 000202
000202
38 000202 023737 000000 000002
000210 001007
000212 023737 000004 000006
000220 001403
39 000222 013737 000012 000014
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
MOV B,A
;IF EXAMPLES
IF A IS TRUE
TST A
BEQ L0
MOV 23,D
END ;OF IF A
L0:
IF B IS FALSE
TST B
BNE L1
MOV 34,E
END ;OF IF B
L1:
IF A EQ B THEN LET C := D
CMP A,B
BNE L2
MOV D,C
L2:
IF A LT B
CMP A,B
BGE L3
MOV C,D
ELSE
BR L4
L3:
MOV E,D
END ;OF IF A
L4:
IF A EQ B AND C NE D
CMP A,B
BNE L5
CMP C,D
BEQ L5
MOV F,G
END ;OF IF A

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

000230
41 000230 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
000264 001010
000266 023727 000004 000001
000274 001004
45 000276 013737 000016 000022
46 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 000403
000344
52 000344 013737 000010 000022
53 000352 000352
54 000352 001003
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 032737 000040 000000
000400

L6:
MOV F,G
END ;OF IF A
L7:
IFB A EQ B AND C EQ 1
CMPB A,B
BNE L10
CMP C, 1
BNE L10
MOV H,J
ELSE
BR L11
L10:
MOV E,J
END ;OF IFB A
L11:
IFB A EQ B ANDB C EQ 1
CMPB A,B
BNE L12
CMPB C, 1
BNE L12
MOV H,J
ELSE
BR L13
L12:
MOV E,J
END ;OF IFB A
L13:
IF RESULT IS EQ
BNE L14
MOV A,B
END ;OF IF RESULT
L14:
IF BITS SET.IN A
BIT BITS,A
BEQ L15
MOV B,C
END ;OF IF BITS
L15:
IF BITS OFF.IN A
BIT BITS,A

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

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

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

```

3797      000532  003771
3798      000534
3799      83 000534
3800      000534  012737  000133  000020
3801      000542
3802      84 000542  063737  000000  000002
3803      85 000550
3804      000550  162737  000002  000020
3805      000556  023727  000020  000003
3806      000564  002366
3807      000566
3808      86
3809      87 000566
3810      000566
3811      88 000566
3812      000566  005000
3813      000570
3814      89 000570  116037  000000  000002
3815      90 000576
3816      000576  005737  000002
3817      000602  002415
3818      91 000604
3819      000604  005200
3820      000606  020027  000167
3821      000612  003766
3822      000614
3823      92 000614
3824      000614  012700  000400

```

```

E1:      BLE B1
          FOR I := 133 DOWNT0 3 BY 2
          MOV 133,I
B2:      ADD A,B
          END ;OF FOR I
          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

```

3825
3826
3827
3828
3829
3830
3831      000620
3832      93 000620
3833      000620  005737  000002
3834      000624  002004
3835      94 000626
3836      000626  005200
3837      000630  020027  000567
3838      000634  003771
3839      000636
3840      95 000636
3841      000636
3842      96
3843      97 000636
3844      000636  000207
3845      98 000640
3846      000640  000261
3847      000642  000207
3848      99 000644
3849      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

```

```

3851      100
3852      101 000650 013700 000000
3853      102 000654
3854          000654 010046
3855          000656 006316
3856          000660 004737 000700
3857      103 000664 000000
3858      104 000666 000002
3859      105 000670 000004
3860      106 000672 000006
3861      107 000674 000010
3862      108 000676 000012
3863      109 000700
3864          000700
3865          000700 062616
3866          000702 013646
3867          000704 004736
3868      110
3869      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 11/23B) -

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

3900

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

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

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

```

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

```

```

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

```



```

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

```


4056
4057

104513
104514

CBREG =104513
SYNREG =104514

;ENABLES CHECKBIT REGISTER
;ENABLES SYNDROME BIT REGISTER

```

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

```

```

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

```

```

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

```



```

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

```

```

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

```

```

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

```

```

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

```

```

4355
4356      .SBTTL DEFINE CONTROL STATUS REGISTERS
4357      172100 CSRADD=172100

```

```

4358
4359      .SBTTL DEFINE PARAMETERS
4360      060000 FIRST=60000 ;START OF THE 16K TEST PATTERN AREA
4361      157776 LAST=157776 ;END OF THE 16K TEST PATTERN AREA
4362      040000 SIZE=40000 ;SIZE OF THE 16K TEST PATTERN AREA (FOR SOB INSTRUCTIONS)

```

```

4365          .LIST MD                      ;BE NICE TO SEE MY DEFINITIONS
4366          .SBTTL MACRO FATAL
4367          ;***** FATAL *****
4368          ;
4369          ;FATAL IS USED TO REPORT FATAL ERRORS (ERRORS THAT PREVENT
4370          ; THE PROGRAM FROM CONTINUING).
4371          ;
4372          ;*****
4373          .MACRO FATAL ARG                ;***MACRO***MACRO***MACRO***
4374          .NLIST
4375          .DSABL CRF
4376          .IIF DF LST$$ .LIST ME
4377          .ENABL CRF
4378          .LIST
4379          INC FATAL$                      ;SET FATAL INDICATOR
4380          ERROR +ARG
4381          .DSABL CRF
4382          .IIF DF LST$$ .NLIST ME
4383          .ENABL CRF
4384          .ENDM FATAL
4385
4386          .SBTTL MACRO TYPE
4387          .MACRO TYPE ARG
4388          .NLIST
4389          .DSABL CRF
4390          .IIF DF LST$$ .LIST ME
4391          .ENABL CRF
4392          .LIST
4393          .IF B ARG
4394          TYPEIT
4395          .IFF
4396          TYPEIT ,ARG
4397          .ENDC
4398          .DSABL CRF
4399          .IIF DF LST$$ .NLIST ME
4400          .ENABL CRF
4401          .ENDM TYPE

```


4404
 4405
 4406
 4407
 4408
 4409
 4410
 4411
 4412
 4413
 4414
 4415
 4416
 4417
 4418
 4419
 4420
 4421
 4422
 4423
 4424
 4425
 4426
 4427
 4428
 4429
 4430
 4431
 4432
 4433
 4434
 4435
 4436
 4437
 4438
 4439
 4440
 4441
 4442
 4443
 4444
 4445
 4446
 4447
 4448
 4449
 4450
 4451

```

.SBTTL MACRO NEWTST
;***** NEWTST *****
;NEWTST IS USED AS THE FIRST INSTRUCTION OF A TEST.
;IT WILL:
;1) GENERATE A TEST NUMBER FOR THE LABEL OF THIS TEST
;2) PUT STARS BEFORE AND AFTER A MESSAGE
;ARGUMENTS
;1) ASCII -- THIS IS THE MESSAGE THAT WILL APPEAR
;           ON THE LISTING
;2) ICOUNT -- IF NON-BLANK AND BIT 11 OF $SWR = 1 IT WILL BE
;            THE NUMBER OF ITERATIONS TO MAKE ON THIS TEST
;3) RETURN -- IF NON-BLANK WILL BE THE ADDRESS TO
;            WHICH THE NEXT SCOPE STATEMENT WILL
;            LOOP BACK TO.
;4) COMAND -- IF NON-BLANK WILL BE THE FIRST
;            INSTRUCTION OF THE TEST
;            IF BLANK SCOPE WILL BE THE
;            FIRST INSTRUCTION
;*****
.MACRO NEWTST ASCII,ICOUNT,RETURN,COMAND
$STN=1
$NWTST=0
.NLIST MC
.IF B <COMAND>
$$NEWTST \ $TN,<ASCII>,SCOPE
.IFF
$$NEWTST \ $TN,<ASCII>,<COMAND>
.ENDC
.NLIST
.LIST ME
.LIST
.IF NE 4000&$SWR
.IF NB ICOUNT
.IF LE <ICOUNT-1>
MOV #1,$TIMES ;;DO 1 ITERATION
.IFF
MOV #ICOUNT,$TIMES ;;DO ICOUNT ITERATIONS
.ENDC
.ENDC
.IF NB RETURN
MOV #RETURN,$LPADR ;;SET SCOPE LOOP ADDRESS
.ENDC
.ENDC
.NLIST
.LIST MC
.LIST
.NLIST ME
.ENDM NEWTST
    
```

4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474
4475
4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490
4491
4492
4493
4494
4495
4496
4497
4498
4499
4500
4501
4502
4503
4504
4505
4506

```

.SBTTL MACRO $$NEWTEST
.MACRO $$NEWTEST A,ASC,COMND
.IRP ASCI,<ASC>
.IF EQ $NWTST
$NWTST=1
.SBTTL T'A' ASCI
.NLIST
.LIST ME
.LIST
;*****
;*TEST A ASCI
.IFF
ASCI
.ENDC
.ENDM
;*****
TST'A: COMND
.NLIST ME
$TN=$TN+1
.ENDM $$NEWTEST

.SBTTL MACRO SUBTST
;***** SUBTST *****
;
;THIS MACRO WILL FORMAT A SUBTEST HEADING WITH STARS
;A .SBTTL WILL BE FORCED & .NLISTED FOR THE TABLE OF CONTENTS.
;
;ARGUMENT:
;1) TXT -- THIS IS THE MESSAGE THAT WILL APPEAR IN THE TABLE OF CONTENTS & LISTING.
;
;EXAMPLE: SUBTST <<THIS IS A FUN SUBTST>>
;
;*****

.MACRO SUBTST ASCII
.NLIST MC
$SUBTST <ASCII>
.LIST MC
.ENDM SUBTST

.SBTTL MACRO $SUBTST
.MACRO $SUBTST ASC
.IRP ASCI,<ASC>
.SBTTL ASCI
.NLIST
.LIST ME
.LIST
;*****
;*SUBTEST ASCI
.ENDM
;*****
.NLIST ME
.ENDM $SUBTST

```

4509
4510
4511
4512
4513
4514
4515
4516
4517
4518
4519
4520
4521
4522
4523
4524
4525
4526
4527
4528
4529
4530
4531
4532
4533
4534
4535
4536
4537
4538
4539
4540
4541
4542
4543
4544
4545
4546

```

.SBTTL MACRO TYPOCT
;***** TYPOCT *****
;
;TYPOCT IS USED TO CHANGE A BINARY NUMBER
; TO A 6 DIGIT OCTAL NUMBER AND TYPE IT
;
; ARGUMENTS:
;
;1) NUM THE NUMBER TO BE TYPED
;
;2) REMARK ALLOWS A COMMENT TO BE MADE
;
; ROUTINES REQUIRED
;
;1) CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
;
;2) TYPE AN ASCIZ STRING (.$TYPE)
;
; EXAMPLES:
;
;1) TYPOCT HILMT,<TYPES THE CONTENTS OF HILMT>
;2) TYPOCT #5,<TYPES ' 000005'>
;
;*****
.MACRO TYPOCT NUM,REMARK
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM,-(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPOCT

```

4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562
4563
4564
4565
4566
4567
4568
4569
4570
4571
4572
4573
4574
4575
4576
4577
4578
4579
4580
4581
4582
4583
4584
4585
4586
4587
4588
4589
4590
4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602

```

.SBTTL MACRO TYPOCS
;***** TYPOCS *****
;
; TYPOCS IS USED TO CHANGE A BINARY NUMBER TO AN OCTAL
; NUMBER AND TYPE 1 TO 6 DIGITS
; WITH OR WITHOUT LEADING ZEROS.
;
; ARGUMENTS:
;
; 1) NUM NUMBER TO BE TYPED
;
; 2) REMARK ALLOWS A COMMENT TO BE MADE
;
; 3) N NUMBER OF DIGITS (1 TO 6) TO BE TYPED
;
; 4) Z BLANK=SUPPRESS LEADING ZEROS (TYPES SPACES)
; NON-BLANK=TYPE LEADING ZEROS
;
; ROUTINES REQUIRED
;
; 1) CONVERT BINARY TO OCTAL AND TYPE (.$TYPOCT)
;
; 2) TYPE AN ASCIZ STRING (.$TYPE)
;
; EXAMPLES:
; 1) TYPOCS #12345,<TYPES "5">,1
; 2) TYPOCS #004,<TYPES "04">,2,X
; 3) TYPOCS #004,<TYPES " 4">,2
;
;*****
.MACRO TYPOCS NUM,REMARK,N,Z
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV NUM,-(SP) ;;SAVE NUM FOR TYPEOUT
.IIF NB <REMARK>, ;;REMARK
TYPOS ;;GO TYPE--OCTAL ASCII
.IF NB N
.BYTE N ;;TYPE N DIGIT(S)
.IFF
.BYTE 6 ;;TYPE 6 DIGITS
.ENDC
.IF NB Z
.BYTE 1 ;;TYPE LEADING ZEROS
.IFF
.BYTE 0 ;;SUPPRESS LEADING ZEROS
.ENDC
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM TYPOCS

```



```

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

```

4647
4648
4649
4650
4651
4652
4653
4654
4655
4656
4657
4658
4659
4660
4661
4662
4663
4664
4665
4666
4667
4668
4669
4670
4671
4672
4673
4674
4675
4676
4677
4678
4679
4680
4681
4682
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697
4698
4699
4700
4701
4702
4703

```

.SBTTL MACRO BMOV
;***** BMOV *****
;
; THIS MACRO MOVES A BLOCK OF DATA.
;
; ARGUMENTS:
;
; 1) FROMHERE THE FIRST ADDRESS OF THE SOURCE BLOCK.
;
; 2) TOHERE THE FIRST ADDRESS OF THE DESTINATION BLOCK.
; IF BLANK THE 1ST ADDRESS OF THE USER INSTRUCTION
; PAR'S IS USED (FASTCITY).
;
; 3) SIZE THE SIZE OF THE SOURCE BLOCK.
; IF BLANK A 16 WORD TRANSFER IS ASSUMED.
; "WHY DEFAULT TO 16 WORDS?" YOU ASK!
; "BECAUSE THAT'S HOW MANY WORDS TO THE USER PAR
; REGISTERS & THAT'S WHERE I INTEND TO MOVE LOTS
; OF STUFF." I REPLY!
;
;*****
.MACRO BMOV FROMHERE,TOHERE,SIZE
  .IF B TOHERE
    .NLIST
    .DSABL CRF
    .IIF DF LST$$ .LIST ME
    .ENABL CRF
    .LIST
    JSR R5,BLOCK1
    FROMHERE
    .DSABL CRF
    .IIF DF LST$$ .NLIST ME
    .ENABL CRF
    .MEXIT
  .ENDC
  .IF B SIZE
    .NLIST
    .DSABL CRF
    .IIF DF LST$$ .LIST ME
    .ENABL CRF
    .LIST
    JSR R5,BLOCK2
    TOHERE
    FROMHERE
    .DSABL CRF
    .IIF DF LST$$ .NLIST ME
    .ENABL CRF
    .MEXIT
  .IFF
    .NLIST
    .DSABL CRF
    .IIF DF LST$$ .LIST ME
    .ENABL CRF
    .LIST
    JSR R5,BLOCK3
    SIZE

```

4704
4705
4706
4707
4708
4709
4710

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

4713
4714
4715
4716
4717
4718
4719
4720
4721
4722
4723
4724
4725
4726
4727
4728
4729
4730
4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747
4748
4749

```

.SBTTL MACRO MAP
;***** MAP *****
;
; THIS MACRO MAPS A MEMORY BANK (16K) INTO THE
; TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000-157777)).
;
; ARGUMENTS:
;
; 1) BANK THE BANK OF 16K WORDS TO BE MAPPED.
; THERE ARE 120 BANKS OF 16K WORDS
;
; EXAMPLES
;
; MAP LOC ;LOCATION "LOC" CONTAINS THE # OF THE BANK TO MAP
;
; MAP #28. ;BANK 34 (OCTAL) WILL BE MAPPED
;
;*****

.MACRO MAP BANK
PUSH R3
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
.IF B BANK
MOV #200,R3
.IFF
MOV BANK,R3
.ENDC
CALL MAPPER
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
POP R3
.ENDM MAP

```


4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793

```

.SBTTL MACRO SUPERVISOR
;***** SUPERVISOR *****
;
; THIS MACRO SWITCHES TO SUPERVISOR MODE.
;
; ARGUMENTS: NONE.
;
;*****

.MACRO SUPERVISOR
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM SUPERVISOR

.SBTTL MACRO USER
;***** USER *****
;
; THIS MACRO SWITCHES TO USER MODE.
;
; ARGUMENTS: NONE.
;
;*****

.MACRO USER
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
BIS #BIT15!BIT14,PSW ;GO TO USER MODE
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM USER

```

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

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

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

4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836
4837
4838
4839
4840
4841
4842
4843
4844
4845
4846
4847
4848
4849
4850
4851
4852
4853
4854
4855
4856
4857
4858
4859

```

.SBTTL MACRO SET4 & RES4
;***** SET4 & RES4 *****
;
; THESE MACROS SET & RESTORE VECTOR 4(TIMEOUT TRAP)
;
; IN IT'S RESTORED MODE TRAPS ARE REPORTED AS SUCH.
;
; ARGUMENTS:  LOC      ;THE LOCATION TO VECTOR TO (ONLY USED IN "SET4" NOT "RES4")
;
; I USE THE SET4 AND RES4 MACROS AROUND CODE THAT I EXPECT TO TRAP TO 4
; LIKE LOOKING FOR ALL POSSIBLE CSR'S AND ETC.  WHENEVER CODE IS NOT
; SURROUNDED BY SET4 AND RES4 THEN ANY TRAPS TO 4 WILL CAUSE AN ERROR
; PRINTOUT THAT SAYS "UNEXPECTED TRAP TO 4" AND ALL THE ASSOCIATED REGISTER JUNK
;*****
.MACRO SET4 ARG
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV ARG,4
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM SET4

.MACRO RES4
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
MOV #TIMEOUT,4
CMP #5,PROTYP ;IS THIS AN 11/83/84 ?
BNE 101$ ;BRANCH IF NOT
CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS
;THAT A EXPECTED TRAP COULD HAVE SET
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM RES4
101$:

```

4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899

```
.SBTTL MACRO DLEFT
;***** DLEFT *****
;
; THIS MACRO DOES A DOUBLE WORD LEFT SHIFT
;
; ARGUMENTS: LOC ;THE LOCATION TO BE SHIFTED LEFT (CARRY TO LOC+2)
;
;*****
```

```
.MACRO DLEFT ARG
.NLIST
.DSABL CRF
.IIF DF LST$$ .LIST ME
.ENABL CRF
.LIST
ROL ARG
ROL ARG+2
.DSABL CRF
.IIF DF LST$$ .NLIST ME
.ENABL CRF
.ENDM DLEFT
.NLIST MD ;DON'T NEED TO SEE THEM ANY MORE
```

```
*****
.SBTTL ABORT ROUTINE FOR ORION UFD MODE (VMK)
*****
```

```
ABORT: TSTB $DDW7 ;TEST FOR USER FRIENDLY MODE
        BEQ ABORT2 ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
        MOV SAV30,30 ;RESTORE EMT LOCATION (30)
        MOV SAV32,32 ;RESTORE EMT PRIORITY LOCATION (32)
        EMT +42 ;GET DCA LOCATION INTO RO FROM MONITOR
        MOV #-1,42(RO) ;SET A -1 INTO LOCATION DRSEFF IN MONITOR
ABORT1: MOV 42,RO ;AND PUT THE MONITOR RETURN ADDRESS IN RO
        CLR 42 ;CLEAR MONITOR RETURN FLAG
        JMP $ENDAD ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
ABORT2: RTS PC ;IF NOTUFD RETURN TO MAINLINE
```

```
000000 105737 056732
000004 001420
000006 013737 003706 000030
000014 013737 003710 000032
000022 104042
000024 012760 177777 000042
000032 013700 000042
000036 005037 000042
000042 000137 014042
000046 000207
```



```

4902
4903
4904 000000 000000 000000
4905          000177
4909
4910
4911
4912
4913
4914
4915
4916
4917
4918
4919
4920
4921
4922
4923
4924
4925          000046
4926 000046 014042
4927          000052
4928 000052 000020
4929
4930          000024
4931 000024 000200
4932          000042
4933 000042 002000
4934          000044
4935 000044 056750
4936          000200
4937 000200 000437
4938 000202 000442
4939          000300
4940 000300 005037 002626
4941 000304 000137 003670
4942 000310
      000310 012737 177777 002626
4943 000316 000137 003670
4944          002000
    
```

```

.SBTTL TRAP CATCHER
.=0
.WORD 0,0
.REPT 177      ;.WGRD .+2,HALT

.SBTTL ACT11 HOOKS
;*THE HOOKS REQUIRED BY ACT11 ARE DEFINED AND SETUP BELOW:
;*
;* DEFINITIONS:
;*
;* 1)LOC.46      "END-OF-PASS" HOOK
;*              =ADDRESS OF END OF PASS ROUTINE
;*              MODIFIED BY ACT11.
;*
;* 2)LOC.52      PROGRAM NEEDS HOOK
;*              BIT 15=1 PROGRAM SHOULD BE POWER
;*              FAILED WHILE RUNNING
;*              =0 NO POWER FAIL
;*              BIT 14=1 PROGRAM MEMORY SIZE DEPENDENT
;*              =0 NOT MEMORY SIZE DEPENDENT
;*              BIT 13=1 PROGRAM REQUIRES MANUAL INTERVENTION
;*              =0 MANUAL INTERVENTION NOT REQUIRED
;*              BITS 12-0 MUST BE ZERO'S
;*
.=46
$ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
.=52
.WORD BIT4      ;;2)SET LOC.52 TO INDICATE MEMORY SIZE DEPENDANT
.SBTTL APT11 HOOKS
.=24
200            ;;SET POWER FAIL TO POINT TO START OF PROGRAM
               ;;FOR APT START UP
.=42
STACK          ;SO RT11 CAN START WITH RUN COMMAND
.=44           ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR        ;;POINT TO APT HEADER BLOCK
.=200
START3: BR     START1      ;"NORMAL" START
           BR     START2   ;RESTART (SAVE ERROR ACCOUNTING)
.=300
START1: CLR   RESTART
           JMP  START
START2: SET   RESTART
           JMP  START
           MOV #-1,RESTART
.=STACK
    
```

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

VARIABLES

INITIALIZED TO ZERO

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

::I.L.C.:REV

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

5114 002532 000000 000000
5115 002536 000000
5116 002540 000000
5117 002542 000000
5118 002544

PGMCSR: .WORD 0,0 ;POINTS TO PROGRAM CSR
INHECC: .WORD 0 ;FLAGS INHIBIT ECC TESTS ON RELOCATION
INHBANK: .WORD 0 ;
FULLREL: .WORD 0 ;
\$CMTGE: ;*END OF COMMON TAGS

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

```

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

```

***** MAIN *****

```

5195          .SBTTL ***** MAIN *****
5196 003670  START: SUBTST <<INITIALIZE VARIABLES TO ZERO>>
;*****
;*SUBTEST    INITIALIZE VARIABLES TO ZERO
;*****

5197          SUBTST <<SAVEMT>>
5198 003670  ;*****
;*SUBTEST    SAVEMT
;*****
;SAVEMT SAVES THE EMULATOR AND PRIORITY LOCATION
;UNDER THE NAMES OF SAV30 AND SAV32.

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

```



```

5231 004006          SUBTST  <<TYPE OF SYSTEM SIZER>>
;*****
;*SUBTEST          TYPE OF SYSTEM SIZER
;*****
5232 004006 000401  BR      SYSSIZ          ;SKIP OVER VARIABLE LOCATION
5233 004010 000000  PROTYT: .WORD 0
5234 004012 012737 004074 000004 SYSSIZ: SET4 #3$
      004012 012737 004074 000004 MOV #3$,4
      .DSABL CRF
5235 004020 012737 004032 000010 MOV #1$,10          ;TRAPS TO 10 = BAD PROCESSOR TYPE
5236 004026 000007  MFPT          ;TYPE OF PROCESSOR TEST: THIS INSTRUCTION
5237          ;(AVAILABLE ON NEWER PROCESSORS ONLY) PLACES
5238          ;A CODE IN THE LOWER BYTE OF R0 THAT
5239          ;INDICATES THE PROCESSOR TYPE. 1=11/44
5240          ;3=11/23, 5=11/83/84 (Orion)
5241 004030 000406  BR      2$
5242 004032 012737 047122 000034 1$: MOV #TYPE,34          ;LOAD TRAP VECTOR
5243 004040 004040 104401 071474 TYPE MSG130          ;TELL THEM BAD PROCESSOR TYPE
      .DSABL CRF
5244 004044 000000  HALT          ;NO NEED TO GO ON
5245 004046 012737 000012 000010 2$: MOV #12,10          ;RESTORE TRAPS TO 10
5246 004054 110037 004010 MOVB R0,PROTYP      ;MOV THE CODE TO PROTYP
5247 004060 012737 004074 000004 SET4 #3$
      MOV #3$,4
      .DSABL CRF
5248 004066 005737 177746 TST CONTRL          ;SEE IF CACHE REGISTER RESPONDS
5249 004072 000447 BR      6$          ;BRANCH IF CACHE AVAILABLE
5250 004074 005037 002544 3$: CLR CACHKN          ;NO CACHE ON SYSTEM
5251 004100 012737 002336 060342 MOV #ZEROS,DT14     ;DO NOT PRINT CONTRL ERROR MESSAGES
5252 004106 022737 000005 004010 CMP #5,PROTYP       ;IS THIS A 11/83/84
5253 004114 001436 BEQ 6$          ;YES - BRANCH
5254 004116 012737 004146 000004 SET4 #4$
      MOV #4$,4
      .DSABL CRF
5255 004124 005037 172516 CLR MMR3          ;SEE IF THERE IS 22-BIT ADDRESSING
5256 004130 052737 000020 172516 BIS #BIT4,MMR3
5257 004136 032737 000020 172516 BIT #BIT4,MMR3
5258 004144 001005 BNE 5$          ;BRANCH IF 22-BIT RELOCATION
5259 004146 005237 002454 4$: INC NO22BIT          ;SET FOR NO 22 BIT ADDRESSING
5260 004152 012737 000007 002556 MOV #7,LASTBANK     ;HIGHEST BANK OF MEMORY
5261 004160 012737 140000 002552 5$: MOV #140000,TESTMODE ;MAKE TESTMODE USER
5262 004166 005237 002456 INC NOSUPER
5263 004172 005037 060212 CLR DT5+10          ;CLEAR SOME ERROR DATA TAGS
5264 004176 005037 060352 CLR DT14+10
5265 004202 005037 060214 CLR DT5+12
5266 004206 005037 060354 CLR DT14+12
5267 004212 022737 000005 004010 6$: CMP #5,PROTYP       ;CPU TYPE = 11/83/84
5268 004220 001003 BNE 22$          ;NO - BRANCH
5269 004222 052737 000020 172516 BIS #BIT4,MMR3     ;SET UP 22 BIT ADDRESSING
5270 004230 012737 004254 000004 22$: SET4 #7$
      MOV #7$,4
      .DSABL CRF
5271 004236 005037 052314 CLR CPERRF        ;CLEAR THE FLAG
5272 004242 005737 177766 TST @#177766      ;IS THERE A CPU ERROR REGISTER?
5273 004246 012737 177777 052314 MOV #-1,CPERRF    ;YES-TRAPPED
5274 004254 7$: RES4          ;ENABLE TRAPS TO 4

```

004254 012737 034114 000004
004262 022737 000005 004010
004270 001002
004272 005037 177766
004276

1014:

MOV #TIMEOUT,4
CMP #5,PROTYP
BNE 1014
CLR CPUERR

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

.DSABL CRF

5275

5278 004276

SUBTST <<INITIALIZE VARIABLES TO NON ZERO>>

;SUBTEST INITIALIZE VARIABLES TO NON ZERO

5279 004276

SET WORST

MOV #-1,WORST

004276 012737 177777 002600

5280 004304 012737 000003 002616

MOV #3,FLIPLC

5281 004312

SET HEADER

MOV #-1,HEADER

004312 012737 177777 002612

5282 004320 012737 176543 002606

MOV #176543,MSEEDH

5283 004326 012737 123456 002610

MOV #123456,MSEEDL

5284 004334 013737 002606 002602

MOV MSEEDH,SEEDHI ;PRIME THE RANDOM NUMBER GENERATOR

5285 004342 013737 002610 002604

MOV MSEEDL,SEEDLO ;BOTH HIGH AND LOW WORDS

5286 004350 012737 000377 002632

MOV #377,BAKPAT

5287 004356 012737 177400 002634

MOV #177400,SWAPAT

5288 004364

SUBTST <<INITIALIZE VECTORS>>

;SUBTEST INITIALIZE VECTORS

5289 004364 012737 051146 000020

MOV #IOTVEC, IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE

5290 004372 012737 000340 000022

MOV #340, IOTVEC+2 ;LEVEL 7

5291 004400 012737 051502 000030

MOV #ERROR, EMTVEC ;EMT VECTOR FOR ERROR ROUTINE

5292 004406 012737 000340 000032

MOV #340, EMTVEC+2 ;LEVEL 7

5293 004414 012737 056764 000034

MOV #TRAP, TRAPVEC ;TRAP VECTOR FOR TRAP CALLS

5294 004422 012737 000340 000036

MOV #340, TRAPVEC+2 ;LEVEL 7

5295 004430 012737 045352 000024

MOV #PWRDN, PWRVEC ;POWER FAILURE VECTOR

5296 004436 012737 000340 000026

MOV #340, PWRVEC+2 ;LEVEL 7

5297 004444 012737 033744 000114

MOV #PARITY, PARVEC ;GET READY FOR PARITY ERRORS

5298 004452 012737 000340 000116

MOV #340, PARVEC+2

5299 004460 012737 034140 000010

MOV #PDP1105, RESVEC ;RESERVED INSTRUCTION TRAP

5300 004466 012737 000340 000012

MOV #340, RESVEC+2

5301 004474 012737 034114 000004

MOV #TIMEOUT, ERRVEC ;SETUP TIMEOUT ERRORS

5302 004502 012737 000340 000006

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

5303 004510 012737 034126 000250

MOV #MMTRAP, MMVEC ;VECTOR FOR MEMORY MANAGEMENT

5304 004516 012737 000340 000252

MOV #340, MMVEC+2

5305 004524 104423

CACHON ;TURN CACHE ON

5308 004526

SUBTST <<INITIALIZE PATTERNS>>

;*SUBTEST INITIALIZE PATTERNS

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

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

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

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

;*SUBTEST SUBR PLUG IN NULL PATTERNS

FOR I := #1 TO R2

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

ROR R1
ON.NOERROR ;IF CARRY CLEAR
MOV #MT0999,(R3)
END ;OF ON.ERROR
ADD #2,R3
END ;OF FOR
RETURN

MOV #1,I
B0:;;;;;
BCS L0
L0:;;;;;
INC I
CMP I,R2
BLE B0
E0:;;;;;

5348 004702

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

5349
5350
5351

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

5352 004702

.ENABLE LSB
IF RESTART IS FALSE

TST RESTART
BNE L1

004702 005737 002626

004706 001006

5353 004710 012700 002664

5354 004714 005020

5355 004716 022700 003670

5356 004722 001374

5357 004724

004724

1\$: MOV #CONFIG,R0
CLR (R0)+
CMP #CONFIEND,R0
BNE 1\$
END ;OF IF RESTART

L1:::~::~

5358

5359 004724 012737 000002 002106

5360 004732

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

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

5361

5362

5363

5364 004732

004732 012737 004766 000004

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

.ENABL LSB
SET4 #3\$;TRAPS TO 4 GOTO 3\$

5365 004740 012737 177570 002636

5366 004746 012737 177570 002640

5367 004754

004754 022777 177777 175654

004762 001023

5368 004764 000403

5369 004766 012716 004774

5370 004772 000002

5371 004774

004774 012737 034114 000004

005002 022737 000005 004010

005010 001002

005012 005037 177766

005016

3\$: BR 2\$;;BRANCH IF NO TIMEOUT
MOV #2\$,(SP) ;;SET UP FOR TRAP RETURN
RTI
RES4 ;RESET TRAPS TO 4 TO DEFAULT

2\$: 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

5372 005016 012737 000176 002636

5373 005024 012737 000174 002640

5374 005032

005032

5375

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

L2:::~::~

.DSABL LSB


```

5392 005146
      005146 000430
      005150
5393 005150
      005150 023727 000042 002000
      005156 001424
      005160 005737 000042
      005164 001421
5394 005166
      005166 012737 177777 002346
      005174 012737 177777 002062
5395 005202
      005202 023727 000042 014042
      005210 001004
5396 005212
      005212 012737 177777 002350
5397 005220
      005220 000403
      005222
5398 005222
      005222 012737 177777 002354
5399 005230
      005230
5400 005230
      005230
5401 005230
      005230

```

```

ELSE
      IF 42 NE #STACK AND 42 NE #0
          SET QVFLAG,$AUTO
      IF 42 EQ #ENDAD
          SET ACTFLAG
      ELSE
          SET XXDPCHAIN
      END ;OF IF 42
      END ;OF IF 42
      END ;OF IFB $ENV

```

```

BR L6
L5:~~~~~:
      CMP 42,#STACK
      BEQ L7
      TST 42
      BEQ L7
      MOV #-1,QVFLAG
      MOV #-1,$AUTO
      CMP 42,#ENDAD
      BNE L10
      MOV #-1,ACTFLAG
      BR L11
L10:~~~~~:
      MOV #-1,XXDPCHAIN
L11:~~~~~:
L7:~~~~~:
L6:~~~~~:

```


5403 005230

```

SUBTST <<PROTECT PROGRAM & LOADERS>>
;*****
;*SUBTEST    PROTECT PROGRAM & LOADERS
;*****
BIS    #BIT7,CONFIG    ;PROTECT PROGRAM SPACE (BANK 0)
BIS    #BIT7,CONFIG+4  ;PROTECT LOADER SPACE (BANK 1)
IF    #$ENDAD NE 42    ;NOT ACT-11?

                                CMP    #$ENDAD,42
                                BEQ    L12
IF NO22BIT NE #0        ;
                                TST    NO22BIT
                                BEQ    L13
SET    MONFLG           ;RETURN TO XXDP MONITOR
                                MOV    #-1,MONFLG
ERROR +64               ;ILLEGAL PROCESSOR
ELSE
                                BR    L14
                                L13:::
;TYPE PROGRAM TITLE
                                L14:::
                                L12:::
TYPE    MSG000
TYPEIT ,MSG000
.DSABL CRF
END
END ;OF IF #$ENDAD
    
```

5404 005230 052737 000200 002664
5405 005236 052737 000200 002670
5406 005244 022737 014042 000042
005252 001412
5407 005254 005737 002454
005260 001405
5408 005262 012737 177777 002276
005262 104064
5409 005270 104064
5410 005272 000402
005272 000402
005274
5411 005274 104401 071422
005274 104401 071422
5412 005300
005300
5413 005300
005300
5414
5415 005300

```

SUBTST <<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
    
```

5416
5417
5418 005300 012737 005500 000004
005300 012737 005500 000004
5419 005306 005737 177746
5420 005312 012737 005412 000004
005312 012737 005412 000004
5421 005320 005737 172516
5422 005324 013746 000004
5423 005330 012737 005400 000004
5424 005336 032737 001000 177750
5425 005344 001415
5426 005346 104401 071100
005346 104401 071100
5427 005352 052737 000400 177730 12\$:
5428 005360 042737 000077 177734
5429 005366 104401 071547
005366 104401 071547
5430 005372 012637 000004
5431 005376 000411
5432 005400 012637 000004 14\$:
5433 005404 104401 071112
005404 104401 071112

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

```

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

5484 005630

005630 000004

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

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

```

NEWSTST <<BIT TEST OF ALL CSR'S>>
;*****
;*TEST 1 BIT TEST OF ALL CSR'S
;*****
TST1: SCOPE
;* THE FIRST PART OF THE CONFIGURATION ANALYSIS DOES THE FOLLOWING:
;* 1) FINDS WHICH CSR'S RESPOND, AND PUTS THEM INTO THE CSR INFORMATION
;* TABLE, AND STORES ANOTHER BIT FOR "TOTCSRS".
;* 2) TESTS THE CSR BITS COMMON TO ALL CSR'S.
;* 3) FIGURES OUT IF THE MODULE IS A ECC, OR PARITY MEMORY
;* 4) TESTS THE BITS PARTICULAR TO THAT TYPE OF CSR.
;* 5) IF ANY BITS TEST BAD IN THE CSR UNDER TEST, THE CSR OK BIT IN THE
;* CSR INFORMATION TABLE IS CLEARED.
;* THE INFORMATION BITS ONE THROUGH THREE FORM A CODE WHICH GIVES THE TYPE
;* OF CSR:
;*
;* TYPE ECC
;* BIT0
;*
;* MSV11-L/P 0 PARITY
;* MSV11-J 1 ECC
;*
;* THIS MEMORY CODE WILL BE USED IN THE SECOND PART OF THIS ANALYSIS
;*
CLR R5 ;R5 IS THE TOTAL CSR NUMBER
CLR R0 ;R0 IS A TABLE INDEX
MOV #CSRADD,R3 ;R3 HAS THE CSR ADDRESS
MOV #1,NOPAR ;IGNORE PARITY ERRORS
SET4 #CSRBMP
MOV #CSRBMP,4
.DSABL CRF
REPEAT
;*****
; B1:;;;;;
TST (R3) ;DOES THIS CSR RESPOND???
BIS #1,R5 ;MARK IT IN CSR MAP
CLR R4 ;CLEAR THE LAST CSR INDICATOR
BIC #6,CSRINFO(R0) ;CLEAR UNUSED BITS
BIS #BIT4:BIT3,CSRINFO(R0) ;YES-MARK IT IN CSR INFORMATION TABLE
CLR (R3) ;CLEAR THE CSR UNDER TEST
LET (R3) := #BIT13 ;IS THIS AN ECC MEMORY???
MOV #BIT13,(R3)
IF #BIT13 SET.IN (R3);IS BIT 13 SET
BIT #BIT13,(R3)
BEQ L17
BIS #BIT0,CSRINFO(R0);MARK IT IN THE TABLE AS BEING A ECC MEMORY
END ;
; L17:;;;;;
CLR (R3) ;CLEAR CSR UNDER TEST
CALL RWCSR ;BIT TEST OF ALL BITS IN CSR'S
IF CSRINFO(R0) MI #30 ;DO WE HAVE A LEGAL CONFIGURATION?
CMP CSRINFO(R0),#30
BPL L20
MOV CSRINFO(R0),BAD ;MOVE IN BAD DATA
ERROR +21 ;
END ;
; L20:;;;;;
NXTCSR: ADD #2,R0 ;GO TO NEXT CSR
ADD #2,R3 ;GO TO NEXT CSR

```



```

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

```


5544 006034

CSRMAP: SUBTST <<PRINT CSR REGISTER MAP>>

;SUBTEST PRINT CSR REGISTER MAP

5545 006034 005000
5546 006036 104401 071370

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

5547 006042 104401 066064

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

5548 006046 005001
5549 006050 006050

CLR R1 ;
REPEAT ;

B2:::~::~

5550 006050 010102
5551 006052 022702 000011
5552 006056 100002
5553 006060 062702 000007
5554 006064 062702 000060
5555 006070 110237 066062
5556 006074 104401 066062

1\$:

MOV R1,R2 ;
CMP #9.,R2 ;
BPL 1\$;JUMP AROUND NEXT INSTRUCTION
ADD #7,R2 ;
ADD #60,R2 ;MAKE IT ASCII
MOVB R2,MSG015 ;
TYPE MSG015 ;

5557 006100 104401 066060

TYPEIT ,MSG014 ;TYPE SINGLE SPACE
.DSABL CRF

5558 006104 005201
5559 006106 020127 000020
006112 001356
006114

INC R1 ;
UNTIL R1 EQ #16. ;

CMP R1,#16.
BNE B2

E2:::~::~

5560 006114 104401 065544

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

B3:::~::~

5561 006120 005760 002462
5562 006120 001414

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

TST CSRINFO(R0)
BEQ L24

5563 006126 032760 000001 002462
006134 001404

IF #BIT0 SET.IN CSRINFO(R0)

BIT #BIT0,CSRINFO(R0)
BEQ L25

5564 006136 112737 000105 066062
5565 006144 000403
006146

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

BR L26

5566 006146 112737 000120 066062
5567 006154 006154

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

L25:::~::~

5568 006154 000403
006156

ELSE

L26:::~::~

BR L27

5569 006156 112737 000040 066062
5570 006164 006164

MOVB #' ,MSG015
END

L24:::~::~

5571 006164 104401 066062

TYPE MSG015 ;TYPE MEMORY TYPE
TYPEIT ,MSG015

L27:::~::~

5572	006170			.DSABL CRF		
	006170	104401	066060	TYPE MSG014		;TYPE SPACE
				TYPEIT ,MSG014		
5573	006174	000240		.DSABL CRF		
				NOP		;POINT TO NEXT ENTRY
5574	006176	062700	000002	ADD #2,R0		
5575	006202			UNTIL R0 EQ #40		
	006202	020027	000040			CMP R0,#40
	006206	001344				BNE B3
	006210					E3:::~::~
5576	006210			TYPE MSG129		
	006210	104401	071471	TYPEIT ,MSG129		
				.DSABL CRF		
5577	006214	000207		RETURN		
5578	006216	000000		TRACE: .WORD 0		

5580 006220

SUBTST <<READ AND WRITE ALL CSR BITS>>

;SUBTEST READ AND WRITE ALL CSR BITS

5581
5582
5583
5584
5585
5586
5587
5588
5589

; 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

5590 006220
5591 006220

RWCSR:

PUSH R4,R5,UIPARO ;SAVE R4,R5, AND UIPARO ON STACK

MOV R4,-(SP)
MOV R5,-(SP)
MOV UIPARO,-(SP)

006220 010446
006222 010546
006224 013746 177640

LET R5 := R0 ;GET CSR NUMBER FOR POSSIBLE ERROR

MOV R0,R5

5592 006230
006230 010005
5593 006232 006205

ASR R5 ;
LET CSRNO := R5 ;

MOV R5,CSRNO

006234 010537 002152
5595 006240
006240 010337 002034

LET ADDRESS := R3 ;GET ADDRESS FOR POSSIBLE ERROR

MOV R3,ADDRESS

5596 006244
006244 032760 000001 002462
006252 001403

IF #BIT0 SET.IN CSRINFO(R0) ;WHAT KIND OF MEMORY IS THIS??? ;GET BIT MASKS FOR D
BIT #BIT0,CSRINFO(R0)
BEQ L31

5597 006254
006254 012705 017740

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

MOV #017740,R5

5598 006260
006260 000402

ELSE ;IT IS A MSV11-L/P

BR L32

5599 006262
006262 012705 070032

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

L31:::::;

MOV #070032,R5

5600 006266
006266

END ;

L32:::::;

5601 006266
006266 012737 177777 002322

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

MOV #177777,CSR1S

5602 006274 040537 002322
5603 006300
006300 005013

BIC R5,CSR1S ;CLEAR BITS FOR GOOD DATA
LET (R3) := #0 ;0----->CSR

CLR (R3)

5604 006302
006302 011304

LET R4 := (R3) ;MASK OUT UNWANTED BITS

MOV (R3),R4

5605 006304 040504
5606 006306
006306 005704

BIC R5,R4 ;
IF R4 NE #0 ;DO WE HAVE A CORRECT READ

TST R4
BEQ L33

5607 006312
006312 005037 002044

LET GOOD := #0 ;GOOD DATA=0'S

CLR GOOD

5608 006316
006316 010437 002150

LET CSR := R4 ;BAD DATA=CSR

MOV R4,CSR

5609 006322 104035
5610 006324 042760 000010 002462

ERROR +35 ;BIT SET ERROR
BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT

5611 006332
006332

END

L33:::::;

5612 006332

LET (R3) := CSR1S ;ONES--->(R3)


```

5613 006332 013713 002322          LET R4 := (R3)          ;MASK OUT CORRECT FIELD          MOV CSR1S,(R3)
      006336 011304          ;CLEAR OUT CSR          MOV (R3),R4
5614 006340 005013          CLR (R3)
5615 006342 040504          BIC R5,R4
5616 006344 020437 002322          IF R4 NE CSR1S          ;WAS PATTERN WRITTEN CORRECTLY?
      006350 001411          ;GOOD DATA = ALL LEGAL BITS SET
5617 006352 013737 002322 002044          LET GOOD := CSR1S          ;GOOD DATA = ALL LEGAL BITS SET          IN CSR
      006352 010437 002150          LET CSR := R4          ;BAD DATA=CSR          MOV CSR1S,GOOD
5618 006360 010437 002150          ERROR +10          ;BIT CLEAR ERROR          MOV R4,CSR
5619 006364 104010          BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT
5620 006366 042760 000010 002462          END
5621 006374          ;
      006374          ;
5622 006374          LET PASFLG := #0          ;SET UP LOOP COUNTER          L34:::
      006374 005037 002264          REPEAT          ;REPEAT WITH A FIELD OF 1'S THROUGH 0'S          CLR PASFLG
5623 006400          ;
      006400          ;
5624          ;
5625 006400 005237 002264          INC PASFLG          ;INCREMENT LOOP COUNTER          O'S THROUGH 1'S
5626 006404          LET UIPARO := #-1          ;USE USER PAR FOR BIT COUNTER
5627 006412 012737 177777 177640          IF PASFLG EQ #1          ;PASS 1          MOV #-1,UIPARO
      006412 023727 002264 000001          ;
      006420 001003          LET R2 := #1          ;1----->FIELD OF ZEROS          CMP PASFLG,#1
5628 006422          ;
      006422 012702 000001          ELSE          ;PASS 2          BNE L35
5629 006426          ;
      006426 000402          ;
      006430          ;
5630 006430          LET R2 := #177776          ;0----->FIELD OF ONES          L35:::
      006430 012702 177776          ;
5631 006434          ;
      006434          ;
5632 006434          REPEAT          ;DO BITS 0-4 AND 13-15          L36:::
      006434          ;
5633 006434 005237 177640          INC UIPARO          ;INCREMENT BIT POINTER          B5:::
5634 006440          IF PASFLG EQ #1 AND #BIT0 OFF.IN CSRINFO(R0) ;
      006440 023727 002264 000001          ;
      006446 001006          ;
      006450 032760 000001 002462          ;
      006456 001002          ;
5635 006460 042702 040000          BIC #BIT14,R2          ;IF THIS IS PASS 1 ON A MSV11-L/P,CLEAR BIT 14
5636 006464          ;
      006464          ;
5637 006464          IF PASFLG EQ #2 AND #BIT0 OFF.IN CSRINFO(R0) ;          L37:::
      006464 023727 002264 000002          ;
      006472 001006          ;
      006474 032760 000001 002462          ;
      006502 001002          ;
5638 006504 042702 040004          BIC #BIT14!BIT2,R2          ;IF THIS IS PASS 2 ON A MSV11-L/P, CLEAR ECRSRD BIT AND
5639 006510          ;
      006510          ;
5640 006510          LET (R3) := R2          ;WRITE DATA          L40:::
      006510 010213          ;
      006510          MOV R2,(R3)

```



```

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

```

```

006654 001411
5668 006656 012737 140005 002044 LET GOOD := #140005 ;GOOD DATA BEQ L51
006656 012737 140005 002044 MOV #140005,GOOD
5669 006664 010237 002150 LET CSR := R2 ;BAD DATA MOV R2,CSR
006664 010237 002150 ERROR +10 ;BIT CLEAR ERROR
5670 006670 104010 BIC #BIT3,CSRINFO(R0) ;CLEAR CSR OK BIT!
5671 006672 042760 000010 002462 END ;
5672 006700 006700 DONE: LET (R3) := #0 ;CLEAR OUT CSR L51:::::
5673 006700 005013 POP UIPARO,R5,R4 ;RESTORE UIPARO,R4, AND R5 CLR (R3)
5674 006702 012637 177640 MOV (SP)+,UIPARO
006702 012637 177640 MOV (SP)+,R5
006706 012605 MOV (SP)+,R4
006710 012604
5675 006712 000207 RETURN

```

```

5678                                     ;THE FOLLOWING ROUTINE DETERMINES WHICH CSR CONTROLS PROGRAM SPACE
5679                                     ;
5680 006714 104424                       ;TEST: CACHOFF
5681 006716 012737 177777 002532        MOV #177777,PGMCSR
5682 006724 012737 002000 172350        MOV #2000,KIPAR4           ;SET UP MAP REGISTER
5683 006732 012701 002412                MOV #TESTADD,R1
5684 006736 012737 100000 002412        MOV #100000,TESTADD
5685 006744 012737 100002 002414        MOV #100002,TESTADD+2
5686 006752 005000                        CLR R0           ;CLEAR CSR COUNTER
5687 006754 005037 002152                CLR CSRNO
5688 006760 013703 002224                MOV TOTCSRS,R3      ;OBTAIN CSR MAP
5689 006764 000240                        NOP              ;DEBUG AID
5690 006766 006303                        4$: ASL R3         ;PUT HIGH ORDER BIT INTO C BIT
5691 006770 103407                        BCS 2$          ;BRANCH IF CSR EXISTS
5692 006772 062700 000002                1$: ADD #2,R0     ;UPDATE CSR COUNTER
5693 006776 010037 002152                MOV R0,CSRNO
5694 007002 005703                        TST R3          ;IS MAP EMPTY?
5695 007004 001465                        BEQ 3$          ;BRANCH IF SO
5696 007006 000767                        BR 4$
5697 007010 000240                        2$: NOP         ;DEBUG AID
5698 007012 000241                        CLC            ;CLEAR CARRY
5699 007014 032760 000001 002462        BIT #BIT0,CSRINFO(R0)  ;IS THIS PARITY MEMORY?
5700 007022 001014                        BNE 5$         ;BRACH IF NOT
5701 007024 052760 000004 172100        BIS #BIT2,CSRADD(R0)  ;SET WRITE WRONG PARITY
5702 007032 012771 123456 000000        MOV #123456,@(R1)    ;WRITE DATA
5703 007040 012771 123456 000002        MOV #123456,@2(R1)
5704 007046 005060 172100                CLR CSRADD(R0)      ;RESTORE CSR
5705 007052 000414                        BR 6$
5706 007054 012760 000000 172100        5$: MOV #0,CSRADD(R0) ;CLEAR THE CSR UNDER TEST
5707 007062 012771 123456 000000        MOV #123456,@(R1)   ;WRITE DATA
5708 007070 012771 123456 000002        MOV #123456,@2(R1)
5709 007076 012760 020006 172100        MOV #20006,CSRADD(R0) ;SET DIAG CHECK MODE
5710 007104 005771 000000                6$: TST @(R1)     ;WRITE CHECKBITS TO CSR
5711 007110 016004 172100                MOV CSRADD(R0),R4   ;WRITE CSR TO R4
5712 007114 032760 000001 002462        BIT #BIT0,CSRINFO(R0) ;PARITY MEMORY?
5713 007122 001003                        BNE 7$         ;BRANCH IF NOT
5714 007124 005704                        TST R4          ;PARITY ERROR?
5715 007126 100412                        BMI 8$         ;BRACH IF SO
5716 007130 000720                        BR 1$          ;TRY NEXT CSR

```

```

5717 007132 000240          7$:  NOP          ;DEBUG AID
5718 007134 072427 177773    ASH        #-5,R4
5719 007140 042704 177700    BIC        #+C77,R4
5720 007144 0127C2 000040    MOV        #40,R2          ;LOAD IN CORRECT CHECK BITS FOR MSV11-J
5721 007150 020204          CMP        R2,R4          ;CORRECT CHECKBITS?
5722 007152 001307          BNE        1$             ;BRANCH IF NOT
5723 007154 010037 002532    8$:  MOV        R0,PGMCSR
5724 007160 000240          3$:  NOP
5725 007162 104502          CLRCSR
5726 007164 012771 000000 000000  MOV        #0,@(R1)        ;DEBUG AID
5727 007172 012771 000000 000002  MOV        #0,@2(R1)        ;CLEAR ALL CSR'S
5728 007200 023727 002532 177777  CMP        PGMCSR,#177777    ;RESTORE TEST LOCATIONS
5729 007206 001402          BEQ        FINT$
5730 007210 000137 007232          JMP        CLRMEM
5731 007214 012737 001000 172350 FINT$: MOV        #1000,KIPAR4
5732 007222          TYPE        MSG126
          007222 104401 071264    TYPEIT     ,MSG126
          .DSABL CRF
          CLR        PGMCSR          ;IF PROGRAM CSR NOT FOUND GO TO FINT
          ;GO TO SIZING ROUTINE IF FOUND
          ;
          ;ERROR - PROGRAM CSR NOT FOUND!

5733 007226 005037 002532          ;SET TO DEFAULT OF 0

```


5735 007232

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

;SUBTEST CLEAR ALL MEMORY SPACE FROM BANK 2 ON

5736
5737
5738
5739
5740
5741

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

5742 007232 012737 007342 000004

CLRMEM: SET4 #CLREX ;NONEM TRAPS GO TO CLREX
MOV #CLREX,4

5743 007240 005037 006216
5744 007244 012737 000001 002076

.DSABL CRF
CLR TRACE
MOV #1,NOPAR ;IGNORE PARITY ERRORS

5745 007252 012737 002000 172350
5746 007260 012701 100000

MOV #2000,KIPAR4 ;SET UP MAP TO START AT BANK 2
MOV #100000,R1 ;R1 MAPS TO KIPAR4

5747 007264 020127 117776
5748 007270 001003

1\$: CMP R1,#117776 ;WHOLE 16K BANK DONE?
BNE 2\$;KEEP GOING IF NOT

5749 007272 012737 177777 006216
5750 007300 005021

2\$: MOV #-1,TRACE ;USE TRACE FLAG TO FLAG END OF BANK
CLR (R1)+ ;CLEAR CONTENTS & INCREMENT

5751 007302 005737 006216
5752 007306 001001

TST TRACE ;EOB FLAG SET?
BNE 3\$;GO TO NEXT BANK IF SO

5753 007310 000765
5754 007312 062737 000200 172350

3\$: BR 1\$
ADD #200,KIPAR4 ;SET MAP FOR NEXT BANK
CMP #177600,KIPAR4 ;ARE WE AT THE PERIPHERAL PAGE

5755 007320 022737 177600 172350
5756 007326 001405

BEQ CLREX ;YES-GO ON
CLR TRACE ;RESET FLAG

5757 007330 005037 006216
5758 007334 012701 100000

MOV #100000,R1 ;RESET R1
BR 1\$;CLEAR NEXT BANK

5759 007340 000751
5760 007342 000240

CLREX: NOP
CLR TRACE

5761 007344 005037 006216
5762 007350

RES4
MOV #TIMEOUT,4
CMP #5,PROTYP ;IS THIS AN 11/83/84 ?

007356 022737 000005 004010
007364 001002

BNE 101\$;BRANCH IF NOT
CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS

007366 005037 177766
007372

101\$: ;THAT A EXPECTED TRAP COULD HAVE SET
.DSABL CRF

5765 007372

```

ANA2:  SUBTST  <<MATCH ALL CSR'S WITH MEMORY>>
;*****
;SUBTEST    MATCH ALL CSR'S WITH MEMORY
;*****
;* THE SECOND PART OF THE ANALYSIS MATCHES UP THE CSR'S WITH THE MEMORY, AND
;* INSTALLS ALL THE INFORMATION FOUND IN THE CONFIGURATION TABLE.  FOR ECC,
;* THIS IS DONE BY TAKING EACH CSR FOUND IN THE PREVIOUS SECTION SEQUENTIALLY
;* AND CHECKING THROUGH ALL OF MEMORY, ONE BANK AT AT 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.
; *

```

5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789

5790 007372 012737 010444 000004

5791 007400 005037 002314
5792 007404 012701 002412
5793 007410 013703 002224
5794 007414 005000
5795 007416 005005
5796 007420 005737 002454
5797 007424 001403
5798 007426 005037 002560
5799 007432 000413
5800 007434 022737 000177 002556 7\$:
5801 007442 001407
5802 007444 013702 002556
5803 007450 005202
5804 007452 072227 000011
5805 007456 010237 002560
5806 007462 012702 000004 1\$:
5807 007466 005037 002564
5808 007472 012737 001000 172350
5809 007500 012737 001000 172352
5810 007506 006303 2\$:
5811 007510 103420
5812 007512 062700 000002
5813 007516 010037 002152
5814 007522 005703
5815 007524 001370
5816 007526 012737 001000 172350

```

SET4  #100$ ;NE MEMORY TRAPS GO TO 100$
MOV   #100$,4
.DSABL CRF
CLR   CHECK ;CLEAR CHECK
MOV   #TESTADD,R1 ;SET UP THE VIRTUAL ADDR. POINTER
MOV   TOTCSRS,R3 ;MOVE CSR MAP INTO R3
CLR   R0 ;CLEAR THE CSR POINTER
CLR   R5 ;CLEAR THE PROGRAM CSR STATUS POINTER
TST   NO22BIT ;DO WE HAVE 22 BIT ADDRESSING ?
BEQ   7$ ;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

```



```

5817 007534 012737 001200 172352      MOV      #1200,KIPAR5      ;RESTORE KIPAR5
5818 007542 013706 002574              MOV      KSTACK,SP        ;RESTORE STACK
5819 007546 000137 010454              JMP      SUBAAS            ;JUMP TO SUBAAS IF ALL CSR'S ARE DONE
5820 007552 010037 002152      3$:     MOV      RO,CSRNO        ;MAKE SURE CSRNO IS UPDATED
5821 007556 104424      13$:     CACHOFF                    ;TURN THE CACHE OFF
5822 007560 000240              NOP
5823 007562 012737 100000 002412 45$:     MOV      #100000,TESTADD   ;SET UP VIRTUAL ADDRESS TO KIPAR4
5824 007570 012737 120002 002414      MOV      #120002,TESTADD+2 ;SET UP VIRTUAL ADDRESS TO KIPAR5
5825 007576 032762 000040 002664      BIT      #BIT5,CONFIG(R2)  ;IS THIS A BANK TO SKIP ECC/LOGIC TESTS?
5826 007604 001402              BEQ      43$              ;NO - BRANCH AROUND NEXT INSTRUCTION
5827 007606 000137 010342              JMP      6$              ;YES - GO TO END OF BANK
5828 007612 005037 002452      43$:     CLR      I                ;CLEAR THE MEMORY CONFIGURATION COUNTER
5829 007616 005771 000000      4$:     TST      @ (R1)          ;TEST TO SEE THAT THERE IS MEMORY PRESENT
5830 007622 005237 002452              INC      I                ;MEMORY PRESENT
5831 007626              PUSH     @ (R1),@2(R1)    ;SAVE THE LOCATIONS UNDER TEST
      007626 017146 000000              MOV @ (R1),-(SP)
      007632 017146 000002              MOV @2(R1),-(SP)
5832 007636 032760 000001 002462      BIT      #BIT0,CSRINFO(RO) ;IS THIS PARITY MEMORY?
5833 007644 001014              BNE     34$              ;NO - BRANCH
5834 007646 052760 000004 172100      BIS      #BIT2,CSRADD(RO) ;SET WRITE WRONG PARITY
5835 007654 012771 123456 000000      MOV      #123456,@ (R1)   ;SET THE FIRST LOCATION UNDER TEST
5836 007662 012771 123456 000002      MOV      #123456,@2(R1)  ;SET THE SECOND LUT
5837 007670 005060 172100              CLR      CSRADD(RO)      ;CLEAR THE CSR
5838 007674 000411              BR      41$              ;TEST LOCATIONS
5839 007676 012771 123456 000000 34$:     MOV      #123456,@ (R1)   ;SET THE FIRST LOCATION UNDER TEST
5840 007704 012771 123456 000002      MOV      #123456,@2(R1)  ;SET THE SECOND LUT
5841 007712 104503              CLR1CSR                    ;RESET CSR
5842 007714 104475              CB1CSR                    ;SET DIAG. CHECK MODE IN CSR UNDER TEST
5843 007716 000240              NOP
5844 007720 005771 000000      41$:     TST      @ (R1)          ;READ THE FIRST LUT TO WRITE CKBITS. INTO CSR
5845 007724 104426              READCSR                    ;READ THE CSR UNDER TEST
5846 007726 000240              NOP
5847 007730 013704 002150              MOV      CSR,R4           ;GET THE CHECKBITS FROM THE CSR
5848 007734 000240              NOP
5849 007736 010437 002434              MOV      R4,TEMP         ;SAVE IN TEMP FOR LATER
5850 007742 104503              CLR1CSR                    ;RESET CSR
5851 007744              POP      @2(R1),@ (R1)   ;RESTORE LOCATIONS UNDER TEST
      007744 012671 000002              MOV (SP)+,@2(R1)
      007750 012671 000000              MOV (SP)+,@ (R1)
5852 007754 032760 000001 002462      BIT      #BIT0,CSRINFO(RO) ;IS THIS PARITY MEMORY?
5853 007762 001004              BNE     42$              ;NO - BRANCH
5854 007764 005704              TST      R4              ;DID WE GET A PARITY ERROR?
5855 007766 100420              BMI     25$              ;YES - FILL IN CONFIG TABLE
5856 007770 000137 010342              JMP      6$              ;NO - JUMP TO END OF BANK
5857 007774 072427 177773      42$:     ASH      #-5,R4          ;MANIPULATE THE CSR BITS
5858 010000 042704 177700              BIC      #+C77,R4        ;INTO A USABLE FORM.
5859 010004 012737 000040 002316      MOV      #40,CBITS       ;MSV11-J CHECK BITS
5860 010012 000240              NOP
5861 010014 023704 002316      77$:     CMP      CBITS,R4        ;DO THE CHECKBITS COMPARE TO WHAT WAS WRITTEN?
5862 010020 000240              NOP
5863 010022 001402              BEQ     25$              ;DEBUG AIDE
5864 010024 000137 010170              JMP      22$              ;BRANCH IF THERE IS A MATCH
5865              ;*
5866              ;* WE COME HERE IF THERE IS A MATCH
5867              ;*
5868 010030 010004      25$:     MOV      RO,R4           ;GET THE CSR NUMBER
5869 010032 000240              NOP

```

```

5870 010034 006204          ASR      R4          ;SET IT UP FOR USE IN THE
5871 010036 000304          SWAB     R4          ;CONFIGURATION TABLE.
5872 010040 042704 170377    BIC      #170377,R4   ;CLEAR OFF EXTRANEIOUS BITS
5873 010044 050462 002664    15$:    BIS      R4,CONFIG(R2) ;PUT CSR NUMBER IN CONFIG. TABLE
5874 010050 016004 002462    MOV      CSRINFO(R0),R4 ;GET MEMORY TYPE
5875 010054 042704 177770    BIC      #+C7,R4     ;CLEAR OFF THE EXTRANEIOUS BITS
5876 010060 000304          SWAB     R4          ;MOVE INTO PROPER POSITION
5877 010062 050462 002666    BIS      R4,CONFIG+2(R2) ;SET IT INTO THE CONFIG TABLE
5878
5879
5880
5881 010066 022737 001000 172350 24$:    CMP      #1000,KIPAR4   ;IS THIS BANK 1 ?
5882 010074 001402          BEQ      30$          ;BRANCH IF TRUE
5883 010076 000137 010342    JMP      6$          ;ELSE JUMP TO END OF THIS BANK
5884 010102 032737 100020 002434 30$:    BIT      #BIT15!BIT4,TEMP ;WAS THERE A SBE OR DBE?
5885 010110 001417          BEQ      10$          ;BRANCH IF NOT
5886 010112 013704 002434    MOV      TEMP,R4      ;GET CSR CONTENTS
5887 010116 072427 177767    ASH     #-9.,R4      ;MAKE ERROR ADDRESS INTO BANK #
5888 010122 022704 000001    CMP      #1,R4        ;ERROR IN BANKS 0 OR 1?
5889 010126 003010          BGT      10$          ;BRANCH IF NOT
5890 010130 052762 000001 002664    BIS      #BIT0,CONFIG(R2) ;SET ERROR FLAG IN CONFIG TABLE
5891 010136 105262 002666    INCB    CONFIG+2(R2)  ;ADD ONE TO BANK ERROR COUNT
5892 010142          SET      CONFGERROR   ;PRINT CONFIG TABLE
5893 010150 053737 002670 002664 10$:    BIS      CONFIG+4,CONFIG ;SET UP INFORMATION IN BANK ZERO
5894 010156 053737 002672 002666    BIS      CONFIG+6,CONFIG
5895 010164 000240          NOP
5896 010166 000465          BR      6$          ;DEBUG AID
5897
5898
5899
5900 010170 032737 100020 002150 22$:    BIT      #BIT15!BIT4,CSR ;SBE OR DBE FLAGS SET?
5901 010176 001001          BNE     8$          ;BRANCH IF TRUE
5902 010200 000460          BR      6$          ;CHECK TO SEE IF IT IS MSV11-J
5903 010202 013704 002152          MOV     CSRNO,R4     ;GET CSRNO
5904 010206 042764 000006 172100 8$:    BIC     #6,CSRADD(R4) ;TURN OFF DIAG CHECK & ECC DISABLE
5905 010214          PUSH    R0,R1      ;SAVE R0 & R1
5906 010214 010046          MOV     CSRADD(R4),R1 ;GET CSR INFORMATION
5907 010216 010146          MOV     R0,-(SP)    ;MOV R1,-(SP)
5908 010220 016401 172100          MOV     R1,-(SP)
5909 010224 072127 177773    ASH     #-5,R1      ;SET UP ERROR ADDRESS
5910 010230 042701 177600    BIC     #+C177,R1
5911 010234 052764 040000 172100 27$:    BIS     #BIT14,CSRADD(R4) ;GET EXTENDED ERROR ADDRESS BITS
5912 010242 016400 172100    MOV     CSRADD(R4),R0 ;READ FROM CSR
5913 010246 042764 040000 172100 27$:    BIC     #BIT14,CSRADD(R4) ;TURN OFF EQB BIT
5914 010254 042700 177037    BIC     #+C740,R0    ;SET UP EXTENDED BITS
5915 010260 006300          ASL     R0
5916 010262 006300          ASL     R0
5917 010264 060001          ADD     R0,R1      ;SET UP TOTAL ERROR ADDRESS
5918 010266 010104          MOV     R1,R4      ;SAVE IN R4
5919 010270 012601          POP     R1,R0     ;RESTORE R0 & R1
5920 010272 012600          MOV     (SP)+,R1
5921 010274 072427 000005          MOV     (SP)+,R0
5922 010276 020437 172350    ASH     #5,R4      ;SET ERROR ADDRESS UP IN PAR NOTATION
5923 010278 020437 172350    CMP     R4,KIPAR4   ;DOES IT EQUAL KIPAR4?
5924 010280 001001          BNE     28$        ;BRANCH IF FALSE
5925 010282 000403          BR      35$        ;YES - MARK INFO IN CONFIG TABLE

```



```

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

```

5952 010454 104423
5953 010456 104472
5954 010460

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

;*TEST 2 TEST BANK 0 ACCESSES

010460 000004

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

TST2: SCOPE
;THIS DOES A "TST" INSTRUCTION ON EVERY LOCATION IN BANK #0 TO SEE
;IF IT GETS ANY PARITY TRAPS.
;SINCE EVERY LOCATION IS EITHER LOADED OR WRITTEN INTO BY THE PROGRAM
;PRIOR TO THIS POINT - THEN A PARITY ERROR IMPLIES THAT THERE IS A
;HARDWARE FAILURE IN THE MEMORY.
;THESE ERRORS ARE COUNTED AND A FATAL ACTION IS TAKEN
CLR PARCNT ;CLEAR PARITY ERROR COUNTER
MOV #1,NOPAR ;SET THE NO PARITY ERROR FLAG
CLR NEMCNT ;CLEAR NON-EXISTANT MEMORY ERROR COUNTER
MOV #1,NONEM ;SET THE NON-EXISTANT MEMORY ERROR MODE TO COUNT
SET4 #NONEXIST ;TRAPS TO 4 GOTO NONEXIST
MOV #NONEXIST,4
.DSABL CRF

5966 010514 005000
5967 010516 012701 040000
5968 010522 104424
5969 010524 005720
5970 010526 077102
5971 010530 104423
5972
5973 010532 005737 002072
5974 010536 001403
5975 010540
010540 005237 002064
010544 104003

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

5976 010546 005737 002070
5977 010552 001406
5978 010554 162737 000002 002034
5979 010562
010562 005237 002064
010566 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

5980 010570 053737 002106 002664
5981 010576
010576 012737 034114 000004
010604 022737 000005 004010
010612 001002
010614 005037 177766
010620

3\$: BIS CPUBIT,CONFIG ;SET CORRECT ACCESSED BIT ON BANK 0
RES4 ;RESET TRAPS TO 4 TO DEFAULT
MOV #TIMEOUT,4
CMP #5,PROTYP ;IS THIS AN 11/83/84 ?
BNE 101\$;BRANCH IF NOT
CLR CPUERR ;CLEAR OUT THE CPU ERROR REGISTER BITS

5982
5983 010620

101\$: ;THAT A EXPECTED TRAP COULD HAVE SET
.DSABL CRF

SUBTST <<ENABLE ECC FOR CORRECT TRAPS>>

;*SUBTEST ENABLE ECC FOR CORRECT TRAPS

5984 010620
010620 032777 000001 172010
010626 001003

IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
BIT #SWO,@SWR
BNE L52

```

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

```

```

ENASBE
ELSE

ECCINIT
END ;OF IF #SWO

```

```

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

```


5991 010644

NEWSTST <<TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES>>

;*TEST 3 TEST BANKS 1-200 (OCTAL) FOR ZEROS & ONES

010644 000004

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!

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

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

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

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)
CALL MAPPER
.DSABL CRF

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

MOV (SP)+,R3
TSTB CONFIG(R1) ;IS THIS BANK PROTECTED?
BMI TSTBANK ;YES - GO TEST BANK SPECIAL
MOV #207,@LINK1 ;PUT "RETURN" INSTRUCTION AFTER WRITE ROUTINE
MOV #FIRST,R0
MOV R0,R4
MOV #SIZE,R1
MOV R1,R3
CLR R2 ;DATA IS ZEROS
CACHOFF ;TURN CACHE OFF
TESTAREA ;ENTER SUPERVISOR MODE
BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF

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

CALL MTST3
2\$: KERNEL
CACHON ;ENTER KERNEL MODE
NOP ;TURN CACHE ON
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 ?


```

011066 001002          BNE      101$      ;BRANCH IF NOT
011070 005037 177766   CLR      CPUERR    ;CLEAR OUT THE CPU ERROR REGISTER BITS
011074          101$:          ;THAT A EXPECTED TRAP COULD HAVE SET

        .DSABL CRF
6033 011074 005037 002076 CLR      NOPAR      ;INDICATE DEFAULT PARITY ACTION
6034 011100 000557          BR      SUBAAI
6035 011102 005737 002070 TAG3$: TST     NEMCNT    ;ANY TRAPS?
6036 011106 001401          BEQ     1$         ;NO - SKIP
6037 011110 000677          BR      TAG9$      ;NOW - TRY NEXT BANK
6038 011112 104424          1$: CACHOFF ;TURN CACHE OFF
6039 011114          TESTAREA ;ENTER SUPERVISOR MODE
        011114 053737 002552 177776 BIS     TESTMODE,PSW ;GO TO SYSTEM TEST MODE
        .DSABL CRF
6040 011122 004777 171376 CALL     @LINK2    ;FINISH PATTERN
6041 011126 104417          KERNEL ;ENTER KERNEL MODE
6042 011130 104423          CACHON  ;TURN CACHE ON
6043 011132 000240          NOP     ;DEBUG AID
6044 011134 005737 002074 TST     PATERR     ;ANY PATTERN ERRORS
6045 011140 001032          BNE     2$         ;YES - SKIP
6046 011142 005737 002072 TST     PARCNT    ;ANY PARITY ERRORS
6047 011146 001027          BNE     2$         ;YES - SKIP
6048 011150 005737 002070 TST     NEMCNT    ;ANY NON EXISTANT MEMORY
6049 011154 001024          BNE     2$         ;YES - SKIP
6050 011156 012700 060000 MOV     #FIRST,R0
6051 011162 010004          MOV     R0,R4
6052 011164 012701 040000 MOV     #SIZE,R1
6053 011170 010103          MOV     R1,R3
6054 011172 013702 002614 MOV     ONES,R2
6055 011176 012777 000240 171316 MOV     #000240,@LINK1 ;DATA IS ONES
6056 011204 104424          CACHOFF ;PUT "NOP" INSTRUCTION BACK IN SUBROUTINE
6057 011206          TESTAREA ;TURN CACHE OFF
        011206 053737 002552 177776 BIS     TESTMODE,PSW ;ENTER TEST MODE
        .DSABL CRF ;GO TO SYSTEM TEST MODE
        CALL MTST3
6058 011214 004737 011414          KERNEL ;DO IN MEMORY IF NOT
6059 011220 104417          CACHON  ;ENTER KERNEL MODE
6060 011222 104423          NOP     ;TURN CACHE ON
6061 011224 000240          MOV     BANKINDEX,R0 ;DEBUG AID
6062 011226 013700 002104 2$: TST     PATERR     ;ANY PATTERN ERRORS?
6063 011232 005737 002074          BNE     3$         ;YES - SKIP
6064 011236 001006          TST     PARCNT    ;ANY PARITY ERRORS?
6065 011240 005737 002072          BNE     3$         ;YES - SKIP
6066 011244 001003          TST     NEMCNT    ;ANY HOLES?
6067 011246 005737 002070          BEQ     4$         ;NONE - SKIP
6068 011252 001406          3$: BIS     #BIT0,CONFIG(R0) ;SET ERROR BIT IN THIS BANK
6069 011254 052760 000001 002664 SET     CFGERROR    ;FORCE PRINTING OF CONFIGURATION TABLE
6070 011262          MOV     #-1,CFGERROR
        011262 012737 177777 002450
6071 011270 053760 002106 002664 4$: BIS     CPUBIT,CONFIG(R0) ;SET ACCESSED BIT
6072 011276 000137 010710          JMP     TAG9$
6073
6074          ;TEST A PROTECTED BANK
6075          TSTBANK:PUSH R1
        011302 010146
6076 011304 012737 000001 002100 MOV     #1,NONEM   ;SET NON-EXISTANT MEMORY TO COUNT
6077 011312 012700 060000          MOV     #FIRST,R0
6078 011316 012701 020000          MOV     #20000,R1
    
```

```

6079 011322 104424          CACHOFF          ;TURN CACHE OFF
6080 011324          TESTAREA        ;ENTER TEST MODE
      011324 053737 002552 177776  BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
      .DSABL CRF
6081 011332 005720          TST (R0)+
6082 011334 077102          SOB R1,4$
6083 011336 104417          KERNEL
6084 011340 104423          CACHON          ;ENTER KERNEL MODE
6085 011342 012737 000002 002100  MOV #2,NONEM    ;TURN CACHE ON
6086 011350          POP R1                ;RESET NON-EXISTANT MEMORY TO EXIT TEST LOOP
      011350 012601
6087 011352          IF PARCNT NE #0
      011352 005737 002072
      011356 001406
6088 011360 052761 000001 002664  BIS #BIT0,CONFIG(R1) ;ERROR BANK
6089 011366          SET CONFGERROR
      011366 012737 177777 002450  MOV #-1,CONFGERROR
6090 011374          END ;OF IF PARCNT
      011374
6091 011374          IF NEMCNT EQ #0
      011374 005737 002070
      011400 001003
6092 011402 053761 002106 002664  BIS CPUBIT,CONFIG(R1) ;ACCESSED BANK
6093 011410          END ;OF IF NEMCNT
      011410
6094 011410 000137 010710          JMP TAG9$
6095 011414 010220          MTST3: MOV R2,(R0)+
6096 011416 077102          SOB R1,MTST3
6097 011420 000240          NOP
6098 011422 012401          2$: MOV (R4)+,R1
6099 011424 020102          CMP R1,R2
6100 011426 001402          BEQ 3$
6101 011430 005237 002074          INC PATERR
6102 011434 077306          3$: SOB R3,2$
6103 011436 000207          RETURN

```

```

MOV (SP)+,R1
TST PARCNT
BEQ L55
MOV #-1,CONFGERROR
L55:;;;;;
TST NEMCNT
BNE L56
L56:;;;;;
;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

```

```

6105 011440      SUBAAI: SUBTST  <<FIND SHADOW INHIBIT MODE POINTERS>>
;*****
;*SUBTEST      FIND SHADOW INHIBIT MODE POINTERS
;*****
;* THIS SECTION LOOKS FOR INTERLEAVED MSV11-J MEMORIES AND FIGURES OUT
;* WHERE THE SHADOW INHIBIT MODE POINTERS ARE LOCATED.  THESE AREAS
;* ARE THEN MARKED AS PROGRAM SPACE.
6109 011440 005037 002102      CLR      BANK      ;RESET BANK TO ZERO
6110 011444 004737 040052      SHADL1: CALL  EXBANK ;SET BANK PARAMETERS
6111 011450 013700 002104      MOV      BANKINDEX,R0
6112 011454      IF ACFLAG IS TRUE AND INTFLAG IS TRUE
        011454 005737 002116      TST ACFLAG
        011460 001414      BEQ L57
        011462 005737 002136      TST INTFLAG
        011466 001411      BEQ L57
6113 011470 062702 000040      ADD      #40,R2      ;POINT TO BANKINDEX + 8
6114 011474 062737 000020 002102  ADD      #20,BANK    ;POINT TO BANK + 16
6115 011502 052760 000200 002664  BIS      #BIT7,CONFIG(R0) ;MAKE NEW BANK PROGRAM SPACE
6116 011510      ELSE
        011510 000402      BR L60
        011512      L57:;;;;;
6117 011512 005237 002102      INC      BANK      ;GO TO NEXT BANK
6118 011516      END; OF IF ACFLAG
        011516      L60:;;;;;
6119 011516 023737 002556 002102  CMP      LASTBANK,BANK ;HAVE WE DONE ALL THE BANKS?
6120 011524 002347      BGE      SHADL1      ;BRANCH IF NOT

```


6123 011526

NEWST <<ECC INHIBIT MODE POINTER TEST>>

 ;*TEST 4 ECC INHIBIT MODE POINTER TEST

 TST4: SCOPE

011526 000004

6124
 6125
 6126
 6127
 6128
 6129
 6130
 6131
 6132
 6133
 6134
 6135
 6136
 6137
 6138
 6139
 6140
 6141 011530 104424
 6142 011532 012737 177777 002156
 6143 011540
 011540 005037 002102
 011544
 6144 011544 012701 060000
 6145 011550 004737 040052
 6146 011554 013700 002104
 6147 011560
 011560 005737 002116
 011564 001436
 6148 011566
 011566 005737 002120
 011572 001433
 6149 011574
 011574 005737 002342
 011600 001030
 6150 011602 012703 000002
 6151 011606 116002 002665
 6152 011612 006302
 6153 011614 042702 177741
 6154 011620 010237 002152
 6155 011624
 011624 023737 002152 002156
 011632 001413
 6156 011634 013737 002152 002156
 6157 011642
 011642 005737 002122
 011646 001003
 6158 011650 052760 000100 002664
 6159 011656
 011656
 6160 011656 004737 011750
 6161 011662
 011662

;THE MSV11-J OR MF11S-K INHIBIT ECC DISABLE AND DIAGNOSTIC CHECK MODE
 ;ON THE BOTTOM FIRST OR SECOND 16K WORDS CONTROLLED BY A CSR. THIS
 ;IS CONSIDERED TO BE A PROTECTED BANK BY THE PROGRAM. IT MAY BE
 ;QUITE COMPLEX TO DETERMINE ON A GIVEN SYSTEM CONFIGURATION WHICH
 ;BANKS CAN BE PROTECTED;
 ;SO
 ;THIS ROUTINE ATTEMPS TO CREATE A DOUBLE BIT ERROR IN ADDRESS 0 & 2
 ;OF EVERY ECC BANK. ECC HARDWARE WILL PREVENT THIS FROM HAPPENING
 ;IN PROTECTED BANKS WHICH SHOULD ALWAYS INCLUDE BANK ZERO - WHERE
 ;THE PROGRAM IS.

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

CACHOFF ;TURN CACHE OFF

MOV #-1,OLDCSR
 FOR BANK := #0 TO LASTBANK

CLR BANK

B6:;;;;;;
 ;SET UP VIRT ADDR POINTER

MOV #FIRST,R1
 CALL EXBANK
 MOV BANKINDEX,R0
 IF ACFLAG IS TRUE

TST ACFLAG
 BEQ L61

IF MKFLAG IS TRUE

TST MKFLAG
 BEQ L62

IF SKIPMK IS FALSE

TST SKIPMK
 BNE L63

MOV #2,R3
 MOV CONFIG+1(R0),R2
 ASL R2
 BIC #C36,R2
 MOV R2,CSRNO
 IF CSRNO NE OLDCSR

;SET INDEX COUNTER

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:;;;;;;


```

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

```

6176 011750 005004          IMPTEST:CLR      R4
6177 011752          MAP BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      011752 010346          MOV R3,-(SP)
      011754 013703 002102    MOV BANK,R3
      011760 004737 035716    CALL MAPPER
      .DSABL CRF
      MOV (SP)+,R3
6178 011764 012603          CLR R5
6179 011766 005005          MOV #BIT13,CSR
6180 011770 012737 020000 002150  TESTAREA          ;ENTER TEST MODE
6180 011776          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
      011776 053737 002552 177776  .DSABL CRF
      PUSH (R1)          ;SAVE TEST LOCATION
      MOV (R1),-(SP)
6181 012004          ADD R3,R1          ;INDEX TO NEXT LOCATION
      012004 011146          PUSH (R1)          ;SAVE TEST LOCATION
6182 012006 060301          MOV (R1),-(SP)
6183 012010 011146          CHK1DIS          ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
      012010 104505          MOV R4,(R1)          ;WRITE CHECKBITS (ALL ZEROS)
6184 012012 104505          SUB R3,R1
6185 012014 010411          MOV R4,(R1)
6186 012016 160301          CLR1CSR          ;CLEAR CSR
6187 012020 010411          TST (R1)          ;READ CHECKBITS INTO REAL CSR
6188 012022 104503          WAS1DBE          ;WAS THERE A DOUBLE BIT ERROR
6189 012024 005711
6190 012026 104501
6191
6192          ;THIS MAKES SURE THAT SBE'S DON'T LOOK LIKE PROTECTED AREAS
6193
6194 012030          ON.NOERROR ;1
      012030 103435          BCS L71
6195 012032 012737 020000 002150    MOV #BIT13,CSR
6196 012040 104505          CHK1DIS          ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6197 012042 013711 002614          MOV ONES,(R1)
6198 012046 104503          CLR1CSR          ;CLEAR CSR
6199 012050 005711          TST (R1)
6200 012052 104501          WAS1DBE          ;WAS THERE A DOUBLE BIT ERROR
6201 012054          ON.NOERROR ;2
      012054 103423          BCS L72
6202 012056 104513          CBREG          ;ENABLE CHECK/SYNDROME BIT REGISTER
6203 012060 012737 023140 002150    MOV #23140,CSR          ;WRITE DBE'S IN CSR
6204 012066 104505          CHK1DIS          ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6205 012070 010411          MOV R4,(R1)
6206 012072 104503          CLR1CSR          ;CLEAR CSR
6207 012074 005711          TST (R1)
6208 012076 104501          WAS1DBE          ;WAS THERE A DOUBLE BIT ERROR
6209 012100          ON.NOERROR ;3
      012100 103411          BCS L73
6210 012102 104513          CBREG          ;ENABLE CHECK/SYNDROME BIT REGISTER
6211 012104 012737 023604 002150    MOV #23604,CSR          ;WRITE DBE'S IN CSR
6212 012112 104505          CHK1DIS          ;DISABLE ECC & WRITE CHECKBITS FOR 1 CSR
6213 012114 010411          MOV R4,(R1)
6214 012116 104503          CLR1CSR          ;CLEAR CSR
6215 012120 005711          TST (R1)
6216 012122 104501          WAS1DBE          ;WAS THERE A DOUBLE BIT ERROR
6217 012124          END ;OF ON.NOERROR ;3
6218 012124          END ;OF ON.NOERROR ;2
      012124          L73:::
      012124          L72:::

```

```

6219 012124          END ;OF ON.NOERROR ;1
      012124
6220 012124          ON.ERROR
      012124 103001
6221 012126 005205  INC R5
      012130          END ;OF ON.ERROR
      012130
6223 012130 104471  ECC1DIS
6224 012132 010411  MOV R4,(R1)
6225 012134 060301  ADD R3,R1
6226 012136 010411  MOV R4,(R1)
6227 012140 104503  CLR1CSR
6228 012142 005705  TST R5
6229 012144 001405  BEQ 1$
6230 012146 050560 002664  BIS R5,CONFIG(R0)
6231 012152 105260 002666  INCB CONFIG+2(R0)
6232 012156 104036  ERROR +36
6233 012160          1$: POP (R1)
      012160 012611
6234 012162 160301  SUB R3,R1
6235 012164          POP (R1)
      012164 012611
6236 012166 104417  KERNEL
6237 012170 000207  RETURN
6238
6239 012172          SUBAAR: SET STOPOK
      012172 012737 177777 002420

```

L71:;;;;;;
BCC L74
;IDENTIFY AS BAD BANK

L74:;;;;;;
;DISABLE ERROR CORRECTION
;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

6242 012200

SUBTST <<LEGAL CONFIGURATION CHECK>>

;SUBTEST LEGAL CONFIGURATION CHECK

6243 012200 012700 000020

MOV #16.,R0

6244 012204 012701 002462

MOV #CSRINFO,R1

6245 012210 005021

1\$: CLR (R1)+

6246 012212 077002

SOB RO,1\$

6247 012214

FOR BANK := #0 TO LASTBANK

012214 005037 002102

CLR BANK

012220

B7:::~::~

6248 012220 004737 040052

CALL EXBANK

6249 012224 013700 002104

MOV BANKINDEX,R0

6250

6251 012230

IF ACFLAG IS TRUE

012230 005737 002116

TST ACFLAG

012234 001444

BEQ L75

6252 012236 116003 002665

MOVB CONFIG+1(R0),R3

6253 012242 042703 177760

BIC #+C17,R3

6254 012246 006303

ASL R3

6255 012250 005263 002462

INC CSRINFO(R3)

6256 012254

IF MKFLAG IS TRUE

012254 005737 002120

TST MKFLAG

012260 001432

BEQ L76

6257

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

6258 012262

BEGIN LEGALCSR

012262

B10:::~::~

6259 012262

IF INTFLAG IS TRUE

012262 005737 002136

TST INTFLAG

012266 001423

BEQ L77

6260 012270 116003 002665

MOVB CONFIG+1(R0),R3

6261 012274 010304

MOV R3,R4

6262 012276 042703 177760

BIC #+C17,R3

6263 012302 072427 177774

ASH #-4,R4

6264 012306 042704 177760

BIC #+C17,R4

6265 012312

IF R3 EQ R4

012312 020304

CMP R3,R4

012314 001007

BNE L100

6266 012316 042760 014000 002666

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

6267 012324 042760 170000 002664

BIC #170000,CONFIG(R0)

6268 012332

LEAVE LEGALCSR

012332 000405

BR E10

6269 012334

END; OF IF R3

012334

L100:::~::~

6270 012334

ELSE

012334 000401

BR L101

6271 012336

LEAVE LEGALCSR

012336 000403

L77:::~::~

6272 012340

END; OF IF INTFLAG

012340

BR E10

6273 012340 012737 177777 002450

SET CONFGERROR

L101:::~::~

6274 012346

END LEGALCSR

012346

MOV #-1,CONFGERROR

6275 012346

END ;OF IF MKFLAG

012346

E10:::~::~

L76:::~::~


```

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

```

L75:::~::~

INC BANK
CMP BANK, LASTBANK
BLE B7

E7:::~::~

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

;CLEAR REGISTERS

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

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

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

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

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

```

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

```

6356 012766

SUBAAP: SUBTST <<PRINT CONFIGURATION DETAILS>>

;SUBTEST PRINT CONFIGURATION DETAILS

6357 012766

CLEAR LSIZE,MSIZE

CLR LSIZE
CLR MSIZE

012766 005037 002400

012772 005037 002402

6358 012776 013702 002556

6359 013002 006302

6360 013004 006302

6361 013006 005001

013010

6362 013010

IF CPUBIT SET.IN CONFIG(R1)

CLR R1
B11:::;;

013010 033761 002106 002664

013016 001411

BIT CPUBIT,CONFIG(R1)
BEQ L102

6363 013020

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

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

013020 032761 000400 002666

013026 001403

6364 013030

LET MSIZE := MSIZE + #1

INC MSIZE

6365 013034 005237 002402

013034 000402

ELSE

BR L104

013036

6366 013036

LET LSIZE := LSIZE + #1

L103:::;;

013036 005237 002400

6367 013042

END;IF BIT8

INC LSIZE

013042

6368 013042

END; OF IF CPUBIT

L104:::;;

013042

6369 013042

END ;OF FOR ALL BANKS IN TABLE

L102:::;;

013042 062701 000004

013046 020102

013050 003757

013052

ADD #4,R1
CMP R1,R2
BLE B11

E11:::;;

6370

6371 013052 005037 002452

6372 013056 005001

013060

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

CLR R1
B12:::;;

6373 013060 006361 002374

6374 013064 006361 002374

6375 013070 006361 002374

6376 013074 006361 002374

6377 013100 066137 002374 002452

6378 013106 062701 000002

013112 020127 000010

013116 003760

013120

6379 013120

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

E12:::;;

013120 005001

013122

6380 013122 033761 002106 002664

013130 001402

6381 013132

IF CPUBIT SET.IN CONFIG(R1)

CLR R1
B13:::;;

LET UNITOP := UNITOP + #1

BIT CPUBIT,CONFIG(R1)
BEQ L105

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

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

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

;UNITOP := UNITOP * 16.

; SUBTRACT 4K FOR THE I/O PAGE

::SAVE LSIZE FOR TYPEOUT
::GO TYPE--DECIMAL ASCII WITH SIGN


```

6401 013266 162737 000004 002402      SUB    #4,MSIZE      ; SUBTRACT 4K FOR THE I/O PAGE
6402 013274      TYPDEC MSIZE
      013274 013746 002402      13$:  MOV    MSIZE,-(SP)   ;;SAVE MSIZE FOR TYPEOUT
      013300 104405      TYPDS  MSIZE,-(SP)   ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
6403 013302      TYPE  MSG113
      013302 104401 071057      TYPEIT ,MSG113
      .DSABL CRF
6404 013306 022737 004000 002452 5$:  CMP    #2048.,I
6405 013314 001003      BNE    6$
6406 013316 162737 000004 002452      SUB    #4,I          ; SUBTRACT 4K FOR THE I/O PAGE
6407 013324      TYPDEC I
      013324 013746 002452      6$:  MOV    I,-(SP) ;;SAVE I FOR TYPEOUT
      013330 104405      TYPDS  I,-(SP) ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
6408 013332      TYPE  MSG070
      013332 104401 070127      TYPEIT ,MSG070
      .DSABL CRF
6409 013336      IF #SW6 OFF.IN @SWR
      013336 032777 000100 167272      BIT #SW6,@SWR
      013344 001002      BNE L107
6410 013346 004737 032722      CALL PCONFIG
6411 013352      END; OF IF #SW6
      013352

```

L107:;;;;;

6414 013352

SUBTST <<CHECK APT SIZING>>

;*SUBTEST CHECK APT SIZING

IF APTFLAG IS TRUE AND APTSIZE IS TRUE

6415 013352 005737 002352

TST APTFLAG
BEQ L110
TST APTSIZE
BEQ L110

013356 001475
013360 005737 002444
013364 001472

6416 013366 005037 002434

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

6417 013372 012700 056676

6418 013376 005002
013400

CLR R2
B14:::~::~

6419 013400 105760 000001

IFB 1(R0) NE #0

TSTB 1(R0)
BEQ L111

013404 001441

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

6420 013406 111001
6421 013410 042701 177400

TST 2(R0)
BGE L112

6422 013414 005760 000002
013420 002002

SEC
ELSE

BR L113
L112:::~::~

6423 013422 000261

6424 013424 000401
013426

CLC
END ;OF IF 2(R0)

L113:::~::~

6425 013426 000241

6426 013430 006101

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

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

6427 013430 005201
6428 013432 006301
6429 013434 006301

CMPB 1(R0),#3
BNE L114

6430 013436 006301
6431 013440 006301

6432 013442 006301
6433 013444 163701 002434

6434 013450 010137 002434

6435 013454 126027 000001 000003
013462 001002

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

L114:::~::~

6436 013464 060137 002422

6437 013470 126027 000001 000004

IFB 1(R0) EQ #4

CMPB 1(R0),#4
BNE L115

6438 013470 001002
6439 013500 060137 002424

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

L115:::~::~

6440 013504 062700 000004

6441 013504 000004
6442 013510

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

L111:::~::~

6443 013510 005202
013512 020227 000004

END ;OF FOR R2

INC R2
CMP R2,#4
BLE B14

013516 003730
013520

E14:::~::~

```

6444 013520
      013520 023737 002422 002400
      013526 001010
      013530 023737 002424 002402
      013536 001004
      013540 023737 002424 002404
      013546 001401
      013550
6445 013550 104046
6446 013552
      013552
6447 013552
      013552

```

IF APTPAR NE LSIZE OR APTECC NE MSIZE OR APTECC NE PSIZE

```

CMP APTPAR,LSIZE
BNE L116
CMP APTECC,MSIZE
BNE L116
CMP APTECC,PSIZE
BEQ L117

```

L116:::~::~

```

      ERROR      +46
      END ;OF IF APTPAR

```

L117:::~::~

END ;OF IF APTFLAG

L110:::~::~

```

6449 013552
      013552 000004
6450 013554 005037 002222
6451 013560 017700 167052
6452 013564 042700 177761
6453 013570 004770 013600
6454 013574 000137 013620
6455 013600 014262
6456 013602 014370
6457 013604 014476
6458 013606 014626
6459 013610 014756
6460 013612 015106
6461 013614 015260
6462 013616 015410
6463
6464 013620 004737 014162
6465
6466 013624
      013624 000004
6467
6468
6469 013626
      013626 005737 002002
      013632 001015
6470 013634
      013634 012737 177777 002612
      013642 012737 177777 002110
6471 013650 004737 021610
6472 013654
      013654 012737 177777 002612
6473 013662 005037 002110
6474 013666
      013666
6475 013666 004737 014162
6479
6480 013672
      013672 004737 022200

```

```

LOOP:  NEWTST  <<DIAGNOSTIC MODE DISPATCH ROUTINE>>
;*****
;*TEST 5      DIAGNOSTIC MODE DISPATCH ROUTINE
;*****
TST5:  SCOPE
      CLR     CONTFLAG
      MOV     @SWR,R0      ;GET SWITCHES
      BIC     #+C16,R0     ;MASK TO ONLY MODE BITS
      CALL    @DISPTBL(R0) ;DISPATCH TO ROUTINE THROUGH NEXT TABLE
      JMP     MEMDONE      ;GO TO NEXT TEST
DISPTBL:BAFPF  ;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
      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 L120
          MOV  #-1,HEADER
          MOV  #-1,MUT
          CALL MT0027
          SET  HEADER
          MOV  #-1,HEADER
          CLR  MUT
          END ;OF IF SELONLY
          CALL DOBACK      ;RESTORE BACKROUND PATTERN
          L120:::;
FLUSH: SUBTST  <<FLUSH OUT DBE'S>>
;*****
;*SUBTEST    FLUSH OUT DBE'S
;*****
      CALL    MT0030

```



```

6484
6485
6486
6487
6488
6489
6490
6491 013676 005037 002442
6492 013702 012700 002666
6493 013706 042710 020000
6494 013712 062700 000004
6495 013716 020027 003620
6496 013722 003771
6497 013724 013737 002630 002016
6498 013732 005237 056654
6499 013736 042737 100000 056654
6500 013744
    013744 104401 070251
6501 013750
    013750 032777 004000 166660
    013756 001007
    013760 005737 002346
    013764 001004
    013766 023727 056654 000001
    013774 001004
    013776
6502 013776
    013776 104401 067273
6503 014002 005037 002346
6504 014006
    014006
6505 014006
    014006 013746 056654
    014012 104405
6506 014014 013700 000042
6507 014020 001456
6508 014022 022700 002000
6509 014026 001453
6510
6511 014030
    014030 010046
6512 014032 004737 040636
6513 014036
    014036 012600
6514 014040 000005
6515 014042 004710
6516 014044 000240
6517 014046 000240
6518 014050 000240
6519 014052
6520
6521
6522
6523

```

```

.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
$EOP: CLR FSINFLAG
      MOV #CONFIG+2,R0 ;MOVE 2ND WORD OF CONFIG TO R0
1$:   BIC #BIT13,(R0) ;CLEAR BACKGROUND VALID BIT
      ADD #4,R0 ;INCREMENT TO NEXT BANK
      CMP R0,#3620 ;DONE?
      BLE 1$ ;NO - BRANCH
      MOV $ERTTL,LASTERROR
      INC $PASS ;;INCREMENT THE PASS NUMBER
      BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
      TYPE MSG077 ;;TYPE "END PASS #"
      TYPEIT ,MSG077
      .DSABL CRF
      IF #SW11 SET.IN @SWR OR QVFLAG IS TRUE OR $PASS EQ #1
                                          BIT #SW11,@SWR
                                          BNE L121
                                          TST QVFLAG
                                          BNE L121
                                          CMP $PASS,#1
                                          BNE L122
                                          L121:;;;;;
      TYPE MSG035 ;QV
      TYPEIT ,MSG035
      .DSABL CRF
      CLR QVFLAG
      END ;OF IF SW11
                                          L122:;;;;;
      TYPDEC $PASS
      MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
      TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
      .DSABL CRF
      MOV 42,R0 ;;GET MONITOR ADDRESS
      BEQ $DOAGAIN ;;BRANCH IF NO MONITOR
$ZAP42: CMP #STACK,R0 ;ARE WE UNDER RT11
      BEQ $DOAGAIN ;YES - BRANCH
      ;WE ARE UNDER (HEAVEN HELP US) XXDP!
      PUSH R0
                                          MOV R0,-(SP)
      CALL SHUTUP
      POP R0
                                          MOV (SP)+,R0
$ENDAD: RESET ;;CLEAR THE WORLD
      CALL (R0) ;;GO TO MONITOR
      NOP ;;SAVE ROOM
      NOP ;;FOR
      NOP ;;ACT11
$DOAGN: ;UNDO SHUTUP STUFF
      ; RESTORE STACK
      ; ENERGIZE Q-BUS MAP & 22 BIT ADDRESSING
      ; ENERGIZE MEMORY MANAGEMENT
      ; PUT LOADERS BACK HOME

```

```

6524 014052 013706 002574      MOV      KSTACK,SP
6525 014056 005737 002454      TST      NO22BIT          ;IS THIS AN 11/83,11/23-B OR 11/23?
6526 014062 001003              BNE      1$
6527 014064 052737 000060 172516  BIS      #BIT5:BIT4,MMR3
6528 014072 104420              1$: ENERGIZE          ;TURN ON MEMORY MANAGEMENT
6529 014074 013700 002576      MOV      LOADHOME,RO     ;DESTINATION BANK
6530 014100 012701 000001      MOV      #1,R1          ;SOURCE BANK
6531 014104 004737 037442      CALL    BANKMOV
6532 014110              IF APTFLAG IS TRUE
        014110 005737 002352
        014114 001420
6533 014116              IF $USWR EQ $PASS
        014116 023737 056672 056654
        014124 001014
6534 014126 012701 000050      APTHANG: MOV #50,R1
6535 014132 077001              2$: SOB RO,2$
6536 014134 062737 000001 056656  ADD #1,$DEVCT
6537 014142 005537 056660      ADC $UNIT
6538 014146 077107              SOB R1,2$
6539 014150 005237 056654      INC $PASS
6540 014154 000764              BR APTHANG
6541 014156              END ;OF IF $USWR
        014156
6542 014156              END ;OF IF APTFLAG
        014156
6543 014156 000137 013552      $DOAGAIN: JMP LOOP          ;RETURN

```

TST APTFLAG
BEQ L123
CMP \$USWR,\$PASS
BNE L124
L124:;;;;;
L123:;;;;;

6546 014162

DOBACK: SUBTST <<WRITE BACKGROUND PATTERNS>>

;*SUBTEST WRITE BACKGROUND PATTERNS

6547 014162 005037 002112

CLR PATTERN
FOR BANK := #0 TO LASTBANK

6548 014166 005037 002102

014172

6549 014172 004737 040052

6550 014176 005737 002116

014202 001420

014204 005737 002124

014210 001015

6551 014212

014212 012737 177777 002612

014220 012737 177777 002110

6552 014226 004737 017104

6553 014232 005037 002110

6554 014236

014236 012737 177777 002612

6555 014244

014244

6556 014244

014244 005237 002102

014250 023737 002102 002556

014256 003745

014260

6557 014260 000207

CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE

SET HEADER,MUT

CALL MKTEST
CLR MUT
SET HEADER

END ;OF IF ACFLAG

END ;OF FOR BANK

RETURN

CLR BANK
B15:::~::~

TST ACFLAG
BEQ L125
TST RRFLAG
BNE L125

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

;CALL MJTEST WOULD ALSO WORK

MOV #-1,HEADER

L125:::~::~

INC BANK
CMP BANK, LASTBANK
BLE B15

E15:::~::~

```

6560
6561
6562 014262

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

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

```

```

.SBTTL MTEST MODES
BAFPAF: SUBTST <<BANKS FORWARD,PATTERNS FORWARD **RECURSIVE*>>
;*****
;*SUBTEST BANKS FORWARD,PATTERNS FORWARD **RECURSIVE**
;*****
CLR BANK ;SET BANK TO 0
;START OF BANK LOOP
1$: CALL EXBANK ;EXAMINE BANK
TST ACFLAG ;CAN WE ACCESS THIS BANK?
BEQ 4$ ;NO - GO TO BANK LOOP TERMINATION
TST RRFLAG ;RELOCATION REQUIRED?
BNE 4$ ;YES - GO TO BANK LOOP TERMINATION
CLR PATTERN ;SET PATTERN TO 0
;START OF PATTERN LOOP
2$: CALL MTEST ;GO TEST CORRECT MEMORY
;TERMINATION OF PATTERN LOOP
CALL INCPAT ;GO SEE IF THIS IS THE LAST PATTERN
BNE 2$ ;NO - LOOP ON THIS PATTERN
;TERMINATION OF BANK LOOP
4$: CLR CONTFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 1$ ;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 L126
RTS PC
L126:::::

```


6592 014370

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

6593 014370 005037 002102

CLR BANK ;SET BANK TO 0

6594

;START OF BANK LOOP

6595 014374 004737 040052

1\$: CALL EXBANK ;EXAMINE BANK

6596 014400 005737 002116

TST ACFLAG ;CAN WE ACCESS THIS BANK?

6597 014404 001412

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

6598 014406 005737 002124

TST RRFLAG ;RELOCATION REQUIRED?

6599 014412 001007

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

6600 014414 004737 040510

CALL SETPAT ;SET HIGH PATTERN FOR CORRECT MEMORY

6601

;START OF PATTERN LOOP

6602 014420 004737 015562

2\$: CALL MTEST ;GO TEST CORRECT MEMORY

6603

;TERMINATION OF PATTERN LOOP

6604 014424 005337 002112

DEC PATTERN ;IS THIS THE LAST PATTERN?

6605 014430 100373

BPL 2\$;NO - LOOP ON THIS PATTERN

6606

;TERMINATION OF BANK LOOP

6607 014432 005037 002222

4\$: CLR CONTFLAG

6608 014436 004737 040520

CALL INCBNK ;NEXT HIGHER BANK

6609 014442 002354

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

6610

;END OF LOOPS

6611 014444 005737 002126

TST RLFLAG ;HAVE WE BEEN RELOCATED?

6612 014450 001401

BEQ 5\$;NO - SKIP

6613 014452 000207

RETURN ;YES - RETURN

6614 014454 004737 036402

5\$: CALL RELOCATE ;MOVE & MAP PROGRAM

6615 014460

ON.ERROR THEN \$RETURN

014460 103001

014462 000207

014464

6616

;**NOTE** RECURSIVE CALL

6617 014464 004737 014370

CALL BAFPAR ;CALL SELF

6618 014470 004737 037214

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

6619 014474 000207

RETURN

BCC L127
RTS PC

L127:::~::~

6622 014476

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

6623 014476 005037 002102
6624
6625 014502 004737 040052
6626 014506 005737 002116
6627 014512 001415
6628 014514 005737 002130
6629 014520 001412
6630 014522 005737 002124
6631 014526 001007
6632 014530 005037 002112
6633
6634 014534 004737 015562
6635
6636 014540 004737 040474
6637 014544 001373
6638
6639 014546 005037 002222
6640 014552 004737 040520
6641 014556 002351
6642
6643 014560 005137 002600
6644 014564 001003
6645
6646 014566 004737 014476
6647 014572 000207
6648 014574 005737 002126
6649 014600 001401
6650 014602 000207
6651 014604 004737 036402
6652 014610
014610 103001
014612 000207
014614
6653
6654 014614 004737 014476
6655 014620 004737 037214
6656 014624 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 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 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 L130
RTS PC
L130:;;;;;

```

6659 014626      BAWPAR: SUBTST <<BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**>>
;*****
;*SUBTEST      BANKS WORST FIRST,PATTERNS REVERSE **RECURSIVE**
;*****
6660 014626  005037  002102      CLR      BANK      ;SET BANK TO 0
6661          ;START OF BANK LOOP
6662 014632  004737  040052      1$: CALL      EXBANK      ;EXAMINE BANK
6663 014636  005737  002116      TST      ACFLAG      ;CAN WE ACCESS THIS BANK?
6664 014642  001415          BEQ      4$           ;NO - GO TO BANK LOOP TERMINATION
6665 014644  005737  002130      TST      BMFLAG      ;IS THIS BAD MEMORY (WORST FIRST)
6666 014650  001412          BEQ      4$           ;NO - GO TO BANK LOOP TERMINATION
6667 014652  005737  002124      TST      RRFLAG      ;RELOCATION REQUIRED?
6668 014656  001007          BNE      4$           ;YES - GO TO BANK LOOP TERMINATION
6669 014660  004737  040510      CALL     SETPAT      ;SET HIGH PATTERN FOR CORRECT MEMORY
6670          ;START OF PATTERN LOOP
6671 014664  004737  015562      2$: CALL     MTEST      ;GO TEST CORRECT MEMORY
6672          ;TERMINATION OF PATTERN LOOP
6673 014670  005337  002112      DEC     PATTERN      ;IS THIS THE LAST PATTERN?
6674 014674  100373          BPL     2$           ;NO - LOOP ON THIS PATTERN
6675          ;TERMINATION OF BANK LOOP
6676 014676  005037  002222      4$: CLR     CONTFLAG
6677 014702  004737  040520      CALL     INCBNK      ;NEXT HIGHER BANK
6678 014706  002351          BGE     1$           ;IF NOT DONE - LOOP ON THIS BANK
6679          ;END OF LOOPS
6680 014710  005137  002600      COM     WORST        ;IS THIS AN EVEN NUMBERED PASS?
6681 014714  001003          BNE     5$           ;YES - SKIP
6682          ;**NOTE** RECURSIVE CALL
6683 014716  004737  014626      CALL     BAWPAR      ;CALL SELF
6684 014722  000207          RETURN
6685 014724  005737  002126      5$: TST     RLFLAG      ;HAVE WE BEEN RELOCATED?
6686 014730  001401          BEQ     6$           ;NC - SKIP
6687 014732  000207          RETURN      ;YES - RETURN
6688 014734  004737  036402      6$: CALL     RELOCATE    ;MOVE & MAP PROGRAM
6689 014740          ON.ERROR THEN $RETURN
        014740  103001
        014742  000207
        014744
;*****
;**NOTE** RECURSIVE CALL
6690          ;**NOTE** RECURSIVE CALL
6691 014744  004737  014626      CALL     BAWPAR      ;CALL SELF
6692 014750  004737  037214      CALL     UNRELOCATE  ;UNMOVE & UNMAP PROGRAM
6693 014754  000207          RETURN

```

BCC L131
RTS PC

L131:::~::~

6696 014756

6697 014756 005037 002112
6698
6699 014762 005037 002102
6700
6701 014766 004737 040052
6702 014772 004737 040456
6703 014776 001010
6704 015000 005737 002116
6705 015004 001405
6706 015006 005737 002124
6707 015012 001002
6708 015014 004737 015562
6709
6710 015020 005037 002222
6711 015024 004737 040520
6712 015030 002356
6713
6714 015032 004737 040474
6715 015036 001351
6716
6717 015040 005137 002134
6718
6719 015044 001403
6720
6721 015046 004737 014756
6722 015052 000207
6723 015054 005737 002126
6724 015060 001401
6725 015062 000207
6726 015064 004737 036402
6727 015070
015070 103001
015072 000207
015074

6728
6729 015074 004737 014756
6730 015100 004737 037214
6731 015104 000207

```

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

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

```

BCC L132
RTS PC
L132:::~::~


```

6734 015106
6735 015106 005037 002112
6736
6737 015112 005037 002102
6738
6739 015116 004737 040052
6740 015122 004737 040456
6741 015126 001013
6742 015130 005737 002116
6743 015134 001410
6744 015136 005737 002130
6745 015142 001405
6746 015144 005737 002124
6747 015150 001002
6748 015152 004737 015562
6749
6750 015156 005037 002222
6751 015162 004737 040520
6752 015166 002353
6753
6754 015170 004737 040474
6755 015174 001346
6756
6757 015176 005137 002134
6758
6759 015202 001403
6760
6761 015204 004737 015106
6762 015210 000207
6763 015212 005137 002600
6764 015216 001003
6765
6766 015220 004737 015106
6767 015224 000207
6768 015226 005737 002126
6769 015232 001401
6770 015234 000207
6771 015236 004737 036402
6772 015242
      015242 103001
      015244 000207
      015246
6773
6774 015246 004737 015106
6775 015252 004737 037214
6776 015256 000207

```

```

PAFBAW: SUBTST <<PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**>>
;*****
;*SUBTEST PATTERNS FORWARD,BANKS WORST FIRST **RECURSIVE**
;*****
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$: CLR CONTFLAG
CALL INCBNK ;NEXT HIGHER BANK
BGE 2$ ;IF NOT DONE - LOOP ON THIS BANK
;TERMINATION OF PATTERN LOOP
CALL INCRPT ;NEXT HIGHER PATTERN
BNE 1$ ;OK - LOOP; ELSE CONTINUE
;END OF LOOPS
COM TMFLAG ;COMPLEMENT TYPE OF MEMORY
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

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

```

```

BCC L133
RTS PC
L133:::

```

6779 015260

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

6780 015260 004737 040510

CALL HIPAT ;SET HIGHEST PATTERNS

6781

;START OF PATTERN LOOP

6782 015264 005037 002102

1\$: CLR BANK ;SET BANK TO 0

6783

;START OF BANK LOOP

6784 015270 004737 040052

2\$: CALL EXBANK ;EXAMINE BANK

6785 015274 004737 040456

CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?

6786 015300 001010

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

6787 015302 005737 002116

TST ACFLAG ;CAN WE ACCESS THIS BANK?

6788 015306 001405

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

6789 015310 005737 002124

TST RRFLAG ;RELOCATION REQUIRED?

6790 015314 001002

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

6791 015316 004737 015562

CALL MTEST ;GO TEST CORRECT MEMORY

6792

;TERMINATION OF BANK LOOP

6793 015322 005037 002222

4\$: CLR CONTFLAG

6794 015326 004737 040520

CALL INCBNK ;NEXT HIGHER BANK

6795 015332 002356

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

6796

;TERMINATION OF PATTERN LOOP

6797 015334 005337 002112

DEC PATTERN ;NEXT LOWER PATTERN

6798 015340 100351

BPL 1\$;OK - LOOP; ELSE CONTINUE

6799

;END OF LOOPS

6800 015342 005137 002134

COM TMFLAG ;COMPLEMENT TYPE OF MEMORY

6801

;IS THIS AN EVEN NUMBER PASS?

6802 015346 001403

BEQ 5\$;YES - SKIP

6803

;**NOTE** RECURSIVE CALL

6804 015350 004737 015260

CALL PARBAF ;CALL SELF

6805 015354 000207

RETURN

6806 015356 005737 002126

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

6807 015362 001401

BEQ 6\$;NO - SKIP

6808 015364 000207

RETURN ;YES - RETURN

6809 015366 004737 036402

6\$: CALL RELOCATE ;MOVE & MAP PROGRAM

6810 015372

ON.ERROR THEN \$RETURN

015372 103001

015374 000207

015376

BCC L134
RTS PC

L134:::~::~

6811

;**NOTE** RECURSIVE CALL

6812 015376 004737 015260

CALL PARBAF ;CALL SELF

6813 015402 004737 037214

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

6814 015406 000207

RETURN

6817 015410

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

6818 015410 004737 040510

CALL HIPAT ;SET HIGHEST PATTERN

6819

;START OF PATTERN LOOP

6820 015414 005037 002102

1\$: CLR BANK ;SET BANK TO 0

6821

;START OF BANK LOOP

6822 015420 004737 040052

2\$: CALL EXBANK ;EXAMINE BANK

6823 015424 004737 040456

CALL BANKOK ;CORRECT MEMORY FOR THIS BANK?

6824 015430 001013

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

6825 015432 005737 002116

TST ACFLAG ;CAN WE ACCESS THIS BANK?

6826 015436 001410

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

6827 015440 005737 002130

TST BMFLAG ;IS THIS BAD MEMORY (WORST FIRST)

6828 015444 001405

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

6829 015446 005737 002124

TST RRFLAG ;RELOCATION REQUIRED?

6830 015452 001002

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

6831 015454 004737 015562

CALL MTEST ;GO TEST CORRECT MEMORY

6832

;TERMINATION OF BANK LOOP

6833 015460 005037 002222

4\$: CLR CONTFLAG

6834 015464 004737 040520

CALL INCBNK ;NEXT HIGHER BANK

6835 015470 002353

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

6836

;TERMINATION OF PATTERN LOOP

6837 015472 005337 002112

DEC PATTERN ;NEXT LOWER PATTERN

6838 015476 100346

BPL 1\$;OK - LOOP; ELSE CONTINUE

6839

;END OF LOOPS

6840 015500 005137 002134

COM TMFLAG ;COMPLEMENT TYPE OF MEMORY

6841

;IS THIS AN EVEN NUMBER PASS?

6842 015504 001403

BEQ 5\$;YES - SKIP

6843

;**NOTE** RECURSIVE CALL

6844 015506 004737 015410

CALL PARBAW ;CALL SELF

6845 015512 000207

RETURN

6846 015514 005137 002600

5\$: COM WORST ;4TH PASS?

6847 015520 001003

BNE 6\$;YES - SKIP

6848

;**NOTE** RECURSIVE CALL

6849 015522 004737 015410

CALL PARBAW ;CALL SELF

6850 015526 000207

RETURN

6851 015530 005737 002126

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

6852 015534 001401

BEQ 7\$;NO - SKIP

6853 015536 000207

RETURN ;YES - RETURN

6854 015540 004737 036402

7\$: CALL RELOCATE ;MOVE & MAP PROGRAM

6855 015544

ON.ERROR THEN \$RETURN

015544 103001

015546 000207

015550

BCC L135
RTS PC

L135:::~::~

6856

;**NOTE** RECURSIVE CALL

6857 015550 004737 015410

CALL PARBAW ;CALL SELF

6858 015554 004737 037214

CALL UNRELOCATE ;UNMOVE & UNMAP PROGRAM

6859 015560 000207

RETURN


```

6862 015562          MTEST: SUBTST <<SUBR SETUP MEMORY TEST>>
;*****
;*SUBTEST          SUBR  SETUP MEMORY TEST
;*****
6863 015562          SET    HEADER          ;INITIALIZE HEADER MESSAGE TYPEOUT
015562 012737 177777 002612          MOV    #-1,HEADER
6864 015570          SET    MUT              ;INDICATE THERE IS A MEMORY UNDER TEST
015570 012737 177777 002110          MOV    #-1,MUT
6865 015576          CLR    PASFLG
6866 015602          TST    MKFLAG          ;ECC?
6867 015606          BEQ    MT1              ;NO - SKIP
001413
6868 015610          BEGIN  HOLDLOOP
015610
6869 015610          IF CONTFLAG IS TRUE THEN LEAVE HOLDLOOP
015610 005737 002222          TST CONTFLAG
015614 001005          BNE  E16
6870 015616          IF SKIPMK IS FALSE
015616 005737 002342          TST SKIPMK
015622 001002          BNE  L136
6871 015624          CALL    MKCONTROL
004737 015656          END; OF IF SKIPMK
6872 015630
015630
6873 015630          END    HOLDLOOP
015630
6874 015630          CALL    MKTEST          ;YES - DO ECC TESTS
004737 017104          BR    MT2
6875 015634          000402
6876 015636          004737 017324          MT1: CALL    MJTEST
6877 015642          005037 002110          MT2: CLR    MUT
6878 015646          SET    HEADER          ;DO PARITY TESTS
015646 012737 177777 002612          ;NOW - NO MEMORY UNDER TEST
015654 000207          RETURN          ;ALLOW HEADERS NORMAL
MOV    #-1,HEADER

```


6882 015656

MKCONTROL:SUBTST <<SUBR TEST ECC CSR LOGIC DISPATCH>>
;*****
;*SUBTEST SUBR TEST ECC CSR LOGIC DISPATCH
;*****

6883
6884
6885
6886 015656 005737 002002
015656 001401
015662 000207
015664
015666

;THE NEXT TWO MODULES SOLVE THE PROBLEM OF
;HOW TO RUN THE CSR TESTS ON EACH ECC MEMORY
;
IF SELONLY IS TRUE THEN \$RETURN

TST SELONLY
BEQ L137
RTS PC
L137:::~::~

6887 015666 005737 002536
015666 001401
015672 000207
015674
015676

IF INHECC IS TRUE THEN \$RETURN

TST INHECC
BEQ L140
RTS PC
L140:::~::~

6888 015676 013746 002102
015676 010046
015702 010146
015704 010246
015706 010346

PUSH BANK,R0,R1,R2,R3

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

6889 015712 012737 060000 002232
6890 015720 012737 157776 002234

MOV #FIRST,CSRFBANK
MOV #LAST,CSRLBANK

;SET FIRST TEST ADDRESS TO FIRST ADDR.

6891 015726 005037 002236
6892 015732 005037 002240
6893 015736 005037 002332
6894 015742 013700 002104
6895 015746 016001 002664

CLR CSRINT
CLR SPLTCSR
CLR CSRLOOP
MOV BANKINDEX,R0
MOV CONFIG(R0),R1

; AND ZERO THE LOOP COUNTER
;GET THE BANK INDEX
;GET CSR NUMBER

6896 015752 000301
6897 015754 042701 177760
6898 015760 006301
6899 015762 010137 002526
6900 015766 005737 002136

SWAB R1
BIC #C17,R1
ASL R1
MOV R1,CSRHOLD
TST INTFLAG

;STORE IN THE LOW BYTE
;IS THIS BANK INTERLEAVED?
;BRANCH IF NOT INTERLEAVED

6901 015772 001421
6902 015774 005237 002240
6903 016000 012737 120000 002234

BEQ 1\$
INC SPLTCSR
MOV #120000,CSRLBANK

;WE MUST LOOP TWICE FOR AN INTERLEAVED BANK

6904 016006 005237 002332
6905 016012 005237 002236
6906 016016 016001 002664
6907 016022 072127 177775
6908 016026 042701 160777

INC CSRLOOP
INC CSRINT
MOV CONFIG(R0),R1
ASH #-3,R1
BIC #C17000,R1

;GET THE INTERLEAVE CSR NUMBER

6909 016032 050137 002526
6910 016036 005003
6911 016040 116337 002526 002152
6912 016046 042737 177741 002152

BIS R1,CSRHOLD
CLR R3
1\$: MKLOOP: MOVB CSRHOLD(R3),CSRNO

;STORE IT IN CSRHOLD'S UPPER BYTE

6913 016054
016054 005037 016476
016060

BIC #C36,CSRNO
FOR MKCNT := #0 TO CSRINT

;CLEAR ANY UNNECESSARY BITS

6914 016060
016060 013737 002232 002226
016066

FOR CSRFIRST := CSRFBANK TO CSRLBANK BY #4000

CLR MKCNT
B17:::~::~

6915 016066
016066 010346
016070 013703 002102
016074 004737 035716

MAP BANK
MOV BANK,R3
CALL MAPPER

MOV CSRFBANK,CSRFIRST
B20:::~::~
;MAP TEST SPACE TO BANK
MOV R3,-(SP)

```

        .DSABL CRF
6916 016100 012603
6917 016102 104511
        INVALIDATE
        BEGIN CSRSTUFF
        MOV (SP)+,R3
        ;INVALIDATE BACKGROUND PATTERN
        B21:;
6918 016104 005037 002334
6919 016110
        CLR SUCCESS
        IF ACFLAG IS TRUE AND RRFLAG IS FALSE
        TST ACFLAG
        BEQ L141
        TST RRFLAG
        BNE L141
        016114 005737 002116
        016116 001503
        016122 005737 002124
        016122 001100
6920 016124 013737 002226 002230
        MOV CSRFIRST,CSRLAST
6921 016132 062737 004000 002230
        ADD #4000,CSRLAST
        FOR TESTADD := CSRFIRST TO CSRLAST BY #4
        MOV CSRFIRST,TESTADD
        B22:;
6923 016140 013737 002226 002412
        MOV TESTADD,TESTADD+2
        TST SPLTCSR
        BEQ 1$
        ADD #40000,TESTADD+2
        BR 2$
        016146 013737 002412 002414
        1$: ADD #2,TESTADD+2
        2$: CALL SBETEST
        ON.NOERROR
        6924 016154 005737 002240
        6925 016160 001404
        6926 016162 062737 040000 002414
        6927 016170 000403
        6928 016172 062737 000002 002414
        6929 016200 004737 016500
        6930 016204
        016204 103440
        6931 016206 104424
        6932 016210 005037 002076
        6933 016214
        CACHOFF
        CLR NOPAR
        FOR I := #0 TO #27
        ;TURN CACHE OFF
        ;INDICATE PARITY ACTION
        CLR I
        B23:;
        6934 016220
        SET HEADER
        MOV #-1,HEADER
        016220 012737 177777 002612
        6935 016226 005037 002264
        CLR PASFLG
        LET R0 := I
        6936 016232
        PUSH R3
        ;SAVE LOOP COUNTER
        MOV I,R0
        016232 013700 002452
        MOV R3,-(SP)
        ;SAVE VECTOR IN CSR OF +K
        6937 016236
        MOV SP,CTLKVEC
        SUB #2,CTLKVEC
        CALL CSRCASE
        POP R3
        ;RESTORE LOOP COUNTER
        MOV (SP)+,R3
        016236 010346
        6938 016240 010637 002146
        6939 016244 162737 000002 002146
        6940 016252 004737 017004
        END ;OF FOR I
        6941 016256
        016256 012603
        6942 016260
        016260 005237 002452
        016264 023727 002452 000027
        016272 003752
        016274
        6943 016274 104423
        CACHON
        SET SUCCESS
        ;TURN CACHE ON
        6944 016276
        LEAVE CSRSTUFF
        MOV #-1,SUCCESS
        016276 012737 177777 002334
        6945 016304
        LEAVE CSRSTUFF
        BR E21
        016304 000407
        6946 016306
        END ;OF ON.NOERROR
        016306
        6947 016306
        END ;OF FOR TESTADD
        L142:;
        016306 062737 000004 002412
        016314 023737 002412 002230
        ADD #4,TESTADD
        CMP TESTADD,CSRLAST
    
```

```

016322 003711
016324
6948 016324
016324
6949 016324
016324
6950 016324
016324 005737 002334
016330 001012
6951 016332
016332 104401 067207
6952 016336
016336 013746 002102
016342 104403
016344 003
016345 000
6953 016346
016346 104401 067245
6954 016352 004737 050446
6955 016356
016356
6956 016356
016356 062737 004000 002226
016364 023737 002226 002234
016372 003635
016374
6957 016374 005237 002240
6958 016400
016400 005237 016476
016404 023737 016476 002236
016412 003622
016414
6959 016414 062737 000002 002232
6960 016422 012737 000001 002240
6961 016430 005203
6962 016432 020337 002332
6963 016436 003002
6964 016440 000137 016040
6965 016444 104472
6966 016446
016446 012737 177777 002222
6967 016454 005037 002240
6968 016460
016460 012603
016462 012602
016464 012601
016466 012600
016470 012637 002102
6969 016474 000207
6970 016476 000000

```

```

END ;OF IF
END CSRSTUFF
IF SUCCESS IS FALSE
TYPEIT TYPE MSGA34
.DSABL ,MSGA34
CRF
MOV TYPOCS BANK,<TYPES BANK NUMBER>,3
BANK,-(SP) ;;SAVE BANK FOR TYPEOUT
;;TYPES BANK NUMBER
TYPOS ;;GO TYPE--OCTAL ASCII
.BYTE 3 ;;TYPE 3 DIGIT(S)
.BYTE 0 ;;SUPPRESS LEADING ZEROS
.DSABL CRF
TYPEIT TYPE MSGB34
.DSABL ,MSGB34
CRF
CALL PERBNK
END ;OF IF SUCCESS
END; OF FOR CSRFIRST
INC SPLTCSR
END; OF FOR MKCNT
ADD #2,CSRFBANK
MOV #1,SPLTCSR
I R3
CMP R3,CSRLOOP
BGT 1$
JMP MKLOOP
ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
SET CONTFLAG
CLR SPLTCSR
POP R3,R2,R1,RO,BANK
RETURN
MKCNT: .WORD 0

```

```

BLE B22
E22:;;;;;
L141:;;;;;
E21:;;;;;
TST SUCCESS
BNE L143
L143:;;;;;
ADD #4000,CSRFIRST
CMP CSRFIRST,CSRLBANK
BLE B20
E20:;;;;;
INC MKCNT
CMP MKCNT,CSRINT
BLE B17
E17:;;;;;
MOV #-1,CONTFLAG
MOV (SP),R3
MOV (SP),R2
MOV (SP),R1
MOV (SP),RO
MOV (SP),BANK
;COUNTER FOR MKLOOP

```


6973 016500

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

6974
6975
6976
6977
6978
6979
6980
6981
6982
6983
6984
6985
6986
6987
6988
6989
6990
6991
6992
6993
6994
6995
6996
6997
6998 016500
016500 010046
016502 010146
016504 010446
6999 016506 013701 002412
7000 016512 013704 002414
7001 016516
016516 053737 002552 177776
7002 016524 104424
7003 016526 104471
7004 016530
016530 005011
016532 005014
7005 016534 005711
7006 016536 001107
7007 016540 005714
7008 016542 001105
7009
7010 016544 104503
7011 016546
016546 005011
016550 005014
7012 016552 005711
7013 016554 001100
7014 016556 005714
7015 016560 001076
7016
7017 016562 104510

;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 (R4)
BNE SBENT
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)


```

7018 016564          IF #BIT15!BIT4 SET.IN CSR
      016564 032737 100020 002150          BIT #BIT15!BIT4,CSR
      016572 001415          BEQ L144
7019 016574          SET SKPERR          ;DISABLE ERRGEN'S ERROR PRINTOUT
      016574 012737 177777 002066          MOV #-1,SKPERR
7020 016602          ERRGEN
7021 016604 013700 002460          MOV ERRADD,R0
7022 016610 072027 177774          ASH #-4,R0
7023 016614 042700 177600          BIC #+C177,R0
7024 016620          IF BANK EQ R0 THEN GOTO SBENT
      016620 023700 002102          CMP BANK,R0
      016624 001454          BEQ SBENT
7025 016626          END; OF IF #BIT15
      016626          L144:::~::~:
7026 016626 104471          ECC1DIS          ;DISABLE ECC ON 1 SELECTED CSR
7027 016630 005111          COM (R1)
7028 016632 005114          COM (R4)
7029 016634 023711 002614          CMP ONES,(R1)
7030 016640 001046          BNE SBENT
7031 016642 023714 002614          CMP ONES,(R4)
7032 016646 001043          BNE SBENT
7033
7034 016650 104503          CLR1CSR          ;CLEAR 1 SELECTED CSR
7035 016652 005011          CLR (R1)
7036 016654 012714 100000          MOV #BIT15,(R4)
7037 016660 005711          TST (R1)
7038 016662 001035          BNE SBENT
7039 016664 022714 100000          CMP #BIT15,(R4)
7040 016670 001032          BNE SBENT
7041
7042 016672 104510          TSTREAD          ;TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
7043 016674          IF #BIT15!BIT4 SET.IN CSR
      016674 032737 100020 002150          BIT #BIT15!BIT4,CSR
      016702 001415          BEQ L145
7044 016704          SET SKPERR          ;DISABLE ERRGEN'S ERROR PRINTOUT
      016704 012737 177777 002066          MOV #-1,SKPERR
7045 016712          ERRGEN
7046 016714 013700 002460          MOV ERRADD,R0
7047 016720 072027 177774          ASH #-4,R0
7048 016724 042700 177600          BIC #+C177,R0
7049 016730          IF BANK EQ R0 THEN GOTO SBENT
      016730 023700 002102          CMP BANK,R0
      016734 001410          BEQ SBENT
7050 016736          END; OF IF #BIT15
      016736          L145:::~::~:
7051
7052 016736 104417          KERNEL          ;ENTER KERNEL MODE
7053 016740 104473          ECC1INIT          ;INITIALIZE 1 SELECTED CSR
7054 016742 104423          CACHON          ;TURN CACHE ON
7055 016744          POP R4,R1,R0          ;POP R0,R1 & R4 FROM STACK
      016744 012604          MOV (SP)+,R4
      016746 012601          MOV (SP)+,R1
      016750 012600          MOV (SP)+,R0
7056 016752          $RETURN NOERROR
      016752 000241          CLC
      016754 000207          RTS PC
7057

```

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

7068 017004

CSRCASE:SUBTST <<CSR PATTERN CASE STATEMENT>>

;SUBTEST CSR PATTERN CASE STATEMENT

7069 017004

017004 010046
017006 006316
017010 004737 017074

MOV RO,-(SP)
ASL @SP
JSR PC,L146

7070

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

7071

7072

7073

7074

7075

7076

7077

7078

7079 017016 024024

7080 017020 020354

7081 017022 024104

7082 017024 023560

7083 017026 020456

7084 017030 023622

7085 017032 023660

7086 017034 023740

7087 017036 023774

7088 017040 024134

7089 017042 024164

7090 017044 020320

7091 017046 024220

7092 017050 024220

7093 017052 024220

7094 017054 024220

7095 017056 024220

7096 017060 024220

7097 017062 024220

7098 017064 024220

7099 017066 024220

7100 017070 024220

7101 017072 024220

7102 017074

017074

017074 062616

017076 013646

017100 004736

7103 017102 000207

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

END ;OF CASE RO

RETURN

L146:::~::~
ADD (SP)+,@SP
MOV @ (SP)+,-(SP)
JSR PC,@(SP)+


```

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

7107 017104      032777 000001 163524      BIT #SWO,@SWR
017104      001003      BNE L147
017112      005737 002350      TST ACTFLAG
017120      001402      BEQ L150
017122      L147:;;;;;

7108 017122      104470      ECCDIS      ;DISABLE ERROR CORRECTION
7109 017124      000401      ELSE
017124      000401      BR L151
017126      L150:;;;;;

7110 017126      104502      CLRCSR      ;CLEAR ALL CSR'S
7111 017130      END ;OF IF
017130      L151:;;;;;

7112 017130      012737 000002 002076      MOV #2,NOPAR      ;INDICATE PARITY ACTION
7113 017136      012737 000002 002326      MOV #2,PCBUMP      ;TRAPS ADD 2 TO PC
7114 017144      013700 002112      MOV PATTERN,RO      ;GET PATTERN NUMBER
7115 017150      006300      ASL RO      ;MAKE IT A WORD ADDRESS
7116 017152      IF MKPAT(RO) NE #MT0034 AND MKPAT(RO) NE #MT0999
017152      026027 017244 023346      CMP MKPAT(RO),#MT0034
017160      001405      BEQ L152
017162      026027 017244 024220      CMP MKPAT(RO),#MT0999
017170      001401      BEQ L152
7117 017172      104511      INVALIDATE      ;INVALIDATE BACKGROUND PATTERN ON "BANK"
7118 017174      END ;OF IF MKPAT(RO)
017174      L152:;;;;;

7119 017174      010637 002146      MOV SP,CTLKVEC      ;SAVE VECTOR IN CASE OF tK
7120 017200      162737 000002 002146      SUB #2,CTLKVEC
7121 017206      004770 017244      CALL @MKPAT(RO)      ;INDEX OFF TABLE
7122 017212      IF #SWO SET.IN @SWR OR ACTFLAG IS TRUE
017212      032777 000001 163416      BIT #SWO,@SWR
017220      001003      BNE L153
017222      005737 002350      TST ACTFLAG
017226      001402      BEQ L154
017230      L153:;;;;;

7123 017230      104506      ENASBE      ;TRAP ON SINGLE BIT ERRORS
7124 017232      ELSE
017232      000401      BR L155
017234      L154:;;;;;

7125 017234      104472      ECCINIT      ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
7126 017236      END ;OF IF #SWO
017236      L155:;;;;;

7127 017236      005037 002076      CLR NOPAR      ;INDICATE PARITY ACTION
7128 017242      000207      RETURN

7129
7130      ;WARNING IF YOU CHANGE THIS TABLE ALSO
7131      ;CHANGE "$DDW0" - "$DDW5" (THE PATTERN BIT MAP)
7132      ;PAT TIME DISCRPTION
7133 017244      MKPAT: ;NOTE MT0034 MUST BE FIRST & LAST
7134 017244      023346      MT0034 ;<1 SEC ;SOFT ERROR - BACKGROUND PATTERN TEST
7135 017246      020434      MT0017 ;<1 SEC ;HOLDING 1'S & 0'S TEST
7136 017250      020256      MT0007 ;<1 SEC ;ADDRESS BIT TEST
7137 017252      017544      MT0001 ;<1 SEC ;ADDRESS TEST
7138 017254      017640      MT0002 ;<1 SEC ;COMPLEMENT ADDRESS TEST

```


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

7161 017324

MJTEST: SUBTST <<SUBR PARITY TEST DISPATCH>>

;SUBTEST SUBR PARITY TEST DISPATCH

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

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

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

7168 017362 026027 017430 023346
017370 001405
017372 026027 017430 024220
017400 001401

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

7169 017402 104511
7170 017404 017404
7171 017404 010637 002146
7172 017410 162737 000002 002146
7173 017416 004770 017430
7174 017422 005037 002076
7175 017426 000207

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

7176
7177
7178
7179
7180

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

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

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

7208
7209
7210
7211 017510

.SBTTL PATTERNS

.SBTTL MEMORY TEST SETUP ROUTINES

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

;*SUBTEST MT0000 SETUP DATA PATTERN TEST

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

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

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

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

7236 017640

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

;*SUBTEST MT0002 SETUP COMPLEMENT ADDRESS TEST

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

7259 017754

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

;SUBTEST MT0003 SETUP 3 XOR 9 WORST CASE NOISE TEST

IF EQFLAG IS TRUE THEN \$RETURN

7260 017754 005737 002132
017754 001401
017760 000207
017762
017764

TST EQFLAG
BEQ L157
RTS PC

L157:;;;;;

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

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

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

;SUBTEST MT0004 SETUP ROTATING ZEROS TEST

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

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

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

;SUBTEST MT0005 SETUP ROTATING ONES TEST

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

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

7304 020222

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

;*****

;*SUBTEST MT0006 SETUP INITIAL DATA TEST

;*****

7305 020222 012737 000006 002300
7306 020230 012737 000004 002326
7307 020236 012701 002412
7308 020242 012737 025216 002262
7309 020250 004737 024446
7310 020254 000207
7311 020256

MOV #6,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV #TESTADD,R1
MOV #MTP006,SUPDOADD
CALL SUPD03 ;DO IT IN SUPERVISOR MODE
RETURN

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

;*****

;*SUBTEST MT0007 SETUP ADDRESS BIT TEST

;*****

7312 020256 012737 000007 002300
7313 020264 005002
7314 020266 004737 032630
7315 020272 012701 060000
7316 020276 012702 000001
7317 020302 050201
7318 020304 012737 025416 002262
7319 020312 004737 024446
7320 020316 000207
7321 020320

MOV #7,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR R2
CALL BACKGND ;OF ZEROS
MOV #FIRST,R1
MOV #1,R2
BIS R2,R1
MOV #MTP007,SUPDOADD
CALL SUPD03 ;DO IT IN SUPERVISOR MODE
RETURN

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

;*****

;*SUBTEST MT0010 SETUP BYTE ADDRESSING TEST

;*****

7322 020320 012737 000010 002300
7323 020326 012737 000004 002326
7324 020334 013704 002412
7325 020340 012737 025516 002262
7326 020346 004737 024446
7327 020352 000207

MOV #10,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV #4,PCBUMP ;TRAPS ADD 4 TO PC
MOV TESTADD,R4
MOV #MTP010,SUPDOADD
CALL SUPD03 ;DO IT IN SUPERVISOR MODE
RETURN

```

7330 020354          MT0014: SUBTST  <<MT0014      SETUP BASIC DOUBLE BIT ERROR TEST>>
;*****
;#SUBTEST          MT0014  SETUP BASIC DOUBLE BIT ERROR TEST
;*****
IF ACTFLAG IS TRUE OR APTFLAG IS TRUE

7331 020354          005737  002350          TST ACTFLAG
020354          001003          BNE L160
020360          005737  002352          TST APTFLAG
020366          001404          BEQ L161
020370          L160:::
7332 020370          IF $PASS NE #0 THEN $RETURN          TST $PASS
020370          005737  056654          BEQ L162
020374          001401          RTS PC
020376          000207          L162:::
020400          END; OF IF ACTFLAG          L161:::
7333 020400          MOV #14,REALPAT          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
020400          CALL MAPKERNAL          ;MAP KERNAL SPACE
7334 020400          012737  000014  002300          LET R1 := #100000          ;SETUP TEST ADDRESS
7335 020406          004737  037740          CALL GETCSR          ;GET CSR INFO FROM CONFIGURATION TABLE
7336 020412          012701  100000          CALL MTP014          ;DO BASIC DOUBLE BIT ERROR TEST
020412          004737  032674          CALL UNMAP          ;UNMAP KERNAL SPACE
7337 020416          004737  025624          RETURN
7338 020422          004737  040026
7339 020426          004737
7340 020432          000207

```

7343
7344 020434

```

MT0017: SUBTST <<MT0017      SETUP HOLDING 1'S & 0'S>>
;*****
;*SUBTEST      MT0017  SETUP HOLDING 1'S & 0'S
;*****
MOV      #17,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
MOV      #MTP017,SUPDOADD
CALL     SUPD03           ;DO IT IN SUPERVISOR MODE
RETURN

```

7345	020434	012737	000017	002300
7346	020442	012737	026050	002262
7347	020450	004737	024446	
7348	020454	000207		


```

7351 020456          MT0020: SUBTST  <<MT0020      SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR>>
;*****
;*SUBTEST          MT0020  SETUP SYNDROMES TO CSR ON SINGLE BIT ERROR
;*****
IF ACTFLAG IS TRUE OR APTFLAG IS TRUE

7352 020456          005737  002350          TST ACTFLAG
020462          001003          BNE L163
020464          005737  002352          TST APTFLAG
020470          001404          BEQ L164
020472          L163:;;;;;

7353 020472          IF $PASS NE #0 THEN $RETURN
020472          005737  056654          TST $PASS
020476          001401          BEQ L165
020500          000207          RTS PC
020502          L165:;;;;;

7354 020502          END; OF IF ACTFLAG
020502          L164:;;;;;

7355 020502          012737  000020  002300  MOV    #20,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7356 020510          004737  037740          CALL  MAPKERNAL        ;MAP KERNAL SPACE
7357 020514          012701  100000          LET R1 := #100000     ;SETUP TEST ADDRESS
020514          004737  032674          CALL  GETCSR           ;GET CSR INFO FROM CONFIGURATION TABLE
7358 020520          004737  026126          CALL  MTPO20          ;DO SYNDROMES TO CSR ON SINGLE ERROR TEST
7359 020524          004737  040026          CALL  UNMAP           ;UNMAP KERNAL SPACE
7360 020530          000207          RETURN
7361 020534

```

7363 020536

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

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

SET NOSCOPE

7364	020536	012737	177777	002440					
7365	020544	012737	000021	002300		MOV	#21,REALPAT		MOV #-1,NOSCOPE
7366	020552	013702	002632			MOV	BAKPAT,R2		;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7367	020556	004737	032630			CALL	BACKGND		
7368	020562	010203				MOV	R2,R3		
7369	020564	000303				SWAB	R3		
7370	020566	012701	160000			MOV	#LAST+2,R1		
7371	020572	010105				MOV	R1,R5		
7372	020574	012704	060000			MOV	#FIRST,R4		
7373	020600	022737	000005	004010		CMP	#5,PROTYP		;IS THIS AN 11/83
7374	020606	001450				BEQ	1\$;BRANCH IF IT IS
7375	020610	022737	000003	004010		CMP	#3,PROTYP		;IS THIS AN 11/24?
7376	020616	001407				BEQ	3\$;BRANCH IF SO
7377	020620	022737	000007	002102		CMP	#7,BANK		
7378	020626	001003				BNE	3\$		
7379	020630	012701	140000			MOV	#140000,R1		
7380	020634	010105				MOV	R1,R5		
7381	020636	022737	000177	002102	3\$:	CMP	#177,BANK		
7382	020644	001003				BNE	5\$		
7383	020646	012701	140000			MOV	#140000,R1		
7384	020652	010105				MOV	R1,R5		
7385	020654	012737	026442	002262	5\$:	MOV	#MTPA21,SUPDOADD		
7386	020662	004737	024446			CALL	SUPD03		
7387	020666	012737	026472	002262		MOV	#MTPB21,SUPDOADD		
7388	020674	004737	024462			CALL	SUPD04		
7389	020700	010401				MOV	R4,R1		
7390	020702	012737	026526	002262		MOV	#MTPC21,SUPDOADD		
7391	020710	004737	024462			CALL	SUPD04		
7392	020714	012737	026562	002262		MOV	#MTPD21,SUPDOADD		
7393	020722	004737	024462			CALL	SUPD04		
7394	020726	000434				BR	2\$		
7395	020730	022737	000177	002102	1\$:	CMP	#177,BANK		
7396	020736	001003				BNE	4\$		
7397	020740	012701	140000			MOV	#140000,R1		
7398	020744	010105				MOV	R1,R5		
7399	020746	012737	026442	002262	4\$:	MOV	#MTPA21,SUPDOADD		
7400	020754	004737	024446			CALL	SUPD03		
7401	020760	012737	026472	002262		MOV	#MTPB21,SUPDOADD		
7402	020766	004737	024462			CALL	SUPD04		
7403	020772	010401				MOV	R4,R1		
7404	020774	012737	026526	002262		MOV	#MTPC21,SUPDOADD		
7405	021002	004737	024462			CALL	SUPD04		
7406	021006	012737	026562	002262		MOV	#MTPD21,SUPDOADD		
7407	021014	004737	024462			CALL	SUPD04		
7408	021020	005037	002440		2\$:	CLR	NOSCOPE		
7409	021024	000207				RETURN			

```

7411 021026      MT0022: SUBTST <<MT0022      SETUP REFRESH & SHIFTING DIAGONAL TEST>>
;*****
;*SUBTEST      MT0022 SETUP REFRESH & SHIFTING DIAGONAL TEST
;*****
7412 021026      004737 024234      CALL      KAMITEST      ;CHECK FOR KAMIKAZE MODE
7413 021032      103001      ON.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
      021032      000207      ;*****
      021034      000207      ;*****
      021036      ;*****
7414 021036      012737 000022 002300      MOV      #22,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7415 021044      012737 026612 002262      MOV      #MTP022,SUPDOADD
7416 021052      004737 024446      CALL     SUPD03      ;DO IT IN SUPERVISOR MODE
7417 021056      000207      RETURN
7418
7419 021060      MT0023: SUBTST <<MT0023      SHIFTING DIAGONAL TEST>>
;*****
;*SUBTEST      MT0023 SHIFTING DIAGONAL TEST
;*****
7420 021060      004737 024234      CALL     KAMITEST      ;CHECK FOR KAMIKAZE MODE
7421 021064      103001      ON.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
      021064      000207      ;*****
      021066      000207      ;*****
      021070      ;*****
7422 021070      012737 000023 002300      MOV      #23,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7423 021076      012737 026612 002262      MOV      #MTP022,SUPDOADD
7424 021104      012737 177777 002004      SET      DIAGFLAG      ;IDENTIFY DIAGONAL TEST TO MTP022
      021104      004737 024446      CALL     SUPD03      ;DO IT IN SUPERVISOR MODE
7425 021112      005037 002004      CLR      DIAGFLAG
7426 021116      000207      RETURN
7427 021122

```



```

7429 021124
MT0024: SUBTST <<MT0024          SETUP FAST GALLOPING PATTERN TEST>>
;*****
;*SUBTEST      MT0024  SETUP FAST GALLOPING PATTERN TEST
;*****
7430 021124 004737 024234          CALL    KAMITEST          ;CHECK FOR KAMIKAZE MODE
7431 021130          ON.ERROR THEN $RETURN          ;IF NOT IN KAMIKAZE MODE RETURN
          021130 103001          BCC L170
          021132 000207          RTS PC
          021134          L170:::;
7432 021134          SET      NOSCOPE
          021134 012737 177777 002440          MOV    #-1,NOSCOPE
7433 021142 012737 000024 002300          ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7434 021150 013702 002632          MOV    #24,REALPAT
7435 021154 004737 032630          MOV    BAKPAT,R2
7436 021160 010203          CALL   BACKGND
7437 021162 010304          MOV    R2,R3
7438 021164 000304          MOV    R3,R4
7439 021166 012701 060000          SWAB   R4
7440 021172 012705 157776          MOV    #FIRST,R1
7441 021176 022737 000005 004010          MOV    #LAST,R5
7442 021204 001426          CMP    #5,PROTYP
7443 021206 022737 000003 004010          BEQ   1$
7444 021214 001406          CMP    #3,PROTYP
7445 021216 022737 000007 002102          BEQ   3$
7446 021224 001002          CMP    #7,BANK
7447 021226 012705 137776          BNE   3$
7448 021232 022737 000177 002102 3$:  MOV    #137776,R5
7449 021240 001003          CMP    #177,BANK
7450 021242 012701 137776          BNE   7$
7451 021246 010105          MOV    #137776,R1
7452 021250 104415          MOV    R1,R5
7453 021252 012737 027326 002262 7$:  SAVREG
7454 021260 000412          MOV    #MTPB24,SUPDOADD
7455 021262 022737 000177 002556 1$:  BR     2$
7456 021270 001002          CMP    #177,LASTBANK
7457 021272 012705 137776          BNE   4$
7458 021276 104415          MOV    #137776,R5
7459 021300 012737 027326 002262 4$:  SAVREG
7460 021306 004737 024462          MOV    #MTPB24,SUPDOADD
7461          CALL   SUPD04
          ;DO IT AGAIN FOR COMPLEMENT DATA
7462 021312 104416          RESREG
7463 021314 000302          SWAB   R2
7464 021316 000303          SWAB   R3
7465 021320 004737 024462          CALL   SUPD04
7466 021324 005037 002440          CLR    NOSCOPE
7467 021330 000207          RETURN
    
```


7470 021332

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

 ;SUBTEST MT0026 SETUP RANDOM DATA TEST

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

7516 021610

MT0027: SUBTST <<MT0027 UNIQUE BANK TEST>>

;SUBTEST MT0027 UNIQUE BANK TEST

7517
7518
7519 021610 012737 000027 002300
7520 021616 104502
7521 021620 012737 024446 002522
7522 021626
021626 012737 177777 002426
7523 021634
021634 012737 000001 002452
021642
7524 021642
021642 005037 002102
021646
7525 021646 004737 040052
7526 021652
021652 005737 002116
021656 001436
021660 005737 002124
021664 001033
7527 021666 104511
7528 021670
021670 013702 002102
7529 021674 012700 060000
7530 021700 010004
7531 021702 012701 040000
7532 021706 010103
7533 021710
021710 023727 002452 000001
021716 001005
7534 021720 012737 027704 002262
7535 021726 004777 160570
7536 021732
021732
7537 021732
021732 023727 002452 000002
021740 001005
7538 021742 012737 027712 002262
7539 021750 004737 024446
7540 021754
021754
7541 021754
021754
7542 021754
021754 005237 002102
021760 023737 002102 002556
021766 003727
021770
7543 021770
021770 005237 002452
021774 023727 002452 000002
022002 003717
022004
7544 022004

;MAKE SURE THAT EACH BANK CAN HAVE UNIQUE DATA
;WRITE AND READ THE BANK NUMBER IN EACH BANK (EXCEPT WHERE THE PROGRAM IS)
MOV #27,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLRCSR ;CLEAR CSRS
MOV #SUPD03,LINK1 ;SET UP LINK
SET NOFSMODE
STAR27: FOR I := #1 TO #2
FOR BANK := #0 TO LASTBANK
CALL EXBANK
IF ACFLAG IS TRUE AND RRFLAG IS FALSE
INVALIDATE ;INVALIDATE BACKGROUND PATTERN ON "BANK"
LET R2 := BANK
MOV #FIRST,R0
MOV R0,R4
MOV #SIZE,R1
MOV R1,R3
IF I EQ #1
MOV #MTP034,SUPDOADD
CALL @LINK1
END ;OF IF
IF I EQ #2
MOV #MTP034+6,SUPDOADD
CALL SUPD03
END ;OF IF
END ;OF FOR BANK
END ;OF FOR I
IF FS7FLAG IS TRUE

MOV #-1,NOFSMODE
MOV #1,I
B24:;;;;;
CLR BANK
B25:;;;;;
TST ACFLAG
BEQ L171
TST RRFLAG
BNE L171
MOV BANK,R2
CMP I,#1
BNE L172
L172:;;;;;
CMP I,#2
BNE L173
L173:;;;;;
L171:;;;;;
INC BANK
CMP BANK, LASTBANK
BLE B25
E25:;;;;;
INC I
CMP I,#2
BLE B24
E24:;;;;;

```

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

          CLR  NOFSMODE
          RETURN
        END ;OF IF FS7FLAG

        FOR I := #1 TO #2

          FOR BANK := LASTBANK DOWNT0 #0

            CALL EXBANK
            IF ACFLAG IS TRUE AND RRFLAG IS FALSE

              LET R2 := BANK

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

                CMP I,#1
                BNE L176

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

              IF I EQ #2

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

            END ;OF IF

          END ;OF FOR BANK

        END ;OF FOR I

          CLR  NOFSMODE
          RETURN
        END ;OF IF FS7FLAG

        L174:
          MOV #1,I
        B26:
          MOV LASTBANK,BANK
        B27:

          TST ACFLAG
          BEQ L175
          TST RRFLAG
          BNE L175

          MOV BANK,R2

          CMP I,#1
          BNE L176

        L176:
          CMP I,#2
          BNE L177

        L177:
        L175:
          DEC BANK
          CMP BANK,#0
          BGE B27
        E27:
          INC I
          CMP I,#2
          BLE B26
        E26:

```



```

7573 022200
7574 022200 005037 002264
7575 022204 012737 177777 002542
7576 022212 012737 000030 002300
7577 022220 012737 000001 002076
7578 022226 012737 024446 002522
7579 022234 012737 027466 002262
7580 022242 104470
7581 022244 012737 177777 002426
      022252 012737 177777 002440
7582 022260 005037 002102
      022264
7583 022264 004737 040052
7584 022270 005737 002120
      022274 001414
7585 022276 005737 002116
      022302 001411
      022304 005737 002124
      022310 001006
7586 022312 012701 040000
7587 022316 012700 060000
7588 022322 004777 160174
7589 022326
      022326
7590 022326
      022326
7591 022326 005237 002102
      022332 023737 002102 002556
      022340 003751
      022342
7592 022342 005737 002264
      022346 001032
7593 022350 012737 177777 002264
      022350 104502
7594 022356 004737 036402
7595 022360 103010
7596 022364 104472
7597 022370 005037 002426
7598 022374 005037 002440
      022400 005037 002542
7599 022404 000207
7600 022406
      022406
7601 022406 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      #MTPO30,SUPDOADD
      ECCDIS           ;DISABLE ERROR CORRECTION
      SET      NOFSMODE,NOSCOPE
      MOV      #-1,FULLREL
      MOV      #-1,NOFSMODE
      MOV      #-1,NOSCOPE
      CLR      BANK
B30:;;;;;
      CALL     EXBANK
      IF MKFLAG IS TRUE
      TST     MKFLAG
      BEQ    L200
      IF ACFLAG IS TRUE AND RRFLAG IS FALSE
      TST     ACFLAG
      BEQ    L201
      TST     RRFLAG
      BNE    L201
      MOV     #SIZE,R1
      MOV     #FIRST,R0
      CALL   @LINK1
      END ;OF IF ACFLAG
      END ;OF IF MKFLAG
      END ;OF FOR
      L201:;;;;;
      L200:;;;;;
      INC     BANK
      CMP     BANK,LASTBANK
      BLE    B30
E30:;;;;;
      IF PASFLG IS FALSE
      TST     PASFLG
      BNE    L202
      SET     PASFLG
      MOV     #-1,PASFLG
      CLRCR          ;CLEAR CSRS
      CALL   RELOCATE
      ON.ERROR
      BCC    L203
      ECCINIT           ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
      CLEAR     NOFSMODE,NOSCOPE,FULLREL
      CLR     NOFSMODE
      CLR     NOSCOPE
      CLR     FULLREL
      RETURN
      END ;OF ON.ERROR
      L203:;;;;;
      MOV     NEWBANK,BANK

```


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

```

7613 022454
7614 022454 004737 024234
7615 022460 103001
      022462 000207
      022464
7616 022464 012737 177777 002440
      022464 012737 000031 002300
7617 022472 012737 000031 002300
7618 022500 005037 002076
7619 022504 010346
      022506 013703 002102
      022512 004737 035716
7620 022516 012603
      022520
      022520 053737 002552 177776
7621 022526 004537 041024
      022526 000027
      022532 060000
      022534 027476
7622 022540 104417
7623 022542 013702 002572
7624 022546 010200
7625 022550 012701 100776
7626 022554 012705 060056
7627 022560 012737 060002 002262
7628 022566 012737 160000 002522
7629 022574 005737 002456
7630 022600 001005
7631 022602 023737 172252 172254
7632 022610 001405
7633 022612 000407
7634 022614 023737 177652 177654 1$:
7635 022622 001003
7636 022624 012737 140000 002522 2$:
7637 022632 004737 024462 3$:
7638 022636 005037 002440
7639 022642 000207

```

```

MT0031: SUBTST <<MT0031      SETUP SOB-A-LONG TEST>>
;*****
;*SUBTEST      MT0031  SETUP SOB-A-LONG TEST
;*****
      CALL      KAMITEST      ;CHECK FOR KAMIKAZE MODE
      ON.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
                                      BCC L204
                                      RTS PC
                                      L204:::
7616      SET      NOSCOPE
7617      MOV      #31,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
7618      CLR      NOPAR      ;SETUP PARITY ACTION
7619      MAP      BANK      ;MAP FIRST SO BLOCK MOVE WORKS
                                      MOV R3,-(SP)
7620      TESTAREA
7621      BIS      TESTMODE,PSW      ;ENTER TEST MODE
                                      ;GO TO SYSTEM TEST MODE
      .DSABL CRF
      BMOV      MTP031,FIRST,SOBLENGTH/2
      JSR      R5,BLOCK3
      SOBLENGTH/2
      FIRST
      MTP031
      .DSABL CRF
7622      KERNEL
7623      MOV      SOBK,R2
7624      MOV      R2,R0
7625      MOV      #100776,R1
7626      MOV      #FIRST+SOBLENGTH,R5
7627      MOV      #FIRST+2,SUPDOADD
7628      MOV      #LAST+2,LINK1
7629      TST      NOSUPER
7630      BNE      1$
7631      CMP      SIPAR5,SIPAR6
7632      BEQ      2$
7633      BR      3$
7634      CMP      UIPAR5,UIPAR6
7635      BNE      3$
7636      MOV      #140000,LINK1
7637      CALL      SUPD04
7638      CLR      NOSCOPE
7639      RETURN

```

7642 022644

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

;*SUBTEST MT0032 SETUP WRITE RECOVERY TEST

7643 022644 004737 024234

7644 022650 103001
022652 000207
022654

CALL KAMITEST ;CHECK FOR KAMIKAZE MODE
ON.ERROR THEN \$RETURN ;IF NOT IN KAMIKAZE MODE RETURN
BCC L205
RTS PC

L205:::~::~

7645 022654 012737 177777 002440

7646 022662 012737 000032 002300

7647 022670 005037 002076

7648 022674 010346

022676 013703 002102

022702 004737 035716

SET NOSCOPE
MOV #32,REALPAT ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
CLR NOPAR ;SETUP PARITY ACTION
MAP BANK ;MAP FIRST SO THAT THE BLOCK MOVE WORKS
MOV R3,-(SP)

7649 022710 012700 010247

7650 022714 012701 177667

7651 022720 012702 020000

7652 022724 010237 002522

7653 022730 012703 060000

7654 022734 012704 160000

7655 022740 005037 002524

7656 022744 005737 002456

7657 022750 001005

7658 022752 023737 172252 172254

7659 022760 001405

7660 022762 000415

7661 022764 023737 177652 177654 1\$:

7662 022772 001011

7663 022774 012704 140000 2\$:

7664 023000 012702 014000

7665 023004 010237 002522

7666 023010 012737 000001 002524

7667

7668 023016 053737 002552 177776 3\$:

023016 053737 002552 177776

7669

7670 023024 010023 4\$:

7671 023026 010144

7672 023030 077203

7673

7674 023032 022737 000005 004010

7675 023040 001003

7676

7677 023042 004537 040770

023042 027554

023046

7678 023050 104417 5\$:

7679

7680 023052 012702 005141

7681 023056 012700 023156

MOV #10247,R0 ;OP CODE OF INSTRUCTION "MOV R2,-(PC)"
MOV #177667,R1 ;OP CODE OF COMPLEMENT OF INSTRUCTION "JMP (R0)"
MOV #SIZE/2,R2 ;USED FOR 1/2 BANK LOOP
MOV R2,LINK1
MOV #FIRST,R3
MOV #LAST+2,R4
CLR LINK2
TST NOSUPER
BNE 1\$
CMP SIPAR5,SIPAR6
BEQ 2\$
BR 3\$
CMP UIPAR5,UIPAR6
BNE 3\$
MOV #140000,R4
MOV #14000,R2
MOV R2,LINK1
MOV #1,LINK2
TESTAREA ;ENTER TEST MODE
BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
;MOVE TEST TO MEMORY UNDER TEST
MOV R0,(R3)+
MOV R1,-(R4)
SOB R2,4\$
CMP #5,PROTYP
BNE 5\$
;MOVE LAST PART OF TEST TO FASTCITY
BMOV MTP032
JSR R5,BLOCK1
MTP032
.DSABL CRF
KERNEL ;ENTER KERNEL MODE
MOV #5141,R2 ;OP CODE OF INSTRUCTION "COM -(R1)"
MOV #10\$,R0 ;ADDRESS TO RETURN TO IN R0

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

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


```

7702 023160
7703 023160 004737 024234
7704 023164
      023164 103001
      023166 000207
      023170
7705 023170
      023170 012737 177777 002440
7706 023176 012737 000033 002300
7707 023204 005037 002076
7708 023210
      023210 010346
      023212 013703 002102
      023216 004737 035716
      023222 012603
7709
7710 023224
      023224 053737 002552 177776
7711 023232
      023232 004537 041024
      023236 000037
      023240 060000
      023242 027606
7712 023244 104417
7713
7714 023246 012705 060076
7715 023252 012737 060004 002262
7716 023260 012701 060002
7717 023264 012702 060003
7718 023270 012737 160000 002522
7719 023276 005737 002456
7720 023302 001005
7721 023304 023737 172252 172254
7722 023312 001405
7723 023314 000407
7724 023316 023737 177652 177654 1$:
7725 023324 001003
7726 023326 012737 140000 002522 2$:
7727
7728 023334 004737 024462 3$:
7729 023340 005037 002440
7730 023344 000207
7731
7732 023346

```

```

MT0033: SUBTST <<MT0033      SETUP BRANCH GOBBLE TEST>>
;*****
;*SUBTEST      MT0033      SETUP BRANCH GOBBLE TEST
;*****
      CALL      KAMITEST      ;CHECK FOR KAMIKAZE MODE
      ON.ERROR THEN $RETURN      ;IF NOT IN KAMIKAZE MODE RETURN
                                      BCC L206
                                      RTS PC
                                      L206:::;
      SET      NOSCOPE
                                      MOV #-1,NOSCOPE
      MOV      #33,REALPAT      ;SETUP PATTERN NUMBER FOR TYPEOUT & DISPLAY
      CLR      NOPAR      ;SETUP PARITY ACTION
      MAP      BANK      ;MAP FIRST SO THAT BLOCK MOVE WORKS
                                      MOV R3,-(SP)
      MOV      BANK,R3
      CALL     MAPPER
      .DSABL   CRF
                                      MOV (SP)+,R3
      TESTAREA      ;ENTER TEST MODE
      BIS      TESTMODE,PSW      ;GO TO SYSTEM TEST MODE
      .DSABL   CRF
      BMOV     MTP033,FIRST,GBLENGTH/2
      JSR     R5,BLOCK3
      GBLENGTH/2
      FIRST
      MTP033
      .DSABL   CRF
      KERNEL      ;ENTER KERNEL MODE
      MOV      #FIRST+GBLENGTH,R5
      MOV      #FIRST+4,SUPDOADD
      MOV      #FIRST+2,R1
      MOV      #FIRST+3,R2
      MOV      #LAST+2,LINK1
      TST     NOSUPER
      BNE     1$
      CMP     SIPAR5,SIPAR6
      BEQ     2$
      BR      3$
      CMP     UIPAR5,UIPAR6
      BNE     3$
      MOV     #140000,LINK1
      1$:
      CALL     SUPD04
      CLR     NOSCOPE
      RETURN

```

```

7733 023346 012737 000034 002300
7734 023354 012700 060000
7735 023360 012701 040000
7736 023364 013702 002620

```

```

MT0034: SUBTST <<MT0034      SOFT ERROR - BACKGROUND PATTERN TEST>>
;*****
;*SUBTEST      MT0034      SOFT ERROR - BACKGROUND PATTERN TEST
;*****
      MOV      #34,REALPAT
      MOV      #FIRST,R0
      MOV      #SIZE,R1
      MOV      SOFTPA^,R2

```

```

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

```

7773 023560

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

;*SUBTEST MT0036 SETUP CORRECTION CODE TEST

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

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

7784 023622

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

;*SUBTEST MT0037 SETUP ECC DISABLE TEST

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

MOV #37,REALPAT ;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

7794 023660

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

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

7795 023660 012737 000041 002300

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

7796 023666 004737 032674

CALL GETCSR ;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE

7797 023672

LET SUPDOADD := #MTP041 ;SET UP TEST ADDRESS

023672 012737 030370 002262

MOV #MTP041,SUPDOADD

7798 023700

LET R1 := #FIRST ;SET UP FIRST ADDRESS

023700 012701 060000

MOV #FIRST,R1

7799 023704

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

023704 023727 002102 000177

CMP BANK,#177

023712 001004

BNE L212

7800 023714

LET PASCNT := #12.

023714 012737 000014 002570

MOV #12.,PASCNT

7801 023722

ELSE

BR L213

023722 000403

L212:::;::;::;

7802 023724

LET PASCNT := #16.

023724 012737 000020 002570

MOV #16.,PASCNT

7803 023732

END

L213:::;::;::;

023732

7804 023732 004737 024446

CALL SUPD03 ;EXECUTE ADDRESS TO CSR TEST IN SUPVISIOR MODE

7805 023736 000207

RETURN ;

7807 023740

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

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

MOV #42,REALPAT ;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

```

7816 023774      MT0043: SUBTST <<MT0043      SETUP WRITE BYTE CLEARS SBE TEST>>
;*****
;*SUBTEST      MT0043 SETUP WRITE BYTE CLEARS SBE TEST
;*****
7817 023774 012737 000043 002300      MOV #43,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7818 024002 004737 037740      CALL MAPKERNAL ;MAP TO KERNEL SPACE
7819 024006      LET R1 := #100000 ;SET UP TEST ADDRESS
024006 012701 100000      MOV #100000,R1
7820 024012 004737 031030      CALL MTP043 ;PERFORM WRITE BYTE TEST
7821 024016 004737 040026      CALL UNMAP ;REMAP KERNEL SPACE
7822 024022 000207      RETURN
7823 024024      MT0044: SUBTST <<MT0044      SETUP SHIFTING 1/0'S THROUGH THE CHECK BITS TEST>>
;*****
;*SUBTEST      MT0044 SETUP SHIFTING 1/0'S THROUGH THE CHECK BITS TEST
;*****
7824 024024 012737 000044 002300      MOV #44,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7825 024032 004737 032674      CALL GETCSR ;GET CSR NUMBER AND ADDRESS FROM CONFIGURATION TABLE
7826 024036      LET SUPDOADD := #MTP044 ;SET UP TEST ADDRESS ;:IL
024036 012737 031224 002262      MOV #MTP044,SUPDOADD
7827 024044      LET R1 := #FIRST ;SET UP FIRST ADDRESS ;:IL
024044 012701 060000      MOV #FIRST,R1
7828 024050      IF BANK EQ #177 ;ARE WE AT BANK 177?
024050 023727 002102 000177      CMP BANK,#177
024056 001004      BNE L214
7829 024060      LET ENDADD := #120000
024060 012737 120000 002562      MOV #120000,ENDADD
7830 024066      ELSE
024066 000403      BR L215
024070      L214:::;:;:;
7831 024070      LET ENDADD := #160000
024070 012737 160000 002562      MOV #160000,ENDADD
7832 024076      END
024076      L215:::;:;:;
7833 024076 004737 024446      CALL SUPD03 ;EXECUTE ADDRESS TO CSR TEST IN SUPVISIOR MODE
7834 024102 000207      RETURN
7835 024104      MT0045: SUBTST <<MT0045      SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR>>
;*****
;*SUBTEST      MT0045 SETUP SYNDROMES TO CSR ON DOUBLE BIT ERROR
;*****
7836 024104 012737 000045 002300      MOV #45,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7837 024112 004737 037740      CALL MAPKERNAL ;MAP TO KERNEL SPACE
7838 024116      LET R1 := #100000 ;SET UP TEST ADDRESS
024116 012701 100000      MOV #100000,R1
7839 024122 004737 031546      CALL MTP045 ;PERFORM SYNDROMES TO CSR ON DOUBLE BIT ERROR
7840 024126 004737 040026      CALL UNMAP ;REMAP KERNEL SPACE
7841 024132 000207      RETURN
7842 024134      MT0046: SUBTST <<MT0046      SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET>>
;*****
;*SUBTEST      MT0046 SETUP CHECK SINGLE BIT ERRORS WITH ECC DISABLED TET
;*****
7843 024134 012737 000046 002300      MOV #46,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY
7844 024142 004737 037740      CALL MAPKERNAL ;MAP TO KERNEL SPACE
7845 024146      LET R1 := #100000 ;SET UP TEST ADDRESS
024146 012701 100000      MOV #100000,R1
7846 024152 004737 031734      CALL MTP046 ;PERFORM TRAPS DECTECTED ON SBE WITH ECC DISABLED TE
7847 024156 004737 040026      CALL UNMAP ;REMAP KERNEL SPACE
7848 024162 000207      RETURN

```

7849 024164

MT0047: SUBTST <<MT0047 SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST>>

;*SUBTEST MT0047 SETUP NO CSR UPDATE ON SBE WITH EXSISTING DBE TEST

7850 024164 012737 000047 002300

MOV #47,REALPAT ;SET UP TEST NUMBER FOR TYPEOUT AND DISPLAY

7851 024172 004737 037740

CALL MAPKERNAL ;MAP TO KERNEL SPACE

7852 024176

LET R1 := #100000 ;SET UP TEST ADDRESS

024176 012701 100000

MOV #100000,R1

7853 024202

LET R2 := #120000 ; " " SECOND TEST ADDRESS

024202 012702 120000

MOV #120000,R2

7854 024206 004737 032274

CALL MTP047 ;PERFORM NO UPDATE TO CSR ON SBE WITH DBE

7855 024212 004737 040026

CALL UNMAP ;REMAP KERNEL SPACE

7856 024216 000207

RETURN


```

7859 024220          MT0999: SUBTST <<MT0999          SETUP NULL TEST>>
;*****
;*SUBTEST          MT0999 SETUP NULL TEST
;*****
7860 024220 005037 002300          CLR          REALPAT
7861 024224          SET          NULLFLAG
024224 012737 177777 002344          MOV          #-1,NULLFLAG
7862 024232 000207          RETURN
7863
7864 024234          KAMITEST:SUBTST <<CHECK FOR KAMIKAZE MODE>>
;*****
;*SUBTEST          CHECK FOR KAMIKAZE MODE
;*****
7865 024234          IF KAMIKAZE IS TRUE OR ACTFLAG IS TRUE OR APTFLAG IS TRUE
024234 005737 002006          TST KAMIKAZE
024240 001006          BNE L216
024242 005737 002350          TST ACTFLAG
024246 001003          BNE L216
024250 005737 002352          TST APTFLAG
024254 001403          BEQ L217
024256          L216:::;:;:;
7866 024256          $RETURN NOERROR          ;RUN THE TEST
024256 000241          CLC
024260 000207          RTS PC
7867 024262          ELSE          BR L220
024262 000402          L217:::;:;:;
7868 024264          $RETURN ERROR          ;DON'T RUN THE TEST
024264 000261          SEC
024266 000207          RTS PC
7869 024270          END ;OF IF KAMIKAZE
024270          L220:::;:;:;

```

7872 024270

SUPD01: SUBTST <<SUBR EXECUTE PATTERN IN SUPERVISOR>>

;*SUBTEST SUBR EXECUTE PATTERN IN SUPERVISOR

7873 024270 010346
024270 013703 002102
024276 004737 035716

MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
MOV R3,-(SP)

7874 024302 012603
024304 004737 051430
7875 024310

SUPD02: CALL GETDIS
PUSH \$LPERR,\$LPADR

MOV (SP)+,R3

024310 013746 002624
024314 013746 002622
7876 024320 010037 002160
7877 024324 012700 002162

MOV \$LPERR,-(SP)
MOV \$LPADR,-(SP)

7878 024330 010120
7879 024332 010220
7880 024334 010320
7881 024336 010420
7882 024340 010520
7883 024342 010620
7884 024344 013700 002160
7885 024350 012737 024364 002622
7886 024356 013737 002622 002624

MOV R0,SUPDRO
MOV #SUPDR1,R0
MOV R1,(R0)+
MOV R2,(R0)+
MOV R3,(R0)+
MOV R4,(R0)+
MOV R5,(R0)+
MOV SP,(R0)+
MOV SUPDRO,R0
MOV #TAG4\$, \$LPADR
MOV \$LPADR,\$LPERR
TAG4\$: MOV #SUPDR6+2,R0

7888 024370 014006
7889 024372 014005
7890 024374 014004
7891 024376 014003
7892 024400 014002
7893 024402 014001
7894 024404 014000

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

7895 024406 052737 040000 177776

SUPERVISOR ;ENTER SUPERVISOR MODE
BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
.DSABL CRF

7896 024414 012706 000740
7897 024420 104424
7898 024422 004737 177640
7899 024426 104423
7900 024430 104417
7901 024432 000004
7902 024434

MOV #SUPSTK,SSP
CACHOFF ;TURN CACHE OFF
CALL FASTCITY ;CALL TO THE USER INSTRUCTION PAR'S
CACHON ;TURN CACHE ON
KERNEL ;ENTER KERNEL MODE
SCOPE
POP \$LPADR,\$LPERR

024434 012637 002622
024440 012637 002624
7903 024444 000207

MOV (SP)+,\$LPADR
MOV (SP)+,\$LPERR

RETURN

```

7906 024446          SUPD03: MAP      BANK      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      024446 010346          MOV      BANK,R3      MOV R3,-(SP)
      024450 013703 002102    CALL     MAPPER
      024454 004737 035716    .DSABL   CRF

      024460 012603          MOV (SP)+,R3
7907 024462 004737 051430    SUPD04: CALL  GETDIS
7908 024466          PUSH     $LPERR,$LPADR
      024466 013746 002624    MOV $LPERR,-(SP)
      024472 013746 002622    MOV $LPADR,-(SP)
7909 024476 010037 002160    MOV     R0,SUPDRO
7910 024502 012700 002162    MOV     #SUPDR1,R0
7911 024506 010120          MOV     R1,(R0)+
7912 024510 010220          MOV     R2,(R0)+
7913 024512 010320          MOV     R3,(R0)+
7914 024514 010420          MOV     R4,(R0)+
7915 024516 010520          MOV     R5,(R0)+
7916 024520 010620          MOV     SP,(R0)+
7917 024522 013700 002160    MOV     SUPDRO,R0
7918 024526 012737 024542 002622    MOV     #TBG4$, $LPADR
7919 024534 013737 002622 002624    MOV     $LPADR,$LPERR
7920 024542 012700 002176    TBG4$: MOV     #SUPDR6+2,R0
7921 024546 014006          MOV     -(R0),SP
7922 024550 014005          MOV     -(R0),R5
7923 024552 014004          MOV     -(R0),R4
7924 024554 014003          MOV     -(R0),R3
7925 024556 014002          MOV     -(R0),R2
7926 024560 014001          MOV     -(R0),R1
7927 024562 014000          MOV     -(R0),R0
7928 024564          TESTAREA
      024564 053737 002552 177776    BIS     TESTMODE,PSW      ;ENTER SUPERVISOR MODE
                                          ;GO TO SYSTEM TEST MODE
                                          .DSABL   CRF
7929 024572 005737 002456    TST     NOSUPER
7930 024576 001403          BEQ     1$
7931 024600 012706 000700    MOV     #USESTK,USP
7932 024604 000402          BR      2$
7933 024606 012706 000740    1$: MOV     #SUPSTK,SSP
7934 024612 104424          2$: CACHOFF
7935 024614 004777 155442    CALL   @SUPDOADD      ;TURN CACHE OFF
7936 024620 104423          CACHON      ;TURN CACHE ON
7937 024622 104417          KERNEL    ;ENTER KERNEL MODE
7938 024624 000004          SCOPE
7939 024626          POP     $LPADR,$LPERR
      024626 012637 002622    MOV (SP)+,$LPADR
      024632 012637 002624    MOV (SP)+,$LPERR
7940 024636 000207          RETURN

```



```

7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953 024640

7954 024640 010220
7955 024642 077102
7956 024644 000240
7957 024646 012401
7958 024650 020102
7959 024652 001402
7960 024654 104430
7961 024656 000240
7962 024660 077306
7963 024662 000207
7964 024664

7965 024664 010220
7966 024666 062702 000002
7967 024672 077104
7968 024674 000240
7969 024676 012400
7970 024700 020005
7971 024702 001401
7972 024704 104427
7973 024706 062705 000002
7974 024712 077307
7975 024714 000207
7976 024716

7977 024716 010540
7978 024720 062705 000002
7979 024724 077104
7980 024726 000240
7981 024730 162702 000002
7982 024734 012401
7983 024736 020102
7984 024740 001401
7985 024742 104430
7986 024744 077307
7987 024746 000207

```

```

.SBTTL MEMORY TEST PATTERN ROUTINES
;*****
; PATTERN REGISTER CONVENTIONS
; R0 FIRST ADDRESS OF PATTERN (FIRST, LAST+2, ETC)
; R1 NUMBER OF ADDRESSES IN PATTERN (SIZE)
; R2 DATA FOR PATTERN (ONES, 52525, ETC)
; R3 COPY OF R1 (IF NECESSARY)
; R4 COPY OF R0 (IF NECESSARY)
; R5 COPY OF R2 (IF NECESSARY)
;*****
MTP000: SUBTST <<MTP000 BASIC DATA TEST>>
;*****
;*SUBTEST MTP000 BASIC DATA TEST
;*****
1$: MOV R2, (R0)+ ;V177640
SOB R1, MTP000 ;V177642
NOP ;V177644
2$: MOV (R4)+, R1 ;V177646
CMP R1, R2 ;V177650
BEQ 3$ ;V177652
PERR02 ;V177654
NOP ;V177656
3$: SOB R3, 2$ ;V177660
RETURN ;V177662
MTP001: SUBTST <<MTP001 ADDRESS TEST>>
;*****
;*SUBTEST MTP001 ADDRESS TEST
;*****
3$: MOV R2, (R0)+ ;V177640
ADD #2, R2 ;V177642
SOB R1, 3$ ;V177646
NOP ;V177650
1$: MOV (R4)+, R0 ;V177652
CMP R0, R5 ;V177654
BEQ 2$ ;V177656
PERR01 ;V177660
2$: ADD #2, R5 ;V177662
SOB R3, 1$ ;V177666
RETURN ;V177672
MTP002: SUBTST <<MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)>>
;*****
;*SUBTEST MTP002 COMPLEMENT ADDRESS TEST (WRITE DOWN, READ UP)
;*****
3$: MOV R5, -(R0) ;V177640
ADD #2, R5 ;V177642
SOB R1, 3$ ;V177646
NOP ;V177650
1$: SUB #2, R2 ;V177652
MOV (R4)+, R1 ;V177656
CMP R1, R2 ;V177660
BEQ 2$ ;V177662
PERR02 ;V177664
2$: SOB R3, 1$ ;V177666
RETURN ;V177670

```


7990 024750

MTPA03: SUBTST <<MTPA03 3 XOR 9 WORST CASE NOISE TEST (WRITE)>>
;*****
;*SUBTEST MTPA03 3 XOR 9 WORST CASE NOISE TEST (WRITE)
;*****

7991
7992
7993
7994
7995
7996
7997 024750 010421
7998 024752 010421
7999 024754 077203
8000 024756 005104
8001 024760 052704
8002 024762 000401
8003 024764 012702 000004
8004 024770 077511
8005 024772 005104
8006 024774 052704
8007 024776 000401
8008 025000 012705 000100
8009 025004 077317
8010 025006 000207
8011
8012
8013 025010

;R1 = ADDRESS
;R2 = SMALL LOOP CONSTANT
;R3 = NUM OF ADD TO TEST (LARGE LOOP)
;R4 = GOOD DATA
;R5 = MEDIUM LOOP CONSTANT
.ENABL LSB
1\$: MOV R4,(R1)+ ;V177640
MOV R4,(R1)+ ;V177642
SOB R2,1\$;V177644
COM R4 ;V177646
BIS (PC)+,R4 ;V177650
WARN2: 401 ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #4,R2 ;V177654
SOB R5,1\$;V177660
COM R4 ;V177662
BIS (PC)+,R4 ;V177664
WARN3: 401 ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #64,R5 ;V177670
SOB R3,1\$;V177674
RETURN ;V177676
.DSABL LSB

MTPB03: SUBTST <<MTPB03 3 XOR 9 WORST CASE NOISE TEST (READ)>>
;*****
;*SUBTEST MTPB03 3 XOR 9 WORST CASE NOISE TEST (READ)
;*****

8014
8015 025010 000137 025050
8016 025014 077203
8017 025016 005104
8018 025020 052704
8019 025022 000401
8020 025024 012702 000004
8021 025030 077511
8022 025032 005104
8023 025034 052704
8024 025036 000401
8025 025040 012705 000100
8026 025044 077317
8027 025046 000207
8028

.ENABL LSB
1\$: JMP @#MTPC03 ;V177640 GO TO V172360
SOB R2,1\$;V177644
COM R4 ;V177646
BIS (PC)+,R4 ;V177650
WARN4: 401 ;V177652 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #4,R2 ;V177654
SOB R5,1\$;V177660
COM R4 ;V177662
BIS (PC)+,R4 ;V177664
WARN5: 401 ;V177666 WARNING LOCATION IS MODIFIED BEFORE LOADING
MOV #64,R5 ;V177670
SOB R3,1\$;V177674
RETURN ;V177676
.DSABL LSB

8031 025050

MTPC03: SUBTST <<MTPC03 TEST DATA SUBPROGRAM>>
;*****
;*SUBTEST MTPC03 TEST DATA SUBPROGRAM
;*****

8032 025050 020421
8033 025052 001401
8034 025054 104431
8035 025056 005141
8036 025060 005111
8037 025062 000137 025066
8038
8039 025066

1\$: CMP R4,(R1)+ ;V172360
BEQ 1\$;V172362
PERR03 ;V172364
COM -(R1) ;V172366
COM (R1) ;V172370
JMP @#MTPD03 ;V172372 GO TO V172260

MTPD03: SUBTST <<MTPD03 TEST DATA SUBSUBPROGRAM>>
;*****
;*SUBTEST MTPD03 TEST DATA SUBSUBPROGRAM
;*****

8040 025066 020421
8041 025070 001401
8042 025072 104431
8043 025074 005127
8044 025076 000000
8045 025100 001363
8046 025102 000137 025014

1\$: CMP R4,(R1)+ ;V172260
BEQ 1\$;V172262
PERR03 ;V172264
COM (PC)+ ;V172266
0 ;V172270
BNE MTPC03 ;V172272 GO TO V172360
JMP @#MTPB03+4 ;V172274 GO TO V177644

8049 025106

MTPA04: SUBTST <<MTPA04 ROTATING ZEROS TEST>>

;*SUBTEST MTPA04 ROTATING ZEROS TEST

8050 025106 012705 000010
8051 025112 010504
8052 025114 000241
8053 025116 000137 025142
8054 025122 016004 177776
8055 025126 103402
8056 025130 020204
8057 025132 001401
8058 025134 104432
8059 025136 077115
8060 025140 000207
8061
8062 025142

1\$: MOV #8.,R5 ;V177640
MOV R5,R4 ;V177644
CLC ;V177646
JMP @#MTPB04 ;V177650
MOV -2(R0),R4 ;V177654
BCS 2\$;V177660
CMP R2,R4 ;V177662
BEQ 3\$;V177664
2\$: PERR04 ;V177666
3\$: SOB R1,1\$;V177670
RETURN ;V177672

MTPB04: SUBTST <<MTPB04 SUBR ROTATING BIT>>

;*SUBTEST MTPB04 SUBR ROTATING BIT

8063 025142 106110
8064 025144 077502
8065 025146 106120
8066 025150 106110
8067 025152 077402
8068 025154 106120
8069 025156 000137 025122
8070
8071 025162

1\$: ROLB (R0) ;V172360
SOB R5,1\$;V172362
ROLB (R0)+ ;V172364
2\$: ROLB (R0) ;V172366
SOB R4,2\$;V172370
ROLB (R0)+ ;V172372
JMP @#MTPA04+14 ;V172374

MTP005: SUBTST <<MTP005 ROTATION ONES TEST>>

;*SUBTEST MTP005 ROTATION ONES TEST

8072 025162 012705 000010
8073 025166 010504
8074 025170 000261
8075 025172 000137 025142
8076 025176 016004 177776
8077 025202 103002
8078 025204 020204
8079 025206 001401
8080 025210 104432
8081 025212 077115
8082 025214 000207

1\$: MOV #8.,R5 ;V177640
MOV R5,R4 ;V177644
SEC ;V177646
JMP @#MTPB04 ;V177650
MOV -2(R0),R4 ;V177654
BCC 2\$;V177660
CMP R2,R4 ;V177662
BEQ 3\$;V177664
2\$: PERR04 ;V177666
3\$: SOB R1,1\$;V177670
RETURN ;V177672

IF THIS HAPPENS THE GOOD & BAD MATCH

8085 025216

MTP006: SUBTST <<MTP006 INITIAL DATA TEST>>

;SUBTEST MTP006 INITIAL DATA TEST

8086

8087

8088 025216 012737 000001 002242

8089 025224 005037 002244

8090 025230 013771 002242 000000 1\$:

8091 025236 013771 002244 000002

8092 025244 017102 000000

8093 025250 023702 002242

8094 025254 001401

8095 025256 104433

8096

8097 025260 017102 000002 2\$:

8098 025264 023702 002244

8099 025270 001401

8100 025272 104434

8101

8102 025274 005737 002244 3\$:

8103 025300 100405

8104 025302

025302 006137 002242

025306 006137 002244

8105 025312 000746

8106

8107 025314 012737 177776 002242 4\$:

8108 025322 012737 177777 002244

8109 025330 013771 002242 000000 5\$:

8110 025336 013771 002244 000002

8111 025344 017102 000000

8112 025350 023702 002242

8113 025354 001401

8114 025356 104433

8115

8116 025360 017102 000002 6\$:

8117 025364 023702 002244

8118 025370 001401

8119 025372 104434

8120

8121 025374 005737 002244 7\$:

8122 025400 100005

8123 025402

025402 006137 002242

025406 006137 002244

8124 025412 000746

8125 025414 000207 8\$:

; THIS TEST CHECKS THE DI/DO LINES BY
; SHIFTING A 1 THROUGH THE WORD.

MOV #1,DATBUF ;SET THE FIRST TEST BIT

CLR DATBUF+2 ;CLEAR 2ND WORD

MOV DATBUF,@(R1) ;WRITE TEST WORD 1

MOV DATBUF+2,@2(R1) ;AND TEST WORD 2

MOV @(R1),R2

CMP DATBUF,R2 ;NOW READ THEM

BEQ 2\$;BR IF FIRST 16 OK

PERR07 ;ERROR TRAP

2\$: MOV @2(R1),R2

CMP DATBUF+2,R2 ;NOW READ SECOND WORD

BEQ 3\$;BR IF OK

PERR10 ;ERROR TRAP

3\$: TST DATBUF+2 ;HAS LAST BIT BEEN TESTED ?

BMI 4\$;MINUS MEANS BIT 31

DLEFT DATBUF ;NO, SHIFT TEST BIT LEFT

ROL DATBUF

ROL DATBUF+2

.DSABL CRF

BR 1\$;GO WRITE NEW TEST DATA

;NOW GOING TO SHIFT A 0 IN DATA DIRECTION

4\$: MOV #177776,DATBUF ;PUT A 0 IN BIT 0

MOV #-1,DATBUF+2 ;AND 1'S IN ALL OTHERS

5\$: MOV DATBUF,@(R1) ;WRITE THE DATA

MOV DATBUF+2,@2(R1) ;2 WORDS WORTH

MOV @(R1),R2

CMP DATBUF,R2 ;NOW READ FIRST WORD

BEQ 6\$;BR IF OK

PERR07

6\$: MOV @2(R1),R2

CMP DATBUF+2,R2 ;NOW, READ SECOND WORD

BEQ 7\$;BR IF OK

PERR10

7\$: TST DATBUF+2 ;TESTED BIT 31 YET?

BPL 8\$;BR IF YES, WE'RE DONE

DLEFT DATBUF

ROL DATBUF

ROL DATBUF+2

.DSABL CRF

BR 5\$;KEEP GOING

8\$: RETURN

8128 025416

```
MTP007: SUBTST <<MTP007 ADDRESS BIT TEST>>
;*****
;*SUBTEST MTP007 ADDRESS BIT TEST
;*****
```

```
8129
8130
8131
8132
8133 025416 111100
8134 025420 105700
8135 025422 001401
8136 025424 104435
8137
8138 025426 105111
8139 025430 111100
8140 025432 105700
8141 025434 001001
8142 025436 104436
8143
8144 025440 040201
8145 025442 006302
8146 025444 050201
8147 025446 011100
8148 025450 005700
8149 025452 001401
8150 025454 104437
8151
8152 025456 005111
8153 025460 011100
8154 025462 005700
8155 025464 001001
8156 025466 104440
8157
8158 025470 022702 100000
8159 025474 001407
8160 025476 022702 010000
8161 025502 001356
8162 025504 006302
8163 025506 012701 160000
8164 025512 000752
8165 025514 000207
```

```

; 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.
;
;
; MOVB (R1),R0
; TSTB R0 ;READ AND COMPARE FOR ZEROS
; BEQ 1$ ;BR IF OK
; PERR11
;
; 1$: COMB (R1) ;COMPLEMENT THE BYTE
; MOVB (R1),R0
; TSTB R0 ;READ FOR NON ZEROS
; BNE 2$ ;BR IF OK
; PERR12
;
; 2$: BIC R2,R1 ;MASK OFF THE ASSERTED BIT
; ASL R2 ;SHIFT R2 FOR NEXT BIT
; BIS R2,R1 ;SET THE NEW BIT INTO R1
; MOV (R1),R0
; TST R0 ;READ THE NEW ADDRESS
; BEQ 3$ ;READ FOR ZEROS
; PERR13
;
; 3$: COM (R1) ;COMPL THE WORD
; MOV (R1),R0
; TST R0 ;READ IT AGAIN
; BNE 4$
; 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
```

8168 025516

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

8169
8170
8171 025516 010402
8172 025520 010403
8173 025522 062702 000004
8174 025526 012713 177777
8175 025532 012763 177777 000002
8176 025540 105013
8177 025542 010401
8178 025544 020201
8179 025546 001420
8180 025550 020301
8181 025552 001007
8182 025554 111100
8183
8184 025556 022700 000000
8185 025562 001401
8186 025564 104435
8187
8188 025566 005201
8189 025570 000765
8190 025572 111100
8191 025574 122700 177777
8192 025600 001401
8193 025602 104436
8194
8195 025604 005201
8196 025606 000756
8197 025610 112713 177777
8198 025614 005203
8199 025616 020302
8200 025620 001347
8201 025622 000207
8202

;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
1\$: CLR B (R3) ;CLEAR A BYTE
; INITIALIZE R1 FOR EACH PASS
2\$: MOV R4,R1 ;IF EQUAL, JUST READ LAST BYTE
CMP R2,R1 ;BR IF EQUAL
BEQ 6\$;IS THIS THE BYTE OF ZEROS
;BR IF NOT
CMP R3,R1
BNE 4\$
MOV B (R1),R0
;WARNING IF YOU OPTOMIZE CHANGE THE PCBUMP FOR THIS ERROR INCASE OF TRAPS
CMP #0,R0 ;IT IS, COMPARE FOR ZEROS
BEQ 3\$
PERR11
3\$: INC R1 ;NEXT BYTE
BR 2\$;RETURN
4\$: MOV B (R1),R0
CMPB #-1,R0 ;ITS NOT THE BYTE OF 0'S, READ 1'S
BEQ 5\$
PERR12
5\$: INC R1 ;MOVE TO NEXT BYTE
BR 2\$
6\$: MOV B #-1,(R3) ;RESTORE 1'S TO BYTE JUST TESTED
INC R3 ;INC TO NEXT BYTE
CMP R3,R2 ;WAS THAT JUST THE LAST ONE?
BNE 1\$;BR IF NO
RETURN

8205 025624

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

8206
8207
8208
8209
8210

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

8211 025624 104424
8212 025626 005037 002072

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

8213 025632 012737 000001 002076

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

8214 025640 012737 060000 002034

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

8215 025646 104513

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

8216 025650 012737 003145 002150

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

8217 025656 104425

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

8218 025660 012737 103145 002044

TST (R1) ;READ A=0 TO GET DOUBLE BIT ERROR
IF PARCNT NE #1 ;WAS BUSPBL ASSERTED????
CMP PARCNT,#1
BEQ L221

8219 025666 005011

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

8220 025670 005711

READCSR ;READ CSR FOR CORRECT CHECK BITS AND DBE INDICATOR
BIC #BIT13,CSR ;CLEAR INHIBIT MODE POINTER FROM DATA IF IT EXISTS!
IF CSR NE GOOD THEN ;CHECK IF DOUBLE ERROR BIT IS SET
CMP CSR,GOOD
BEQ L222

8221 025672 023727 002072 000001

SET HEADER ;
LET BAD := CSR ;BAD DATA
MOV #-1,HEADER

8222 025702 104055

ERROR +65 ;
END ;

8223 025704 025704

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

8224 025704 104426

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

8225 025706 042737 020000 002150

TSTB (R1) ;READ R1 TO SEE IF IT IS STILL 0
IF PARCNT NE #1 ;WAS WRITE ABORTED????
CMP PARCNT,#1
BEQ L223

8226 025714 023737 002150 002044

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

8227 025724 012737 177777 002612

ERROR +65 ;
END ;

8228 025732 013737 002150 002052

END ;

8229 025740 104065

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

8230 025742 025742

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

8231 025742 104473

TSTB (R1) ;READ R1 TO SEE IF IT IS STILL 0
IF PARCNT NE #1 ;WAS WRITE ABORTED????
CMP PARCNT,#1
BEQ L223

8232 025744 005037 002266

SET HEADER ;

8233 025750 025750

SET HEADER ;

8234 025750 104473

SET HEADER ;

8235 025752 005237 002266

SET HEADER ;

8236 025756 005037 002072

SET HEADER ;

8237 025762 112711 000377

SET HEADER ;

8238 025766 105711

SET HEADER ;

8239 025770 023727 002072 000001

SET HEADER ;

8240 025776 001411

SET HEADER ;

8240 026000

SET HEADER ;


```

8241 026000 012737 177777 002612          LET GOOD := #0      ;GOOD DATA      MOV #-1,HEADER
      026006 005037 002044          LET BAD := #377    ;BAD DATA      CLR GOOD
8242 026012 012737 000377 002052          ERROR +56        ;
      026012 104056          END                ;
8243 026020 005201          INC R1            ;AND REPEAT ON HIGH BYTE L223:::::
8244 026022 002266 000002          UNTIL PASSNO EQ #2 ;
      026024 023727 001346          ;
      026032 005041          ;
      026034 104503          ;
8247 026034 005041          CLR -(R1)         ;CLEAR LUT
8248 026036 005037          CLR1CSR          ;CLEAR CSR
8249 026040 104423          CLR PARCNT      ;CLEAR PARITY TRAP COUNTER
8250 026044 000207          CACHON          ;TURN ON CACHE
8251 026046 000207          RETURN        ;
8252

```


8255 026050

```

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$:     MOVB   RO,(R1)+ ;WRITE A BYTE
        MOVB   R3,(R1)+ ;WRITE THE MEMORY WITH THE BYTE STORED IN BAKPAT+1
        CMP    R1,R5    ;COMPARE TEST LOC TO TOP + 2
        BLO   1$       ;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

```

```

8256
8257
8258
8259
8260
8261
8262 026050 012701 060000
8263 026054 010104
8264 026056 012705 160000
8265 026062 012700 000377
8266 026066 010003
8267 026070 000303
8268 026072 110021
8269 026074 110321
8270 026076 020105
8271 026100 103774
8272
8273 026102 014102
8274 026104 020002
8275
8276 026106 001401
8277 026110 104446
8278
8279 026112 020104
8280 026114 101372
8281 026116 000303
8282 026120 000300
8283 026122 001763
8284
8285 026124 000207
8287

```

8289 026126

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

8290
8291
8292
8293
8294
8295 026126 104424
8296 026130 005000
8297 026132 105037 002264
8298 026136 104513
8299 026140
026140
8300 026140
026140 105237 002264
8301 026144
026144 012704 177777
8302 026150
026150 005037 002324
8303 026154
026154 123727 002264 000001
026162 001003
8304 026164
026164 012705 000001
8305 026170
026170 000402
026172
8306 026172
026172 012705 177776
8307 026176
026176
8308 026176
026176
8309 026176 005237 002324
8310 026202
026202 005204
8311 026204
026204 116402 030276
8312 026210 072227 000005
8313 026214 052702 000004
8314 026220
026220 010237 002150
8315 026224 104425
8316 026226
026226 010011
8317 026230 104503
8318 026232 005711
8319 026234 104426
8320 026236 042737 177757 002150
8321 026244
026244 023727 002150 000020
026252 001407
8322 026254
026254 012737 000020 002044
8323 026262

```

;
; 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 RO ;CLEAR DATA
CLRB PASFLG ;CLEAR PASFLG
CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT ;
;
; B32:::
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
; CMPB PASFLG,#1
; BNE L225
; DATA=0;BIT TO BE CORRECTED IS A ONE
; MOV #1,R5
ELSE ;
; BR L226
;
; L225:::
LET R5 := #177776 ;DATA=177776;BIT TO BE CORRECTED IS A ZERO
; MOV #177776,R5
END ;
;
; L226:::
;
; B33:::
REPEAT ;INCREMENT BIT POINTER
; POINT TO NEXT SET OF CHECK BITS
; INC R4
LET R2 :B= PTABLE(R4) ;GET NEXT SET OF CHECK BITS
; MOV B 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) := RO ;WRITE DATA TO TEST ADDRESS
; MOV RO,(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 L227
LET GOOD := #20 ;
; MOV #20,GOOD
LET BAD := CSR ;
;

```

```

8324 026262 013737 002150 002052          MOV CSR,BAD
8325 026270 104060          ERROR +60          ;NO ERROR
                                ;
                                L227:;;;;;
8326 026272 104514          SYNREG             ;ENABLE SYNDROME BIT REGISTER
8327 026274 104426          READCSR           ;GET SYNDROMES FROM CSR
8328 026276 042737 174033 002150          BIC #1C3744,CSR  ;MASK SYNDROME BITS
8329 026304          LET R3 :B= SBESYN(R4) ;GET GOOD SYNDROMES
                                MOV# SBESYN(R4),R3
8330 026310 116403 026422          ASH #5,R3         ;SHIFT INTO POSITION
8331 026314 072327 000005          BIS #BIT2,R3     ;SET DIAG MODE IN DATA
8332 026320 052703 000004          IF R3 NE CSR    ;DO SYNDROME BITS AGREE
                                CMP R3,CSR
                                BEQ L230
8333 026326 020337 002150          SET HEADER
                                ;
                                MOV #-1,HEADER
8334 026326 012737 177777 002612          LET GOOD := R3
                                ;
                                MOV R3,GOOD
8335 026334 010337 002044          LET BAD := CSR
                                ;
                                MOV CSR,BAD
8336 026340 013737 002150 002052          ERROR +42
8337 026346 104042          END
                                ;
                                L230:;;;;;
8338 026350 005011          CLR (R1)         ;CLEAR LUT
8339 026352 123727 002264 000001          IFB PASFLG EQ #1 ;SHIFT NEW DATA DEPENDING ON PASFLG
                                CMPB PASFLG,#1
                                BNE L231
8340 026362 006305          ASL R5           ;SHIFT BITNO TO THE LEFT
8341 026364 000402          ELSE
                                ;
                                BR L232
8342 026366 000261          SEC             L231:;;;;;
8343 026370 006105          ROL R5          ;SET CARRY BIT AND.....
8344 026372          END           ;ROTATE LEFT
                                ;
                                L232:;;;;;
8345 026372          UNTIL BITNO EQ #16. ;UNTIL ALL BITS ARE DONE
                                CMP BITNO,#16.
                                BNE B33
8346 026402 005100          COM R0          E33:;;;;;
8347 026404          UNTILB PASFLG EQ #2 ;COMPLEMENT DATA AND REPEAT
                                ;UNTIL 2 PASSES ARE COMPLETE!
                                CMPB PASFLG,#2
                                BNE B32
8348 026414 104503          CLR1CSR         E32:;;;;;
8349 026416 104423          CACHON         ;CLEAR CSR
8350 026420 000207          RETURN        ;TURN CACHE
8351          ;
8352          ;
8353          ;
8354 026422 016 013 023  SBESYN: .BYTE 16,13,23,25,26,31,32,34,43,45,46,51,52,54,61,64
      026425 025 026 031
      026430 032 034 043
      026433 045 046 051
      026436 052 054 061
      026441 064
8355

```


8357 026442

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

8358
8359 026442 014100
8360 026444 020200
8361 026446 001401
8362 026450 104443
8363
8364 026452 000311
8365 026454 011100
8366 026456 020300
8367 026460 001401
8368 026462 104444
8369
8370 026464 020401
8371 026466 001365
8372 026470 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

8373
8374 026472
8375 026472 011100
8376 026474 020300
8377 026476 001401
8378 026500 104444
8379
8380 026502 000311
8381 026504 011100
8382 026506 020200
8383 026510 001401
8384 026512 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
3\$: ADD #2,R1 ;V177662
CMP R5,R1 ;V177666 ;DONE?
BNE 1\$;V177670 ;NO - LOOP
RETURN ;V177672 ;YES - RETURN

8385
8386 026514 062701 000002
8387 026520 020501
8388 026522 001363
8389 026524 000207
8390
8391 026526
8392 026526 011100
8393 026530 020200
8394 026532 001401
8395 026534 104443
8396
8397 026536 000311
8398 026540 011100
8399 026542 020300
8400 026544 001401
8401 026546 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
3\$: ADD #2,R1 ;V177662
CMP R5,R1 ;V177666 ;DONE?
BNE 1\$;V177670 ;NO - LOOP
RETURN ;V177672 ;YES - RETURN

8402
8403 026550 062701 000002
8404 026554 020501
8405 026556 001363
8406 026560 000207


```

8409 026562
8410 026562 014100
8411 026564 020300
8412 026566 001401
8413 026570 104444
8414
8415 026572 000311
8416 026574 011100
8417 026576 020200
8418 026600 001401
8419 026602 104443
8420
8421 026604 020401
8422 026606 001365
8423 026610 000207
8424

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

```

```

8427 026612 MTP022: SUBTST <<MTP022 REFRESH & SHIFTING DIAGONAL TEST>>
;*****
;*SUBTEST MTP022 REFRESH & SHIFTING DIAGONAL TEST
;*****
8428 ;(1) WE WRITE A DIAGONAL PATTERN IN MEMORY (WITH CACHE ON).
8429 ;(2) IF A REFRESH TEST WE DISTURB ALL ROWS FOR > 2 MS (WITH CACHE ON).
8430 ;(3) WE READ & CHECK FOR CORRECTNESS THE DIAGONAL PATTERN
8431 ; (WITH CACHE OFF).
8432 KDIAG=8. ;HOW OFTEN A DIAGONAL STRIPE OCCURS (MUST BE A POWER OF 2)
8433 026612 000010 FOR EVEN := #1 TO #2 ;FOR DATA & COMPLEMENT DATA
026612 012737 000001 002364 MOV #1,EVEN
026620 B34:;;;;;
8434 026620 IF EVEN EQ #1 CMP EVEN,#1
026620 023727 002364 000001 BNE L235
026626 001005 MOV ZEROS,R2
8435 026630 LET R2 := ZEROS MOV ONES,R3
8436 026634 LET R3 := ONES MOV ONES,R3
8437 026640 ELSE BR L236
026640 000404 L235:;;;;;
026642 LET R2 := ONES MOV ONES,R2
8438 026642 013702 002614 LET R3 := ZEROS MOV ZEROS,R3
8439 026646 013703 002336 END ;OF IF EVEN
8440 026652 FOR STRIPES := #0 TO #KDIAG-1 ;FOR THE NUMBER OF STRIPES
026652 005037 002366 CLR STRIPES
026656 B35:;;;;;
8442 ;WRITE LOOP
8443 CACHON ;TURN CACHE ON
8444 026656 104423 LET COUNT := STRIPES MOV STRIPES,COUNT
8445 026660 013737 002366 002370 LET R1 := #FIRST MOV #FIRST,R1
8446 026666 012701 060000 WHILE R1 LOS #LAST
8447 026672 B36:;;;;;
026672 020127 157776 CMP R1,#LAST
026676 101032 BHI L237
8448 026700 IF COUNT LT #0 THEN LET COUNT := #KDIAG-1 TST COUNT
026700 005737 002370 BGE L240
026704 002003 MOV #KDIAG-1,COUNT
026706 012737 000007 002370 L240:;;;;;
026714 IF #374 OFF.IN R1 THEN LET COUNT := COUNT - #1 BIT #374,R1
8449 026714 032701 000374 BNE L241
026720 001002 DEC COUNT
026722 005337 002370 L241:;;;;;
026726 IF COUNT NE #0 TST COUNT
8450 026726 005737 002370 BEQ L242
026732 001404 LET (R1) := R2
8451 026734 010211 MOV R2,(R1)
026734
  
```

```

8452 026736          LET 2(R1) := R2
      026736 010261 000002
8453 026742          ELSE
      026742 000403
      026744
8454 026744          LET (R1) := R3
      026744 010311
8455 026746          LET 2(R1) := R3
      026746 010361 000002
8456 026752          END ;OF IF COUNT
      026752
8457 026752          LET COUNT := COUNT - #1
      026752 005337 002370
8458 026756          LET R1 := R1 + #4
      026756 062701 000004
8459 026762          END ;OF WHILE
      026762 000743
      026764
8460
8461
8462 026764          ;END OF WRITE LOOP
      026764 005737 002004
      026770 001002
      026772 004737 027166
      026776
8463
8464 026776          IF DIAGFLAG IS FALSE THEN $CALL REFRESH
      026776 013737 002366 002370
      027004
8465 027004          ;READ LOOP
      027004 012701 060000
      104424
8466 027010          LET COUNT := STRIPES
      CACHOFF          LET R1 := #FIRST
                        ;TURN CACHE OFF

```

```

MOV R2,2(R1)
BR L243
L242:;;;;;
MOV R3,(R1)
MOV R3,2(R1)
L243:;;;;;
DEC COUNT
ADD #4,R1
BR B36
L237:;;;;;
E36:;;;;;
TST DIAGFLAG
BNE L244
JSR PC,REFRESH
L244:;;;;;
MOV STRIPES,COUNT
MOV #FIRST,R1

```

8468	027012			WHILE R1 LOS #LAST		B37:::~::~:
	027012					CMP R1,#LAST
	027012	020127	157776			BHI L245
	027016	101046				TST COUNT
8469	027020			IF COUNT LT #0 THEN LET COUNT := #KDIAG-1		BGE L246
	027020	005737	002370			MOV #KDIAG-1,COUNT
	027024	002003				L246:::~::~:
	027026	012737	000007	002370		- #1
	027034					BIT #374,R1
8470	027034			IF #374 OFF.IN R1 THEN LET COUNT := COUNT		BNE L247
	027034	032701	000374			DEC COUNT
	027040	001002				L247:::~::~:
	027042	005337	002370			TST COUNT
	027046					BEQ L250
8471	027046			IF COUNT NE #0		MOV (R1),R0
	027046	005737	002370			CMP R2,R0
	027052	001412				BEQ L251
8472	027054			LET R0 := (R1)		
	027054	011100				
8473	027056			IF R2 NE R0		
	027056	020200				
	027060	001401				
8474	027062			PERR17		
8475	027064			END ;OF IF R2		
	027064					
8476	027064			LET R0 := 2(R1)		L251:::~::~:
	027064	016100	000002			MOV 2(R1),R0
8477	027070			IF R2 NE R0		
	027070	020200				
	027072	001401				
8478	027074			PERR17		
8479	027076			END ;OF IF R2		
	027076					
8480	027076			ELSE		L252:::~::~:
	027076	000411				BR L253
	027100					L250:::~::~:
8481	027100			LET R0 := (R1)		MOV (R1),R0
	027100	011100				
8482	027102			IF R3 NE R0		
	027102	020300				
	027104	001401				
8483	027106			PERR20		
8484	027110			END ;OF IF R3		
	027110					
8485	027110			LET R0 := 2(R1)		L254:::~::~:
	027110	016100	000002			MOV 2(R1),R0
8486	027114			IF R3 NE R0		
	027114	020300				
	027116	001401				
8487	027120			PERR20		
8488	027122			END ;OF IF R3		
	027122					
8489	027122			END ;OF IF COUNT		L255:::~::~:
	027122					L253:::~::~:
8490	027122			LET COUNT := COUNT - #1		
	027122	005337	002370			DEC COUNT
8491	027126			LET R1 := R1 + #4		

8492 027126 062701 000004
027132
027132 000727
027134
027134

END ;OF WHILE

ADD #4,R1

BR B37

L245:::;
E37:::;

8493
8494

;END OF READ LOOP

8495 027134
027134 005237 002366
027140 023727 002366 000007
027146 003643
027150

END ;OF FOR STRIPES

INC STRIPES
CMP STRIPES,#KDIAG-1
BLE B35

E35:::;

8496 027150
027150 005237 002364
027154 023727 002364 000002
027162 003616
027164

END ;OF FOR EVEN

INC EVEN
CMP EVEN,#2
BLE B34

E34:::;

8497 027164 000207
8498
8499 027166

RETURN

REFRESH:SUBTST <<SUBR REFRESH DELAY>>

;SUBTEST SUBR REFRESH DELAY

8500
8501 027166 012700 060000
027172

;DISTURB EACH ROW FOR > 3.2 MS
FOR RO := #FIRST TO #FIRST+374 BY #4

MOV #FIRST,RO

B40:::;

8502 027172 004737 027236
8503 027176
027176 062700 000004
027202 020027 060374
027206 003771
027210

CALL REFSUB
END ;OF FOR RO

ADD #4,RO
CMP RO,#FIRST+374
BLE B40

E40:::;

8504 027210
027210 012700 120000
8505 027214
027214

LET RO := #FIRST+BIT14

MOV #FIRST+BIT14,RO

027214 020027 020372
027220 101005
8506 027222 004737 027236
8507 027226
027226 062700 000004
8508 027232
027232 000770
027234
027234

WHILE RO LOS #LAST+BIT14+374

B41:::;

CMP RO,#LAST+BIT14+374
BHI L256

8509 027234 000207
8510 027236 012704 000640
8511 027242 062700 000002
8512 027246 005140
8513 027250 005120
8514 027252 005110
8515 027254 005110
8516 027256 077405
8517 027260 162700 000002
8518 027264 000207

CALL REFSUB
LET RO := RO + #4

ADD #4,RO

END ;OF WHILE

BR B41

L256:::;
E41:::;

REFSUB: RETURN
MOV #640,R4
ADD #2,RO
1\$: COM -(RO)
COM (RO)+
COM (RO)
COM (RO)
SOB R4,1\$
SUB #2,RO
RETURN

;TIME FOR A > 3.2 MS LOOP

8521 027266

MTPA24: SUBST <<MTPA24 FAST GALLOPING PATTERN TEST>>

;*SUBTEST MTPA24 FAST GALLOPING PATTERN TEST

8522
8523
8524
8525
8526
8527
8528
8529
8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8540
8541
8542
8543
8544
8545
8546

;THE TOTAL TEST (INCLUDING SETUP) IS AS FOLLOWS
;*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
;* STORED AT LOCATION BAKPAT
;*(2) TEST BEGINS AT LOWEST LOCATION BEING TESTED
;* (LETS NAME IT 'A')
;*(3) LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
;*(4) SWAPS BYTES FOR LOCATION 'A'.
;*(5) READS 'A', READS 'B'
;*(6) 'B' = 'B'+400 (ADDS 64 DOUBLE WORDS TO 'B')
;*(7) REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
;*(8) END OF THE BANK A+2
;*(9) REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
;*(10) AFTER EXECUTING THE TEST DATA IS COMPLEMENTED
;* AND STEPS 1-9 ARE REPEATED
;REGISTERS ARE USED AS FOLLOWS
;R0 TEST DATA
;R1 'A'
;R2 'B'
;R3 BAKPAT
;R4 SWAPAT
;R5 LAST

;NOTE THE PATTERN STARTS AT MTPB24!!!!!!!!!!!!!!!!!!!!

8547 027266 011100
8548 027270 020004
8549 027272 001401
8550 027274 104447
8551
8552 027276 011200
8553 027300 020003
8554 027302 001401
8555 027304 104450
8556
8557 027306 062702 000400
8558 027312 020205
8559 027314 101764
8560
8561 027316 062701 000002
8562 027322 000137 027326

;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

8565 027326

MTPB24: SUBTST <<MTPB24 FAST GALLOP PART B>>

;*****
;*SUBTEST MTPB24 FAST GALLOP PART B
;*****

8566

8567 027326 010411
8568 027330 020105
8569 027332 001001
8570 027334 000207
8571 027336 000137 027342
8572
8573 027342;SDPAR'S
MOV R4,(R1) ;V172260 ;WRITE 'A'
CMP R1,R5 ;V172262 ;DONE?
BNE 1\$;V172264 ;BR IF NO
RETURN ;V172266 ;YES - RETURN
1\$: JMP @#MTPC24 ;V172270 ;GOTO V172360

MTPC24: SUBTST <<MTPC24 FAST GALLOP PART C>>

;*****
;*SUBTEST MTPC24 FAST GALLOP PART C
;*****

8574

8575 027342 010102
8576 027344 011100
8577 027346 020004
8578 027350 001401
8579 027352 104447
8580 027354 000137 027306
8581;KDPAR'S
MOV R1,R2 ;V172360 ;RESET 'B' <--- 'A'
MOV (R1),R0 ;V172362 ;READ 'A'
CMP R0,R4 ;V172364 ;CHECK 'A'
BEQ 1\$;V172366 ;BR IF OK
PERR23 ;V172370 ;REPORT ERROR
1\$: JMP @#MTPA24+20 ;V172372 ;GOTO V177660

8584 027360

MTPA26: SUBTST <<MTPA26 RANDOM DATA (WRITE)>>
;*****

;*SUBTEST MTPA26 RANDOM DATA (WRITE)
;*****

8585 027360 000137 027430
8586 027364 010221
8587 027366 010321
8588 027370 077005
8589 027372 000207
8590
8591 027374

1\$: JMP @#MTPC26 ;V177640 GOTO V172360
MOV R2,(R1)+ ;V177644
MOV R3,(R1)+ ;V177646
SOB R0,1\$;V177650
RETURN ;V177652

MTPB26: SUBTST <<MTPB26 RANDOM DATA (READ)>>
;*****

;*SUBTEST MTPB26 RANDOM DATA (READ)
;*****

8592
8593
8594 027374 000137 027430
8595 027400 020221
8596 027402 001401
8597 027404 104451
8598 027406 005127
8599 027410 000000
8600 027412 020321
8601 027414 001401
8602 027416 104451
8603 027420 005167 177764
8604 027424 077015
8605 027426 000207
8606
8607
8608
8609 027430

.DSABL AMA
.ENABL LSB
1\$: JMP @#MTPC26 ;V177640 GOTO V172360
CMP R2,(R1)+ ;V177644
BEQ 2\$;V177646
PERR25 ;V177650
2\$: COM (PC)+ ;V177652
RANODD: 0 ;V177654 FOR ERROR REPORTING
CMP R3,(R1)+ ;V177656
BEQ 3\$;V177660
PERR25 ;V177662
3\$: COM RANODD ;V177664
SOB R0,1\$;V177670
RETURN ;V177672
.DSABL LSB
.ENABL AMA

MTPC26: SUBTST <<RANDOM NUMBER SUBPROGRAM>>
;*****

;*SUBTEST RANDOM NUMBER SUBPROGRAM
;*****

8610
8611
8612
8613
8614
8615 027430 073427 000007
8616 027434 060305
8617 027436 005504
8618 027440 060204
8619 027442 062705 001057
8620 027446 000240
8621
8622 027450

;CALLER MUST SETUP
; MOV SEEDLO,R3
; MOV SEEDHI,R2
; MOV R3,R5
; MOV R2,R4
ASHC #7,R4 ;V172360
ADD R3,R5 ;V172364
ADC R4 ;V172366
ADD R2,R4 ;V172370
ADD #1057,R5 ;V172372
NOP ;V172376 GOTO V172260

MTPD26: SUBTST <<RANDOM NUMBER SUBSUBPROGRAM>>
;*****

;*SUBTEST RANDOM NUMBER SUBSUBPROGRAM
;*****

8623 027450 005504
8624 027452 062704 047401
8625 027456 010503
8626 027460 010402
8627 027462 000137 027364

ADC R4 ;V172260
ADD #47401,R4 ;V172262
MOV R5,R3 ;V172266
MOV R4,R2 ;V172270
JMP @#MTPA26+4 ;V172272 GOTO V177644

8630 027466

```

MTP030: SUBTST <<MT0030      FLUSH OUT DBE'S>>
;*****
;*SUBTEST      MT0030  FLUSH OUT DBE'S
;*****

```

```

8631 027466 011002
8632 027470 010220
8633 027472 077103
8634 027474 000207
8635
8636 027476

```

```

1$:  MOV      (R0),R2      ;V177640
      MOV      R2,(R0)+    ;V177642
      SOB      R1,1$      ;V177644
      RETURN                      ;V177646

```

```

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

```

```

8637
8638 027476 000000
8639 027500 077001
8640 027502 005167 177772
8641 027506 020167 177766
8642 027512 001403
8643 027514 104454
8644 027516 010167 177756
8645 027522 005167 177752
8646 027526 010200
8647
8648 027530 010503
8649 027532 005725
8650 027534 010504
8651 027536 020537 0C2522
8652 027542 001001
8653 027544 000207
8654
8655 027546 014344
8656 027550 001376
8657 027552 000752
8658 000056
8659

```

```

      .DSABL  AMA
      O
;MOVE TERMINATOR
1$:  SOB      R0,1$      ;SOB TILL R0 UNDERFLOWS
      COM      1$      ;WRITE COMPLEMENT OF SOB
      CMP      R1,1$    ;READ & CHECK FOR NOT "SOB R0, DOT"
      BEQ      2$      ;OK - SKIP
      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

```

8687 027554

MTP032: SUBTST <<MTP032 WRITE RECOVERY TEST>>

;*SUBTEST MTP032 WRITE RECOVERY TEST

8688
8689
8690
8691
8692

;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.

8693 027554 012401
8694 027556 020102
8695 027560 001401
8696 027562 104430
8697 027564 077305
8698 027566 013703 002522
8699 027572 012400
8700 027574 020005
8701 027576 001401
8702 027600 104427
8703 027602 077305
8704 027604 000207

1\$: MOV (R4)+,R1 ;V177640 ;GET DATA FROM LOWER 1/2 BANK
CMP R1,R2 ;V177642 ;IS IT #5141?
BEQ 2\$;V177644 ;YES - SKIP
PERR02 ;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
PERR01 ;V177664 ;NO- TAKE ERROR TRAP
4\$: SOB R3,3\$;V177666 ;LOOP FOR 1/2 BANK
RETURN

8707 027606

```

MTP033: SUBTST <<MTP033          BRANCH GOBBLE TEST>>
;*****
;*SUBTEST      MTP033  BRANCH GOBBLE TEST
;*****
      .DSABL  AMA
      0
BGTEST: 0          ;MOVE TERMINATOR
BRGOBB: SEC          ;TEST WORD (TWO BYTES)
      ADCB   (R1)    ;SET CARRY (TO BE ADDED TO "BGTEST")
      BMI   1$      ;INCREMENT LOW BYTE OF "BGTEST"
      INCB  (R2)    ;BRANCH WHEN BIT7 IS SET
      BR    BRGOBB ;INCREMENT HIGH BYTE OF "BGTEST"
                       ;LOOP 128 TIMES

      ;NOW CHECK FOR CORRECT CONDITION CODES
1$:   BVS    2$      ;BR IF V-BIT SET (SHOULD BE)
      PERR35        ;NO - REPORT ERROR AND ABORT TEST
                       ;COND CODES NOT EQUAL TO 1010
2$:   CLV
      INCB  (R2)    ;CLEAR V-BIT
      BCS   3$      ;INCREMENT HIGH BYTE OF "BGTEST" ONCE MORE
      BVC   3$      ;BR IF C-BIT SET (SHOULD NOT BE)
      BMI   4$      ;BR IF V-BIT CLEAR (SHOULD NOT BE)
      PERR35        ;BR IF N-BIT SET (SHOULD BE)
                       ;NO - REPORT ERROR AND ABORT TEST
                       ;COND CODES NOT EQUAL TO 1010

      ;UPDATE TEST POINTERS
4$:   MOV    PC,R1
5$:   SUB    #5$-BGTEST,R1
      MOV    R1,R2
      INC    R2

      ;UPDATE MOVE REGISTERS
      MOV    R5,R3
      TST   (R5)+
      MOV    R5,R4          ;BUMP (SAFELY) BY 2

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

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

GBLENGTH=.-MTP033
      .ENABL  AMA

```

```

8708
8709 027606 000000
8710 027610 000000
8711 027612 000261
8712 027614 105511
8713 027616 100402
8714 027620 105212
8715 027622 000773
8716
8717
8718 027624 102401
8719 027626 104461
8720
8721 027630 000242
8722 027632 105212
8723 027634 103402
8724 027636 102001
8725 027640 100401
8726 027642 104461
8727
8728
8729
8730 027644 010701
8731 027646 162701 000036
8732 027652 010102
8733 027654 005202
8734
8735
8736 027656 010503
8737 027660 005725
8738 027662 010504
8739
8740
8741 027664 020537 002522
8742 027670 001001
8743 027672 000207
8744
8745
8746 027674 014344
8747 027676 001376
8748 027700 005011
8749 027702 000743
8750 000076
8751

```

8753 027704

```

MTP034: SUBTST <<MTP034          SOFT ERROR - BACKGROUND PATTERN TEST>>
;*****
;*SUBTEST      MTP034  SOFT ERROR - BACKGROUND PATTERN TEST
;*****

```

```

8754 027704 010220
8755 027706 077102
8756 027710 000207
8757 027712 012401
8758 027714 020102
8759 027716 001402
8760 027720 104430
8761 027722 000240
8762 027724 077306
8763 027726 000207

```

```

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

```


8765 027730

MTP035:SUBTST <<MTP035 WORST CASE NOISE PARITY TEST>>
;*****
;*SUBTEST MTP035 WORST CASE NOISE PARITY TEST
;*****

8766 027730 012737 000003 002076
8767

MOV #3,NOPAR ;SET PARITY TRAPS TO RETURN TO "PARTHERE"

8768 027736 012700 060000
027742

FOR RO := #FIRST TO #LAST BY #4000

MOV #FIRST,RO

8769 027742 012737 000005 002150

MOV #BIT2!BIT0,CSR ;SET WRITE WRONG PARITY & PAR. TRAPS INTO CSR
LOADCSR

8770 027750 104425

8771 027752 012737 030006 002304

MOV #1\$,PARTHERE ;WWP TEST LOCATION
MOV (RO),(RO)

8772 027760 011010

TST (RO)

8773 027762 005710

MOV RO,ADDRESS

8774 027764 010037 002034

ERROR +50

8775 027770 104050

CALL PERBNK

8776 027772 004737 050446

BIT #BIT10,CONFIG+2(R3)

8777 027776 032763 002000 002666

BNE 2\$

8778 030004 001002

1\$: READCSR

8779 030006 104426

ERRGEN

8780 030010 104512

8781

2\$: CLR1CSR

8782 030012 104503

MOV (RO),(RO) ;CLEAR WRONG PARITY IN MEMORY

8783 030014 011010

MOV #BIT0,CSR

8784 030016 012737 000001 002150

LOADCSR

8785 030024 104425

MOV #3\$,PARTHERE

8786 030026 012737 030040 002304

TST (RO)

8787 030034 005710

BR 4\$

8788 030036 000405

3\$: MOV RO,ADDRESS

8789 030040 010037 002034

ERROR +50

8790 030044 104050

CALL PERBNK

8791 030046 004737 050446

4\$: END; OF FOR

8792 030052

ADD #4000,RO

CMP RO,#LAST

BLE B42

030052 062700 004000

030056 020027 157776

030062 003727

E42:;;;;;;

030064

8793

CLR NOPAR ;RESET PARITY TRAP ACTION

8794 030064 005037 002076

8795 030070 000207

RETURN

8797 030072

MTP036: SUBTST <<MTP036 CORRECTION CODE TEST>>
;*****
;#SUBTEST MTP036 CORRECTION CODE TEST
;*****

8798
8799
8800
8801
8802 030072 104424
8803 030074 105037 002264
8804 030100 104513
8805 030102
030102
8806 030102 105237 002264
030102
8807 030106 012704 177777
030106
8808 030112 005037 002324
030112
8809 030116 123727 002264 000001
030116
030124 001003
8810 030126 012705 000001
030126
8811 030132 000402
030132
030134
8812 030134 012705 177776
030134
8813 030140
030140
8814 030140
030140
8815
8816 030140 005237 002324
8817 030144 005204
030144
8818 030146 116402 030276
030146
8819 030152 072227 000005
8820 030156 052702 000004
030156
8821 030162 010237 002150
030162
8822 030166 104425
030166
8823 030170 010011
030170
8824 030172 005711
030172
8825 030174 021105
030174
030176 001412
8826 030200
030200 012737 060000 002034
8827 030206 010237 002314
030206
8828 030212 010537 002246
030212
8829 030216

; 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 ;
; B43:;;;;;;
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
; CMPB PASFLG,#1
; BNE L257
LET R5 := #1 ;DATA=0;BIT TO BE CORRECTED IS A ONE
; MOV #1,R5
ELSE ;
; BR L260
LET R5 := #177776 ;DATA=177776;BIT TO BE CORRECTED IS A ZERO
; MOV #177776,R5
END ;
; L260:;;;;;;
REPEAT ;
; B44:;;;;;;
INC BITNO ;INCREMENT BIT POINTER
LET R4 := R4 + #1 ;POINT TO NEXT SET OF CHECK BITS
; INC R4
LET R2 :B= PTABLE(R4) ;GET NEXT SET OF CHECK BITS
; 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)
TST (R1) ;CORRECT SBE
IF (R1) NE R5 ;WAS DATA CORRECTED???
; CMP (R1),R5
; BEQ L261
LET ADDRESS := #60000 ;MOV ERROR INFORMATION IN
; MOV #60000,ADDRESS
LET CHECK := R2 ;
; MOV R2,CHECK
LET TSTDAT := R5 ;
; MOV R5,TSTDAT
LET TSTDAT+2 := (R1) ;

```

      030216 011137 002250
8830 030222 104052          ERROR +52          ;NO ERROR          MOV (R1),TSTDAT+2
8831 030224          END                      ;
      030224          CLR (R1)                ;CLEAR LUT          L261:;;;;;;;;
8832 030224 005011          IFB PASFLG EQ #1      ;SHIFT NEW DATA DEPENDING ON PASFLG
8833 030226 123727 002264 000001          ;SHIFT BITNO TO THE LEFT      CMPB PASFLG,#1
      030226 001002          ASL R5                ;
      030234 006305          ELSE                      ;
8834 030236 000402          SEC R5                ;SET CARRY BIT AND.....
8835 030240 000261          ROL R5                ;ROTATE LEFT
      030240 006105          END                      ;
      030242          UNTIL BITNO EQ #16.          ;UNTIL ALL BITS ARE DONE      L262:;;;;;;;;
8836 030244 023727 002324 000020          ;UNTIL ALL BITS ARE DONE      BR L263
      030246 001331          COM R0                ;COMPLEMENT DATA AND REPEAT  L263:;;;;;;;;
8837 030246 005100          UNTILB PASFLG EQ #2      ;UNTIL 2 PASSES ARE COMPLETE!
8838 030246 123727 002264 000002          ;UNTIL 2 PASSES ARE COMPLETE!  E44:;;;;;;;;
      030246 001305          CLR1CSR                ;CLEAR CSR          E43:;;;;;;;;
8839 030246 104503          CACHON                ;TURN CACHE
8840 030256 104423          RETURN
8841 030260 000207          ;
8842 030270 000207          ;
8843 030272 000207          ;
8844 030274 000207          ;
8845          ;
8846          ;
8847          ;
8848 030276 002 007 037 PTABLE: .BYTE 2,7,37,31,32,25,26,20,57,51,52,45,46,40,75,70
      030301 031 032 025
      030304 026 020 057
      030307 051 052 045
      030312 046 040 075
      030315 070

```


8850 030316

MTP037: SUBTST <<MTP037 CHECK ECC DISABLE TEST>>
;*****
;*SUBTEST MTP037 CHECK ECC DISABLE TEST
;*****

8851
8852
8853
8854
8855 030316 104424
8856 030320 005037 002044
8857 030324 005037 002314
8858 030330 104475
8859 030332 012737 000100 002314
8860 030340 104475
8861 030342 005011
8862 030344 005711
8863 030350 011137 002052
8864 030354 012737 060000 002034
8865 030362 104037
8866 030364
8867 030364 104423
8868 030366 000207

;
; THIS TEST CHECKS THAT ECC CAN BE DISABLED AND THAT
; NO CORRECTION TAKES PLACE WITH ECC DISABLED.
;
;CACHOFF ;TURN OFF CACHE
LET GOOD := #0 ;GOOD DATA FOR ERROR PRINT OUT CLR GOOD
LET CHECK := #0 ;CLEAR CHECK BIT FIELD CLR CHECK
CB1CSR ;ENABLE SYNDROME/CHECK BIT REGISTER
LET CHECK := #100 ;SBE CHECK BITS MOV #100,CHECK
CB1CSR ;WRITE CHECK BITS TO CB REGISTER
LET (R1) := #0 ;WRITE CHECK BITS TO MEMORY CLR (R1)
IF (R1) NE #0 ;WAS CORRECTION MADE???? TST (R1)
; BEQ L266
LET BAD := (R1) ;YES IT WAS.....ERROR MOV (R1),BAD
LET ADDRESS := #60000 ; MOV #60000,ADDRESS
ERROR +37 ;
END ;
CACHON ;TURN ON CACHE L266:::;;
RETURN ;

8871 030370

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

8872
8873
8874
8875

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

8876 030370 013704 002102

LET R4 := BANK ;GET STARTING BANK NUMBER
ASH #9.,R4 ;SHIFT INTO POSTION TO MATCH ADDRESS IN CSR
BIC #+C7740,R4 ;CLEAR OFF EXTRANEIOUS BITS
LET R0 := #-40 ;INIT CSR ADDRESS TO 0 - 1K (BIT 5 = 1K ADD.)
MOV BANK,R4
MOV #-40,R0

8877 030374 072427 000011

8878 030400 042704 170037

8879 030404 012700 177740

8880 030410 012701 060000

8881 030414 162701 004000

8882 030420 105037 002264

8883 030424 005037 002314

8884 030430 104475

8885 030432 105237 002264

8886 030436 062700 000040

8887 030442 062701 004000

8888 030446 012737 001340 002314

8889 030454 104475

8890 030456 005011

8891 030460 104503

8892 030462 005711

8893 030464 104426

8894 030466 013705 002150

8895 030472 042705 170037

8896 030476 010002

8897 030500 060402

8898 030502 000240

8899 030504 020205

8900 030510 010237 002052

8901 030514 010537 002044

8902 030520 104455

8903 030522

8904 030522 005011

8905 030524 104475

8906 030526

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
B45:;;;;;
INCB PASFLG ;INC LOOP COUNTER
LET R0 := R0 + #40 ;INC CSR ADDRESS TO BE EXPECTED
ADD #40,R0
LET R1 := R1 + #4000
ADD #4000,R1
LET CHECK := #1340 ;DOUBLE ERROR CHECK BITS
MOV #1340,CHECK
CB1CSR ;WRITE DOUBLE ERROR CHECK BITS
LET (R1) := #0 ;WRITE DATA AND D.E. CHK BITS AT A=0
CLR (R1)
CLR1CSR ;CLEAR CSR
TST (R1) ;READ ADDRESS TO GET DOUBLE ERROR
READCSR ;READ CSR FOR CORRECT ADDRESS
LET R5 := CSR
MOV CSR,R5
BIC #+C7740,R5 ;
LET R2 := R0 ;GET CORRECT ADDRESS
MOV R0,R2
ADD R4,R2 ;ADD STARTING BANK TO DOUBLE BIT ADDRESS
NOP ;DEBUG AIDE
IF R2 NE R5 ;DO ADDRESSES AGREE?
CMP R2,R5
BEQ L267
LET BAD := R2 ;
MOV R2,BAD
LET GOOD := R5 ;
MOV R5,GOOD
PERR31 ;NO ERROR
END ;
L267:;;;;;
LET (R1) := #0 ;
CLR (R1)
CB1CSR ;ENABLE CHECK/SYNDROME BIT REGISTER
UNTILB PASFLG EQ PASCNT ;DO 16K AT A TIME

E4

030526 123737 002264 002570
030534 001336
030536
8907 030536 104503
8908 030540 000207
8909

CLR1CSR
RETURN

:
:

CMPB PASFLG,PASCNT
BNE B45
E45: :::::

```

8912 030542      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.
;
8913
8914
8915
8916
8917
8918 030542      104424      CACHOFF      ;TURN OFF CACHE MEMORY
8919 030544      013704      002102      LET R4 := BANK      ;GET BANK NUMBER TO FIGURE OUT EXTENDED ADDRESS
;                               MOV BANK,R4
8920 030550      023727      002102      000177      IF BANK EQ #177
;                               CMP BANK,#177
;                               BNE L271
8921 030556      001004
;                               LET PASCNT := #1      ;DO ONCE IN LAST BANK
;                               MOV #1,PASCNT
8922 030560      012737      000001      002570      ELSE
;                               BR L272
;                               L271:::
8923 030570      000403
;                               LET PASCNT := #2
;                               MOV #2,PASCNT
8924 030576      012737      000002      002570      END
;                               L272:::
8925 030576      042704      177607      BIC #+C170,R4      ;CLEAR OFF LOWER BITS
8926 030602      072427      000002      ASH #2,R4          ;SHIFT TO LINE UP WITH CSR
8927 030606      052704      040000      BIS #BIT14,R4      ;SET EXTENDED ADDRESS BIT
8928 030612      062737      000400      172352      ADD #400,KIPARS    ;SET UP PAR TO POINT TO TOP OF A BANK
8929 030620      105037      002264      LET PASFLG := #0   ;INIT LOOP COUNTER
;                               CLR B PASFLG
8930 030624      013705      002102      LET R5 := BANK     ;R5 GETS THE BANK NUMBER
;                               MOV BANK,R5
8931 030630      042705      177770      BIC #+C7,R5        ;CLEAR ALL BUT THE LOWER BITS
8932 030634      072527      000011      ASH #9.,R5         ;ROTATE INTO POSTION
8933 030640      052705      000020      BIS #BIT4,R5       ;SET UP SBE INDICATOR ;;DATA TO BE EXPECTED
8934 030644      104513      CBREG           ;ENABLE CHECK/SYNDROME BIT REGISTER
8935 030646      105237      002264      REPEAT
;                               B46:::
8936 030652      012737      000104      002150      INCB PASFLG       ;INCR LOOP COUNTER
8937 030652      104425      000104      002150      LET CSR := #104   ;WRITE CHECK BITS TO CSR WITH DIAG MODE
;                               MOV #104,CSR
8938 030660      005011      LOADCSR        ;LOAD CSR WITH DATA
8939 030662      104503      LET (R1) := #0 ;WRT ZEROS AT A=0 AND SINGLE ERROR BITS
;                               CLR (R1)
8940 030664      005711      CLR1CSR        ;CLEAR CSR
8941 030670      104426      TST (R1)       ;READ A=0;DATA BIT 0 SHOULD BE CORRECTED TO A 1
8942 030672      042737      READCSR        ;READ CSR FOR DATA
8943 030700      020000      002150      BIC #BIT13,CSR    ;CLEAR POSSIBLE INHIBIT MODE IN DATA "CSR"
8944 030700      023705      002150      IF CSR NE R5 THEN ;HAS SINGLE ERROR BITS SET IN CSR?
;                               CMP CSR,R5
;                               BEQ L273
8945 030706      001406      LET BAD := CSR  ;
;                               MOV CSR,BAD
8946 030714      013737      002150      002052      LET GOOD := R5  ;
;                               MOV R5,GOOD
8947 030720      010537      002044      ERROR +23      ;
8948 030722      104023      END            ;

```



```

      030722
8949 030722 012737 040000 002150      LET CSR := #40000      ;WRITE EQB BIT TO CSR      L273:;;;;;
      030722 104425                                ;                                MOV #40000,CSR
8950 030730 104425                                ;                                ;
8951 030732 104426                                ;                                ;
8952 030734 042737 020000 002150      LOADCSR                ;READ FOR CORRECT EXTENDED Q-BUS ADDRESS
8953 030742 042737 020000 002150      READCSR                ;CLEAR INHIBIT MODE POINTER IN DATA
      030742 023704 002150                        BIC #BIT13,CSR        ;READ EQB ADDRESS
      030746 001411                                ;                                ;
8954 030750 013737 002150 002052      LET BAD := CSR        ;                                ;                                CMP CSR,R4
      030750 013737 002150 002052      ;                                ;                                BEQ L274
8955 030756 010437 002044                                ;                                ;                                MOV CSR,BAD
      030756 010437 002044                                ;                                ;                                MOV R4,GOOD
8956 030762 012737 177777 002612      SET HEADER            ;                                ;                                MOV #-1,HEADER
8957 030770 104023                                ;                                ;                                ;
8958 030772 104023                                ;                                ;                                ;
      030772                                ;                                ;                                ;
8959 030772 005011                                ;                                ;                                ;
      030772 005011                                ;                                ;                                ;
8960 030774 012701 137776                                ;                                ;                                ;
      030774 012701 137776                                ;                                ;                                ;
8961 031000 062705 000740                                ;                                ;                                ;
8962 031004 052704 001000                                ;                                ;                                ;
8963 031010 104513                                ;                                ;                                ;
8964 031012 123737 002264 002570      UNTILB PASFLG EQ PASCNT ;LOOP 2 TIMES                                ;                                ;
      031012 123737 002264 002570                                ;                                ;                                ;
      031020 001312                                ;                                ;                                ;
      031022                                ;                                ;                                ;
8965 031022 104503                                ;                                ;                                ;
8966 031024 104423                                ;                                ;                                ;
8967 031026 000207                                ;                                ;                                ;
      CLR1CSR                ;CLEAR CSR                                ;                                ;
      CACHON                 ;TURN ON CACHE                                ;                                ;
      RETURN

```


8969 031030

MTP043: SUBTST <<MTP043 WRITE BYTE CLEARS SINGLE BIT ERROR TEST>>

;SUBTEST MTP043 WRITE BYTE CLEARS SINGLE BIT ERROR TEST

8970
8971
8972
8973
8974
8975 031030 104424
8976 031032 104513
8977 031034 105037 002264
8978 031040
031040 010102
031042 005202
8979 031044
031044 012704 000001
8980 031050
031050
8981 031050 105237 002264
8982 031054
031054 012737 000604 002150
8983 031062 104425
8984 031064
031064 010411
8985 031066 104503
8986 031070
031070 112712 000377
8987 031074 104426
8988 031076 042737 177757 002150
8989 031104
031104 023727 002150 000020
031112 001407
8990 031114
031114 012737 000020 002044
8991 031122
031122 013737 002150 002052
8992 031130 104060
8993 031132
031132
8994 031132 104513
8995 031134 005711
8996 031136 104426
8997 031140 042737 174037 002150
8998 031146
031146 023727 002150 000300
031154 001412
8999 031156
031156 012737 177777 002612
9000 031164
031164 012737 000300 002044
9001 031172
031172 013737 002150 002052
9002 031200 104061
9003 031202
031202
9004 031202 005302

;
; THIS TEST CHECKS TO SEE IF A SINGLE BIT ERROR WILL BE CORRECTED DURING
; THE READ PORTION OF A WRITE BYTE AND THAT THE CORRECT CHECK BITS WILL
; BE GENERATED ON A WRITE.
;
CACHOFF ;TURN OFF CACHE
CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
CLRB PASFLG ;CLEAR LOOP COUNTER
LET R2 := R1 + #1 ;R2 POINTS TO HIGH BYTE

MOV R1,R2
INC R2

LET R4 := #1 ;INITIAL DATA = 1
MOV #1,R4

REPEAT ;
; B47:::;;
INCB PASFLG ;INCREMENT LOOP COUNTER
LET CSR := #604 ;WRITE CHECK BITS CORRESPONDING TO DATA OF 0
MOV #604,CSR

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

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

READCSR ;READ CSR
BIC #+C20,CSR ;SEE IF SBE INDICATOR IS SET
IF CSR NE #20 ;IS SBE SET?????
CMP CSR,#20
BEQ L276

LET GOOD := #20 ;
MOV #20,GOOD

LET BAD := CSR ;
MOV CSR,BAD

ERROR +60 ;
END ;
; L276:::;;
CBREG ;WRITE CSR TO DIAG MODE
TST (R1) ;READ SAO FOR CORRECT CHECK BITS
READCSR ;READ CSR
BIC #+C3740,CSR ;MASK OUT CHECK BIT FIELD
IF CSR NE #300 ;WERE CORRECT CHECK BITS GENERATED?????
CMP CSR,#300
BEQ L277

SET HEADER ;
MOV #-1,HEADER

LET GOOD := #300 ;
MOV #300,GOOD

LET BAD := CSR ;
MOV CSR,BAD

ERROR +61 ;
END ;
; L277:::;;
DEC R2 ;POINT TO HIGH BYTE AND REPEAT

```
9005 031204          LET R4 := #400          ;BIT 0 OF HIGH BYTE
      031204 012704 000400
9006 031210          UNTILB PASFLG EQ #2      ;DO HIGH AND LOW BYTE
      031210 123727 002264 000002
      031216 001314
      031220
9007 031220          CACHON                    ;TURN ON CACHE
9008 031222          RETURN                    ;
                                     E47:;;;;;
```

```

9010 031224 MTP044: SUBTST <<MTP044 SHIFTING CHECK BITS THROUGH THE CSR TEST>>
;*****
;#SUBTEST MTP044 SHIFTING CHECK BITS THROUGH THE CSR TEST
;*****
; THIS TEST CHECKS THE ABILITY TO READ AND WRITE CHECKBITS INTO MEMORY
; BY SHIFTING A ONE BIT THROUGH A FIELD OF ZEROS. THE CSR IS READ FOR THE
; CORRECT PATTERNS. THE TEST IS THEN REPEATED ON A ZERO BIT THROUGH A
; FIELD OF ALL ONES.
;
; REPEAT ;ILC;;REV B
;***** B50:;;;;;
9018 031224 104424 CACHOFF ;TURN OFF CACHE
9019 031226 105037 002264 LET PASFLG :B= #0 ;INIT PASFLG
9020 031232 012705 174037 LET R5 := #174037 ;CHECK BIT MASK FOR CSR CLR B PASFLG
9021 031236 104475 CB1CSR ;ENABLE CHECK/SYNDROME BIT REGISTER MOV #174037,R5
9022 031240 012702 000046 LET R2 := #46 ;SET UP INITIAL CSR DATA
;***** MOV #46,R2
9023 031244 REPEAT
;***** B51:;;;;;
9024 031244 105237 002264 LET PASFLG :B= PASFLG + #1 ;INC LOOP COUNTER INCB PASFLG
;*****
9025 ;CHK BITS = 1
9026 ;DISABLE ECC;DIAG CHK SET
9027 031250 005037 002266 LET PASSNO := #0 ;INIT PASSNO(INNER LOOP COUNTER) CLR PASSNO
;*****
9028 031254 REPEAT
;***** B52:;;;;;
9029 031254 005237 002266 LET PASSNO := PASSNO + #1 ;INC LOOP COUNTER INC PASSNO
9030 031260 010204 LET R4 := R2 ;COPY R2 TO R4
9031 031262 010237 002150 LET CSR := R2 ;GET CSR DATA TO BE WRITTEN MOV R2,R4
;***** MOV R2,CSR
9032 031266 104425 LOADCSR ;WRITE SBE CHECK BITS TO CSR
9033 031270 005011 LET (R1) := #0 ;WRITE DATA AND CHECK BITS AT A=0 CLR (R1)
;*****
9034 031272 005105 COM R5 ;COMPLEMENT MASK
9035 031274 010546 MOV R5,-(SP) ;SAVE R5 ON STACK
9036 031276 040416 BIC R4,(SP) ;CREATE AN XOR FUNCTION
9037 031300 040504 BIC R5,R4
9038 031302 052604 BIS (SP)+,R4
9039 031304 010437 002150 LET CSR := R4
;*****
9040 031310 104425 LOADCSR ;LOAD CSR WITH COMPLEMENT CHECK BITS MOV R4,CSR
9041 031312 104426 READCSR ;READ CSR FOR COMPLEMENT CHECK BITS
9042 031314 013703 002150 LET R3 := CSR ;COPY CSR DATA TO R3
;***** MOV CSR,R3
9043 031320 042703 020000 BIC #BIT13,R3 ;CLEAR ANY POSSIBLE INHIBIT MODE POINTER
9044 031324 020304 IF R3 NE R4 THEN ;READ CSR FOR PROPER CHECK BITS
;***** CMP R3,R4
;***** BEQ L301
9045 031330 010137 002034 LET ADDRESS := R1 ;
;***** MOV R1,ADDRESS
9046 031334 LET GOOD := R4 ;

```



```

9047 031334 010437 002044          LET BAD := R3          ;          MOV R4,GOOD
9048 031340 010337 002052          SET HEADER            ;          MOV R3,BAD
9049 031344 012737 177777 002612  PERR37                ;ERROR CALL          MOV #-1,HEADER
9050 031352 104463                END                    ;                    ;:ILC::REVB
9051 031354 005105                COM R5                ;                    L301:::
9052 031356 005711                TST (R1)              ;COMPLEMENT MASK
9053 031360 000240                NOP                   ;READ CHECK BITS AT A=0 INTO CSR
9054 031362 104426                READCSR               ;READ CSR FOR CORRECT CHECK BITS
9055 031364 040537 002150                BIC R5,CSR            ;MASK OUT CHECK BIT FIELD
9056 031370 010204                LET R4 := R2          ;GET CHECK BITS THAT WERE WRITTEN
9057 031372 040504                BIC R5,R4             ;MAYBE CHECK BIT FIELD
9058 031374 020437 002150                IF R4 NE CSR          ;ARE CHECK BITS THE SAME?          MOV R2,R4
9059 031402 010437 002044          LET GOOD := R4        ;          CMP R4,CSR
9060 031406 013737 002150 002052    LET BAD := CSR        ;          BEQ L302
9061 031414 010137 002034          LET ADDRESS := R1    ;          MOV R4,GOOD
9062 031420 012737 177777 002612    SET HEADER            ;          MOV CSR,BAD
9063 031426 104464                PERR40                ;          MOV R1,ADDRESS
9064 031430 010102                END                    ;ERROR CALL          MOV #-1,HEADER
9065 031430 040502                BIC R5,R2             ;                    ;:ILC::REVB
9066 031432 123727 002264 000001    IFB PASFLG EQ #1     ;                    L302:::
9067 031442 006302                ASL R2                ;SHIFT CHECK BITS AND CREATE NEW DATA FOR CSR
9068 031444 000413                ELSE                  ;SELECT FUNCTION
9069 031446 005105                COM R5                ;                    CMPB PASFLG,#1
9070 031450 010546                MOV R5,-(SP)          ;                    BNE L303
9071 031452 040216                BIC R2,(SP)           ;DO A FIELD OF ZEROS--->ONES
9072 031454 040502                BIC R5,R2             ;SHIFT CHECK BITS
9073 031456 052602                BIS (SP)+,R2         ;DO A FIELD OF ONES --->ZEROS
9074 031460 006302                ASL R2                ;                    BR L304
9075 031462 010546                MOV R5,-(SP)          ;                    L303:::
9076 031464 040216                BIC R2,(SP)           ;TAKE OUT CHECK BIT FIELD
9077 031466 040502                BIC R5,R2             ;
9078 031470 052602                BIS (SP)+,R2         ;
9079 031472 005105                COM R5                ;SHIFT CHECK BITS
9080 031474 010546                MOV R5,-(SP)          ;PUT BACK CHECK BIT FIELD
9081 031474 040216                BIC R2,(SP)           ;
9082 031474 062702 000006          LET R2 := R2 + #6    ;                    ;
9083 031500 123727 002266 000006    UNTILB PASSNO EQ #6 ;                    ;COMPLEMENT DATA PATTERN
9083 031506 001262                ;                    L304:::
                                ;ADD 6 SO THAT WRITE ON CSR WILL ENABLE DIAG MODE
                                ADD #6,R2
                                ;DO ALL CHECK BITS
                                CMPB PASSNO,#6
                                BNE B52

```



```

9084 031510          LET R2 := #3706          ;REPEAT WITH FIELD OF ONES
031510 012702 003706          ;E2:;
9085 031514          UNTILB PASFLG EQ #2      ;MOV #3706,R2
031514 123727 002264 000002          ;CMPB PASFLG,#2
031522 001250          ;BNE B51
031524          ;E1:;
9086 031524 104503          CLR1CSR          ;
9087 031526 005011          CLR (R1)          ;
9088 031530 062701 000100          ADD #100,R1          ;ALL 256 TO ADDRESS
9089 031534          UNTIL R1 EQ ENDADD      ;
031534 020137 002562          ;CMP R1,ENDADD
031540 001231          ;BNE B50
031542          ;E0:;
9090 031542 104423          CACHON          ;TURN ON CACHE
9091 031544 000207          RETURN          ;

```

```

9093 031546      MTP045: SUBST  <<MTP045      SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST>>
;*****
;*SUBTEST      MTP045  SYNDROMES TO CSR ON DOUBLE BIT ERROR TEST
;*****
9094      ;
9095      ; THIS TEST CHECKS TO SEE IF THE DOUBLE BIT ERROR INDICATOR IS SET
9096      ; ON A DOUBLE BIT ERROR AND THE CORRECT SYNDROMES ARE LATCHED INTO THE
9097      ; CSR. THIS TEST IS THEN REPEATED WITH MULTIPLE ERROR CHECK/SYNDROME BITS
9098      ;
9099 031546 104424 CACHOFF      ;TURN OFF CACHE
9100 031550 104513 CBREG      ;ENABLE CHECK/SYNDROME BIT REGISTER
9101 031552      LET PASSNO := #0      ;CLEAR LOOP COUNTER
9102 031552 005037 002266      CLR PASSNO
9103 031556 012737 003744 002044      LET GOOD := #3744      ;GOOD DATA
9104 031564 012737 003144 002150      LET CSR := #3144      ;DBE CHECK BITS FOR CSR
9105 031572      REPEAT      ;
9106 031572 005237 002266      INC PASSNO      ;
9107 031576 104425      LOADCSR      ;WRITE DBE CHECK BITS TO CSR
9108 031600 005011      LET (R1) := #0      ;WRITE ZEROS AND DBL ERROR CHK BITS A=0
9109 031602 104503      CLR1CSR      ;CLEAR CSR OUT
9110 031604 005711      TST (R1)      ;READ A=0 TO GET DOUBLE BIT ERROR
9111 031606 104426      READCSR      ;WAS UNCORRECTABLE ERROR BIT SET??
9112 031610      IF #BIT15 OFF.IN CSR ;
9113 031610 032737 100000 002150      BIT #BIT15,CSR
9114 031616 001007      BNE L310
9115 031620      SET HEADER      ;
9116 031620 012737 177777 002612      LET BAD := CSR      ;
9117 031626 013737 002150 002052      ERROR +63      ;BIT NOT SET
9118 031634 104063      END      ;
9119 031636      SYNREG      ;ENABLE SYNDROME BIT REGISTER
9120 031636 104514      READCSR      ;READ CSR FOR CORRECT SYNDROME BITS
9121 031640 104426      NOP      ;DEBUG AIDE
9122 031642 000240      BIC #+C3744,CSR      ;MASK SYNDROMES OUT
9123 031644 042737 174033 002150      IF CSR NE GOOD THEN ;CHECK IF DOUBLE ERROR BIT IS SET
9124 031652 023737 002150 002044      CMP CSR,GOOD
9125 031660 001407      BEQ L311
9126 031662      LET BAD := CSR      ;BAD DATA
9127 031662 013737 002150 002052      SET HEADER      ;
9128 031670 012737 177777 002612      ERROR +42      ;
9129 031676 104042      END      ;
9130 031700      CLR (R1)      ;CLEAR LUT
9131 031700 005011      LET GOOD := #3604      ;REPEAT WITH MULTIPLE ERROR SYNDROMES
9132 031702 012737 003604 002044      LET CSR := #3004      ;MULTIPLE ERROR CHECK BITS
9133 031710 012737 003004 002150      UNTIL PASSNO EQ #2      ;
9134 031716 023727 002266 000002      CMP PASSNO,#2

```

031724 001322
031726
9129 031726 104503
9130 031730 104423
9131 031732 000207

CLR1CSR
CACHON
RETURN

BNE B53
E53:.....

9133 031734

MTP046: SUBTST <<MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED>>

;SUBTEST MTP046 CHECK SINGLE BIT ERRORS WITH ECC DISABLED

9134
9135
9136
9137
9138

; 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.

9139 031734 005037 002266
9140 031740 104424
9141 031742
031742

CLR PASSNO ;CLEAR OUTER LOOP COUNTER
CACHOFF ;TURN OFF CACHE
REPEAT ;

9142 031742
031742 005237 002266

LET PASSNO := PASSNO + #1 ; B54:;;;;;
; INC PASSNO

9143 031746 005000
9144 031750 105037 002264
9145 031754 104513
9146 031756
031756

CLR R0 ;CLEAR DATA
CLRB PASFLG ;CLEAR PASFLG
CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
REPEAT ;

9147 031756
031756 105237 002264

LET PASFLG :B= PASFLG + #1 ;INCREMENT LOOP COUNTER
; INCB PASFLG

9148 031762
031762 012704 177777

LET R4 := #-1 ;INDEX TO SINGLE BIT ERROR TABLE
; MOV #-1,R4

9149 031766
031766 012737 000001 002076

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

9150 031774
031774 005037 002324

LET BITNO := #0 ;CLEAR INNER LOOP COUNTER
; CLR BITNO

9151 032000
032000 123727 002264 000001
032006 001003

IFB PASFLG EQ #1 ;SELECT DATA TO BE CORRECTED BY PASSNO
; CMPB PASFLG,#1

9152 032010
032010 012705 000001

LET R5 := #1 ;DATA=0;BIT TO BE CORRECTED IS A ONE
; BNE L313

9153 032014
032014 000402
032016

ELSE ;
; BR L314

9154 032016
032016 012705 177776

LET R5 := #177776 ;DATA=177776;BIT TO BE CORRECTED IS A ZERO
; MOV #177776,R5

9155 032022
032022

END ;

9156 032022
032022

REPEAT ; L314:;;;;;
; B56:;;;;;

9157 032022
032022 005037 002072

LET PARCNT := #0 ;CLEAR PARITY COUNTER
; CLR PARCNT

9158 032026 005237 002324
9159 032032
032032 005204

INC BITNO ;INCREMENT BIT POINTER
LET R4 := R4 + #1 ;POINT TO NEXT SET OF CHECK BITS
; INC R4

9160 032034
032034 116402 030276

LET R2 :B= PTABLE(R4) ;GET NEXT SET OF CHECK BITS
; MOV B PTABLE(R4),R2

9161 032040 072227 000005
9162 032044 052702 000006

ASH #5,R2 ;SHIFT TO LINE UP IN CSR
BIS #BIT2!BIT1,R2 ;ENABLE DIAG MODE

9163 032050
032050 010237 002150

LET CSR := R2 ;GET CHECK BITS TO BE WRITTEN
; MOV R2,CSR

9164 032054 104425
9165 032056
032056 010011

LOADCSR ;LOAD CSR WITH DATA
LET (R1) := R0 ;WRITE DATA TO TEST ADDRESS
; MOV R0,(R1)

9166 032060

IF PASSNO EQ #1 ;WRITE CSR


```

032060 023727 002266 000001                                CMP PASSNO,#1
032066 001002                                BNE L315
9167 032070 104471                                ECC1DIS ;FIRST PASS ;ECC DISABLE,NO PBL
9168 032072 000401                                ELSE ;
032074 000401                                BR L316
032074 104507                                L315:;;;;;
9169 032074 104507                                ENA1SBE ;SECOND PASS ;ECC DISABLE,PBL ENABLED
9170 032076 000401                                END ;
032076 005711                                L316:;;;;;
9171 032076 004737 032220                                TST (R1) ;CORRECT SBE
9172 032100 004737 032220                                CALL CHKTRP ;CHECK FOR CORRECT TRAP
9173 032104 104426                                READCSR ;READ THE CSR FOR UNCORRECTABLE ERROR
9174 032106 032737 100000 002150                                IF #BIT15 OFF.IN CSR ;IS UNCORRECTABLE ERROR BIT SET????
032106 001007                                BIT #BIT15,CSR
032114 001007                                BNE L317
9175 032116 013737 002150 002052                                LET BAD := CSR ;
032116 012737 177777 002612                                SET HEADER ;
9176 032124 012737 177777 002612                                ERROR +45 ;
9177 032132 104045                                END ;
032134 104503                                CLR1CSR ;
9179 032134 005011                                CLR (R1) ;CLEAR LUT
9180 032136 005011                                IFB PASFLG EQ #1 ;SHIFT NEW DATA DEPENDING ON PASFLG
032140 123727 002264 000001                                CMPB PASFLG,#1
032146 001002                                BNE L320
9182 032150 006305                                ASL R5 ;SHIFT BITNO TO THE LEFT
9183 032152 000402                                ELSE ;
032154 000402                                BR L321
032154 000261                                L320:;;;;;
9184 032154 000261                                SEC ;SET CARRY BIT AND.....
9185 032156 006105                                ROL R5 ;ROTATE LEFT
9186 032160 000402                                END ;
032160 000402                                UNTIL BITNO EQ #16. ;UNTIL ALL BITS ARE DONE
9187 032160 023727 002324 000020                                CMP BITNO,#16.
032166 001315                                BNE B56
032170 005100                                COM R0 ;COMPLEMENT DATA AND REPEAT
9188 032170 005100                                UNTILB PASFLG EQ #2 ;UNTIL 2 PASSES ARE COMPLETE!
9189 032172 123727 002264 000002                                CMPB PASFLG,#2
032172 001266                                BNE B55
032200 001266                                E55:;;;;;
9190 032202 023727 002266 000002                                UNTIL PASSNO EQ #2 ;
032202 001254                                CMP PASSNO,#2
032210 001254                                BNE B54
032212 104503                                CLR1CSR ;
9191 032212 104423                                CACHON ;TURN CACHE
9192 032214 000207                                RETURN ;
9193 032216 000207                                RETURN ;
9194
9195
9196 032220 023727 002266 000001                                CHKTRP: IF PASSNO EQ #1 ;PASS 1 CHECK FOR NO TRAP
032220 001011                                CMP PASSNO,#1
032226 001011                                BNE L325
9197 032230                                IF PARCNT EQ #1 ;

```


9209 032274

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

;#SUBTEST MTP047 NO CSR UPDATE ON SBE WITH EXSISTING DBE

9210
9211
9212
9213
9214

;
; THIS TEST CHECKS TO SEE THAT THE CSR CONTENTS WILL NOT CHANGE
; WITH A SINGLE BIT ERROR WHEN A DOUBLE BIT ERROR ALREADY
; EXISTS.

9215 032274 104424
9216 032276 013704 002102
9217 032302 072427 000011
9218 032306 042704 170037
9219 032312 052704 100000
9220 032316 104513
9221 032320 012737 003144 002150
9222 032326 104425
9223 032330 005011
9224 032332 012737 000104 002150
9225 032340 104425
9226 032342 005012
9227 032344 104503
9228 032346 005711
9229 032350 104426
9230 032352 042737 020000 002150
9231 032360 023704 002150
9232 032366 013737 002150 002052
9233 032374 010437 002044
9234 032400 012737 177777 002612
9235 032406 104063
9236 032410 052704 000020
9237 032410 005712
9238 032414 104426
9239 032416 042737 020000 002150
9240 032420 023704 002150
9241 032426 001411
9242 032434 013737 002150 002052
9243 032442 010437 002044
9244 032446 012737 177777 002612
9245 032454 104051
9246 032456

;
; CACHOFF ;TURN OFF CACHE
LET R4 := BANK ;GET BANK NUMBER
MOV BANK,R4
ASH #9,R4 ;SHIFT INTO PLACE
BIC #+C7740,R4 ;MASK OUT UNWANTED BITS
BIS #BIT15,R4 ;SET UP GOOD DATA
CBREG ;ENABLE CHECK/SYNDROME BIT REGISTER
LET CSR := #3144 ;CHECK BITS FOR DOUBLE BIT ERROR
MOV #3144,CSR
LOADCSR ;
LET (R1) := #0 ;WRITE DBE CHECK BITS
CLR (R1)
LET CSR := #104 ;WRITE SBE CHECK BITS
MOV #104,CSR
LOADCSR ;
LET (R2) := #0 ;WRITE SBE CHECK BITS AT ADDRESS + 4K
CLR (R2)
CLR1CSR ;CLEAR CSR
TST (R1) ;READ DBE LOCATION
READCSR ;READ FOR CSR DBE INDICATOR
BIC #BIT13,CSR ;CLEAR INHIBIT MODE POINTER
IF CSR NE R4 ;
CMP CSR,R4
BEQ L331
LET BAD := CSR ;
MOV CSR,BAD
LET GOOD := R4 ;
MOV R4,GOOD
SET HEADER ;
MOV #-1,HEADER
ERROR +63 ;
END ;
BIS #20,R4 ;SET BIT IN GOOD DATA L331:::;;
TST (R2) ;READ SBE
READCSR ;READ CSR FOR NO CHANGE
BIC #BIT13,CSR ;CLEAR INHIBIT MODE POINTER
IF CSR NE R4 ;
CMP CSR,R4
BEQ L332
LET BAD := CSR ;
MOV CSR,BAD
LET GOOD := R4 ;
MOV R4,GOOD
SET HEADER ;
MOV #-1,HEADER
ERROR +51 ;
END ;

032456
9247 032456 104503
9248 032460 005011
9249 032462 005012
9250 032464 104423
9251 032466 000207

CLR1CSR
CLR (R1)
CLR (R2)
CACHON
RETURN

L332:::~::~
:CLEAR 1 CSR
:
:
:TURN ON CACHE

.SBTTL MISC SUBROUTINES

9253
9254
9255 032470

REGCOPY:SUBTST <<SUBR COPY R0 TO R4,R1 TO R3, & R2 TO R5>>
;*****
;*SUBTEST SUBR COPY R0 TO R4,R1 TO R3, & R2 TO R5
;*****

9256 032470 010004
9257 032472 010103
9258 032474 010205
9259 032476 000207
9260
9261 032500

MOV R0,R4
MOV R1,R3
MOV R2,R5
RETURN

FLIPWARN:SUBTST <<FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS>>
;*****
;*SUBTEST FLIP WARNING CONSTANTS IN WORST CASE NOISE TESTS
;*****

9262 032500
032500 010046
9263 032502 005237 002616
9264 032506 042737 177774 002616
9265 032514 022737 000001 002616
9266 032522 001414
9267 032524 022737 000002 002616
9268 032532 001413
9269 032534 022737 000003 002616
9270 032542 001414
9271 032544 005000
9272 032546 013704 002614
9273 032552 000414
9274 032554
032554 005000
032556 005004
9275 032560 000411
9276 032562 012700 000401
9277 032566 013704 002614
9278 032572 000404
9279 032574 012700 000401
9280 032600 012704 000401
9281 032604 010037 024762
9282 032610 010037 024776
9283 032614 010037 025022
9284 032620 010037 025036
9285 032624
032624 012600
9286 032626 000207

PUSH R0
MOV R0,-(SP)
INC FLIPLOC
BIC #+C3,FLIPLOC
CMP #1,FLIPLOC
BEQ 1\$
CMP #2,FLIPLOC
BEQ 2\$
CMP #3,FLIPLOC
BEQ 3\$
CLR R0
MOV ONES,R4
BR 4\$
1\$: CLEAR R0,R4
CLR R0
CLR R4
BR 4\$
2\$: MOV #401,R0
MOV ONES,R4
BR 4\$
3\$: MOV #401,R0
MOV #401,R4
4\$: MOV R0,WARN2
MOV R0,WARN3
MOV R0,WARN4
MOV R0,WARN5
POP R0
MOV (SP)+,R0
RETURN

9288 032630

```

BACKGND:SUBTST <<SUBR WRITE BACKGROUND>>
;*****
;*SUBTEST      SUBR      WRITE BACKGROUND
;*****

```

9289

```

9290 032630 104415
9291 032632 012700 060000
9292 032636 012701 040000
9293 032642 012737 000207 024644
9294 032650 012737 024640 002262
9295 032656 004737 024446
9296 032662 012737 000240 024644
9297 032670 104416
9298 032672 000207
9299

```

```

;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

```

9301 032674

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

9302
9303
9304
9305
9306 032674 013702 002104
9307 032700 016203 002664
9308 032704 000303
9309 032706 006303
9310 032710 042703 177741
9311 032714 010337 002152
9312 032720 000207

;INPUTS : NONE
;
;OUTPUT : CSRNO = CSR NUMBER
;
MOV BANKINDEX,R2 ;GET INDEX INTO CONFIG TABLE
MOV CONFIG(R2),R3 ;MOV IT INTO R3
SWAB R3
ASL R3
BIC #+C36,R3 ;CLEAR OFF SOME BITS
MOV R3,CSRNO ;SAVE CSR NUMBER
RETURN

9315 032722

PCONFIG:SUBTST <<SUBR PRINT CONFIGURATION MAP>>

;SUBTEST SUBR PRINT CONFIGURATION MAP

9316 032722

032722 013746 000060

032726 013746 000062

032732 010046

9317 032734

010637 033334

MOV SP,PCONFS

;SAVE LAST GOOD SP

MOV TKVEC,-(SP)

MOV TKVEC+2,-(SP)

MOV RO,-(SP)

9318 032740

012737 033302 000060

MOV #PCONF2,TKVEC

9319 032746

012737 000340 000062

MOV #340,TKVEC+2

9320 032754

017700 147664

MOV @TKB,RO

;KILL ANY OLD INTERRUPT

9321 032760

042737 000200 177776

BIC #BIT7,PSW

;LOWER CPU PRIORITY TO 140

9322 032766

052777 000100 147646

BIS #BIT6,@TKS

;ENABLE KEYBOARD INTERRUPTS

9323

9324 032774

032774 104401 065123

TYPE MSG001

TYPEIT ,MSG001

.DSABL CRF

9325 033000

033000 104401 065205

TYPE MSG002

TYPEIT ,MSG002

.DSABL CRF

9326 033004

033004 104401 065262

TYPE MSG003

TYPEIT ,MSG003

.DSABL CRF

9327 033010

022737 000060 002556

CMP #60,LASTBANK

9328 033016

002006

BGE NOOJ

9329

9330 033020

033020 032777 000020 147610

;IF FAT PAPER ON TERMINAL GOTO 1\$

IF #SW4 SET.IN @SWR THEN JUMPTO PCONF1

BIT #SW4,@SWR

BEQ L333

JMP PCONF1

L333:::::;

9331 033034

012700 000074

NOOJ:

MOV #60.,RO

9332 033040

010004

MOV RO,R4

9333 033042

033042 005001

CLEAR R1,R3

CLR R1

CLP R3

9334 033046

033046 104401 065367

TYPE MSG004

TYPEIT ,MSG004

.DSABL CRF

9335 033052

004737 033336 002556

CALL TCONFIG

;GO TYPE CONFIGURATION (1ST HALF)

9336 033056

022737

CMP #60,LASTBANK

9337 033064

002106

BGE PCONF2

9338 033066

033066 104401 002660

TYPE \$CRLF

TYPEIT , \$CRLF

.DSABL CRF

9339 033072

033072 104401 066076

TYPE MSG017

;PRINT SPACE(S)

TYPEIT ,MSG017

.DSABL CRF

9340 033076

033076 104401 065570

TYPE MSG11A

TYPEIT ,MSG11A

.DSABL CRF

9341 033102

033102 104401 002660

TYPE \$CRLF

TYPEIT , \$CRLF

.DSABL CRF

9342 033106

033106 104401 066076

TYPE MSG017

;PRINT SPACE(S)

TYPEIT ,MSG017

9343	033112				.DSABL	CRF		
	033112	104401	065656		TYPE	MSG011		
					TYPEIT	,MSG011		
9344	033116				.DSABL	CRF		
	033116	104401	002660		TYPE	,\$CRLF		
					TYPEIT	,\$CRLF		
9345	033122				.DSABL	CRF		
	033122	104401	066076		TYPE	MSG017		;PRINT SPACE(S)
					TYPEIT	,MSG017		
9346	033126				.DSABL	CRF		
	033126	104401	065744		TYPE	MSG012		
					TYPEIT	,MSG012		
9347	033132	012701	000360		.DSABL	CRF		
9348	033136	010103			MOV	#60.*2*2,R1		
9349	033140	004737	033336		MOV	R1,R3		
9350	033144	022737	000170	002556	CALL	TCONFIG		
9351	033152	002053		N00J1:	CMP	#170, LASTBANK		
9352	033154				BGE	PCONF2		
	033154	104401	002660		TYPE	,\$CRLF		
					TYPEIT	,\$CRLF		
9353	033160				.DSABL	CRF		
	033160	104401	066076		TYPE	MSG017		;PRINT SPACE(S)
					TYPEIT	,MSG017		
9354	033164				.DSABL	CRF		
	033164	104401	066041		TYPE	MSG11B		
					TYPEIT	,MSG11B		
9355	033170				.DSABL	CRF		
	033170	104401	002660		TYPE	,\$CRLF		
					TYPEIT	,\$CRLF		
9356	033174				.DSABL	CRF		
	033174	104401	066076		TYPE	MSG017		;PRINT SPACE(S)
					TYPEIT	,MSG017		
9357	033200				.DSABL	CRF		
	033200	104401	066043		TYPE	MSG11C		
					TYPEIT	,MSG11C		
9358	033204				.DSABL	CRF		
	033204	104401	002660		TYPE	,\$CRLF		
					TYPEIT	,\$CRLF		
9359	033210				.DSABL	CRF		
	033210	104401	066076		TYPE	MSG017		;PRINT SPACE(S)
					TYPEIT	,MSG017		
9360	033214				.DSABL	CRF		
	033214	104401	066045		TYPE	MSG11D		
					TYPEIT	,MSG11D		
9361	033220	012701	000740		.DSABL	CRF		
9362	033224	010103			MOV	#740,R1		
9363	033226	012700	000010		MOV	R1,R3		
9364	033232	010004			MOV	#8.,R0		
9365	033234	004737	033336		MOV	R0,R4		
9366	033240	000420			CALL	TCONFIG		
9367					BR	PCONF2		
9368	033242	012700	000170	PCONF1:	MOV	#120.,R0		
9369	033246	010004			MOV	R0,R4		
9370	033250				CLEAR	R1,R3		
	033250	005001						CLR R1
	033252	005003						CLR R3

```

9371 033254          TYPE      MSG014          ;SPACE
      033254 104401 066060      TYPEIT   ,MSG014
      .DSABL   CRF
9372 033260          TYPE      MSG011
      033260 104401 065656      TYPEIT   ,MSG011
      .DSABL   CRF
9373 033264          TYPE      MSG004
      033264 104401 065367      TYPEIT   ,MSG004
      .DSABL   CRF
9374 033270          TYPE      MSG012
      033270 104401 065744      TYPEIT   ,MSG012
      .DSABL   CRF
9375 033274 004737 033336      CALL     TCONFIG
9376 033300 000721          BR         NOOJ1
9377
9378 033302 013706 033334      PCONF2: MOV     PCONFS,SP          ;RESTORE STACK
9379 033306 042777 000100 147326 BIC     #BIT6,@TKS
9380 033314 117700 147324      MOVB   @TKB,RO          ;READ CHAR TO KILL FLAG
9381 033320          POP     RO,TKVEC+2,TKVEC
      033320 012600
      033322 012637 000062          MOV (SP)+,RO
      033326 012637 000060          MOV (SP)+,TKVEC+2
9382 033332 000207          RETURN          MOV (SP)+,TKVEC
9383
9384 033334 000000          PCONFS: 0          ;STACK SAVED HERE!

```

9387 033336

SUBTST <<SUBR TYPE CONFIGURATION>>

```

*****
;SUBTEST      SUBR      TYPE CONFIGURATION
*****
;CALL:  MOV      #N,R0          ;N=NUMBER OF CHARACTERS
;        MOV      R0,R4          ;BACKUP
;        MOV      #K,R1          ;INDEX CONSTANT
;        MOV      R1,R3          ;BACKUP
;        CALL     TCONFIG        ;ACTUAL CALL
;        RETURN                    ;ONLY RETURN
*****

```

9388
9389
9390
9391
9392
9393
9394
9395
9396
9397
9398
9399

```

*****
; ** ERROR **
*****
9400 033336 012737 000340 177776 TCONFIG:MOV #340,PSW ;DISABLE INTERUPTS
9401 033344 104401 065475          TYPE MSG005
033344          TYPEIT ,MSG005
          .DSABL CRF
9402 033350 032761 000001 002664 1$: BIT #BIT0,CONFIG(R1) ;ERROR ON THIS BANK?
9403 033356 001403          BEQ 2$ ;NO - SKIP
9404 033360          TYPE MSG013 ;PRINT "X"
033360 104401 066056          TYPEIT ,MSG013
          .DSABL CRF
9405 033364 000402          BR 3$
9406 033366          TYPE MSG014 ;PRINT SPACE
033366 104401 066060          TYPEIT ,MSG014
          .DSABL CRF
9407 033372 062701 000004          3$: ADD #4,R1 ;BUMP POINTER
9408 033376 077014          SOB R0,1$ ;LOOP UNTIL DONE
9409 033400 010400          MOV R4,R0
9410 033402 010301          MOV R3,R1

```

```

9413 ;*****
9414 ;** INTERLEAVE **
9415 ;*****
9416 ;THIS IS AN ENTRY POINT FROM ERROR REPORTS
9417 033404 012737 000340 177776 TCFIG1: MOV #340,PSW ;DISABLE INTERUPTS
9418 033412 112737 000040 066062 MOVB #' ,MSG015 ;MOVE A BLANK IN TO BE PRINTED
9419 033420
      033420 104401 066062
      TYPE MSG015
      TYPEIT ,MSG015
      .DSABL CRF
      IF NOTAB NE #0 THEN $RETURN
9420 033424
      033424 005737 002372
      033430 001401
      033432 000207
      033434
      TST NOTAB
      BEQ L334
      RTS PC
      L334:;;;;;
9421 033434 010400 MOV R4,R0
9422 033436 010301 MOV R3,R1
9423
9424 ;*****
9425 ;** MEMORY TYPE **
9426 ;*****
9427 .ENABL LSB
9428 033440
      033440 104401 065544
      TYPE MSG009
      TYPEIT ,MSG009
      .DSABL CRF
9429 033444 033761 002106 002664 TCFIG2: BIT CPUBIT,CONFIG(R1)
9430 033452 001432 BEQ 17$
9431 033454 016105 002666 MOV CONFIG+2(R1),R5
9432 033460 000305 SWAB R5 ;GET MEMORY TYPE
9433 033462 042705 177770 BIC #C7,R5 ;CLEAR NON INTERESTING BITS
9434 033466 020527 000003 CMP R5,#3 ;IS IT A LEGAL MEMORY TYPE
9435 033472 003022 BGT 17$ ;IF IF SO BRANCH!!!!!!
9436 033474
      033474 032705 000001 IF #BIT0 SET.IN R5 ;IS IT AN ECC MEMORY????
      033500 001413 BIT #BIT0,R5
9437 033502
      033502 032705 000002 IF #BIT1 SET.IN R5 ;IS IT A MSV11-P OR A MSV11-J???
      033506 001404 BIT #BIT1,R5
9438 033510 112737 000120 066062 MOVB #'P,MSG015 ;IT IS A MSV11-P
9439 033516
      033516 000403 ELSE ;
      033520 BR L337
9440 033520 112737 000105 066062 MOVB #'E,MSG015 ;IT IS A MSV11-J
9441 033526
      033526 END ;
      L336:;;;;;
9442 033526
      033526 000403 ELSE ;
      033530 BR L340
9443 033530 112737 000120 066062 MOVB #'P,MSG015 ;IT IS A MSV11-L/P
9444 033536
      033536 END ;
      L335:;;;;;
9445 033536 000403 BR 8$
9446 033540 112737 000040 066062 17$: MOVB #' ,MSG015
9447 033546
      033546 104401 066062 8$: TYPE MSG015
      TYPEIT ,MSG015
      .DSABL CRF
      IF NOTAB NE #0 THEN $RETURN
9448 033552
      033552 005737 002372
      TST NOTAB

```



```

033556 001401
033560 000207
033562
9449 033562 062701 000004
9450 033566 077052
9451 033570 010400
9452 033572 010301
9453
9454
9455
9456
9457
9458 033574
033574 104401 066064
9459 033600 112737 000040 066062 TCFIG3:
9460 033606 016105 002664
9461 033612 032705 000002
9462 033616 001414
9463 033620 042705 170377
9464 033624 000305
9465 033626 022705 000011
9466 033632 100002
9467 033634 062705 000007
9468 033640 062705 000060
9469 033644 110537 066062
9470 033650
033650 104401 066062
9471 033654
033654 005737 002372
033660 001401
033662 000207
033664
9472 033664 062701 000004
9473 033670 077035
9474 033672 010400
9475 033674 010301
9476
9477
9478
9479
9480 033676
033676 104401 065556
9481 033702 105761 002664 11$:
9482 033706 100006
9483 033710 112737 000120 066062
9484 033716
033716 104401 066062 13$:
9485 033722 000402
9486 033724
033724 104401 066060 14$:
9487 033730 062701 000004 15$:
9488 033734 077016
    
```

```

                                BEQ L341
                                RTS PC
                                L341:;;;;;
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
MOVB #' ,MSG015
MOV CONFIG(R1),R5
BIT #BIT1,R5
BEQ 16$
BIC #C7400,R5
SWAB R5
CMP #9.,R5
BPL 10$
ADD #7,R5
ADD #60,R5 ;MAKE ASCII
MOVB R5,MSG015 ;PLUG INTO MEMORY
TYPE MSG015
TYPEIT ,MSG015
.DSABL CRF
IF NOTAB NE #0 THEN $RETURN

                                TST NOTAB
                                BEQ L342
                                RTS PC
                                L342:;;;;;
ADD #4,R1 ;BUMP POINTER
SOB R0,TCFIG3
MOV R4,R0
MOV R3,R1

;*****
; ** PROTECTED **
;*****
TYPE MSG010
TYPEIT ,MSG010
.DSABL CRF
TSTB CONFIG(R1) ;BANK PROTECTED?
BPL 14$ ;NO - SKIP
MOVB #'P,MSG015
TYPE MSG015
TYPEIT ,MSG015
.DSABL CRF
BR 15$
TYPE MSG014 ;PRINT SPACE
TYPEIT ,MSG014
.DSABL CRF
ADD #4,R1 ;BUMP POINTER
SOB R0,11$ ;LOOP UNTIL DONE
    
```

C6

CVMJAAO MSV11-J MEMORY DIAG.
SUBR TYPE CONFIGURATION

MACRO V05.01b Friday 15-Feb-85 16:10 Page 296-2

SEQ 0274

9489 033736 010400
9490 033740 010301
9491 033742 000207

MOV R4,R0
MOV R3,R1
RETURN

```

9494          .SBTTL TRAP PARITY ERROR HANDLER
9495          ;*****
9496          ;VECTOR TO HERE FROM TRAPS TO 114
9497          ;IGNORE ERRORS BUT COUNT IF NOPAR FLAG = 1.
9498          ;*****
9499          ;
9500          ;      CODE      ACTION
9501          ;      ---
9502          ;      0      PRINT UNEXPECTED PARITY TRAP
9503          ;      1      COUNT ERROR
9504          ;      2      SET "ABORT" / SETUP "BADPC" / RETURN VIA PCBUMP
9505          ;      3      RETURN VIA "PARTHERE"
9506          ;
9507 033744 022737 000001 002076 PARITY: CMP #1,NOPAR ;COUNTING PARITY ERRORS?
9508 033752 001003          BNE 1$ ;NO - SKIP
9509 033754 005237 002072          INC PARCNT ;PARITY ERROR COUNTER + 1
9510 033760 000002          RTI
9511 033762 022737 000002 002076 1$: CMP #2,NOPAR ;ACTION CODE = 2 ?
9512 033770 001013          BNE 2$ ;NO - SKIP
9513 033772          SET ABORTFLAG ;YES
9514 034000 012737 177777 002144          MOV #-1,ABORTFLAG
9515 034004 004737 034152          CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
9516 034010 063716 002326          ADD PCBUMP,(SP) ;UPDATE RETURN PC
9517 034016 042766 000004 000002          BIC #BIT2,2(SP) ;SHOW FAILURE BY .NE.
9518 034020 000002          RTI
9519 034026 022737 000003 002076 2$: CMP #3,NOPAR ;ACTION CODE = 3 ?
9520 034030 001003          BNE 3$ ;NO - SKIP
9521 034034 013716 002304          MOV PARTHERE,(SP)
9522 034036 000002          RTI
9523 034042 004737 034152          3$: CALL BADSTACK ;FIND BAD SP,PC,PSW OFF STACK
          FATAL 32
          INC FATAL$ ;SET FATAL INDICATOR
          ERROR +32
          .DSABL CRF
034046 005237 002064
          104032

```



```

9526 .SBTTL TRAP NON-EXISTANT MEMORY (HOLES) HANDLER
9527 ;*****
9528 ;VECTOR TO HERE (SOMETIMES) FROM TRAPS TO 4
9529 ; CODE IN NONEM DETERMINES ACTION AS FOLLOWS:
9530 ; 1) IGNORE ERRORS BUT COUNT IF NONEM (NO NON-EXISTANT MEMORY) FLAG = 1.
9531 ; 2) TO EXIT PATTERN 0 DURING SIZING IF NON-EXIST MEM ERROR
9532 ;*****
9533
9534 034050 022737 000001 002100 NONEXIST: CMP #1, NONEM ;COUNTING NON-EXISTANT MEMORY ERRORS?
9535 034056 001011 BNE 2$ ;NO - SKIP
9536 034060 005237 002070 INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
9537 034064 022737 000001 002070 CMP #1, NEMCNT ;FIRST ERROR?
9538 034072 001002 BNE 1$ ;NO - SKIP
9539 034074 010037 002034 MOV RO, ADDRESS ;ASSUME RO CONTAINS THE ADDRESS ACCESSED
9540 034100 000002 1$: RTI
9541 034102 005237 002070 2$: INC NEMCNT ;BUMP NON-EXISTANT MEMORY COUNTER
9542 034106 012701 000001 MOV #1, R1 ;DUMMY UP R1 FOR A FORCED SOB EXIT
9543 034112 000002 RTI
9544
9545 ;*****
9546 .SBTTL TRAP TIMEOUT (TRAP TO 4) HANDLER
9547 034114 004737 034152 TIMEOUT: CALL BADSTACK ;FIND BAD SP, PC, PSW OFF STACK
9548 034120 FATAL 6
9549 034120 005237 002064 INC FATAL$ ;SET FATAL INDICATOR
9550 034124 104006 ERROR +6
9551 .DSABL CRF
9552 ;*****
9553 .SBTTL TRAP MEMORY MANAGEMENT (TRAP TO 250) HANDLER
9554 034126 004737 034152 MMTRAP: CALL BADSTACK ;FIND BAD SP, PC, PSW OFF STACK
9555 034132 FATAL 7
9556 034132 005237 002064 INC FATAL$ ;SET FATAL INDICATOR
9557 034136 104007 ERROR +7
9558 .DSABL CRF
9559 .SBTTL TRAP RESERVED INSTRUCTION HANDLER
9560 034140 004737 034152 PDP1105: CALL BADSTACK ;FIND BAD SP, PC, PSW OFF STACK
9561 034144 FATAL 5
9562 034144 005237 002064 INC FATAL$ ;SET FATAL INDICATOR
9563 034150 104005 ERROR +5
9564 .DSABL CRF
9565
9566 BADSTACK: SUBTST <<FIND BAD SP, PC, & PSW FROM STACK>>
9567 ;*****
9568 *SUBTEST FIND BAD SP, PC, & PSW FROM STACK
9569 ;*****
9570 MOV SP, BADSP
9571 ADD #2, BADSP
9572 MOV 2(SP), BADPC
9573 MOV 4(SP), BADPSW
9574 RETURN
9575
9576 034152 010637 002026 002026 MOV SP, BADSP
9577 034156 062737 000002 002026 ADD #2, BADSP
9578 034164 016637 000002 002022 MOV 2(SP), BADPC
9579 034172 016637 000004 002032 MOV 4(SP), BADPSW
9580 034200 000207 RETURN

```



```

9571          .SBTTL TRAP   KERNEL TRAP HANDLER
9572          ;*****
9573          ;KERNEL IS A TRAP THAT COMES HERE
9574          ;*****
9575
9576 034202 042766 140000 000002 $KERNEL:   BIC   #140000,2(SP)
9577 034210 000002          RTI
9578          ;*****
9579          .SBTTL TRAP   ENERGIZE TRAP HANDLER
9580 034212 052737 000001 177572 $ENERGIZE:BIS #BITO,MMRO
9581 034220 000002          RTI
9582          ;*****
9583          .SBTTL TRAP   DEENERGIZE TRAP HANDLER
9584 034222 042737 000001 177572 $DEENERGIZE:BIC #BITO,MMRO
9585 034230 000002          RTI
9586          ;*****
9587          .SBTTL TRAP   CACHON TRAP HANDLER
9588 034232 005737 002544 $CACHN:  TST   CACHKN          ;IS THERE A CACHE
9589 034236 001406          BEQ   1$              ;NO - RETURN
9590 034240 013737 002544 177746          MOV   CACHKN,CONTRL ;SETUP CACHE AS PER CONSTANT (USUALLY 1 = FULLY ON)
9591 034246 052737 000001 177746          BIS   #BITO,CONTRL ;DISABLE TRAPS (BUT NOT ABORTS)
9592 034254 000002          1$:  RTI
9593          ;*****
9594          .SBTTL TRAP   CACHOFF TRAP HANDLER
9595 034256 005737 002544 $CACHF:  TST   CACHKN          ;IS THERE A CACHE?
9596 034262 001403          BEQ   1$              ;NO - RETURN
9597          ;DISABLE TRAPS (NOT ABORTS), FORCE MISSES, FLUSH, BYPASS
9598 034264 053737 002550 177746          BIS   CACHKF,CONTRL
9599 034272 000002          1$:  RTI

```

```

9602          .SBTTL TRAP LOAD CSR TRAP HANDLER
9603          ;LOAD CORRECT CSR WITH DATA IN CSR
9604          ;PROGRAM CSR'S ASSERT INHIBIT MODE POINTER WHEN LOADED
9605 034274    $LOADC: PUSH    R0,R1          ;SAVE REGISTERS
          034274    010046
          034276    010146
9606 034300    013700    002152          MOV    CSRNO,R0          ;CREATE CSR ADDRESS
9607 034304    005737    002536          IF INHECC IS TRUE THEN GOTO 3$ ;DON'T WANT INH. MODE POINTER ON
          034304    001021
          034310    001021
9608 034312    005737    002532          TST    PGMCSR          ;PROGRAM IN INTERLEAVED SPACE?
9609 034316    100007          BPL    1$              ;BRANCH IF NOT
9610 034320    113701    002533          MOVB   PGMCSR+1,R1    ;CHECK SECOND CSR
9611 034324    042701    177740          BIC    #+C37,R1      ;CLEAR UNNECESSARY BITS
9612 034330    020137    002152          CMP    R1,CSRNO      ;IS THIS THE CURRENT CSR?
9613 034334    001404          BEQ    2$              ;BRANCH IF IT IS
9614 034336    123737    002532    002152  1$:  CMPB   PGMCSR,CSRNO    ;IS THIS THE CURRENT CSR?
9615 034344    001003          BNE    3$              ;BRANCH IF NOT
9616 034346    052737    020000    002150  2$:  BIS    #BIT13,CSR     ;SET THE INHIBIT MODE POINTER TO 1ST 16K
9617 034354    013760    002150    172100  3$:  MOV    CSR,CSRADD(R0) ;LOAD THE CSR
9618 034362          POP    R1,R0          ;RESTORE REGISTERS
          034362    012601
          034364    012600
9619 034366    000002          RTI
9620
9621          .SBTTL TRAP READ CSR TRAP HANDLER
9622          ;READ THE CORRECT CSR INTO LOCATIONS CSR
9623 034370    $READC: PUSH    R0
          034370    010046
9624 034372    013700    002152          MOV    CSRNO,R0          ;READ IT
9625 034376    016037    172100    002150          MOV    CSRADD(R0),CSR
9626 034404    012600          POP    R0
          034404    012600
9627 034406    000002          RTI
          MOV (SP)+,R1
          MOV (SP)+,R0
          MOV (SP)+,R0

```

```

9629
9630 034410          $TSTRD: .SBTTL TRAP TEST (R1) & READ CSR CAREFULLY
      034410 010046
      034412 010246
      034414 010346
9631 034416 012700 172100          MOV #CSRADD,R0          ;CREATE CSR ADDRESS
9632 034422 063700 002152          ADD CSRNO,R0
9633 034426 005002          CLR R2
9634 034430 005737 002532          TST PGMCSR
9635 034434 100007          BPL 1$
9636 034436 113703 002533          MOVB PGMCSR+1,R3
9637 034442 042703 000200          BIC #BIT7,R3
9638 034446 020337 002152          CMP R3,CSRNO
9639 034452 001404          BEQ 2$
9640 034454 123737 002532 002152 1$: CMPB PGMCSR,CSRNO
9641 034462 001002          BNE 3$
9642 034464 012702 020000          MOV #BIT13,R2
9643 034470 004737 034542          3$: CALL TSTRD1
9644
9645 034474          5$: ;IF SINGLE BIT ERROR ONLY - SET CARRY BIT
      034474 012603          POP R3,R2,R0
      034476 012602
      034500 012600
9646 034502          IF #BIT4 SET.IN CSR AND #BIT15 OFF.IN CSR
      034502 032737 000020 002150
      034510 001410
      034512 032737 100000 002150
      034520 001004
9647 034522 052766 000001 000002          BIS #BIT0,2(SP)
9648 034530          ELSE
      034530 000403
      034532
9649 034532 042766 000001 000002          BIC #BIT0,2(SP)
9650 034540          END ;OF IF #BIT4
      034540
9651 034540 000002          RTI
9652
9653 034542 010210          TSTRD1: MOV R2,(R0)          ;
9654 034544          TESTAREA          ;
      034544 053737 002552 177776          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
      .DSABL CRF
9655 034552 105711          TSTB (R1)          ;
9656 034554 042737 140000 177776          BIC #BIT15!BIT14,PSW          ;
9657 034562 011037 002150          MOV (R0),CSR          ;
9658 034566 000207          RETURN          ;

```

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

MOV (SP)+,R3
MOV (SP)+,R2
MOV (SP)+,R0

BIT #BIT4,CSR
BEQ L343
BIT #BIT15,CSR
BNE L343

BR L344
L343: :::::

L344: :::::

;GO TO SYSTEM TEST MODE

```

9661          .SBTTL TRAP ECC DISABLE ALL CSR'S TRAP HANDLER
9662 034570 012737 000002 002150 $ECCDIS:MOV #BIT1,CSR
9663 034576 004737 035314          CALL CSRROUT
9664 034602 000002          RTI
9665          .SBTTL TRAP ECC DISABLE OF 1 SELECTED CSR TRAP HANDLER
9666 034604 012737 000002 002150 $ECC1DIS:MOV #BIT1,CSR
9667 034612 104425          LOADCSR
9668 034614 000002          RTI
9669          .SBTTL TRAP INITIALIZE ALL CSR'S TRAP HANDLER
9670 034616 012737 000001 002150 $ECCINIT:MOV #BIT0,CSR
9671 034624 004737 035314          CALL CSRROUT
9672 034630 000002          RTI
9673          .SBTTL TRAP INITIALIZE 1 SELECTED CSR TRAP HANDLER
9674 034632 012737 000001 002150 $ECC1INIT:MOV #BIT0,CSR
9675 034640 104425          LOADCSR
9676 034642 000002          RTI
9677          .SBTTL TRAP ENABLE SBE PARITY TRAPS ON ALL CSR'S
9678 034644 012737 000003 002150 $ENASBE:MOV #BIT0!BIT1,CSR
9679 034652 004737 035314          CALL CSRROUT
9680 034656 000002          RTI
9681          .SBTTL TRAP ENABLE SBE PARITY TRAPS ON 1 SELECTED CSR
9682 034660 012737 000003 002150 $ENA1SBE:MOV #BIT0!BIT1,CSR
9683 034666 104425          LOADCSR
9684 034670 000002          RTI
9685          .SBTTL TRAP WRITE CHECKBITS THRU ALL CSR'S TRAP HANDLER
9686 034672 013737 002314 002150 $CBCSR:MOV CHECK,CSR ;BITS 11-5
9687 034700 052737 000006 002150 BIS #BIT1!BIT2,CSR ;CHECK MODE
9688 034706 004737 035314          CALL CSRROUT
9689 034712 000002          RTI
9690          .SBTTL TRAP WRITE CHECKBITS THRU 1 SELECTED CSR TRAP HANDLER
9691 034714 013737 002314 002150 $CB1CSR:MOV CHECK,CSR ;BITS 11-5
9692 034722 052737 000006 002150 BIS #BIT1!BIT2,CSR ;CHECK MODE
9693 034730 104425          LOADCSR
9694 034732 000002          RTI

```



```

9697          .SBTTL TRAP WAS THERE A SBE ON ANY CSR TRAP HANDLER
9698 034734    $WASSBE:PUSH R1,R4
          MOV R1,-(SP)
          MOV R4,-(SP)
9699 034734 010146
          034736 010446
9699 034740 013701 002224    MOV TOTCSRS,R1 ;GET CSR'S BYTE
9700 034744 005004          CLR R4
9701 034746          BEGIN LWSBE
          B57:;;;;;
9702 034746          FOR CSRNO := #0 TO #36 BY #2
          CLR CSRNO
          B60:;;;;;
          034746 005037 002152
          034752 006301          ASL R1
9703 034752          ON.ERROR
          BCC L345
9704 034754 103010          READCSR
          034756 104426          IF #BIT4 SET.IN CSR
          BIT #BIT4,CSR
9705 034756 032737 000020 002150    BEQ L346
          034766 001403          SET R4
          MOV #-1,R4
9707 034770 012704 177777          LEAVE LWSBE
          BR E57
9708 034774 000411          END ;OF IF #BIT4
          L346:;;;;;
9709 034776          END ;OF ON.ERROR
          L345:;;;;;
          034776          IF R1 EQ #0 THEN LEAVE LWSBE
          TST R1
9710 034776          END LWSBE
          BEQ E57
          034776 005701          ADD #2,CSRNO
9711 034776 001407          BEQ CSRNO,#36
          035000          BLE B60
9712 035002          END ;OF FOR CSRNO
          E60:;;;;;
          035002 062737 000002 002152
          035010 023727 002152 000036
          035016 003755          ROR R4 ;SET C BIT FOR ERROR
          E57:;;;;;
          035020          POP R4,R1
          MOV (SP)+,R4
9713 035020          ON.ERROR
          MOV (SP)+,R1
          035020 006004          BCC L347
9714 035022 012604
          035024 012601          BIS #BIT0,2(SP)
          BCC L347
9715 035026 103004          ELSE
          BR L350
9716 035026 052766 000001 000002    BIC #BIT0,2(SP)
          L347:;;;;;
9717 035030 000403          END ;OF ON.ERROR
          L350:;;;;;
          035036 000403          RTI
          035040          .SBTTL TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER
9718 035040 042766 000001 000002    ;ON RETURN IF CARRY IS SET THERE WAS A SBE
          $WAS1SBE:READCSR
9719 035046 000002          BIC #BIT0,2(SP) ;CLR C BIT ON STACK
9720 035050 104426          BIT #BIT4,CSR
9721 035052 042766 000001 000002    BEQ 1$
9722 035052 032737 000020 002150    BIS #BIT0,2(SP) ;SET C BIT ON STACK
9723 035060 001403
9724 035066 052766 000001 000002

```

K6

CVMJAAO MSV11-J MEMORY DIAG. MACRO V05.01b Friday 15-Feb-85 16:10 Page 309-1
TRAP WAS THERE A SBE IN 1 SELECTED CSR TRAP HANDLER

SEQ 0282

9729 035076 000002

14: RTI

```

9732          .SBTTL TRAP WAS THERE A DBE ON ANY CSR TRAP HANDLER
9733          $WASDBE: PUSH R1,R4
                                MOV R1,-(SP)
                                MOV R4,-(SP)
9734 035100 010146
9735 035102 010446
9734 035104 013701 002224      MOV TOTCSRS,R1 ;GET CSR'S BYTE
9735 035110 005004            CLR R4
9736 035112            BEGIN LWDBE
                                B61:;;;;;;
9737 035112            FOR CSRNO := #0 TO #36 BY #2
                                CLR CSRNO
9738 035116 005037 002152      ASL R1
9739 035120            ON.ERROR                                B62:;;;;;;
                                BCC L351
9740 035120 103010
9741 035122 104426            READCSR
                                IF #BIT15 SET.IN CSR
9742 035124 032737 100000 002150      SET R4
9743 035132 001403            LEAVE LWDBE
                                MOV #-1,R4
9744 035134 012704 177777      END ;OF IF #BIT4
                                BR E61
9745 035140 000411            END ;OF ON.ERROR
                                L352:;;;;;;
9746 035142            IF R1 EQ #0 THEN LEAVE LWDBE
                                L351:;;;;;;
9747 035142 005701            TST R1
9748 035144 001407            BEQ E61
9749 035146            END ;OF FOR CSRNO
9750 035146 062737 000002 002152      ADD #2,CSRNO
9751 035154 023727 002152 000036      CMP CSRNO,#36
9752 035162 003755            BLE B62
9748 035164            END LWDBE
                                E62:;;;;;;
9749 035164 006004            ROR R4 ;SET C BIT FOR ERROR
9750 035166            POP R4,R1
                                E61:;;;;;;
9751 035166 012604
9752 035170 012601            ON.ERROR
                                MOV (SP)+,R4
9753 035172 103004            MOV (SP)+,R1
9754 035174 052766 000001 000002      BCC L353
9755 035202            ELSE
9756 035202 000403            BR L354
9757 035204            L353:;;;;;;
9758 035204 042766 000001 000002      BIC #BIT0,2(SP)
9759 035212            END ;OF ON.ERROR
                                L354:;;;;;;
9760 035212 000002            RTI
9761 035214            .SBTTL TRAP WAS THERE A DBE ON 1 SELECTED CSR TRAP HANDLER
9762 035216            ;ON RETURN IF CARRY IS SET THERE WAS A DBE
9763 035218            $WAS1DBE: READCSR
9764 035220            TST CSR ;DBE?
9765 035222            BPL 3$ ;NO - SKIP
9766 035224            BIS #BIT0,2(SP) ;SET C BIT ON STACK
9767 035232            RTI

```

9764 035234 042766 000001 000002 34: BIC #BIT0,2(SP) ;CLR C BIT ON STACK
9765 035242 000002 RTI


```

9768          .SBTTL TRAP CLEAR ALL ECC CSR'S TRAP HANDLER
9769 035244    $CLRCSR: CLEAR CSR                                CLR CSR
          035244 005037 002150
9770 035250    CALL CSROUT
9771 035254    RTI
          035254 000002
9772          .SBTTL TRAP CLEAR 1 SELECTED CSR TRAP HANDLER
9773 035256    $CLR1CSR: CLEAR CSR                                CLR CSR
          035256 005037 002150
9774 035262    LOADCSR
9775 035264    RTI
          035264 000002
9776          .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN ALL CSR'S TRAP HANDLER
9777          ;CHECKBITS ALREADY IN LOC "CSR"
9778 035266    052737 000006 002150 $CHKDIS: BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
9779 035274    004737 035314
          035300 000002
9780          RTI
          035300 000002
9781          .SBTTL TRAP ECC DISABLE, CHECK MODE, & WRITE CHECKBITS IN 1 SELECTED CSR
9782          ;CHECKBITS ALREADY IN LOC "CSR"
9783 035302    052737 000006 002150 $CHK1DIS: BIS #BIT1!BIT2,CSR ;ECC DISABLE & DIAG CHECK MODE
9784 035310    104425
          035312 000002
9785          LOADCSR
          035312 000002
          RTI

```

```

9788 035314          CSROUT: SUBTST <<SUBR WRITE IN ALL CSR'S>>
;*****
;*SUBTEST          SUBR   WRITE IN ALL CSR'S
;*****
9789 035314          PUSH   R1
035314 010146          MOV    TOTCSRS,R1      ;GET CSR'S BYTE
9790 035316 013701 002224  BEGIN LCSROUT
9791 035322          FOR CSRNO := #0 TO #36 BY #2
035322          B63:;;;;;
035322 005037 002152          CLR  CSRNO
035326          B64:;;;;;
9793 035326 006301          ASL   R1
9794 035330          ON.ERROR
035330 103001          BCC  L355
9795 035332 104425          LOADCSR
9796 035334          END ;OF ON.ERROR
035334          L355:;;;;;
9797 035334          IF R1 EQ #0 THEN LEAVE LCSROUT
035334 005701          TST  R1
035336 001407          BEQ  E63
9798 035340          END ;OF FOR CSRNO
035340 062737 000002 002152          ADD  #2,CSRNO
035346 023727 002152 000036          CMP  CSRNO,#36
035354 003764          BLE  B64
035356          E64:;;;;;
9799 035356          END LCSROUT
035356          E63:;;;;;
9800 035356          POP    R1
035356 012601          MOV  (SP)+,R1
9801 035360 000207          RETURN
9802
9803 035362          $INVALID: SUBTST <<TRAP INVALIDATE BACKGROUND PATTERN>>
;*****
;*SUBTEST          TRAP   INVALIDATE BACKGROUND PATTERN
;*****
9804 035362          PUSH   RO,R1
035362 010046          MOV  RO,-(SP)
035364 010146          MOV  R1,-(SP)
9805 035366 013701 002102          MOV  BANK,R1
9806 035372 006301          ASL  R1
9807 035374 006301          ASL  R1
9808 035376 042761 020000 002666          BIC  #BIT13,CONFIG+2(R1)
9809 035404          POP   R1,RO
035404 012601          MOV  (SP)+,R1
035406 012600          MOV  (SP)+,RO
9810 035410 000002          RTI

```

```

9812 035412          $ERRGEN:          SUBTST<<TRAP  GENERATE AND TEST ERROR ADDRESS>>
;*****
;*SUBTEST          TRAP  GENERATE AND TEST ERROR ADDRESS
;*****
9813 035412          PUSH          RO,R1,R2,R3
035412 010046                                MOV RO,-(SP)
035414 010146                                MOV R1,-(SP)
035416 010246                                MOV R2,-(SP)
035420 010346                                MOV R3,-(SP)
9814 035422 013703 002104          MOV          BANKINDEX,R3
9815 035426 005737 002456          TST          NOSUPER
9816 035432 001003                    BNE          6$
9817 035434 013700 172246          MOV          SIPAR3,RO          ;GENERATE WHAT ERROR ADDR SHOULD BE
9818 035440 000402                    BR           7$
9819 035442 013700 177646          6$: MOV          UIPAR3,RO
9820 035446 072027 177773          7$: ASH          #-5,RO
9821 035452 005737 002132          TST          EQFLAG
9822 035456 001002                    BNE          1$
9823 035460 042700 177600          BIC          #+C177,RO
9824 035464 000301          1$: SWAB          R1          ;GET CURRENT ADDRESS BITS 11 AND 12
9825 035466 006201                    ASR          R1
9826 035470 006201                    ASR          R1
9827 035472 006201                    ASR          R1
9828 035474 042701 177775          BIC          #+C2,R1
9829 035500 060100          ADD          R1,RO          ;ADD THEM TO THE ADJUSTED PAR VALUE
9830                    ;GET ERROR ADDRESS FROM CSR UNDER TEST
9831 035502 013701 002150          MOV          CSR,R1
9832 035506 072127 177773          ASH          #-5,R1
9833 035512 042701 177600          BIC          #+C177,R1
9834 035516 005737 002454          TST          NO22BIT          ;IS THIS AN 11/83,11/23-B OR 11/23 ?
9835 035522 001024                    BNE          2$          ;BRANCH IF NOT NECESSARY
9836 035524 005737 002132          TST          EQFLAG          ;IS IT EQB?
9837 035530 001421                    BEQ          2$          ;BRANCH IF NOT
9838 035532          PUSH          RO          ;SAVE GENERATED ERROR ADDRESS
035532 010046                                MOV RO,-(SP)
9839 035534 013702 002152          MOV          CSRNO,R2          ;GET CSR NUMBER
9840 035540 052762 040000 172100        BIS          #BIT14,CSRADD(R2) ;TURN ON EQB BIT CAREFULLY
9841 035546 016200 172100          MOV          CSRADD(R2),RO ;GET CSR CONTENTS
9842 035552 042762 040000 172100        BIC          #BIT14,CSRADD(R2) ;TURN OFF EQB BIT CAREFULLY
9843 035560 042700 177037          BIC          #+C740,RO          ;CLEAR EVERYTHING BUT ERROR ADDR
9844 035564 006300                    ASL          RO
9845 035566 006300                    ASL          RO          ;SHIFT ADDR BITS 18-21 INTO POSITION
9846 035570 060001          ADD          RO,R1          ;ADD TO CURRENT ERROR ADDRESS
9847 035572          POP          RO
035572 012600                                MOV (SP)+,RO
9848 035574 020001          2$: CMP          RO,R1          ;COMPARE REAL AND GENERATED ERR. ADDR.
9849 035576 001420                    BEQ          5$          ;BRANCH IF THEY ARE THE SAME
9850 035600 005737 002136          TST          INTFLAG          ;INTERLEAVED?
9851 035604 001411                    BEQ          3$          ;NO - WE HAVE AN ERROR
9852 035606 062700 000100          ADD          #100,RO
9853 035612 005737 002140          TST          INT64K          ;64K INTERLEAVED MEMORY?
9854 035616 001002                    BNE          4$
9855 035620 062700 000100          ADD          #100,RO
9856 035624 020001          4$: CMP          RO,R1
9857 035626 001404                    BEQ          5$
9858 035630 005737 002066          3$: TST          SKPERR          ;ARE WE SUPPOSED TO SKIP ERROR P.O.?
9859 035634 001001                    BNE          5$          ;YES - SKIP ERROR PRINTOUT

```



```

9860 035636 104462          PERR36          ;ELSE PRINT ERROR ADDRESS ERROR
9861 035640 010137 002460 5$: MOV R1,ERRADD ;SAVE CSR'S ERROR ADDRESS
9862 035644 005037 002066 CLR SKPERR ;ENABLE THE ERROR PRINTOUT AGAIN
9863 035650 POP R3,R2,R1,R0 ;RESTORE REGISTERS
      035650 012603          MOV (SP),R3
      035652 012602          MOV (SP),R2
      035654 012601          MOV (SP),R1
      035656 012600          MOV (SP),R0
9864 035660 000002          RTI
9865
9866 035662          $CBREG: SUBTST <<TRAP          ENABLE CHECKBIT REGISTER>>
      ;*****
      ;*SUBTEST TRAP          ENABLE CHECKBIT REGISTER
      ;*****
9867 035662 005037 002150 CLR CSR ;
9868 035666 052737 000004 002150 BIS #BIT2,CSR ;ENABLE DIAGNOSTIC MODE
9869 035674 104425 LOADCSR ;LOAD CSR REGISTER
9870 035676 000002 RTI ;
9871
9872 035700          $SYNREG: SUBTST <<TRAP          ENABLE SYNDROME BIT REGISTER>>
      ;*****
      ;*SUBTEST TRAP          ENABLE SYNDROME BIT REGISTER
      ;*****
9873 035700 005037 002150 CLR CSR ;
9874 035704 052737 040004 002150 BIS #BIT14!BIT2,CSR ;ENABLE DIAGNOSTIC MODE
9875 035712 104425 LOADCSR ;LOAD CSR REGISTER
9876 035714 000002 RTI ;
9877

```


9880 035716

SUBTST<<SUBR MAPPER>>

;SUBTEST SUBR MAPPER

9881
9882
9883
9884
9885
9886
9887
9888
9889
9890

;THIS SUBROUTINE MAPS THE MEMORY BANK (16K WORDS = 1 BANK)
;IN R3 TO THE TEST PATTERN AREA (SUPERVISOR VIRTUAL (60000 - 157777) FOR
;THE 11/83 ; USER VIRTUAL (60000 - 157777) FOR ALL OTHERS.
;
;

;CALL MOV BANKNO,R3 ;SET UP BANK ARGUMENT
; CALL MAPPER ;ACTUAL CALL
; RETURN ;ONLY RETURN

;SET SUPERVISOR/USER UP FOR 1 TO 1 MAP

9891 035716

MAPPER: PUSH RO,R1,R2,R4,R5

MOV RO,-(SP)
MOV R1,-(SP)
MOV R2,-(SP)
MOV R4,-(SP)
MOV R5,-(SP)

035716 010046
035720 010146
035722 010246
035724 010446
035726 010546
9892 035730 012700 172340
9893 035734 012701 172240
9894 035740 012704 172200
9895 035744 005737 002456
9896 035750 001404
9897 035752 012701 177640
9898 035756 012704 177600
9899 035762 012702 077406
9900 035766 012705 000010
9901 035772 012021
9902 035774 010224
9903 035776 077503
9904 036000 012741 177600
9905 036004 022737 000005 004010 30\$:
9906 036012 001007
9907 036014 012704 172206
9908 036020 012705 000004
9909 036024 052724 100000
9910 036030 077503
9911
9912 036032 022703 000200 40\$:
9913 036036 001516
9914 036040 072327 000011
9915
9916 036044 012701 172246
9917 036050 005737 002456
9918 036054 001402
9919 036056 012701 177646
9920 036062 012702 000004 5\$:
9921 036066 010321 2\$:
9922 036070 062703 000200
9923 036074 077204
9924 036076 005737 002240
9925 036102 001442
9926 036104 162701 000010
9927 036110 010102
9928 036112 062702 000004

MOV #KIPAR0,R0 ;FIRST AREA TO MAP TO
MOV #SIPAR0,R1 ;FIRST ADDRESS REGISTER
MOV #SIPDR0,R4 ;FIRST DESCRIPTOR REGISTER
TST NOSUPER ;CAN WE USE SUPERVISOR MODE?
BEQ 4\$;YES, BRANCH
MOV #UIPAR0,R1 ;FIRST ADDRESS REGISTER
MOV #UIPDR0,R4 ;FIRST DESCRIPTOR REGISTER
4\$: MOV #77406,R2 ;CONSTANT FOR 4K PAGE, UP, R/W
MOV #8.,R5 ;COUNTER
1\$: MOV (R0)+,(R1)+ ;PUT IN SUPERVISOR ADDRESS
MOV R2,(R4)+ ;PUT IN SUPERVISOR DESCRIPTOR
SOB R5,1\$;LOOP TILL DONE
MOV #177600,-(R1) ;CORRECT LAST FIELD FOR PERIPHERALS PAGE
30\$: CMP #5,PROTYP ;IS THIS A 11/73?
BNE 40\$;BRANCH IF NOT
MOV #SIPDR0+6,R4 ;POINT TO PDR 3
MOV #4.,R5 ;COUNTER=4
35\$: BIS #BIT15,(R4)+ ;SET UNCONDITIONAL CACHE BYPASS IN PDR3-6
SOB R5,35\$
;SET UP SUPERVISOR/USER FOR TEST AREA
40\$: CMP #200,R3 ;MAP NOTHING (1 TO 1)?
BEQ 3\$;YES - SKIP
ASH #9.,R3 ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000
MOV #SIPAR3,R1 ;SETUP FOR AUTO INCREMENTING
TST NOSUPER ;DO WE HAVE SUPERVISOR MODE?
BEQ 5\$;YES - BRANCH
MOV #UIPAR3,R1 ;SETUP FOR AUTO INCREMENTING
5\$: MOV #4,R2 ;COUNTER
2\$: MOV R3,(R1)+ ;PLUG IN PAR INFO
ADD #200,R3 ;BUMP ADDRESS 4K
SOB R2,2\$;LOOP TILL DONE
TST SPLTCSR
BEQ 9\$
SUB #10,R1
MOV R1,R2
ADD #4,R2

```

SUBR MAPPER
9929 036116 022737 000001 002240      CMP      #1,SPLTCSR
9930 036124 001403                      BEQ      10$
9931 036126 010200                      MOV      R2,R0
9932 036130 010102                      MOV      R1,R2
9933 036132 010001                      MOV      R0,R1
9934 036134 012122                      10$:    MOV      (R1)+,(R2)+
9935 036136 011112                      MOV      (R1),(R2)
9936 036140 013700 002104              MOV      BANKINDEX,R0
9937 036144 005737 002140              TST      INT64K
9938 036150 001403                      BEQ      11$
9939 036152 012700 004000              MOV      #4000,R0
9940 036156 000402                      BR       12$
9941 036160 012700 010000              11$:    MOV      #10000,R0
9942 036164 005737 002456              12$:    TST      NOSUPER
9943 036170 001403                      BEQ      13$
9944 036172 012701 177652              MOV      #UIPAR5,R1
9945 036176 000402                      BR       14$
9946 036200 012701 172252              13$:    MOV      #SIPAR5,R1
9947 036204 060021                      14$:    ADD      R0,(R1)+
9948 036206 060011                      ADD      R0,(R1)
9949
9950
9951
9952
9953
9954 036210 022737 000007 002556  9$:    CMP      #7,LASTBANK
9955 036216 001010                      BNE      7$
9956 036220 005737 002454              TST      NO22BIT
9957 036224 001423                      BEQ      3$
9958 036226 022737 000007 002102              CMP      #7,BANK
9959 036234 001017                      BNE      3$
9960 036236 000404                      BR       8$
9961 036240 022737 000177 002102  7$:    CMP      #177,BANK
9962 036246 001012                      BNE      3$
9963 036250 005737 002456              8$:    TST      NOSUPER
9964 036254 001404                      BEQ      6$
9965 036256 013737 177652 177654              MOV      UIPAR5,UIPAR6
9966 036264 000403                      BR       3$
9967 036266 013737 172252 172254  6$:    MOV      SIPAR5,SIPAR6
9968 036274
9968 036274 012605                      3$:    POP      R5,R4,R2,R1,R0
9968 036276 012604
9968 036300 012602
9968 036302 012601
9968 036304 012600
9969 036306 000207                      MOV (SP)+,R5
9970
9971 036310
9971 036310 010046                      MOV (SP)+,R4
9971 036312 010146                      MOV (SP)+,R2
9971 036314 010246                      MOV (SP)+,R1
9971 036316 010346                      MOV (SP)+,R0
9971 036320 010446
9972 036322 005000
9973 036324 012701 172340
9974 036330 012702 077406
9975 036334 012703 172300
          RETURN
          .SBTTL TRAP MAP KERNEL (ALMOST 1 TO 1) TRAP HANDLER
          $KMAP: PUSH R0,R1,R2,R3,R4
          MOV R0,-(SP)
          MOV R1,-(SP)
          MOV R2,-(SP)
          MOV R3,-(SP)
          MOV R4,-(SP)
          CLR R0
          MOV #KIPAR0,R1
          MOV #77406,R2
          MOV #KIPDR0,R3
          ;1ST AREA TO MAP TO
          ;FIRST ADDRESS
          ;CONSTANT FOR 4K PAGE,UP,R/W
          ;1ST PAGE DESCRIPTOR REGISTER

```

9976	036340	012704	000010		MOV	#8.,R4		:COUNTER
9977	036344	010021		1\$:	MOV	R0,(R1)+		:PUT IN KERNEL ADDRESS
9978	036346	010223			MOV	R2,(R3)+		:PUT IN KERNEL DISCRIPTOR
9979	036350	062700	000200		ADD	#200,R0		:ADD ADDRESS CONSTANT FOR 4K CHANGE
9980	036354	077405			SOB	R4,1\$:LOOP TILL DONE
9981	036356	012741	177600		MOV	#177600,-(R1)		:THE PERIPHERALS PAGE TO KIPAR7
9982	036362	012741	177400		MOV	#177400,-(R1)		:AND NEXT LOWER PAGE TO KIPAR6
9989	036366				POP	R4,R3,R2,R1,R0		
	036366	012604						MOV (SP)+,R4
	036370	012603						MOV (SP)+,R3
	036372	012602						MOV (SP)+,R2
	036374	012601						MOV (SP)+,R1
	036376	012600						MOV (SP)+,R0
9990	036400	000002			RTI			


```

9993 036402          RELOCATE:SUBTST <<RELOCATE PROGRAM>>
;*****
;*SUBTEST          RELOCATE PROGRAM
;*****
9994 036402          IF #SW12 SET.IN @SWR THEN $RETURN ERROR
036402 032777 010000 144226          BIT #SW12,@SWR
036410 001402          BEQ L356
036412 000261          SEC
036414 000207          RTS PC
036416          L356:;;;;;
9995 036416          IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
036416 005737 002352          TST APTFLAG
036422 001003          BNE L357
036424 005737 002350          TST ACTFLAG
036430 001405          BEQ L360
036432          L357:;;;;;
9996 036432          IF $PASS NE #0 THEN $RETURN ERROR
036432 005737 056654          TST $PASS
036436 001402          BEQ L361
036440 000261          SEC
036442 000207          RTS PC
036444          L361:;;;;;
9997 036444          END; OF IF APTFLAG
036444          L360:;;;;;
9998 036444          BEGIN LOADERBANK
036444          B65:;;;;;
9999 036444          FOR BANK := #1 TO LASTBANK
036444 012737 000001 002102          MOV #1,BANK
036452          B66:;;;;;
10000 036452          CALL EXBANK
10001 036456          IF ACFLAG IS TRUE AND PFLAG IS FALSE AND BMFLAG IS FALSE
036456 005737 002116          TST ACFLAG
036462 001431          BEQ L362
036464 005737 002122          TST PFLAG
036470 001026          BNE L362
036472 005737 002130          TST BMFLAG
036476 001023          BNE L362
10002 036500          MOV BANK,RO
10003 036504          MOV RO,LOADBANK
10004 036510          MOV LOADHOME,R1
10005 036514          CALL BANKMOV
10006 036520          CALL NEWLOAD          ;MAP NEW LOADER BANK IN KERNEL
10007 036524          MOV BANKINDEX,R1
10008 036530          BIS #BIT15,CONFIG+2(R1)          ;MARK LOADER
10009 036536          BIC #BIT13,CONFIG+2(R1)          ;INVALIDATE BACKGROUND PATTERN
10010 036544          LEAVE LOADERBANK
036544 000416          BR E65
10011 036546          END ;OF IF ACFLAG
036546          L362:;;;;;
10012 036546          END ;OF FOR BANK
036546 005237 002102          INC BANK
036552 023737 002102 002556          CMP BANK,LASTBANK
036560 003734          BLE B66
036562          E66:;;;;;
10013 036562          IF #SW13 OFF.IN @SWR
036562 032777 020000 144046          BIT #SW13,@SWR
036570 001002          BNE L363

```



```

10014 036572      TYPE          MSG075          ;RELOCATION NOT POSSIBLE
      036572 104401 070176      TYPEIT ,MSG075
10015 036576      .DSABL CRF
      036576      END ;OF IF #SW13
10016 036576      $RETURN ERROR          L363:;;;;;
      036576 000261      SEC
      036600 000207      RTS PC
10017 036602      END LOADERBANK
      036602      BEGIN FINDBANK          E65:;;;;;
10018 036602      MOV          LASTBANK,R2
      036602 013702 002556      ASL          R2
10019 036606 006302      ASL          R2          ;R2 <- R2 * 4
10020 036610 006302      FOR R1 := #2*2 TO R2 BY #4
10021 036612 012701 000004      MOV #2*2,R1
      036612 012701 000004      B70:;;;;;
      036616      IF #BIT7!BIT0 OFF.IN CONFIG(R1) ;IF NO ERRORS & NOT PROGRAM SPACE
10023 036616 032761 000201 002664      BIT #BIT7!BIT0,CONFIG(R1)
      036624 001035      BNE L364
10024 036626      IF #BIT15 OFF.IN CONFIG+2(R1) ;IF NOT LOADER BANK
      036626 032761 100000 002666      BIT #BIT15,CONFIG+2(R1)
      036634 001031      BNE L365
10025 036636      IF CPUBIT SET.IN CONFIG(R1) ;IF ACCESSABLE
      036636 033761 002106 002664      BIT CPUBIT,CONFIG(R1)
      036644 001425      BEQ L366
10026 036646      IF #BIT8 OFF.IN CONFIG+2(R1) THEN LEAVE FINDBANK ;IF PARITY
      036646 032761 000400 002666      BIT #BIT8,CONFIG+2(R1)
      036654 001460      BEQ E67
10027 036656      IF #BIT6 SET.IN CONFIG(R1) AND #BIT7 OFF.IN CONFIG(R1)
      036656 032761 000100 002664      BIT #BIT6,CONFIG(R1)
      036664 001405      BEQ L367
      036666 032761 000200 002664      BIT #BIT7,CONFIG(R1)
      036674 001001      BNE L367
10028      ;IF 1ST PROTECTABLE ECC BANK
10029 036676      LEAVE FINDBANK
      036676 000447      BR E67
10030 036700      END ;OF IF #BIT6
      036700      IF INHECC IS FALSE          L367:;;;;;
10031 036700      TST INHECC
      036700 005737 002536      BNE L370
      036704 001005      SET INHECC
10032 036706      MOV # -1,INHECC
      036706 012737 177777 002536      MOV R1,INH BANK
10033 036714 010137 002540      END; OF IF INHECC
10034 036720      END ;OF IF CPUBIT          L370:;;;;;
10035 036720      END ;OF IF #BIT15          L366:;;;;;
10036 036720      END ;OF IF #BIT7          L365:;;;;;
10037 036720      END ;OF FOR          L364:;;;;;
10038 036720 062701 000004      ADD #4,R1
      036724 020102      CMP R1,R2

```

```

036726 003733
036730
10039 036730 IF FULLREL IS FALSE BLE B70
036730 005737 002542 E70:;;;;;;
036734 001012 TST FULLREL
10040 036736 IF INHECC IS TRUE BNE L371
036736 005737 002536 TST INHECC
036742 001407 BEQ L372
10041 036744 013701 002540 MOV INHBANK,R1
10042 036750 023727 002300 000030 CMP REALPAT,#30 ;IS THIS PATTERN 30?
10043 036756 001421 BEQ RELENT1 ;YES - SKIP MESSAGE
10044 036760 000420 BR RELENT1
10045 036762 END; OF IF INHECC
036762 L372:;;;;;;
10046 036762 END; OF IF FULLREL
036762 L371:;;;;;;
10047 036762 005037 002536 CLR INHECC ;MAKE SURE FLAG IS TURNED OFF!
10048 036766 IF #SW13 OFF.IN @SWR
036766 032777 020000 143642 BIT #SW13,@SWR
036774 001006 BNE L373
10049 036776 023727 002300 000030 CMP REALPAT,#30 ;IS THIS PATTERN 30?
10050 037004 001402 BEQ SKUB ;YES - SKIP MESSAGE
10051 037006 TYPE MSG075 ;RELOCATION NOT POSSIBLE
037006 104401 070176 TYPEIT ,MSG075
10052 037012 .DSABL CRF
037012 END ;OF IF #SW13 L373:;;;;;;
10053 037012 SKUB: $RETURN ERROR
037012 000261 SEC
037014 000207 RTS PC
10054 037016 END FINDBANK
037016 CLEAR INHECC E67:;;;;;;
10055 037016 005037 002536 ;IF WE RELOCATED PROPERLY, THIS SHOULD BE OFF!
037016 042761 020000 002666 RELENT1: BIC #BIT13,CONFIG+2(R1) CLR INHECC
10056 037022 005000 ;INVALIDATE BACKGROUND PATTERN
10057 037030 071027 000004 CLR RO
10058 037032 RELOC1: LET NEWBANK := RO DIV #4,RO
10059 037036 010037 002310 MOV RO,NEWBANK
037036 013737 002532 002534 MOV PGMCSR,PGMCSR+2 ;SAVE CURRENT PGM. CSR
10060 037042 004737 037610 CALL USERMAP ;MAP NEWBANK TO USER PAR
10061 037050 USER ;ENTER USER MODE
10062 037054 052737 140000 177776 BIS #BIT15!BIT14,PSW ;GO TO USER MODE
037054 .DSABL CRF
10063 037062 BMOV 0,100000,SIZE ;MOVE PROGRAM
037062 004537 041024 JSR R5,BLOCK3
037066 040000 SIZE
037070 100000 100000
037072 000000 0
.DSABL CRF
10064 037074 104417 KERNEL ;ENTER KERNEL MODE
10065 037076 042737 000001 177572 JMPRL1: BIC #BIT0,MMRO ;DEENERGIZE MEMORY MANAGEMENT
10066 037104 004737 037672 CALL NEWKERNEL
10067 037110 013700 002310 MOV NEWBANK,RO
10068 037114 006300 ASL RO
10069 037116 006300 ASL RO ;RO <- RO * 4
10070 037120 016002 002664 MOV CONFIG(RO),R2

```

10071	037124	000302			SWAB	R2		
10072	037126	042702	177760		BIC	#+C17,R2		
10073	037132	006302			ASL	R2		
10074	037134	052737	000001	177572	BIS	#BIT0,MMR0		;ENERGIZE MEMORY MANAGEMENT
10075	037142	010237	002532		MOV	R2,PGMCSR		;PUT NEW PGM. CSR INTO PGMCSR
10076	037146	032760	010000	002666	BIT	#BIT12,CONFIG+2(R0)		;IS THE NEW BANK INTERLEAVED?
10077	037154	001412			BEQ	1\$;BRANCH IF NOT INTERLEAVED
10078	037156	016002	002664		MOV	CONFIG(R0),R2		
10079	037162	042702	007777		BIC	#+C170000,R2		
10080	037166	072227	177775		ASH	#-3,R2		
10081	037172	052702	100000		BIS	#BIT15,R2		
10082	037176	050237	002532		BIS	R2,PGMCSR		
10083	037202				SET	RLFLAG		
	037202	012737	177777	002126				MOV #-1,RLFLAG
10084	037210				\$RETURN	NOERROR		
	037210	000241						CLC
	037212	000207						RTS PC

10087 037214

UNRELOCATE:SUBTST <<UNRELOCATE PROGRAM>>
;*****
;*SUBTEST UNRELOCATE PROGRAM
;*****

```

10088
10089 037214 010046 002432 002102      ;RESTORE LOADERS
10090 037216 013701 002432      PUSH      RO
10091 037222 013700 002576      MOV      LOADBANK,R1
10092 037226 004737 037442      MOV      LOADHOME,RO
10093 037232 004737 037774      CALL     BANKMOV
10094 037236 013746 002102      CALL     NEWLOAD      ;MAP NEW LOADER BANK IN KERNEL SPACE
10095 037242 013737 002432      CALL     BANK        ;MOV BANK,-(SP)
10096 037250 004737 040052      MOV      LOADBANK,BANK
10097 037254 013701 002104      CALL     EXBANK
10098 037260 042761 100000      MOV      BANKINDEX,R1
10099 037266 013737 002576      BIC      #BIT15,CONFIG+2(R1) ;CLEAR LOADER FLAG
10100 037274 004737 040052      MOV      LOADHOME,BANK
10101 037300 013701 002104      CALL     EXBANK
10102 037304 042761 020000      MOV      BANKINDEX,R1
10103 037312 012637 002102      BIC      #BIT13,CONFIG+2(R1) ;INVALIDATE BACKGROUND PATTERN
10104 037316 005037 002536      POP      BANK
10105 037316 005037 002536      CLEAR   INHECC      ;MAKE SURE ECC TESTS ARE NOT INHIBITED!
10106 037316 005037 002536      CLR     INHECC
10107 037322 042737 020000      ;RESTORE BANK 0
10108 037330 005037 002310      BIC      #BIT13,CONFIG+2      ;INVALIDATE BACKGROUND PATTERN
10109 037334 004737 037610      LET     NEWBANK := #0
10110 037340 052737 140000      CALL     USERMAP      CLR NEWBANK
10111 037346 004537 041024      USER    USERMAP      ;MAP NEWBANK TO USER PAR
10112 037346 004537 041024      BIS     #BIT15:BIT14,PSW    ;ENTER USER MODE
10113 037346 004537 041024      .DSABL CRF                ;GO TO USER MODE
10114 037346 004537 041024      BMOV   0,100000,SIZE      ;MOVE PROGRAM
10115 037346 004537 041024      JSR    R5,BLOCK3
10116 037346 004537 041024      SIZE  100000
10117 037346 004537 041024      0
10118 037346 004537 041024      .DSABL CRF
10119 037360 104417 000001      KERNEL
10120 037362 042737 000001      BIC     #BIT0,MMRO          ;ENTER KERNEL MODE
10121 037370 004737 037672      CALL   NEWKERNEL          ;DEENERGIZE MEMORY MANAGEMENT
10122 037374 013737 002534      MOV     PGMCSR+2,PGMCSR    ;RESTORE PREVIOUS PGM. CSR
10123 037402 052737 000001      BIS     #BIT0,MMRO          ;ENERGIZE MEMORY MANAGEMENT
10124 037410 005037 002126      CLR     RLFLAG
10125 037414 012700 002666      1$: MOV     #CONFIG+2,RO      ;MOVE 2ND WORD OF CONFIG TO RO
10126 037420 042710 020000      2$: BIC     #BIT13,(RO)     ;CLEAR BACKGROUND VALID BIT
10127 037424 062700 000004      ADD     #4,RO              ;INCREMENT TO NEXT BANK
10128 037430 020027 003620      CMP     RO,#3620           ;DONE?
10129 037434 003771 000000      BLE    2$                 ;NO - BRANCH
10130 037436 012600 000000      POP     RO
10131 037436 012600 000000      MOV (SP)+,RO
10132 037440 000207 000000      RETURN
10133 037440 000207 000000
10134 037440 000207 000000
10135 037440 000207 000000

```


10128 037442

BANKMOV:SUBTST <<MOVE BANKS>>

;SUBTEST MOVE BANKS

10129
10130
10131
10132
10133 037442 104415
10134 037444 004737 037610
10135 037450 104416
10136 037452 104415
10137 037454 072027 000011
10138 037460 072127 000011
10139 037464 012702 177650
10140 037470 012703 000200

;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

10141
10142 037474 010122
10143 037476 060301
10144 037500 010122
10145 037502 060301
10146
10147 037504 010022
10148 037506 060300
10149 037510 010022
10150 037512 060300

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

10151
10152 037514
037514 052737 140000 177776

USER
BIS #BIT15!BIT14,PSW ;GO TO USER MODE
.DSABL CRF

10153 037522
037522 004537 041024
037526 020000
037530 140000
037532 100000

BMOV 100000,140000,SIZE/2 ;MOV 1ST HALF BANK
JSR R5,BLOCK3
SIZE/2
140000
100000
.DSABL CRF

10154 037534 104417
10155
10156 037536 012702 177650

KERNEL ;ENTER KERNEL MODE
MOV #UIPAR4,R2

10157
10158 037542 010122
10159 037544 060301
10160 037546 010122
10161 037550 060301

MOV R1,(R2)+ ;MAP 2ND HALF BANK
ADD R3,R1 ;BUMP BY 4K
MOV R1,(R2)+
ADD R3,R1

10162
10163 037552 010022
10164 037554 060300
10165 037556 010022
10166 037560 060300

MOV R0,(R2)+
ADD R3,R0
MOV R0,(R2)+
ADD R3,R0

10167
10168 037562
037562 052737 140000 177776

USER
BIS #BIT15!BIT14,PSW ;GO TO USER MODE
.DSABL CRF

10169 037570
037570 004537 041024
037574 010000
037576 140000

BMOV 100000,140000,SIZE/4 ;MOV 3ND FOURTH OF BANK
JSR R5,BLOCK3
SIZE/4
140000

037600 100000
10170 037602 104417
10171
10172 037604 104416
10173 037606 000207

100000
.DSABL CRF
KERNEL ;ENTER KERNEL MODE
RESREG
RETURN

10176 037610

USERMAP:SUBTST <<SUBR MAP USER TO NEW BANK>>

;SUBTEST SUBR MAP USER TO NEW BANK

10177 037610 012701 177640
10178 037614 012702 172340
10179 037620 012703 177600
10180 037624 012704 172300
10181 037630 012705 000004
10182 037634 012221
10183 037636 011423
10184 037640 077503
10185
10186 037642 013700 002310
10187 037646 072027 000011
10188
10189 037652 012705 000004
10190 037656 010021
10191 037660 062700 000200
10192 037664 011423
10193 037666 077505
10194 037670 000207
1C195
10196 037672

MOV #UIPAR0,R1 ;COPY KERNEL PAR'S & PDR'S (0-3)
MOV #KIPAR0,R2
MOV #UIPDR0,R3
MOV #KIPDR0,R4
1\$: MOV #4,R5
MOV (R2)+,(R1)+
MOV (R4),(R3)+
SOB R5,1\$

MOV NEWBANK,R0
ASH #9.,R0 ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000

2\$: MOV #4,R5
MOV R0,(R1)+ ;SETUP UIPAR(4-7)
ADD #200,R0 ;BUMP ADDRESS 4K
MOV (R4),(R3)+ ;SETUP UIPDR(4-7)
SOB R5,2\$
RETURN

NEWKERNEL:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW BANK>>

;SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW BANK

10197 037672
037672 010046
037674 010146
037676 010546
10198 037700 012700 172340
10199 037704 013701 002310
10200 037710 072127 000011
10201
10202 037714 012705 000004
10203 037720 010120
10204 037722 062701 000200
10205 037726 077504
10206 037730
037730 012605
037732 012601
037734 012600
10207 037736 000207
10208
10209 037740

PUSH R0,R1,R5

MOV R0,-(SP)
MOV R1,-(SP)
MOV R5,-(SP)

MOV #KIPAR0,R0
MOV NEWBANK,R1
ASH #9.,R1 ;BANK 1 STARTS AT 100,000 LESS 6 LSB'S
;FOR MEMORY MANAGEMENT = 1000

1\$: MOV #4,R5
MOV R1,(R0)+ ;SETUP KIPAR(0-3)
ADD #200,R1
SOB R5,1\$
POP R5,R1,R0

MOV (SP)+,R5
MOV (SP)+,R1
MOV (SP)+,R0

RETURN

MAPKERNAL:SUBTST <<SUBR MAP KERNAL PARS 4 AND 5 TO A BANK>>

;SUBTEST SUBR MAP KERNAL PARS 4 AND 5 TO A BANK

10210
10211 037740 013705 002102
10212 037744 072527 000011
10213 037750 013737 172350 002272
10214 037756 010537 172350
10215 037762 062705 000200
10216 037766 010537 172352
10217 037772 000207

MOV BANK,R5 ;MOV BANK NUMBER TO R5
ASH #9.,R5 ;R5 ENTERS 100000 LESS SHIFT TO CREATE MAPPING
MOV KIPAR4,SAVPAR ;SAVE OLD PAR
MOV R5,KIPAR4 ;GET NEW PAR'S
ADD #200,R5
MOV R5,KIPAR5
RETURN

10218
10219 037774

NEWLOAD:SUBTST <<SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK>>
;*****
;*SUBTEST SUBR SETUP KERNEL PAR'S FOR NEW LOADER BANK
;*****

10220
10221 037774

;RO CONTAINS THE DESTINATION BANK
PUSH RO,R1

037774 010046
037776 010146

MOV RO,-(SP)
MOV R1,-(SP)

10222 040000 012701 172350
10223 040004 072027 000011

MOV #KIPAR4,R1
ASH #9.,RO

;BANK 1 STARTS AT 100000 LESS 6 LSB'S (1000)
;SETUP KIPAR4

10224 040010 010021
10225 040012 062700 000200

MOV RO,(R1)+
ADD #200,RO

;SETUP KIPARS

10226 040016 010021
10227 040020

MOV RO,(R1)+
POP R1,RO

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

040020 012601
040022 012600

10228 040024 000207
10229

RETURN

10230 040026

UNMAP: SUBTST <<SUBR UNMAP KERNAL PAR'S 4 AND 5>>
;*****
;*SUBTEST SUBR UNMAP KERNAL PAR'S 4 AND 5
;*****

10231 040026 013737 002272 172350
10232 040034 062737 000200 002272

MOV SAVPAR,KIPAR4
ADD #200,SAVPAR

;RESTORE KIPAR4
;ADD 200 FOR NEXT PAR

10233 040042 013737 002272 172352
10234 040050 000207

MOV SAVPAR,KIPARS
RETURN

;RESTORE KIPARS
;

10237 040052

EXBANK: SUBTST <<SUBR EXAMINE BANK>>

;SUBTEST SUBR EXAMINE BANK

10238
10239
10240
10241
10242
10243
10244
10245
10246
10247
10248
10249
10250
10251
10252
10253
10254
10255
10256

;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.

10257 040052

PUSH R0,R1,R2

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

040052 010046
040054 010146
040056 010246

10258 040060

CLEAR MKFLAG,KPFLAG

CLR MKFLAG
CLR KPFLAG

040060 005037 002120
040064 005037 002114

10259 040070

SET ACFLAG

MOV #-1,ACFLAG

040070 012737 177777 002116

10260 040076

CLEAR PFLAG,RRFLAG,BMFLAG

CLR PFLAG
CLR RRFLAG
CLR BMFLAG

040076 005037 002122
040102 005037 002124
040106 005037 002130

10261 040112

CLEAR INTFLAG,INT64K,SKIPMK

CLR INTFLAG
CLR INT64K
CLR SKIPMK

040112 005037 002136
040116 005037 002140
040122 005037 002342

10262 040126

MOV BANK,R1

040126 013701 002102

10263 040132

ASL R1

040132 006301

10264 040134

ASL R1 ;R1 <- R1 * 4

040134 006301

10265 040136

MOV R1,BANKINDEX

040136 010137 002104

10266 040142

BIT #BIT6,CONFIG(R1) ;PROTECTED REGION OF ECC MEMORY?

040142 032761 000100 002664

10267 040150

BEQ 1\$;NO - SKIP

040150 001403

10268 040152

SET KPFLAG

MOV #-1,KPFLAG

040152 012737 177777 002114

10269 040160

1\$: MOV #BIT1,R0

040160 012700 000002

10270 040164

IF R0 SET.IN CPUBIT AND R0 OFF.IN CONFIG(R1)

040164 030037 002106

BIT R0,CPUBIT

040170 001405

BEQ L374

040172 030061 002664

BIT R0,CONFIG(R1)

040176 001002

BNE L374

10271 040200

CLR ACFLAG

040200 005037 002116

10272 040204

END ;OF IF R0

040204

L374:::~::~

```

10273 040204 005737 002116      TST      ACFLAG      ;ACTIVE MEMORY?
10274 040210 001415                BEQ      2$          ;BRANCH IF NOT
10275 040212 016102 002666      MOV      CONFIG+2(R1),R2
10276 040216 000302                SWAB     R2
10277 040220 042702 177770      BIC      #1C7,R2     ;ISOLATE MEM TYPE BITS
10278 040224 020227 000003      CMP      R2,#3       ;IS THIS AN ILLEGAL MEM TYPE?
10279 040230 003405                BLE     2$          ;BRANCH IF NOT
10280 040232                SET      BMFLAG     ;SET BAD BANK FLAG
      040232 012737 177777 002130                MOV     #-1,BMFLAG
10281 040240 000137 040446                JMP     ENEXBK      ;JUMP OVER REST OF FLAG TESTS
10282 040244 032761 000400 002666 2$:      BIT     #BIT8,CONFIG+2(R1) ;IS THERE ECC THERE?
10283 040252 001403                BEQ     3$          ;NO - SKIP
10284 040254                SET      MKFLAG     ;YES - SET MKFLAG
      040254 012737 177777 002120                MOV     #-1,MKFLAG
10285 040262 032761 000200 002664 3$:      BIT     #BIT7,CONFIG(R1) ;BANK = PROGRAM SPACE?
10286 040270 001406                BEQ     5$          ;NO - SKIP
10287 040272                SET      PFLAG,RRFLAG
      040272 012737 177777 002122                MOV     #-1,PFLAG
      040300 012737 177777 002124                MOV     #-1,RRFLAG
10288 040306 005737 002126 5$:      TST     RLFLAG     ;IS PROGRAM RELOCATED?
10289 040312 001402                BEQ     6$          ;NO - SKIP
10290 040314 005137 002124                COM     RRFLAG     ;YES - COMPLEMENT RELOCATION REQUIRED FLAG
10291 040320 032761 000001 002664 6$:      BIT     #BIT0,CONFIG(R1) ;ERRORS PRESENT IN THIS BANK?
10292 040326 001403                BEQ     8$          ;NO - SKIP
10293 040330                SET      BMFLAG
      040330 012737 177777 002130                MOV     #-1,BMFLAG
10294 040336 005737 002600 8$:      TST     WORST     ;IS THIS A WORST FIRST PASS?
10295 040342 001002                BNE     9$          ;YES - SKIP
10296 040344 005137 002130                COM     BMFLAG     ;NO - COMPLEMENT BAD MEMORY FLAG
10297 040350 9$:      IF SELONLY IS TRUE AND #BIT14 OFF.IN CONFIG+2(R1)
      040350 005737 002002                TST     SELONLY
      040354 001407                BEQ     L375
      040356 032761 040000 002666                BIT     #BIT14,CONFIG+2(R1)
      040364 001003                BNE     L375
10298 040366                SET      RRFLAG
      040366 012737 177777 002124                MOV     #-1,RRFLAG
10299 040374                END ;OF IF SELONLY
      040374                L375:;
10300 040374 032761 010000 002666                BIT     #BIT12,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED?
10301 040402 001421                BEQ     ENEXBK     ;BRANCH IF IT IS NOT
10302 040404                SET      INTFLAG
      040404 012737 177777 002136                MOV     #-1,INTFLAG
10303 040412 032761 004000 002666                BIT     #BIT11,CONFIG+2(R1) ;IS THIS BANK INTERLEAVED WITH 64K BOARDS?
10304 040420 001403                BEQ     10$        ;BRANCH IF IT IS NOT
10305 040422                SET      INT64K
      040422 012737 177777 002140                MOV     #-1,INT64K
10306 040430 032761 000040 002664 10$:      BIT     #BIT5,CONFIG(R1) ;SHOULD THIS BANK BE TESTED?
10307 040436 001403                BEQ     ENEXBK     ;BRANCH IF IT SHOULD
10308 040440                SET      SKIPMK
      040440 012737 177777 002342                MOV     #-1,SKIPMK
10309 040446                ENEXBK: POP     R2,R1,R0 ;RESTORE REGISTERS
      040446 012602                MOV     (SP)+,R2
      040450 012601                MOV     (SP)+,R1
      040452 012600                MOV     (SP)+,R0
10310 040454 000207                RETURN

```

10313 040456

BANKOK: SUBTST <<SUBR BANK OK?>>
;*****
;*SUBTEST SUBR BANK OK?
;*****

10314
10315
10316

;TEST TO INSURE THAT THE TYPE OF MEMORY IN THE PRESENT BANK
;IS OF THE TYPE WE ARE TESTING "TMFLAG".
;RESULT IS RETURNED IN THE CONDITION CODES (OK = (=0)).

10317 040456 013700 002134
10318 040462 005100
10319 040464 013701 002120
10320 040470 074001
10321 040472 000207

MOV TMFLAG,R0
COM R0
MOV MKFLAG,R1
XOR R0,R1
RETURN ;OK = (=OK)

10322
10323 040474
10324 040474

INCRPT:
INCPAT: SUBTST <<SUBR INCREMENT PATTERN TESTING >>
;*****
;*SUBTEST SUBR INCREMENT PATTERN TESTING
;*****

10325
10326
10327 040474 005237 002112
10328 040500 022737 000030 002112
10329 040506 000207

;INCREMENT THE PATTERN & SET UP THE CONDITION CODES
;RESULT - Z BIT SET INDICATES OVERFLOW
INC PATTERN
CMP #30,PATTERN ;SET UP CONDITION CODES
RETURN ;NOT EQUAL TO ZERO IS GOOD (NO OVERFLOW)

10330
10331 040510
10332 040510

SETPAT:
HIPAT: SUBTST <<SUBR SET HIGHEST PATTERN TESTING TYPE>>
;*****
;*SUBTEST SUBR SET HIGHEST PATTERN TESTING TYPE
;*****

10333 040510 012737 000027 002112
10334 040516 000207
10335
10336 040520

MOV #27,PATTERN ;SET HIGHEST PATTERN
RETURN

INCBNK: SUBTST <<SUBR INCREMENT BANK & TEST>>
;*****
;*SUBTEST SUBR INCREMENT BANK & TEST
;*****

10337
10338 040520 005237 002102
10339 040524 023737 002556 002102
10340 040532 000207

;RESULTS RETURNED IN CONDITION CODES
INC BANK
CMP LASTBANK,BANK ;TOO FAR?
RETURN

10343 040534

QUIT: SUBTST <<QUIT ROUTINE>>

;SUBTEST QUIT ROUTINE

10344
10345
10346
10347
10348
10349

;INITIALIZE ALL CSR'S
;UNRELOCATE IF NECESSARY
;FLUSH OUT ANY DBE'S
;TURN OFF MEMORY MANAGEMENT
;TURN OFF THE Q-BUS MAP
;HALT

10350 040534 104472
10351 040536 012737 040600 000004

ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
SET4 #QUIT1 ;TRAPS TO 4 GOTO QUIT1
MOV #QUIT1,4
.DSABL CRF

10352 040544
040544 005737 002126
040550 001402
040552 004737 037214
040556

IF RLFLAG IS TRUE THEN \$CALL UNRELOCATE

TST RLFLAG
BEQ L376
JSR PC,UNRELOCATE

L376:::~::~

10353 040556 004737 022200
10354 040562 104421
10355 040564 005737 002454
10356 040570 001003
10357 040572 042737 000040 172516
10358 040600 000005
10359 040602 000000
10360

CALL MT0030 ;FLUSH OUT DBE'S
DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
TST NQ22BIT ;IS THIS AN 11/83,11/23-B OR 11/23?
BNE QUIT1
BIC #BIT5,MMR3 ;TURN OFF THE Q-BUS MAP
QUIT1: RESET
HALT


```

10363 040604          EXIT:  SUBTST  <<HALT PROGRAM>>
;*****
;*SUBTEST            HALT PROGRAM
;*****
10364 040604 004737 040636          CALL  SHUTUP
10365 040610          EXIT2:  IF APTFLAG IS TRUE OR ACTFLAG IS TRUE
      040610 005737 002352          TST  APTFLAG
      040614 001003          BNE  L377
      040616 005737 002350          TST  ACTFLAG
      040622 001402          BEQ  L400
      040624          L377:::;
10366 040624 000777          BR
10367 040626          ELSE
      040626 000403          BR  L401
      040630          L400:::;
10368 040630 000000          $EXHALT: HALT
10369 040632 000137 003670          JMP  START
10370 040636          END ;OF IF APTFLAG
      040636          L401:::;
10371
10372 040636          SHUTUP: SUBTST  <<SHUTDOWN DIAGNOSTIC>>
;*****
;*SUBTEST            SHUTDOWN DIAGNOSTIC
;*****
      ;INITIALIZE ALL CSR'S
10373          ;UNRELOCATE
10374          ;FLUSH OUT DBE'S
10375          ;RESTORE LOADERS
10376          ;TURN OFF MEMORY MANAGEMENT
10377          ;UNMAP THE Q-BUS MAP
10378          ;UNMAP THE Q-BUS MAP
10382 040636 104472          ECCINIT ;TRAP ON DOUBLE BIT ERRORS (NORMAL)
10383 040640          IF RLFLAG IS TRUE THEN $CALL UNRELOCATE
      040640 005737 002126          TST  RLFLAG
      040644 001402          BEQ  L402
      040646 004737 037214          JSR  PC,UNRELOCATE
      040652          L402:::;
10384 040652          IF QUICK IS FALSE
      040652 005737 002436          TST  QUICK
      040656 001002          BNE  L403
10385 040660 004737 022200          CALL  MT0030 ;FLUSH OUT DBE'S
10386 040664          END ;OF IF QUICK
      040664          L403:::;
10387 040664 012700 000001          MOV   #1,R0 ;DESTINATION BANK
10388 040670 013701 002576          MOV   LOADHOME,R1 ;SOURCE BANK
10389 040674 004737 037442          CALL  BANKMOV
10390 040700 104421          DEENERGIZE ;TURN OFF MEMORY MANAGEMENT
10391 040702 005737 002454          TST  NO22BIT ;DOES THIS PDP-11 HAVE 22-BIT ADDR?
10392 040706 001003          BNE  1$ ;BRANCH IF NOT
10393 040710 042737 000040 172516          BIC  #BIT5,MMR3 ;TURN OFF Q-BUS MAP
10397 040716 000207          1$:  RETURN
10398
10399 040720          APTDOWN:SUBTST  <<APT SHUTDOWN SEQUENCE>>
;*****
;*SUBTEST            APT SHUTDOWN SEQUENCE
;*****
10400 040720          MAP   #0 ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK #0
      040720 010346          MOV  R3,-(SP)

```

```

040722 012703 000000      MOV    #0,R3
040726 004737 035716      CALL  MAPPER
                                .DSABL CRF
10401  040732 012603      TESTAREA                      MOV (SP),R3
040734 053737 002552 177776  BIS   TESTMODE,PSW          ;ENTER TEST MODE
                                .DSABL CRF                          ;GO TO SYSTEM TEST MODE
10402 040742 012737 040720 060024  MOV   #APTDOWN,FIRST+24
10403 040750 012737 000340 060026  MOV   #340,FIRST+26
10404 040756 012737 000000 120720  MOV   #0,FIRST+APTDOWN
10405 040764 104417
10406 040766 000000      KERNEL                      ;ENTER KERNEL MODE
                                APTHLT: HALT

```

10409 040770

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

10410
10411
10412
10413
10414
10415
10416

BLOCK1: PUSH R0,R1,R2

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

040770 010046
040772 010146
040774 010246
10417 040776 012702 177640
10418 041002 012701 000020
10419 041006 000413

MOV #FASTCITY,R2
MOV #16.,R1
BR 3\$

10420
10421

BLOCK2: PUSH R0,R1,R2

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

041010 010046
041012 010146
041014 010246
10422 041016 012701 000020
10423 041022 000404

MOV #16.,R1
BR 2\$

10424
10425

BLOCK3: PUSH R0,R1,R2

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

041024 010046
041026 010146
041030 010246
10426 041032 012501
10427 041034 012502
10428 041036 012500

MOV (R5)+,R1
2\$: MOV (R5)+,R2
3\$: MOV (R5)+,R0

10429
10430 041040 012022
10431 041042 077102
10432 041044

1\$: MOV (R0)+,(R2)+
SOB R1,1\$
POP R2,R1,R0

041044 012602
041046 012601
041050 012600
10433 041052 000205
10434

RTS R5
.DSABL LSB

MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0

```

10436
10437
10438 041054

10439 041054 104415
10440 041056 104401 066116
      041056

10441
10442 041062
      041062 005737 002126
      041066 001003
      041070 005737 002426
      041074 001404
      041076
10443 041076 104401 067557
      041076

10444 041102 104416
10445 041104 000207
10446 041106
      041106
10447 041106 005737 002544
10448 041112 001402
10449 041114
      041114 013746 177746
10450 041120
      041120 013746 002152
      041124 013746 002006
10451 041130 104424
10452 041132
      041132 012737 177777 002006
10453 041140
      041140 104401 067007

10454 041144 104414
10455 041146
      041146 012600
10456 041150 020027 000022
10457 041154 101403
10458 041156
      041156 104401 066137

10459 041162 000766
10460 041164
      041164 010046
      041166 006316
      041170 004737 041242
10461 041174 041252
10462 041176 041354
10463 041200 041464
10464 041202 041632
10465 041204 042106
10466 041206 042426
10467 041210 043344

```

```

.SBTTL FIELD SERVICE MODE
FIELDSERVICE:SUBTST <<SUBR FIELD SERVICE COMMAND MODE>>
;*****
;*SUBTEST SUBR FIELD SERVICE COMMAND MODE
;*****
SAVREG
TYPE MSG020 ;FIELD SERVICE COMMAND MODE
TYPEIT ,MSG020
.DSABL CRF

IF RLFLAG IS TRUE OR NOFSMODE IS TRUE

TST RLFLAG
BNE L404
TST NOFSMODE
BEQ L405
L404:;;;;;
TYPE MSG048 ;NOT AVAILABLE NOW - TRY LATER!
TYPEIT ,MSG048
.DSABL CRF
RESREG
RETURN
END ;OF IF RLFLAG
L405:;;;;;
TST CACHKN
BEQ 1$
PUSH CONTRL ;SAVE CACHE STATUS
1$: PUSH CSRNO,KAMIKAZE ;SAVE CSR & KAMIKAZE STATUS
MOV CONTRL,-(SP)
MOV CSRNO,-(SP)
MOV KAMIKAZE,-(SP)
CACHOFF
SET KAMIKAZE ;TURN CACHE OFF
MOV #-1,KAMIKAZE
FS1: TYPE MSG026 ;COMMAND:
TYPEIT ,MSG026
.DSABL CRF
RDDEC ;READ A DECIMAL NUMBER
POP RO ;COMMAND --> RO
MOV (SP)+,RO
CMP RO,#18.
BLOS 1$
TYPE MSG021
TYPEIT ,MSG021
.DSABL CRF
BR FS1
1$: CASE RO
MOV RO,-(SP)
ASL @SP
JSR PC,L406

FSCMD0 ;EXIT FIELD SERVICE COMMANDS
FSCMD1 ;READ CSR
FSCMD2 ;LOAD CSR
FSCMD3 ;EXAMINE MEMORY
FSCMD4 ;MODIFY MEMORY
FSCMD5 ;SELECT BANK & PATTERN
FSCMD6 ;TYPE CONFIGURATION MAP

```


10468 041212 043352
 10469 041214 043644
 10470 041216 044072
 10471 041220 044364
 10472 041222 044412
 10473 041224 044434
 10474 041226 044454
 10475 041230 044476
 10476 041232 044514
 10477 041234 044600
 10478 041236 044642
 10479 041240 044656
 10480 041242
 041242
 041242 062616
 041244 013646
 041246 004736
 10481 041250 000733

FSCMD7
 FSCMD8
 FSCMD9
 FCMD10
 FCMD11
 FCMD12
 FCMD13
 FCMD14
 FCMD15
 FCMD16
 FCMD17
 FCMD18
 END ;OF CASE

;SOB-A-LONG TEST
 ;ERROR SUMMARY
 ;REFRESH TEST
 ;SET FILL COUNT
 ;ENTER KAMIKAZE MODE
 ;EXIT KAMIKAZE MODE
 ;TURN CACHE OFF
 ;TURN CACHE ON
 ;TEST ONLY SELECTED BANKS
 ;RESUME TESTING ALL BANKS
 ;ENABLE TRACE
 ;DISABLE TRACE

L406:::;:
 ADD (SP),@SP
 MOV @ (SP),-(SP)
 JSR PC,@(SP),

BR FS1

```

10484 041252
FSCMD0: SUBTST <<COMMAND 0 EXIT>>
;*****
;*SUBTEST COMMAND 0 EXIT
;*****
10485 041252 104401 070535
041252 TYPE MSG103 ;LEAVING FIELD SERVICE MODE
041252 TYPEIT ,MSG103
10486 041256 062706 000002 .DSABL CRF
10487 041262 ADD #2,SP
041262 IF SKIPKAMI IS TRUE
041262 TST SKIPKAMI
041266 001405 BEQ L407
10488 041270 062706 000002 ADD #2,SP ;THROW AWAY OLD KAMIKAZE FLAG
10489 041274 005037 002010 CLR SKIPKAMI
10490 041300 ELSE
041300 BR L410
041302 L407:::
10491 041302 POP KAMIKAZE ;RESTORE OLD KAMIKAZE FLAG
041302 MOV (SP)+,KAMIKAZE
10492 041306 END ;OF IF SKIPKAMI
041306 L410:::
10493 041306 POP CSRNO MOV (SP)+,CSRNO
041306 012637 002152 TST CACHKN
10494 041312 005737 002544 BEQ RESO
10495 041316 001414 IF CACHKN EQ CACHKF ;IF CACHE IS OFF
10496 041320 023737 002544 002550 CMP CACHKN,CACHKF
041320 001003 BNE L411
10497 041330 062706 000002 ADD #2,SP ;THROW AWAY CACHE STATUS
10498 041334 ELSE
041334 000405 BR L412
041336 L411:::
10499 041336 005737 002544 TST CACHKN
10500 041342 001402 BEQ RESO
10501 041344 POP CONTRL ;RESTORE CACHE STATUS
041344 012637 177746 MOV (SP)+,CONTRL
10502 041350 END ;OF IF CACHKN
041350 L412:::
10503 041350 104416 RESO: RESREG
10504 041352 000207 RETURN
10505
10506 041354
FSCMD1: SUBTST <<FS COMMAND 1 READ CSR>>
;*****
;*SUBTEST FS COMMAND 1 READ CSR
;*****
10507 041354 004737 044670 CALL WHICHCSR
10508 041360 010637 002306 MOV SP,FSSTACK
10509 041364 012737 041430 000004 SET4 #RES1 ;TRAPS TO 4 GOTO RES1
041364 MOV #RES1,4
.DSABL CRF
10510 041372 104426 READCSR
10511 041374 012737 177777 002430 SET NOERROR
041374 MOV #-1,NOERROR
10512 041402 104026 ERROR +26 ;USE ERROR ROUTINE FOR PRINTOUT
10513 041404 012737 034114 000004 RES4 ;RESET TRAPS TO 4 TO DEFAULT
041412 022737 000005 004010 MOV #TIMEOUT,4
041420 001002 CMP #5,PROTYP ;IS THIS AN 11/83/84 ?
BNE 101$ ;BRANCH IF NOT

```

```

041422 005037 177766          CLR    CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
041426          101$:          ;THAT A EXPECTED TRAP COULD HAVE SET

10514 041426 000207          .DSABL CRF
10515 041430          RETURN
041430 104401 066763          RES1: TYPE MSG025          ;THIS CSR DOES NOT EXIST
                                TYPEIT ,MSG025
                                .DSABL CRF
10516 041434 013706 002306          MOV    FSSTACK,SP
10517 041440          RES4          ;RESET TRAPS TO 4 TO DEFAULT
041440 012737 034114 000004          MOV    #TIMEOUT,4
041446 022737 000005 004010          CMP    #5,PROTYP          ;IS THIS AN 11/83/84 ?
041454 001002          BNE    101$          ;BRANCH IF NOT
041456 005037 177766          CLR    CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
041462          101$:          ;THAT A EXPECTED TRAP COULD HAVE SET

10518 041462 000207          .DSABL CRF
                                RETURN

```

```

10521 041464          FSCMD2: SUBTST <<FS  COMMAND 2  LOAD CSR>>
;*****
;*SUBTEST          FS  COMMAND 2  LOAD CSR
;*****
10522 041464 004737 044670          CALL  WHICHCSR
10523 041470 010637 002306          MOV   SP,FSSTACK
10524 041474 012737 041576 000004  SET4  #RES2          ;TRAPS TO 4 GOTO RES2
                                MOV   #RES2,4
                                .DSABL CRF
10525 041502 104426          READCSR
10526 041504 104401 067021          TYPE  MSG027
                                TYPEIT ,MSG027
                                .DSABL CRF
10527 041510 012737 177777 002430  SET   NOERROR
                                MOV   #-1,NOERROR
10528 041516 104026          ERROR +26          ;USE ERROR ROUTINE FOR PRINTOUT
10529 041520 012737 034114 000004  RES4          ;RESET TRAPS TO 4 TO DEFAULT
                                MOV   #TIMEOUT,4
                                CMP   #5,PROTYP          ;IS THIS AN 11/83/84 ?
                                BNE   101$          ;BRANCH IF NOT
                                CLR   CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                101$:
                                ;THAT A EXPECTED TRAP COULD HAVE SET
                                ;FIRST CSR WORD
                                .DSABL CRF
                                TYPE  MSG023
                                TYPEIT ,MSG023
                                .DSABL CRF
10530 041542 104401 066747          RDOCT
                                POP   CSR          ;READ AN OCTAL NUMBER
                                ;PUT IN IN LOC "CSR"
                                MOV (SP)+,CSR
10531 041546 104413          LOADCSR
10532 041550 012637 002150          READCSR
10533 041554 104425          TYPE  MSG028
10534 041556 104426          TYPEIT ,MSG028
10535 041560 104401 067036          .DSABL CRF
                                SET   NOERROR
10536 041564 012737 177777 002430  ERROR +26          ;USE FOR PRINTOUT - NOT AN ERROR
                                RETURN
10537 041572 104026          RES2:
10538 041574 000207          TYPE  MSG025
10539 041576 104401 066763          TYPEIT ,MSG025
                                .DSABL CRF
10540 041602 013706 002306          MOV   FSSTACK,SP
10541 041606 012737 034114 000004  RES4          ;RESET TRAPS TO 4 TO DEFAULT
                                MOV   #TIMEOUT,4
                                CMP   #5,PROTYP          ;IS THIS AN 11/83/84 ?
                                BNE   101$          ;BRANCH IF NOT
                                CLR   CPUERR          ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                101$:
                                ;THAT A EXPECTED TRAP COULD HAVE SET
                                .DSABL CRF
10542 041630 000207          RETURN

```



```

10545 041632          FSCMD3: SUBTST <<FS  COMMAND 3  EXAMINE MEMORY>>
;*****
;*SUBTEST FS  COMMAND 3  EXAMINE MEMORY
;*****
10546 041632          PUSH BANK,NOPAR,PARHERE.4
041632 013746 002102          MOV BANK,-(SP)
041636 013746 002076          MOV NOPAR,-(SP)
041642 013746 002304          MOV PARTHERE,-(SP)
041646 013746 000004          MOV 4,-(SP)
10547 041652 012737 000002 002076  MOV #2,NOPAR ;INDICATE PARITY ACTION
10548 041660 104401 067052          TYPE MSG029 ;EXAMINE MEMORY
                                TYPEIT ,MSG029
                                .DSABL CRF
10549 041664 104401 067111          1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-17775776)??
                                TYPEIT ,MSG031
                                .DSABL CRF
10550 041670 104413          RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
10551 041672 013737 056146 002102  MOV $HIOCT,BANK ;PUT MSB'S IN BANK
10552 041700 012600          POP RO ;PUT LSB'S IN RO
                                MOV (SP)+,RO
10553 041702 000241          CLC
10554 041704 006100          ROL RO
10555 041706 006137 002102          ROL BANK
10556 041712 000241          CLC
10557 041714 006000          ROR RO
10558 041716 023737 002102 002556  CMP BANK, LASTBANK ;CHECK FOR BANK TOO HIGH
10559 041724 003357          BGT 1$ ;BRANCH IF TRUE
10560 041726 062700 060000          ADD #FIRST,RO
10561 041732 032700 000001          BIT #BIT0,RO ;CHECK FOR ODD ADDRESS
10562 041736 001352          BNE 1$ ;BRANCH IF ODD ADDRESS
10563 041740 020027 157776          CMP RO,#LAST ;CHECK FOR ADDRESS OVER 16K
10564 041744 101347          BHI 1$ ;BRANCH IF OVER 16K
10565 041746 012737 042020 002304  MOV #3$,PARTHERE ;INCASE OF ABORTS
10566 041754 012737 042026 000004  SET4 #4$ ;TRAPS TO 4 GOTO 4$
                                MOV #4$,4
                                .DSABL CRF
10567 041762 010346          MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
                                MOV R3,-(SP)
041762 013703 002102          MOV BANK,R3
041770 004737 035716          CALL MAPPER
                                .DSABL CRF
                                MOV (SP)+,R3
10568 041774 012603          TESTAREA ;ENTER TEST MODE
10568 041776 053737 002552 177776  BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
                                .DSABL CRF
10569 042004 011001          MOV (RO),R1
10570 042006 104417          KERNEL ;ENTER KERNEL MODE
10571 042010          TYPOCS R1
                                MOV R1,-(SP) ;SAVE R1 FOR TYPEOUT
                                TYPOS ;GO TYPE--OCTAL ASCII
                                .BYTE 6 ;TYPE 6 DIGITS
                                .BYTE 0 ;SUPPRESS LEADING ZEROS
                                .DSABL CRF
10572 042016 000410          BR EXCMD3
10573
10574 042020 104401 067151          3$: TYPE MSG032 ;PARITY ABORT
                                TYPEIT ,MSG032

```

```

10575 042024 000405          .DSABL  CRF
10576 042024 000405          BR      EXCMD3
10577 042026 062706 000004    4$:    ADD    #4,SP          ;FIX STACK
10578 042032 104401 067170    TYPE   MSG033         ;TIMEOUT TRAP
                                TYPEIT  ,MSG033
                                .DSABL  CRF
10579 042036 000400          BR      EXCMD3
10580 042040 104417          EXCMD3: KERNEL          ;ENTER KERNEL MODE
10582 042042 012637 000004    POP    4,PARTHERE,NOPAR,BANK
                                MOV (SP)+,4
                                MOV (SP)+,PARTHERE
                                MOV (SP)+,NOPAR
                                MOV (SP)+,BANK
                                042042 012637 002304
                                042046 012637 002076
                                042052 012637 002102
10583 042062 012737 034114 000004    RES4          ;RESET TRAPS TO 4 TO DEFAULT
                                MOV    #TIMEOUT,4
                                CMP    #5,PROTYP      ;IS THIS AN 11/83/84 ?
                                BNE    101$           ;BRANCH IF NOT
                                CLR    CPUERR         ;CLEAR OUT THE CPU ERROR REGISTER BITS
                                042070 022737 000005 004010
                                042076 001002
                                042100 005037 177766    101$:
                                ;THAT A EXPECTED TRAP COULD HAVE SET
                                042104
10584 042104 000207          .DSABL  CRF
                                RETURN

```

```

10587 042106          FSCMD4: SUBTST <<FS  COMMAND 4  MODIFY MEMORY>>
;*****
;SUBTEST  FS  COMMAND 4  MODIFY MEMORY
;*****
10588 042106          PUSH  BANK,NOPAR,PARTHERE,4
042106 013746 002102          MOV BANK,-(SP)
042112 013746 002076          MOV NOPAR,-(SP)
042116 013746 002304          MOV PARTHERE,-(SP)
042122 013746 000004          MOV 4,-(SP)
10589 042126 012737 000003 002076  MOV #3,NOPAR ;INDICATE PARITY ACTION
10590 042134 104401 067276  TYPE MSG036 ;MODIFY MEMORY
TYPEIT ,MSG036
.DSABL CRF
10591 042140 104401 067111 1$: TYPE MSG031 ;PHYSICAL ADDRESS (0-17775776)??
TYPEIT ,MSG031
.DSABL CRF
10592 042144 104413 RDOCT ;READ OCTAL NUMBER ONTO STACK & $HIOCT
10593 042146 013737 056146 002102  MOV $HIOCT,BANK ;PUT MSB'S IN BANK
10594 042154 012600 POP RO ;PUT LSB'S IN RO
042154 000241          CLC ;MOV (SP)+,RO
10595 042156 000241          ROL RO
10596 042160 006100          ROL BANK
10597 042162 006137 002102          CLC
10598 042166 000241          ROR RO
10599 042170 006000          IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
10600 042172 023737 002102 002556  CMP BANK, LASTBANK
042172 003357          BGT 1$
10601 042202 062700 060000          ADD #FIRST,RO
10602 042206 032700 000001          IF #BITO SET.IN RO THEN GOTO 1$ ;CHECK FOR ODD ADDRESS
042206 001352          BIT #BITO,RO
042212 001352          BNE 1$
10603 042214 020027 157776          IF RO HI #LAST THEN GOTO 1$ ;CHECK FOR ADDRESS OVER
042214 101347          CMP RO,#LAST
042220 101347          BHI 1$
10604 042222 012737 042270 002304  MOV #3$,PARTHERE ;INCASE OF ABORTS
10605 042230 012737 042276 000004  SET4 #4$ ;TRAPS TO 4 GOTO 4$
MOV #4$,4
.DSABL CRF
10606 042236 010346 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
042236 010346          MOV R3,-(SP)
042240 013703 002102          MOV BANK,R3
042244 004737 035716          CALL MAPPER
.DSABL CRF
042250 012603          MOV (SP)+,R3
10607 042252 104511 INVALIDATE
10608 042254 053737 002552 177776  TESTAREA ;ENTER TEST MODE
BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
.DSABL CRF
10609 042262 011001          MOV (RO),R1
10610          ;GETTING HERE MEANS WE GOT LUCKY - NO TRAPS
10611 042264 104417          KERNEL ;ENTER KERNEL MODE
10612 042266 000410          BR 5$
10613
10614 042270 104401 067151 3$: TYPE MSG032 ;PARITY ABORT
TYPEIT ,MSG032
.DSABL CRF

```



```

10615 042274 000431          BR      EXCMD4          ;EXIT
10616
10617 042276 062706 000004    4$:   ADD      #4,SP          ;FIX STACK
10618 042302 104401 067170    TYPEIT  ,MSG033        ;TIMEOUT TRAP
      042302          .DSABL  CRF
10619 042306 000424          BR      EXCMD4          ;EXIT
10620
10621 042310 104401 067315    5$:   TYPEIT  ,MSG037        ;OLD DATA WAS
      042310          .DSABL  CRF
10622 042314 010146    TYPOCS  R1          ;PRINT IT
      042314 104403    MOV      R1,-(SP)      ;;SAVE R1 FOR TYPEOUT
      042316 104403    TYPOS          ;;GO TYPE--OCTAL ASCII
      042320 006          .BYTE 6          ;;TYPE 6 DIGITS
      042321 000          .BYTE 0          ;;SUPPRESS LEADING ZEROS
      .DSABL  CRF
10623 042322 104401 067352    TYPEIT  ,MSG039        ;INPUT NEW DATA
      042322          .DSABL  CRF
10624 042326 104413          RDOCT          ;READ ON OCTAL NUMBER ONTO THE STACK
10625 042330 012601    POP      R1          ;GET NEW NUMBER
10626 042332 053737 002552 177776  TESTAREA          ;ENTER TEST MODE
      042332          BIS      TESTMODE,PSW      ;GO TO SYSTEM TEST MODE
      .DSABL  CRF
10627 042340 010110    MOV      R1,(R0)      ;PUT IT IN MEMORY
10628 042342 011001    MOV      (R0),R1     ;READ IT AGAIN
10629 042344 104417    KERNEL          ;ENTER KERNEL MODE
10630 042346 104401 067334    TYPEIT  ,MSG038        ;DATA IS NOW
      042346          .DSABL  CRF
10631 042352 010146    TYPOCS  R1          ;PRINT IT
      042352 104403    MOV      R1,-(SP)      ;;SAVE R1 FOR TYPEOUT
      042354 104403    TYPOS          ;;GO TYPE--OCTAL ASCII
      042356 006          .BYTE 6          ;;TYPE 6 DIGITS
      042357 000          .BYTE 0          ;;SUPPRESS LEADING ZEROS
      .DSABL  CRF
10632
10633 042360 104417    EXCMD4: KERNEL          ;ENTER KERNEL MODE
10634 042362 012637 000004    POP      4,PARTHERE,NOPAR,BANK
      042362 012637 002304          MOV (SP)+,4
      042366 012637 002076          MOV (SP)+,PARTHERE
      042372 012637 002102          MOV (SP)+,NOPAR
      042376 012637 000004          MOV (SP)+,BANK
10635 042402 012737 034114 000004    RES4          ;RESET TRAPS TO 4 TO DEFAULT
      042402 022737 000005 004010    MOV      #TIMEOUT,4
      042410 001002          CMP      #5,PROTYP
      042416 005037 177766          BNE     101$
      042420          CLR      CPUERR
      042424          101$:          ;CLEAR OUT THE CPU ERROR REGISTER BITS
      .DSABL  CRF
10636 042424 000207          RETURN          ;THAT A EXPECTED TRAP COULD HAVE SET

```



```

10639 042426 FSCMD5: SUBTST <<FS COMMAND 5 SELECT BANK & PATTERN>>
;*****
;*SUBTEST FS COMMAND 5 SELECT BANK & PATTERN
;*****
10640 042426 PUSH BANK,PATTERN,TESTADD,PCBUMP,TKVEC,TKVEC+2
042426 013746 002102 MOV BANK,-(SP)
042432 013746 002112 MOV PATTERN,-(SP)
042436 013746 002412 MOV TESTADD,-(SP)
042442 013746 002326 MOV PCBUMP,-(SP)
042446 013746 000060 MOV TKVEC,-(SP)
042452 013746 000062 MOV TKVEC+2,-(SP)
10641 042456 010637 002306 MOV SP,FSSTACK ;SAVE LAST GOOD STACK POINTER
10642 042462 TYPE MSG040 ;SELECT BANK & PATTERN TEST
042462 104401 067374 TYPEIT ,MSG040
10643 042466 104401 067072 1$: .DSABL CRF ;BANK(0-177)?
042466 TYPE MSG030
TYPEIT ,MSG030
.DSABL CRF
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP BANK ;PUT IT IN BANK
10644 042472 104413 IF BANK GT LASTBANK THEN GOTO 1$ ;CHECK FOR BANK TOO HIGH
10645 042474 012637 002102 MOV (SP)+,BANK
042474 023737 002102 002556 CMP BANK,LASTBANK
042500 003367 BGT 1$
10647 042510 013701 002102 MOV BANK,R1
10648 042514 006301 ASL R1
10649 042516 006301 ASL R1
10650 042520 IF CPUBIT OFF.IN CONFIG(R1)
10651 042520 033761 002106 002664 BIT CPUBIT,CONFIG(R1)
042526 001003 BNE L413
10652 042530 TYPE MSG041 ;BANK NOT ACCESSABLE
042530 104401 067420 TYPEIT ,MSG041
.DSABL CRF
GOTO 1$
10653 042534 000754 BR 1$
10654 042536 END ;OF IF
042536 L413:;;;;;
10655 042536 2$: TYPE MSG042 ;PATTERN(0-45)?
10656 042536 104401 067445 TYPEIT ,MSG042
.DSABL CRF
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP PATTERN ;PUT IT IN PATTERN
10657 042542 104413 IF PATTERN GT #47 THEN GOTO 2$ ;CHECK FOR PATTERN TO HIGH
10658 042544 012637 002112 MOV (SP)+,PATTERN
042544 023727 002112 000047 CMP PATTERN,#47
042556 003367 BGT 2$
10660 042560 IF PATTERN EQ #0
042560 005737 002112 TST PATTERN
042564 001004 BNE L414
10661 042566 TYPE MSG043 ;PATTERN 0 DATA IS?
042566 104401 067463 TYPEIT ,MSG043
.DSABL CRF
RDOCT ;READ AN OCTAL NUMBER ONTO THE STACK
POP R2 ;PUT IT IN R2
10662 042572 104413 MOV (SP)+,R2
10663 042574 012602
042574

```

```

10664 042576          END ;OF IF
10665 042576          L414:::
10666
10667 042576          MAP      BANK      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
      042576 010346          ;MOV R3,-(SP)
      042600 013703 002102  MOV      BANK,R3
      042604 004737 035716  CALL     MAPPER
      .DSABL CRF
      042610 012603          MOV (SP)+,R3
10668 042612 104511
10669 042614 004737 040052  INVALIDATE
      042620 005737 002124  CALL     EXBANK      ;SET NEW MARGINS
      042624 001404          IF RRFLAG IS TRUE
      042626 104401 067617  TYPE     MSG049      ;BANK REQUIRES RELOCATION
      .DSABL CRF          TYPEIT ,MSG049
      JMP CMD5C          .DSABL CRF
10672 042632 000137 043246  END ;OF IF RRFLAG
10673 042636
10674 042636          TYPE     MSG046      ;TO ESCAPE TYPE ANY KEY!
      042636 104401 067505  TYPEIT ,MSG046
      .DSABL CRF          .DSABL CRF
10675 042642 013737 002152 002154  MOV      CSRNO,SAVCSR ;SAVE OLD CSR NUMBER
10676 042650 013702 002102  MOV      BANK,R2      ;
10677 042654 072227 000002  ASH     #2,R2          ;GENERATE INDEX INTO CONFIGURATION TABLE
10678 042660 016203 002664  MOV     CONFIG(R2),R3 ;R3 = LOW WORD OF CONFIGURATION TABLE FOR THIS BANK
10679 042664 072327 177770  ASH     #-10,R3       ;POSITION CSR CODE IN BITS 0-3
10680 042670 042703 177760  BIC     #1C17,R3      ;CLEAR ALL BUT THE CSR CODE
10681 042674 006303  ASL     R3             ;ADJUST CSR NUMBER
10682 042676 010337 002152  MOV     R3,CSRNO      ;
10683 042702 012737 043246 000060  MOV     #CMD5C,TKVEC
10684 042710 012737 000340 000062  MOV     #340,TKVEC+2
10685 042716 017700 137722  MOV     @TKB,R0
10686 042722 042737 000200 177776  BIC     #BIT7,PSW     ;KILL ANY OLD INTERRUPT
10687 042730 052777 000100 137704  BIS     #BIT6,@TKS    ;LOWER CPU PRIORITY TO 140
10688
10689
10690 042736          SET     HEADER,MUT    ;ENABLE KEYBOARD INTERRUPTS
      042736 012737 177777 002612  MOV     #-1,HEADER
      042744 012737 177777 002110  MOV     #-1,MUT
10691 042752 013701 002102          CMD5B: MOV     BANK,R1
10692 042756 006301  ASL     R1
10693 042760 006301  ASL     R1
10694 042762 005037 002240  CLR     SPLTCSR
10695 042766 005037 002264  CLR     PASFLG
10696 042772 012737 060000 002412  MOV     #FIRST,TESTADD
10697 043000 012737 060002 002414  MOV     #FIRST+2,TESTADD+2
10698 043006          IF #BIT12 SET.IN CONFIG+2(R1)
      043006 032761 010000 002666  BIT     #BIT12,CONFIG+2(R1)
      043014 001413          BEQ     L416
10699 043016 005237 002240          INC     SPLTCSR
10700 043022          MAP     BANK
      043022 010346          MOV R3,-(SP)
      043024 013703 002102  MOV     BANK,R3
      043030 004737 035716  CALL     MAPPER

```

```

      .DSABL CRF
10701 043034 012603          MOV (SP)+,R3
      043036 012737 120000 002414  MOV #120000,TESTADD+2
10702 043044          END; OF IF #BIT12
      043044          L416:;;;;;;
10703 043044          IF #SWO SET.IN @SWR
      043044 032777 000001 137564  BIT #SWO,@SWR
      043052 001402          BEQ L417
10704 043054 104470          ECCDIS          ;DISABLE ERROR CORRECTION
10705 043056          ELSE
      043056 000405          BR L420
      043060          L417:;;;;;;
10706 043060          PUSH CSRNO
      043060 013746 002152          CLRCR          ;CLEAR CSRS
10707 043064 104502          POP CSRNO
10708 043066          MOV (SP)+,CSRNO
      043066 012637 002152          END ;OF IF
10709 043072          L420:;;;;;;
      043072          MOV #2,NOPAR          ;PARITY ACTION
10710 043072 012737 000002 002076  MOV #2,PCBUMP          ;TRAPS ADD 2 TO PC
10711 043100 012737 000002 002326  MOV #2,PATTERN,RO
10712 043106 013700 002112          ASL RO
10713 043112 006300          CALL @FSPAT(RO)
10714 043114 004770 043126          CLR NOPAR
10715 043120 005037 002076          BR CMD5B          ;LOOP TILL KEYBOARD INTERRUPT
10716 043124 000712
10717

```

```

10718 043126 017510          FSPAT: MT0000 ;<1 SEC          DATA PATTERN TEST
10719 043130 017544          MT0001 ;<1 SEC          ADDRESS TEST
10720 043132 017640          MT0002 ;<1 SEC          COMPLEMENT ADDRESS TEST
10721 043134 017754          MT0003 ; 1 SEC          3 XOR 9 WORST CASE NOISE TEST
10722 043136 020070          MT0004 ; 1 SEC          ROTATING ZEROS TEST
10723 043140 020140          MT0005 ; 1 SEC          ROTATING ONES TEST
10724 043142 020222          MT0006 ;<1 SEC          INITIAL DATA TEST
10725 043144 020256          MT0007 ;<1 SEC          ADDRESS BIT TEST
10726 043146 020320          MT0010 ;<1 SEC          BYTE ADDRESSING TEST
10727 043150 024220          MT0999 ;<1 SEC          NULL TEST TO KEEP TABLE ORDER
10728 043152 024220          MT0999 ;<1 SEC          NULL TEST TO KEEP TABLE ORDER
10729 043154 024220          MT0999 ;<1 SEC          NULL TEST TO KEEP TABLE ORDER
10730 043156 020354          MT0014 ; 1 SEC          BASIC DOUBLE BIT ERROR TEST
10731 043160 024220          MT0999 ;<1 SEC          NULL TEST TO KEEP TABLE ORDER
10732 043162 024220          MT0999 ;<1 SEC          NULL TEST TO KEEP TABLE ORDER
10733 043164 020434          MT0017 ;<1 SEC          HOLDING 1'S & 0'S TEST
10734 043166 020456          MT0020 ; 1 SEC          SYNDROMES TO CSR ON SBE TEST
10735 043170 020536          MT0021 ; 1 SEC          MARCHING 0'S & 1'S TEST
10736 043172 021026          MT0022 ;10 SEC          REFRESH & SHIFTING DIAGONAL TEST
10737 043174 021060          MT0023 ;10 SEC          SHIFTING DIAGONAL TEST
10738 043176 021124          MT0024 ;20 SEC          FAST GALLOPING PATTERN TEST
10739 043200 024220          MT0999 ;<1 SEC          NULL TEST TO KEEP TABLE ORDER
10740 043202 021332          MT0026 ;<1 SEC          RANDOM DATA TEST
10741 043204 021610          MT0027 ; 1 SEC          UNIQUE BANK TEST
10742 043206 022200          MT0030 ; 1 SEC          FLUSH OUT DBE'S TEST
10743 043210 022454          MT0031 ; 3 SEC          SOB-A-LONG TEST
10744 043212 022644          MT0032 ;<1 SEC          WRITE RECOVERY TEST
10745 043214 023160          MT0033 ;35 SEC          BRANCH GOBBLE TEST
10746 043216 023346          MT0034 ; 1 SEC          SOFT ERROR TEST
10747 043220 023446          MT0035 ;<1 SEC          WORST CASE NOISE PARITY TEST

```


10748	043222	023560			MT0036	: 1 SEC		CORRECTION CODE TEST
10749	043224	023622			MT0037	: <1 SEC		ECC DISABLE TEST
10750	043226	024220			MT0999	: <1 SEC		NULL TEST TO KEEP TABLE ORDER
10751	043230	023660			MT0041	: 1 SEC		ADDRESS TO CSR ON DOUBLE BIT ERROR TEST
10752	043232	023740			MT0042	: <1 SEC		EXTENDED Q-BUS ADDRESS TEST
10753	043234	023774			MT0043	: 1 SEC		WRITE BYTE CLEARS SBE TEST
10754	043236	024024			MT0044	: 5 SEC		SHIFTING 1/0'S THROUGH THE CHECK BITS
10755	043240	024104			MT0045	: 1 SEC		SYNDROMES TO CSR ON DBE TEST
10756	043242	024134			MT0046	: 1 SEC		CHECK SINGLE BIT ERROR WITH ECC DISABLED TEST
10757	043244	024164			MT0047	: <1 SEC		NO CSR UPDATE WITH SBE ON DBE TEST
10758								
10759	043246	013706	002306		CMD5C: MOV	FSSTACK,SP		;RECOVER OLD STACK POINTER
10760	043252	042777	000100	137362	BIC	#BIT6,@TKS		
10761	043260				POP	TKVEC+2,TKVEC		
	043260	012637	000062					MOV (SP)+,TKVEC+2
	043264	012637	000060					MOV (SP)+,TKVEC
10762	043270	117700	137350		MOVB	@TKB,R0		;GET CHARACTER TO GET RID OF FLAG
10763	043274				POP	PCBUMP,TESTADD		
	043274	012637	002326					MOV (SP)+,PCBUMP
	043300	012637	002412					MOV (SP)+,TESTADD
10764	043304				POP	PATTERN,BANK		
	043304	012637	002112					MOV (SP)+,PATTERN
	043310	012637	002102					MOV (SP)+,BANK
10765	043314				MAP	BANK		;REMAP OLD BANK
	043314	010346						MOV R3,-(SP)
	043316	013703	002102		MOV	BANK,R3		
	043322	004737	035716		CALL	MAPPER		
					.DSABL	CRF		
	043326	012603						MOV (SP)+,R3
10766	043330	004737	040052		CALL	EXBANK		
10767	043334	013737	002154	002152	MOV	SAVCSR,CSRNO		;RESTORE CSRNO.
10768	043342	000207			RETURN			

10770 043344

```

FSCMD6: SUBTST <<FS COMMAND 6 TYPE CONFIGURATION MAP>>
;*****
;*SUBTEST FS COMMAND 6 TYPE CONFIGURATION MAP
;*****
CALL PCONFIG
RETURN

```

10771 043344 004737 032722
10772 043350 000207
10773

```

10776 043352          FSCMD7: SUBTST <<FS  COMMAND 7  SOB-A-LONG TEST>>
;*****
;*SUBTEST  FS  COMMAND 7  SOB-A-LONG TEST
;*****
10777 043352          PUSH  BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
043352 013746 002102          MOV  BANK,-(SP)
043356 013746 002112          MOV  PATTERN,-(SP)
043362 013746 000060          MOV  TKVEC,-(SP)
043366 013746 000062          MOV  TKVEC+2,-(SP)
043372 013746 002076          MOV  NOPAR,-(SP)
10778 043376 010637 002306          MOV  SP,FSSTACK          ;SAVE LAST GOOD STACK POINTER
10779 043402          TYPE  MSG055          ;SOB-A-LONG TEST
043402 104401 067671          TYPEIT ,MSG055
.DSABL CRF

10780
10781 043406          IF #SWO SET.IN @SWR
043406 032777 000001 137222          BIT  #SWO,@SWR
043414 001402          BEQ  L421
10782 043416 104470          ECCDIS          ;DISABLE ERROR CORRECTION
10783 043420          ELSE
043420 000401          BR  L422
043422          L421:;;;;;
10784 043422 104502          CLRCSR          ;CLEAR CSRS
10785 043424          END ;OF IF
043424          L422:;;;;;
10786 043424          TYPE  MSG056          ;BELL = EACH PASS COMPLETE
043424 104401 067712          TYPEIT ,MSG056
.DSABL CRF

10787
10788 043430          TYPE  MSG046          ;TO ESCAPE TYPE ANY KEY!
043430 104401 067505          TYPEIT ,MSG046
.DSABL CRF

10789 043434 012737 043560 000060          MOV  #CMD7C,TKVEC
10790 043442 012737 000340 000062          MOV  #340,TKVEC+2
10791 043450 017700 137170          MOV  @TKB,RO          ;KILL ANY OLD INTERRUPT
10792 043454 042737 000200 177776          BIC  #BIT7,PSW          ;LOWER CPU PRIORITY TO 140
10793 043462 052777 000100 137152          BIS  #BIT6,@TKS          ;ENABLE KEYBOARD INTERRUPTS
10794
10795
10796 043470          SET  HEADER,MUT
043470 012737 177777 002612          MOV  #-1,HEADER
043476 012737 177777 002110          MOV  #-1,MUT

10797
10798 043504          CMD7B: FOR BANK := #0 TO LASTBANK
043504 005037 002102          CLR  BANK
043510          B71:;;;;;
10799 043510 004737 040052          CALL EXBANK
10800 043514          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
043514 005737 002116          TST  ACFLAG
043520 001406          BEQ  L423
043522 005737 002124          TST  RRFLAG
043526 001003          BNE  L423
10801 043530 104511          INVALIDATE
10802 043532 004737 022454          CALL MTO031
10803 043536          END ;OF IF ACFLAG
043536          L423:;;;;;
10804 043536          END ;OF FOR BANK

```

```

043536 005237 002102
043542 023737 002102 002556
043550 003757
043552
10805 043552 TYPE $BELL ;RING BELL E71:.....
043552 104401 002653 TYPEIT , $BELL
.DSABL CRF
GOTO CMD7B
10806 043556 BR CMD7B
043556 000752
10807
10808 043560 013706 002306 CMD7C: MOV FSSTACK, SP ;RECOVER OLD STACK POINTER
10809 043564 042777 000100 137050 BIC #BIT6, @TKS
10810 043572 117700 137046 MOVB @TKB, R0 ;READ CHAR TO KILL FLAG
10811 043576 POP NOPAR, TKVEC+2, TKVEC, PATTERN, BANK
043576 012637 002076 MOV (SP)+, NOPAR
043602 012637 000062 MOV (SP)+, TKVEC+2
043606 012637 000060 MOV (SP)+, TKVEC
043612 012637 002112 MOV (SP)+, PATTERN
043616 012637 002102 MOV (SP)+, BANK
10812 043622 MAP BANK ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
043622 010346 MOV R3, -(SP)
043624 013703 002102 MOV BANK, R3
043630 004737 035716 CALL MAPPER
.DSABL CRF
043634 012603 MOV (SP)+, R3
10813 043636 004737 040052 CALL EXBANK
10814 043642 000207 RETURN

```

10817 043644

FSCMD8: SUBTST <<FS COMMAND 8 ERROR SUMMARY>>

;*SUBTEST FS COMMAND 8 ERROR SUMMARY

10818 043644

043644 010046
043646 010246
043650 010346

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

10819 043656 013737 056654 002102 002434

10820 043664 005337 002434
10821 043670 013746 002434
043674 104405

MOV \$PASS,TEMP
DEC TEMP
TYPDEC TEMP
MOV TEMP,-(SP) ;;SAVE TEMP FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
.DSABL CRF

10822 043676 013737 172350 002270
10823 043704 012737 001000 172350

10824 043712 104401 071242
043712 104401

MOV KIPAR4,SAV4 ;;I.L.C.;REV B
MOV #1000,KIPAR4 ;;I.L.C.;REV B
TYPE MSG125 ;PASSES COMPLETED
TYPEIT ,MSG125
.DSABL CRF

10825 043716 013746 002630
043722 104405

TYPDEC \$ERTTL
MOV \$ERTTL,-(SP) ;;SAVE \$ERTTL FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
.DSABL CRF

10826 043724 104401 070265
043724 104401

TYPE MSG079 ;ERROR(S) DETECTED
TYPEIT ,MSG079
.DSABL CRF
IF \$ERTTL NE #0

10827 043730 005737 002630
043734 001445

TST \$ERTTL
BEQ L424

10828 043736 005037 002334
10829 043742 005037 002102

CLR SUCCESS
FOR BANK := #0 TO LASTBANK

CLR BANK
B72:;;;;;

10830 043746 013703 002102
10831 043752 070327 000004

MOV BANK,R3
MUL #4,R3
IFB CONFIG*2(R3) NE #0

TSTB CONFIG*2(R3)
BEQ L425

10832 043756 105763 002666
043762 001424

IF SUCCESS IS FALSE

TST SUCCESS
BNE L426

10833 043764 005737 002334
043770 001005

10834 043772 104401 070230
043772 104401

TYPE MSG076 ;BANK ERRORS
TYPEIT ,MSG076
.DSABL CRF
SET SUCCESS

MOV #-1,SUCCESS

10835 043776 012737 177777 002334

END ;OF IF SUCCESS

L426:;;;;;

10836 044004 044004

10837 044004 013746 002102
044004 104403

TYPDCS BANK,3
MOV BANK,-(SP) ;;SAVE BANK FOR TYPEOUT
;;3
TYPDS ;;GO TYPE--OCTAL ASCII
.BYTE 6 ;;TYPE 6 DIGITS
.BYTE 0 ;;SUPPRESS LEADING ZEROS
.DSABL CRF

044012 006
044013 000

10838 044014 116300 002666

MOVB CONFIG*2(R3),R0

10839	044020	042700	177400					
10840	044024							
	044024	010046						
	044026	104405						
10841	044030							
	044030	104401	002660					
10842	044034							
	044034							
10843	044034							
	044034	005237	002102					
	044040	023737	002102	002556				
	044046	003737						
	044050							
10844	044050							
	044050							
10845	044050	013737	002270	172350				
10846	044056							
	044056	012637	002102					
	044062	012603						
	044064	012602						
	044066	012600						
10847	044070	000207						


```

BIC #C377,R0
TYPDEC R0
MOV RO,-(SP) ;;SAVE RO FOR TYPEOUT
TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
.DSABL CRF
TYPE $CRLF
TYPEIT $CRLF
.DSABL CRF
END ;OF IFB CONFIG(R3)
END ;OF FOR BANK
L425:;;;;;
INC BANK
CMP BANK, LASTBANK
BLE B72
E72:;;;;;
L424:;;;;;
MOV SAV4, KIPAR4 ;;I.L.C.;;;REV B
POP BANK, R3, R2, R0
MOV (SP)+, BANK
MOV (SP)+, R3
MOV (SP)+, R2
MOV (SP)+, R0
RETURN
    
```

```

10850 044072          FSCMD9: SUBTST <<FS  COMMAND 9  REFRESH TEST>>
;*****
;*SUBTEST          FS  COMMAND 9  REFRESH TEST
;*****
10851 044072          PUSH  BANK,PATTERN,TKVEC,TKVEC+2,NOPAR
044072 013746 002102          MOV  BANK,-(SP)
044076 013746 002112          MOV  PATTERN,-(SP)
044102 013746 000060          MOV  TKVEC,-(SP)
044106 013746 000062          MOV  TKVEC+2,-(SP)
044112 013746 002076          MOV  NOPAR,-(SP)
10852 044116 010637 002306      MOV  SP,FSSTACK          ;SAVE LAST GOOD STACK POINTER
10853 044122          TYPE  MSG073          ;REFRESH TEST
044122 104401 070160      TYPEIT ,MSG073
                                .DSABL CRF
10854
10855 044126          IF #SW0 SET.IN @SWR
044126 032777 000001 136502          BIT  #SW0,@SWR
044134 001402          BEQ  L427
10856 044136 104470          ECCDIS          ;DISABLE ERROR CORRECTION
10857 044140          ELSE
044140 000401          BR  L430
044142          L427:;;;;;
10858 044142 104502          CLRCSR          ;CLEAR CSRS
10859 044144          END ;OF IF
044144          L430:;;;;;
10860 044144          TYPE  MSG056          ;BELL = EACH PASS COMPLETE
044144 104401 067712      TYPEIT ,MSG056
                                .DSABL CRF
10861
10862 044150          TYPE  MSG046          ;TO ESCAPE TYPE ANY KEY!
044150 104401 067505      TYPEIT ,MSG046
                                .DSABL CRF
10863 044154 012737 044300 000060      MOV  #CMD9C,TKVEC
10864 044162 012737 000340 000062      MOV  #340,TKVEC+2
10865 044170 017700 136450          MOV  #TKB,RO
10866 044174 042737 000200 177776      BIC  #JIT7,PSW          ;KILL ANY OLD INTERRUPT
10867 044202 052777 000100 136432      BIS  #BIT6,@TKS          ;LOWER CPU PRIORITY TO 140
10868
10869 044210          SET  HEADER,MUT          ;ENABLE KEYBOARD INTERRUPTS
044210 012737 177777 002612          MOV  #-1,HEADER
044216 012737 177777 002110          MOV  #-1,MUT
10870
10871 044224          CMD9B: FOR BANK := #0 TO LASTBANK
044224 005037 002102          CLR  BANK
044230          B73:;;;;;
10872 044230 004737 040052          CALL EXBANK
10873 044234          IF ACFLAG IS TRUE AND RRFLAG IS FALSE
044234 005737 002116          TST  ACFLAG
044240 001406          BEQ  L431
044242 005737 002124          TST  RRFLAG
044246 001003          BNE  L431
10874 044250 104511          INVALIDATE
10875 044252 004737 021026          CALL MTO022
10876 044256          END ;OF IF ACFLAG
044256          L431:;;;;;
10877 044256          END ;OF FOR BANK
044256 005237 002102          INC  BANK
    
```

```
044262 023737 002102 002556          CMP BANK, LASTBANK
044270 003757          BLE B73
044272          E73: :::::
10878 044272          TYPE      $BELL          ;RING BELL
044272 104401 002653      TYPEIT    , $BELL
          .DSABL  CRF
10879 044276          GOTO     CMD9B
044276 000752          BR      CMD9B
10880          CMD9C: MOV     FSSTACK, SP          ;RECOVER OLD STACK POINTER
10881 044300 013706 002306          BIC     #BIT6, @ $TKS
10882 044304 042777 000100 136330          MOV    @ $TKB, R0          ;READ CHAR TO KILL FLAG
10883 044312 117700 136326          POP
10884 044316          NOPAR, TKVEC+2, TKVEC, PATTERN, BANK
044316 012637 002076          MOV (SP)+, NOPAR
044322 012637 000062          MOV (SP)+, TKVEC+2
044326 012637 000060          MOV (SP)+, TKVEC
044332 012637 002112          MOV (SP)+, PATTERN
044336 012637 002102          MOV (SP)+, BANK
10885 044342          MAP     BANK          ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
044342 010346          MOV    BANK, R3
044344 013703 002102          CALL  MAPPER
044350 004737 035716          .DSABL CRF
          MOV (SP)+, R3
044354 012603
10886 044356 004737 040052          CALL  EXBANK
10887 044362 000207          RETURN
10888
```



```

10891 044364 FCMD10: SUBTST <<FS COMMAND 10 SET FILL COUNT>>
;*****
;*SUBTEST FS COMMAND 10 SET FILL COUNT
;*****
10892 044364 010046 PUSH RO
10893 044366 104401 070311 TYPE MSG085 ;FILL COUNT(OCTAL)? MOV RO,-(SP)
044366 TYPEIT ,MSG085
.DSABL CRF
RDOCT
POP RO
10894 044372 104413
10895 044374 012600 MOV (SP)+,RO
044374 042700 177760 BIC #+C17,RO
10896 044376 110037 002357 MOVB RO,$FILLS
10897 044402 POP RO
10898 044406 012600 MOV (SP)+,RO
044406 000207 RETURN
10899 044410
10900
10901 044412 FCMD11: SUBTST <<FS COMMAND 11 ENTER KAMIKAZE MODE>>
;*****
;*SUBTEST FS COMMAND 11 ENTER KAMIKAZE MODE
;*****
10902 044412 104401 070456 TYPE MSG101 ;ENTERING KAMIKAZE MODE
044412 TYPEIT ,MSG101
.DSABL CRF
SET KAMIKAZE,SKIPKAMI
10903 044416 012737 177777 002006 MOV #-1,KAMIKAZE
044416 012737 177777 002010 MOV #-1,SKIPKAMI
044424 000207 RETURN
10904 044432
10905
10906 044434 FCMD12: SUBTST <<FS COMMAND 12 EXIT KAMIKAZE MODE>>
;*****
;*SUBTEST FS COMMAND 12 EXIT KAMIKAZE MODE
;*****
10907 044434 104401 070506 TYPE MSG102 ;LEAVING KAMIKAZE MODE
044434 TYPEIT ,MSG102
.DSABL CRF
CLR KAMIKAZE
SET SKIPKAMI
10908 044440 005037 002006
10909 044444 012737 177777 002010 MOV #-1,SKIPKAMI
044444 000207 RETURN
10910 044452
10911
10912 044454 FCMD13: SUBTST <<FS COMMAND 13 TURN CACHE OFF>>
;*****
;*SUBTEST FS COMMAND 13 TURN CACHE OFF
;*****
10913 044454 104401 070634 TYPE MSG106 ;CACHE IS OFF
044454 TYPEIT ,MSG106
.DSABL CRF
CACHOFF ;TURN CACHE OFF
MOV CACHKN,CACHKN+2 ;SAVE OLD CACHE ON STATE
CLR CACHKN ;KEEP CACHE OFF
RETURN
10914 044460 104424
10915 044462 013737 002544 002546
10916 044470 005037 002544
10917 044474 000207
10918
10919 044476 FCMD14: SUBTST <<FS COMMAND 14 TURN CACHE ON>>
;*****
;*SUBTEST FS COMMAND 14 TURN CACHE ON

```



```

;*****
10920 044476          TYPE      MSG107          ;CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)
      044476 104401 070652      TYPEIT     ,MSG107
10921 044502 013737 002546 002544 .DSABL    CRF
10922 044510 104423          MOV       CACHKN+2,CACHKN          ;RESTORE OLD CACHE ON STATE
10923 044512 000207          CACHON          ;TURN CACHE ON
10924          RETURN

```

```

10937
10938 044514 FCMD15: SUBTST <<FS COMMAND 15 TEST ONLY SELECTED BANKS>>
;*****
;*SUBTEST FS COMMAND 15 TEST ONLY SELECTED BANKS
;*****
10939 044514 104401 070561 TYPE MSG105 ;ENTER BANKS IN OCTAL - USE NUMBER OUTSIDE RANGE TO TERMINAT
044514 TYPEIT ,MSG105
.DSABL CRF ;ERASE OLD SELECTIONS
10940 044520 004737 044610 CALL CMD16A
10941 044524 BEGIN CMD16LOOP
044524 REPEAT B74:;;;;;
10942 044524 REPEAT B75:;;;;;
044524 TYPE MSG030 ;BANK(0-177)?
10943 044524 104401 067072 TYPEIT ,MSG030
044524 .DSABL CRF ;READ AN OCTAL NUMBER ONTO THE STACK
10944 044530 104413 RDOCT ;PUT IT IN R1
10945 044532 012601 POP R1 MOV (SP)+,R1
10946 044534 IF R1 GT #177 OR R1 LT #0
044534 020127 000177 CMP R1,#177
044540 003002 BGT L432
044542 005701 TST R1
044544 002001 BGE L433
044546 L432:;;;;;
10947 044546 LEAVE CMD16LOOP BR E74
044546 000406
10948 044550 END ;OF IF R1 L433:;;;;;
044550 ASL R1
10949 044550 006301 ASL R1 ;R1 <- R1 * 4
10950 044552 006301 BIS #BIT14,CONFIG+2(R1)
10951 044554 052761 040000 002666 END ;OF REPEAT BR B75
10952 044562 000760 E75:;;;;;
044564 END CMD16LOOP E74:;;;;;
10953 044564 TYPE MSG110 ;ONLY SELECTED BANKS WILL BE TESTED
044564 TYPEIT ,MSG110
.DSABL CRF
10954 044564 104401 070727 SET SELONLY
10955 044570 012737 177777 002002 RETURN MOV #-1,SELONLY
10956 044576 000207
10957
10958 044600 FCMD16: SUBTST <<FS COMMAND 16 RESUME TESTING ALL BANKS>>
;*****
;*SUBTEST FS COMMAND 16 RESUME TESTING ALL BANKS
;*****
10959 044600 TYPE MSG111 ;ALL BANKS WILL BE TESTED
044600 104401 070773 TYPEIT ,MSG111
.DSABL CRF
10960 044604 005037 002002 CLR SELONLY
10961
10962 ;ENTRY POINT FROM CMD15
10963 044610 013702 002556 CMD16A: MOV LASTBANK,R2
10964 044614 006302 ASL R2
10965 044616 006302 ASL R2

```

10966 044620
 044620 005001
 044622
 10967 044622 042761 040000 002666
 10968 044630
 044630 062701 000004
 044634 020102
 044636 003771
 044640
 10969 044640 000207

FOR R1 := #0 TO R2 BY #4

 BIC #BIT14,CONFIG+2(R1)
 END ;OF FOR R1

 RETURN

CLR R1
 B76:::~::~

 ADD #4,R1
 CMP R1,R2
 BLE B76
 E76:::~::~

10972 044642

FCMD17: SUBTST <<FS COMMAND 17 ENABLE TRACE>>
;*****
;*SUBTEST FS COMMAND 17 ENABLE TRACE
;*****

10973 044642
044642 104401 071331

TYPE MSG127
TYPEIT ,MSG127
.DSABL CRF
MOV #-1,TRACE
RETURN

10974 044646 012737 177777 006216
10975 044654 000207

10978 044656

FCMD18: SUBTST <<FS COMMAND 18 DISABLE TRACE>>
;*****
;*SUBTEST FS COMMAND 18 DISABLE TRACE
;*****

10979 044656 104401 071350
044656

TYPE MSG128
TYPEIT ,MSG128
.DSABL CRF
CLR TRACE
RETURN

10980 044662 005037 006216
10981 044666 000207

10984 044670

WHICHCSR:SUBTST <<SUBR DETERMINE CORRECT CSR>>

;*SUBTEST SUBR DETERMINE CORRECT CSR

10985 044670 013700 002224
10986 044674 022700 100000
10987 044700 001003
10988 044702 005037 002152
10989 044706 000207

MOV TOTCSRS,R0 ;GET CSR'S FLAG
CMP #BIT15,R0 ;CSR 0?
BNE 1\$;NO - SKIP
CLR CSRNO ;YES - SET IT UP
RETURN

10990
10991 044710
044710 104401 066725

1\$: TYPE MSG022 ;WHICH CSR(0-F)
TYPEIT ,MSG022
.DSABL CRF

10992 044714 104412
10993 044716
044716 012600

RDLIN ;GET CHARACTER
POP R0 ;PUT IN R0

10994 044720 011000
10995 044722 020027 000106
10996 044726 101370

MOV (R0),R0 ;PUT CHAR IN R0
CMP R0,#106 ;CHECK LIMIT
BHI 1\$;IF BAD LOOP TILL HE TYPES IT RIGHT

10997 044730 022700 000101
10998 044734 103002
10999 044736 162700 000007

CMP #'A,R0
BHIS 2\$

11000 044742 162700 000060
11001 044746 006300
11002 044750 010037 002152
11003 044754 000207

2\$: SUB #7,R0
SUB #60,R0
ASL R0
MOV R0,CSRNO
RETURN

```

11469          .SBTTL  ERROR DATA (SUPERVISOR) SETUP STUFF
11470 044756          $PER25: LET ADDRESS := R1 - #2
          044756 010137 002034          MOV R1,ADDRESS
          044762 162737 000002 002034          SUB #2,ADDRESS
11471 044770          IF ABORTFLAG IS FALSE
          044770 005737 002144          TST ABORTFLAG
          044774 001007          BNE L434
11472 044776          TESTAREA          ;ENTER TEST MODE
          044776 053737 002552 177776          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
          .DSABL CRF
          LET BAD := -2(R1)
11473 045004          KERNEL          ;ENTER KERNEL MODE
          045004 016137 177776 002052          MOV -2(R1),BAD
11474 045012 104417
11475 045014          END ;OF IF ABORTFLAG
          045014          L434:;;;;;
11476 045014          IF 177654 EQ #0
          045014 005737 177654          TST 177654
          045020 001003          BNE L435
11477 045022          LET GOOD := R2
          045022 010237 002044          MOV R2,GOOD
11478 045026          ELSE
          045026 000402          BR L436
          045030          L435:;;;;;
11479 045030          LET GOOD := R3
          045030 010337 002044          MOV R3,GOOD
11480 045034          END ;OF IF
          045034          L436:;;;;;
11481 045034 000137 050212          JMP PERRAW
11482
11483 045040          PERRA3: SUBTST <<DATA WAS 3 WORDS>>
          ;*****
          ;*SUBTEST DATA WAS 3 WORDS
          ;*****
          IF BADPC EQ #0 THEN $CALL BADSTACK
11484 045040          TST BADPC
          045040 005737 002022          BNE L437
          045044 001002          JSR PC,BADSTACK
          045046 004737 034152          L437:;;;;;
          045052
11485 045052          PUSH RO
          045052 010046          MOV RO,-(SP)
11486 045054          CLR CSR          ;MAKE SURE CSR BIT HOLDER IS CLEAR
          045054 005037 002150          CHK1DIS          ;DISABLE ECC & WRITE CHECKBITS FROM 1 SELECTED CSR
11487 045060 104505          TESTAREA
11488 045062          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
          045062 053737 002552 177776          .DSABL CRF
          TST (R1)          ;READ LOCATION TO READ CHECKBITS INTO CSR
11489 045070          KERNEL
          045070 005711          READCSR          ;GET CSR CONTENTS
11490 045072 104417          MOV CSR,RO          ;SAVE CSR CONTENTS IN RO
11491 045074 104426          CLR1CSR          ;RETURN CSR TO NORMAL MODE
11492 045076 013700 002150          ASH #-5,RO          ;MOVE CHECK BITS TO BOTTOM OF WORD
11493 045102 104503          BIC #+C177,RO          ;CLEAR OFF EXTRANEIOUS GARBAGE
11494 045104 072027 177773          LET ADDRESS := R1          ;SAVE VIRTUAL ADDRESS FOR PRINTOUT
11495 045110 042700 177600          MOV R1,ADDRESS
11496 045114          CLR GOOD          ;FIRST TEST WORD WRITTEN SHOULD ALWAYS BE ZERO
          045114 010137 002034          TESTAREA          ;ENTER TEST MODE
11497 045120 005037 002044          BIS TESTMODE,PSW          ;GO TO SYSTEM TEST MODE
11498 045124 053737 002552 177776
    
```

```

11499 045132 011137 002052      .DSABL CRF
11500 045136 011437 002054      MOV (R1),BAD ;GET BAD DATA FROM MUT - FIRST WORD
11501 045142 104417              MOV (R4),BAD2 ;AND SECOND WORD
11502 045144 110037 002056      KERNEL ;ENTER KERNEL MODE
11503 045150 105037 002057      MOVB RO,BAD3 ;MOVE BAD CHECKBITS FOR PRINTOUT
11504 045154 004737 050446      CLRB BAD3+1 ;CLEAR OFF THE OTHER UNUSED BITS
11505 045160 104033              CALL PERBNK ;MARK BANK AS BAD IN CONFIG TABLE
11506 045162 012600              ERROR +33
11507 045164 032777 000001 135444      POP RO ;RESTORE RO
                                MOV (SP)+,RO
                                BIT #SW0,@SWR
                                BEQ L440
11508 045174 104506              ENASBE ;TRAP ON SINGLE BIT ERRORS
11509 045176 000401              ELSE
                                BR L441
11510 045200 104472              ECCINIT ;TRAP ON UNCORRECTABLE ERRORS
11511 045202 000002              END; OF IF #SW0
                                L440:::
11512 045202 000002              RTI ;
                                L441:::

```



```

11515 045204          $PER30: LET GOOD := R1
      045204 010137 002044
11516 045210          LET ADDRESS := (SP) - 16
      045210 011637 002034
      045214 163737 000016 002034
11517 045222          IF ABORTFLAG IS FALSE
      045222 005737 002144
      045226 001007
11518 045230          TESTAREA
      045230 053737 002552 177776
      BIS TESTMODE,PSW ;ENTER TEST MODE
      .DSABL CRF ;GO TO SYSTEM TEST MODE
      LET BAD := @ADDRESS
11519 045236          MOV @ADDRESS,BAD
      045236 017737 134572 002052
11520 045244          KERNEL
      104417
11521 045246          END ;OF IF ABORTFLAG
      045246
11522 045246          JMP PERRAW
      000137 050212
11523
11524 045252          GETDATA:SUBTST <<GET DATA FROM ABORTED AREA IF POSSIBLE>>
      ;*****
      ;*SUBTEST GET DATA FROM ABORTED AREA IF POSSIBLE
      ;*****
11525 045252          PUSH RO,4,114
      045252 010046
      045254 013746 000004
      045260 013746 000114
11526 045264          MOV SP,GETDA1
      010637 045350
11527 045270          MOV #1$,4
      012737 045330 000004
11528 045276          MOV #1$,114
      012737 045330 000114
11529 045304          MOV ADDRESS,RO
      013700 002034
11530 045310          TESTAREA
      045310 053737 002552 177776
      BIS TESTMODE,PSW ;GO TO SYSTEM TEST MODE
      .DSABL CRF
11531 045316          MOV (RO),BAD
      011037 002052
11532 045322          KERNEL
      104417
11533 045324          CLR ABORTFLAG
      005037 002144
11534 045330          1$: MOV GETDA1,SP ;RESTORE KNOWN GOOD STACK POINTER
      013706 045350
11535 045334          POP 114,4,RO
      045334 012637 000114
      045340 012637 000004
      045344 012600
11536 045346          RETURN
      000207
11537 045350          GETDA1: 0
      000000

```

```

11540          .SBTTL POWER FAIL AUTO RESTART
11541          .SBTTL ROUTINE POWER DOWN AND UP
11542          ;*****
11543          ;POWER DOWN ROUTINE
11544 045352    $PWRDN:
11552          ;SAVE CACHE STATUS
11553 045352 005737 002544    TST    CACHKN
11554 045356 001403          BEQ    5$
11555 045360          PUSH   CONTRL
11556 045360 013746 177746          MOV   CONTRL,-(SP)
11557 045364 104423          CACHON
11558 045366 012737 046300 000024 5$:  MOV   #$ILLUP,PWRVEC ;:TURN CACHE ON
11559 045374 012737 000340 000026    MOV   #340,PWRVEC+2 ;:SET FOR FAST UP
11559 045402          PUSH   R0,R1,R2,R3,R4,R5,CSRNO ;:PRIO:7
11559 045402 010046          MOV   R0,-(SP)
11559 045404 010146          MOV   R1,-(SP)
11559 045406 010246          MOV   R2,-(SP)
11559 045410 010346          MOV   R3,-(SP)
11559 045412 010446          MOV   R4,-(SP)
11559 045414 010546          MOV   R5,-(SP)
11559 045416 013746 002152          MOV   CSRNO,-(SP)
11560          ;SAVE USER PAR'S & PDR7
11561 045422 012700 177700    MOV   #177700,R0
11562 045426 012701 000021    MOV   #17.,R1
11563 045432          PUSH   -(R0)
11564 045432 014046          MOV   -(R0),-(SP)
11564 045434 077102          SOB   R1,1$
11565          ;SAVE SUPERVISOR PAR'S
11566 045436 005737 002456    TST   NOSUPER
11567 045442 001013          BNE   PD1
11568 045444 012700 172300    MOV   #172300,R0
11569 045450 012701 000020    MOV   #16.,R1
11570 045454          PUSH   -(R0)
11571 045454 014046          MOV   -(R0),-(SP)
11571 045456 077102          SOB   R1,2$
11572 045460          IF RLFLAG IS TRUE THEN $CALL WOOPS
11572 045460 005737 002126          TST   RLFLAG
11572 045464 001402          BEQ   L443
11572 045466 004737 046306          JSR   PC,WOOPS
11573          L443:::
11573          ;COPY KERNEL MAP TO USER & SUPERVISOR
11574 045472 012700 172300    PD1:  MOV   #KIPDR0,R0
11575 045476 012701 177600    MOV   #UIPDR0,R1
11576 045502 012702 172200    MOV   #SIPDR0,R2
11577 045506 012703 000040    MOV   #32.,R3
11578 045512 011021          3$:  MOV   (R0),(R1)+
11579 045514 012022          MOV   (R0)+,(R2)+
11580 045516 077303          SOB   R3,3$

```

```

11582                                     ;SAVE USER & SUPERVISOR STACK POINTERS
11583 045520                               USER
      045520 052737 140000 177776        BIS #BIT15:BIT14,PSW ;GO TO USER MODE
                                         .DSABL CRF
11584 045526 010600                       MOV USP,R0
11585 045530 104417                       KERNEL ;ENTER KERNEL MODE
11586 045532                               PUSH R0
                                         MOV R0,-(SP)
11587 045534 010046                       TST NOSUPER
11588 045540 005737 002456                BNE 7$
11589 045542 001006                       SUPERVISOR ;ENTER SUPERVISOR MODE
      045542 052737 040000 177776        BIS #BIT14,PSW ;GO TO SUPERVISOR MODE
                                         .DSABL CRF
11590 045550 010600                       MOV SSP,R0
11591 045552 104417                       KERNEL ;ENTER KERNEL MODE
11592 045554                               PUSH R0
                                         MOV R0,-(SP)
11593                                     ;SAVE ECC REGISTERS
11594 045556 013701 002224                7$: MOV TOTCSRS,R1 ;GET CSR'S
11595 045562                               BEGIN LCSRSAVE
11596 045562                               FOR CSRNO := #0 TO #36 BY #2
      045562 005037 002152
11597 045566 006301                       ASL R1
11598 045570                               ON.ERROR
11599 045572 103003                       READCSR
11600 045574 104426                       PUSH CSR
11601 045574 013746 002150                END ;OF ON.ERROR
11602 045600                               IF R1 EQ #0 THEN LEAVE LCSRSAVE
      045600 005701
      045602 001407
11603 045604                               END ;OF FOR CSRNO
      045604 062737 000002 002152
      045612 023727 002152 000036
      045620 003762
11604 045622                               END LCSRSAVE
      045622
11605                                     ;SAVE MMRO,1,2,3
11606 045622                               PUSH MMRO,MMR1,MMR2
      045622 013746 177572
      045626 013746 177574
      045632 013746 177576
11607 045636 005737 002456                TST NOSUPER
11608 045642 001002                       BNE 8$
11609 045644                               PUSH MMR3
      045644 013746 172516
11610                                     ;SAVE KERNEL PAR'S
11611 045650 012700 172400                8$: MOV #172400,R0
11612 045654 012701 000020                MOV #16.,R1
11613 045660                               4$: PUSH -(R0)
      045660 014046
11614 045662 077102                       SOB R1,4$
                                         MOV -(R0),-(SP)

```

```

B77:~~~~~
      CLR CSRNO
B100:~~~~~
      BCC L444
      MOV CSR,-(SP)
L444:~~~~~
      TST R1
      BEQ E77
      ADD #2,CSRNO
      CMP CSRNO,#36
      BLE B100
E100:~~~~~
E77:~~~~~

```

```

11615                                     ;SAVE Q-BUS MAP REGISTERS
11616 045664 022737 000005 004010      CMP      #5,PROTYP      ;IS THIS AN 11/83 ?
11617 045672 001004                                     BNE      9$            ;BRANCH IF NOT
11618 045674                                     PUSH     MAPHO,MAPLO
      045674 013746 170202                                     MOV MAPHO,-(SP)
      045700 013746 170200                                     MOV MAPLO,-(SP)
11619                                     ;SAVE POSSIBLE SOFTWARE SWITCH REGISTER
11620 045704                                     9$:  PUSH     @SWR
      045704 017746 134726                                     MOV @SWR,-(SP)
11621                                     ;SAVE STACK POINTER
11622 045710 010637 046304      MOV      SP,$SAVR6      ;;SAVE SP
11623                                     ;NOW SET UP REAL VECTOR
11624 045714 012737 045726 000024      MOV      #PWRUP,PWRVEC ;;SET UP VECTOR
11625 045722 000000      $DOWN: HALT
11626 045724 000776      BR      $DOWN          ;;HANG UP

```



```

11629 ;*****
11630 ;POWER UP ROUTINE
11631 045726 $PWRUP:
11635 045726 012737 046300 000024 MOV    $$ILLUP,PWRVEC ;;SET FOR FAST DOWN
11636 ;RESTORE STACK POINTER
11637 045734 013706 046304 MOV    $SAVR6,SP      ;;GET SP
11638 045740 005037 046304 CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
11639 045744 005237 046304 1$: INC    $SAVR6        ;;WAIT FOR THE INC
11640 045750 001375 BNE    1$            ;;OF A WORD
11641 ;RESTORE POSSIBLE SOFTWARE SWITCH REGISTER
11642 045752 POP    @SWR
11643 045752 012677 134660 MOV    (SP)+,@SWR
11644 045756 012700 172340 10$: MOV    #172340,R0
11645 045762 012702 172300 MOV    #KIPDR0,R2
11646 045772 012701 000020 6$: MOV    #16.,R1
045772 012620 POP    (R0)+
11647 045774 012722 077406 MOV    #77406,(R2)+
11648 046000 077104 SOB    R1,6$
11649 ;RESTORE MMR3,2,1,0
11650 046002 005737 002456 TST    NOSUPER
11651 046006 001002 BNE    11$
11652 046010 POP    MMR3
11653 046010 012637 172516 11$: POP    MMR2,MMR1,MMR0
046014 012637 177576 MOV    (SP)+,MMR2
046020 012637 177574 MOV    (SP)+,MMR1
046024 012637 177572 MOV    (SP)+,MMR0
11654 ;RESTORE ECC REGISTERS
11655 046030 013701 002224 MOV    TOTCSRS,R1    ;GET CSR'S
11656 046034 042701 177400 BIC    #177400,R1
11657 046040 BEGIN LCSRRESTORE
046040
11658 046040 FOR CSRNO := #36 DOWNT0 #0 BY #2 B101:;;;;;
046040 012737 000036 002152 MOV    #36,CSRNO
046046 B102:;;;;;
11659 046046 006201 ASR    R1
11660 046050 ON.ERROR
046050 103003 BCC   L445
11661 046052 POP    CSR
046052 012637 002150 MOV    (SP)+,CSR
11662 046056 104425 LOADCSR
11663 046060 END ;OF ON.ERROR
046060 L445:;;;;;
11664 046060 IF R1 EQ #0 THEN LEAVE LCSRRESTORE
046060 005701 TST   R1
046062 001407 BEQ   E101
11665 046064 END ;OF FOR CSRNO
046064 162737 000002 002152 SUB   #2,CSRNO
046072 023727 002152 000000 CMP   CSRNO,#0
046100 002362 BGE   B102
046102 E102:;;;;;
11666 046102 END LCSRRESTORE
046102 E101:;;;;;
11667 ;COPY KERNEL MAP TO USER & SUPERVISOR
11668 046102 012700 172300 MOV    #KIPDR0,R0
11669 046106 012701 177600 MOV    #UIPDR0,R1

```

11670	046112	012702	172200		MOV	#SIPDR0,R2
11671	046116	012703	000040		MOV	#32.,R3
11672	046122	011021		3\$:	MOV	(R0),(R1)+
11673	046124	012022			MOV	(R0)+,(R2)+
11674	046126	077303			SOB	R3,3\$

```

11676 ;RESTORE SUPERVISOR & USER STACK POINTERS
11677 046130 005737 002456 TST NOSUPER
11678 046134 001006 BNE 13$
11679 046136 POP RO
                                MOV (SP)+,RO
11680 046136 012600 SUPERVISOR ;ENTER SUPERVISOR MODE
                                ;GO TO SUPERVISOR MODE
046140 052737 040000 177776 BIS #BIT14,PSW
                                .DSABL CRF
11681 046146 010006 MOV RO,SSP
11682 046150 104417 KERNEL ;ENTER KERNEL MODE
11683 046152 13$: POP RO
                                MOV (SP)+,RO
046152 012600 USER
11684 046154 052737 140000 177776 BIS #BIT15!BIT14,PSW ;GO TO USER MODE
                                .DSABL CRF
11685 046162 010006 MOV RO,USP
11686 046164 104417 KERNEL ;ENTER KERNEL MODE
11687 ;RESTORE SUPERVISOR PAR'S
11688 046166 012700 172240 MOV #172240,RO
11689 046172 012701 000020 MOV #16.,R1
11690 046176 7$: POP (RO)+
                                MOV (SP)+,(RO)+
046176 012620 SOB R1,7$
11691 046200 077102 ;RESTORE USER PAR'S & PDR7
11692 MOV #177636,RO
11693 046202 012700 177636 MOV #17.,R1
11694 046206 012701 000021 POP (RO)+
11695 046212 8$: MOV (SP)+,(RO)+
046212 012620 SOB R1,8$
11696 046214 077102 ;RESTORE POSSIBLE SOFTWARE DISPLAY REGISTER
11697 MOV $PATMAR,@DISPLAY
11698 046216 013777 002012 134414 MOV $PATMAR,DISPREG
11699 046224 013737 002012 000174 POP CSRNO,R5,R4,R3,R2,R1,RO
11700 046232 MOV (SP)+,CSRNO
046232 012637 002152 MOV (SP)+,R5
046236 012605 MOV (SP)+,R4
046240 012604 MOV (SP)+,R3
046242 012603 MOV (SP)+,R2
046244 012602 MOV (SP)+,R1
046246 012601 MOV (SP)+,RO
046250 012600 MOV (SP)+,RO
11701 046252 012737 045352 000024 MOV #PWRDN,PWRVEC ;;SET UP THE POWER DOWN VECTOR
11702 046260 TYPE MSG051 ;REPORT THE POWER FAILURE
046260 104401 067652 TYPEIT ,MSG051
                                .DSABL CRF
11703 ;RESTORE CACHE STATUS
11704 046264 005737 002544 TST CACHKN
11705 046270 001402 BEQ 9$
11706 046272 POP CONTRL
                                MOV (SP)+,CONTRL
046272 012637 177746
11707 046276 000002 9$: RTI
11708 046300 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
11709 046302 000776 BR $ILLUP ;; BEFORE THE POWER DOWN WAS COMPLETE
11710 046304 000000 $SAVR6: 0 ;;PUT THE SP HERE
11711 .EVEN

```

```

11723 046306      WOOFS: SUBTST <<POWER FAIL WHILE RELOCATED>>
;*****
;*SUBTEST      POWER FAIL WHILE RELOCATED
;*****
11724 046306      PUSH      BANK
046306 013746 002102      MOV BANK,-(SP)
11725 046312 005037 002102      CLR      BANK
11726 046316      MAP      BANK      ;MAP SUPERVISOR SPACE (TEST AREA) TO BANK
046316 010346      MOV R3,-(SP)
046320 013703 002102      MOV      BANK,R3
046324 004737 035716      CALL     MAPPER
      .DSABL CRF
11727 046330 012603      MOV (SP)+,R3
046332 052737 040000 177776      SUPERVISOR      ;ENTER SUPERVISOR MODE
      BIS      #BIT14,PSW      ;GO TO SUPERVISOR MODE
      .DSABL CRF
11728 046340 013737 060024 046704      MOV      FIRST+PWRVEC,WOOPSAV
11729 046346 013737 060026 046706      MOV      FIRST+PWRVEC+2,WOOPSAV+2
11730 046354      BMOV     FIRST+WOOPUP,WOOPSAV+4,WOOPEND-WOOPUP/2+12.
046354 004537 041024      JSR R5,BLOCK3
046360 000105      WOOPEND-WOOPUP/2+12.
046362 046710      WOOPSAV+4
046364 126472      FIRST+WOOPUP
      .DSABL CRF
11731 046366 012737 046472 060024      MOV      #WOOPUP,FIRST+PWRVEC
11732 046374 012737 000340 060026      MOV      #340,FIRST+PWRVEC+2
11733 046402      BMOV     WOOPUP,FIRST+WOOPUP,WOOPEND-WOOPUP/2
046402 004537 041024      JSR R5,BLOCK3
046406 000071      WOOPEND-WOOPUP/2
046410 126472      FIRST+WOOPUP
046412 046472      WOOPUP
      .DSABL CRF
11734 046414 012700 172340      MOV      #KIPARO,P0
11735 046420 012701 126654      MOV      #FIRST+WOOPEND,R1
11736 046424 012702 000010      MOV      #8,R2
11737 046430 012021      1$: MOV      (R0)+,(R1)+
11738 046432 077202      SOB      R2,1$
11739 046434 005737 002456      TST      NOSUPER
11740 046440 001002      BNE     2$
11741 046442 013721 172516      MOV      MMR3,(R1)+
11742 046446 013721 177576      2$: MOV      MMR2,(R1)+
11743 046452 013721 177574      MOV      MMR1,(R1)+
11744 046456 013721 177572      MOV      MMRO,(R1)+
11745 046462 104417      KERNEL
11746 046464      POP      BANK      ;ENTER KERNEL MODE
046464 012637 002102      MOV (SP)+,BANK
11747 046470 000207      RETURN

```


11750 046472

11751 046472 012700 046654
 11752 046476 012701 172340
 11753 046502 012703 172300
 11754 046506 012702 000010
 11755 046512 012021
 11756 046514 012723 077406
 11757 046520 077204
 11758 046522 005737 002456
 11759 046526 001002
 11760 046530 012037 172516
 11761 046534 012037 177576
 11762 046540 012037 177574
 11763 046544 012037 177572
 11764 046550 013706 046304
 11765 046554
 046554 013746 002102
 11766 046560 005037 002102
 11767 046564
 046564 010346
 046566 013703 002102
 046572 004737 035716
 046576 012603
 11768 046600
 046600 052737 040000 177776
 11769 046606 013737 046704 060024
 11770 046614 013737 046706 060026
 11771
 11772
 11773 046622 012700 046710
 11774 046626 012701 000105
 11775 046632 012702 126472
 11776 046636 012022
 11777 046640 077102
 11778
 11779 046642 104417
 11780 046644
 046644 012637 002102
 11781 046650 000137 045726
 11782 046654 000014
 11785 046704 000107

```

WOOPUP: SUBTST <<POWER UP FROM BANK 0 TO RELOCATION>>
;*****
;*SUBTEST POWER UP FROM BANK 0 TO RELOCATION
;*****
MOV #WOOPEND,R0
MOV #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
    
```

.SBTTL IO SUBROUTINES

.SBTTL ROUTINE TYPE

;ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.

;*
;*CALL:
;*1) USING A TRAP INSTRUCTION
;* TYPE MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;* TYPE
;* MESADR
;*

\$TYPE: TSTB \$TPFLG ;;IS THERE A TERMINAL?
BMI 6\$;BR IF NO
1\$: MOV RO,-(SP) ;;SAVE RO
MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
4\$: MOV (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
BNE 7\$;BR IF IT ISN'T THE TERMINATOR
TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
5\$: MOV (SP)+,RO ;;RESTORE RO
6\$: ADD #2,(SP) ;;ADJUST RETURN PC
RTI ;;RETURN
7\$: CMPB #HT,(SP) ;BRANCH IF NOT <HT>
BNE 11\$
11\$: MOV #',(SP) ;REPLACE TAB WITH SPACE
CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
BNE 8\$
TST (SP)+ ;;POP <CR><LF> EQUIV
TYPE ;;TYPE A CR AND LF
TYPEIT
.DSABL CRF

\$CRLF
CLRB \$CHARCNT ;;CLEAR CHARACTER COUNT
BR 4\$;;GET NEXT CHARACTER
8\$: CALL \$TYPEC ;;GO TYPE THIS CHARACTER
9\$: CMPB \$FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
BNE 4\$;;IF NO GO GET NEXT CHAR.
MOV \$NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
AND THE NULL CHAR.
10\$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
BLT 9\$;;BR IF NO--GO POP THE NULL OFF OF STACK
CALL \$TYPEC ;;GO TYPE A NULL
DECB \$CHARCNT ;;DO NOT COUNT AS A COUNT
BR 10\$;;LOOP

XOCHAR: .WORD 0
\$TYPEC: PUSH R1
MOV R1,-(SP)
MOVB 4(SP),R1
TST CACHKN
11843 047262 001402 BEQ 2\$

11790
11791
11792
11793
11794
11795
11796
11797
11798
11799
11800
11801
11802
11803
11804
11805
11806
11807
11808
11809 047122 105737 002360
11810 047126 100407
11811 047130 010046
11812 047132 017600 000002
11813 047136 112046
11814 047140 001005
11815 047142 005726
11816 047144 012600
11817 047146 062716 000002
11818 047152 000002
11819 047154 122716 000011
11820 047160 001002
11821 047162 112716 000040
11822 047166 122716 000200
11823 047172 001006
11824 047174 005726
11825 047176
047176 104401
11826 047200 002660
11827 047202 105037 047434
11828 047206 000753
11829 047210 004737 047250
11830 047214 123726 002652
11831 047220 001346
11832 047222 013746 002356
11833
11834 047226 105366 000001
11835 047232 002770
11836 047234 004737 047250
11837 047240 105337 047434
11838 047244 000770
11839 047246 000000
11840 047250
047250 010146
11841 047252 116601 000004
11842 047256 005737 002544
11843 047262 001402

```

11844 047264          PUSH   CONTRL
      047264 013746 177746
11845 047270          PUSH   R0
      047270 010046
11846 047272 104424          CACHOFF      ;TURN CACHE OFF
11871 047274 105777 133346  2$: TSTB @ $TPS    ;:WAIT UNTIL PRINTER IS READY
11872 047300 100375          BPL 3$
11873 047302 005037 047246  CLR  X0CHAR
11874 047306 105777 133330  TSTB @ $TKS    ;;CHECK FOR XOFF
11875 047312 100032          BPL NC        ;;SKIP IF NO CHARACTER
11876 047314 117737 133324 047246  MOVB @ $TKB,X0CHAR ;;SAVE THE CHARACTER
11877 047322 042737 177600 047246  BIC #+C177,X0CHAR ;;STRIP OFF ASCII
11878 047330 023727 047246 000023  CMP  X0CHAR,#023  ;;WAS IT A CONTROL S?
11879 047336 001020          BNE NC        ;;BRANCH IF NOT
11880 047340 105777 133276  3$: TSTB @ $TKS    ;:WAIT FOR CHARACTER
11881 047344 100375          BPL CONTS3
11882 047346 117737 133272 047246  MOVB @ $TKB,X0CHAR ;;GET CHARACTER
11883 047354 042737 177600 047246  BIC #+C177,X0CHAR ;;STRIP OFF ASCII
11884 047362          IF X0CHAR EQ #21 ;; IF IT IS A +Q
      047362 023727 047246 000021  CMP X0CHAR,#21
      047370 001002          BNE L446
11885 047372 000402          BR  NC
11886 047374          ELSE
      047374 000401          BR  L447
      047376          L446:;;;;;
11887 047376 000760          BR  CONTS3
11888 047400          END ;OF IF X0CHAR
      047400          L447:;;;;;
11889 047400 110177 133244  000002  NC:  MOVB R1,@ $TPB  ;;LOAD CHAR TO BE TYPED INTO DATA REG.
11893 047404 122766 000015 000002  CMPB #CR,2(SP)  ;;IS CHARACTER A CARRIAGE RETURN?
11894 047412 001003          BNE 1$        ;;BRANCH IF NO
11895 047414 105037 047434  CLRB $CHARCNT  ;;YES--CLEAR CHARACTER COUNT
11896 047420 000406          BR $TYPEX    ;;EXIT
11897 047422 122766 000012 000002  1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
11898 047430 001402          BEQ $TYPEX  ;;BRANCH IF YES
11899 047432 105227          INCB (PC)+  ;;COUNT THE CHARACTER
11900 047434 000000          $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
11901 047436          $TYPEX: POP R0
      047436 012600          MOV (SP)+,R0
11902 047440 005737 002544          TST CACHKN    ;IS THERE A CACHE?
11903 047444 001402          BEQ 2$        ;BRANCH IF NOT
11904 047446          POP  CONTRL  ;POP CACHE STATUS
      047446 012637 177746          MOV (SP)+,CONTRL
11905 047452          2$: POP R1
      047452 012601          MOV (SP)+,R1
11906 047454 000207          RETURN
11907 047456          SUPLIMIT:;!!!!!!!!!!!!!!!!THIS IS THE LIMIT ON SUPERVISOR MAPPED TO MUT SPACE

```


.SBTTL ERROR DATA SETUP

: USE THIS IF THIS CONDITION DISCRIBES THE ERROR

: PERR01 TRAP
: BAD DATA IN R0 UNLESS ABORTED
: THEN BAD DATA IS POINTED TO BY -(R4)
: GOOD DATA IN R5: PERR02 TRAP
: BAD DATA IN R1 UNLESS ABORTED
: THEN BAD DATA IS POINTED TO BY -(R4)
: GOOD DATA IN R2: PERR03 TRAP
: BAD DATA IS POINTED TO BY -(R1)
: GOOD DATA IN R4: PERR04 TRAP
: BAD DATA IN R4 UNLESS ABORTED
: THEN BAD DATA IS POINTED TO BY -2(R0)
: GOOD DATA IN R2: PERR05 JSR PC
: BAD DATA IS POINTED TO BY -(R0)
: GOOD DATA IN R2
: RETURN AFTER SETTING UP GOOD,BAD,ADDRESS: PERR06 JSR PC
: BAD DATA IS POINTED TO BY -(R0)
: GOOD DATA IS ZERO
: RETURN AFTER SETTING UP GOOD,BAD,ADDRESS: PERR07 TRAP
: BAD DATA IN R2 UNLESS ABORTED
: THEN BAD DATA IS POINTED TO BY (R1)
: GOOD DATA IN DATBUF: PERR10 TRAP
: BAD DATA IN R2 UNLESS ABORTED
: THEN BAD DATA IS POINTED TO BY 2(R1)
: GOOD DATA IN DATBUF+2: PERR11 TRAP
: BYTE TEST
: BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
: THEN BAD DATA IS POINTED TO BY (R1)
: GOOD DATA IS A ZERO BYTE: PERR12 TRAP
: BYTE TEST
: BAD DATA IN RIGHT BYTE OF R0 UNLESS ABORTED
: THEN BAD DATA IS POINTED TO BY (R1)
: GOOD DATA IS A BYTE OF ONES: PERR13 TRAP
: BAD DATA IN R0 UNLESS ABORTED12585
12586
12587
12588
12589
12590
12591
12592
12593
12594
12595
12596
12597
12598
12599
12600
12601
12602
12603
12604
12605
12606
12607
12608
12609
12610
12611
12612
12613
12614
12615
12616
12617
12618
12619
12620
12621
12622
12623
12624
12625
12626
12627
12628
12629
12630
12631
12632
12633
12634
12635
12636
12637
12638
12639
12640
12641


```

12699
12700
12701
12702
12703
12704
12705
12706
12707
12708
12709
12710
12711
12712
12713
12714
12715
12716
12717
12718
12719
12720
12721
12722
12723
12724
12725
12726

```

```

:
: PERR27      TRAP
:             BAD DATA IS DOUBLE WORD POINTED TO BY R1 AND IN LOW 7 BITS OF R0
:             GOOD DATA IS 000000,,000000,,077
:
: PERR30      TRAP
:             BAD DATA IS POINTED TO BY -16(SP)
:             GOOD DATA IS IN R1
:
: PERR31      TRAP
:             SPECIAL ECC FAILURE HANDLER
:
: PERR32      TRAP
:             SPECIAL ECC FAILURE HANDLER
:
: PERR33      TRAP
:             SPECIAL ECC FAILURE HANDLER
:
: PERR34      TRAP
:             SPECIAL ECC FAILURE HANDLER
:
: PERR35      TRAP
:             SPECIAL BRANCH GOBBLE FAILURE HANDLER
:
:             CALLING SEQUENCE FOR TRAP TYPES
:             BEQ      2$                ;NO - ERROR,BRANCH FOR CARD
:             PERRXX   ;TRAP TO ERROR ROUTINE
:2$: NEXT      INSTRUCTION            ;CONTINUE TESTING

```

12729	047456	010437	002034		\$PER01: MOV R4, ADDRESS	
12730	047462	162737	000002	002034	SUB #2, ADDRESS	
12731	047470	010037	002052		MOV R0, BAD	
12732	047474	010537	002044		MOV R5, GOOD	
12733	047500	000137	050212		JMP PERRAW	
12734						
12735	047504	010437	002034		\$PER02: MOV R4, ADDRESS	
12736	047510	162737	000002	002034	SUB #2, ADDRESS	
12737	047516	010137	002052		MOV R1, BAD	
12738	047522	010237	002044		MOV R2, GOOD	
12739	047526	000137	050212		JMP PERRAW	
12740						
12741	047532	010137	002034		\$PER03: MOV R1, ADDRESS	
12742	047536	162737	000002	002034	SUB #2, ADDRESS	
12743	047544	010437	002044		MOV R4, GOOD	
12744	047550	016137	177776	002052	MOV -2(R1), BAD	
12745	047556	000137	050212		JMP PERRAW	
12746						
12747	047562	010037	002034		\$PER04: MOV R0, ADDRESS	
12748	047566	162737	000002	002034	SUB #2, ADDRESS	
12749	047574	010437	002052		MOV R4, BAD	
12750	047600	010237	002044		MOV R2, GOOD	
12751	047604	000137	050212		JMP PERRAW	
12752						
12753	047610	010237	002044		PERR05: MOV R2, GOOD	
12754	047614	014037	002052		PERA05: MOV -(R0), BAD	
12755	047620	010037	002034		MOV R0, ADDRESS	
12756	047624	062700	000002		ADD #2, R0	;RESTORE R0
12757	047630	004737	034152		CALL BADSTACK	
12758	047634	000207			RETURN	
12759						
12760	047636	005037	002044		PERR06: CLR GOOD	
12761	047642	000764			BR PERA05	
12762						
12763	047644	010137	002034		\$PER07: MOV R1, ADDRESS	
12764	047650	010237	002052		MOV R2, BAD	
12765	047654	013737	002242	002044	MOV DATBUF, GOOD	
12766	047662	000137	050212		JMP PERRAW	
12767						
12768	047666				\$PER10: LET ADDRESS := R1 + #2	
	047666	010137	002034			MOV R1, ADDRESS
	047672	062737	000002	002034		ADD #2, ADDRESS
12769	047700				LET BAD := R2	
	047700	010237	002052			MOV R2, BAD
12770	047704				LET GOOD := DATBUF+2	
	047704	013737	002244	002044		MOV DATBUF+2, GOOD
12771	047712	000137	050212		JMP PERRAW	
12772						
12773	047716				\$PER11: LET ADDRESS := R1	
	047716	010137	002034			MOV R1, ADDRESS
12774	047722				LET BAD := R0	
	047722	010037	002052			MOV R0, BAD
12775	047726				LET GOOD := #0	
	047726	005037	002044			CLR GOOD
12776	047732	000137	050264		JMP PERRAB	
12777						
12778	047736				\$PER12: LET ADDRESS := R1	

12779	047736	010137	002034	
	047742			
12780	047742	010037	002052	
	047746			
12781	047746	012737	000377	002044
	047754	000137	050264	

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

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

12784	047760			\$PER13: LET ADDRESS := R1	
	047760	010137	002034		MOV R1,ADDRESS
12785	047764			LET BAD := R0	
	047764	010037	002052		MOV R0,BAD
12786	047770			LET GOOD := #0	
	047770	005037	002044		CLR GOOD
12787	047774	000137	050212	JMP PERRAW	
12788					
12789	050000			\$PER14: LET ADDRESS := R1	
	050000	010137	002034		MOV R1,ADDRESS
12790	050004			LET BAD := R0	
	050004	010037	002052		MOV R0,BAD
12791	050010			LET GOOD := ONES	
	050010	013737	002614 002044		MOV ONES,GOOD
12792	050016	000137	050212	JMP PERRAW	
12793					
12794	050022			\$PER15: LET ADDRESS := R1	
	050022	010137	002034		MOV R1,ADDRESS
12795	050026			LET BAD := R0	
	050026	010037	002052		MOV R0,BAD
12796	050032			LET GOOD := TSTDAT	
	050032	013737	002246 002044		MOV TSTDAT,GOOD
12797	050040	000137	050212	JMP PERRAW	
12798					
12799	050044			\$PER16: LET ADDRESS := R1	
	050044	010137	002034		MOV R1,ADDRESS
12800	050050			LET BAD := R0	
	050050	010037	002052		MOV R0,BAD
12801	050054			LET GOOD := TSTDAT+2	
	050054	013737	002250 002044		MOV TSTDAT+2,GOOD
12802	050062	000453		BR PERRAW	
12803					
12804	050064			\$PER17: LET ADDRESS := R1	
	050064	010137	002034		MOV R1,ADDRESS
12805	050070			LET BAD := R0	
	050070	010037	002052		MOV R0,BAD
12806	050074			LET GOOD := R2	
	050074	010237	002044		MOV R2,GOOD
12807	050100	000444		BR PERRAW	
12808					
12809	050102			\$PER20: LET ADDRESS := R1	
	050102	010137	002034		MOV R1,ADDRESS
12810	050106			LET BAD := R0	
	050106	010037	002052		MOV R0,BAD
12811	050112			LET GOOD := R3	
	050112	010337	002044		MOV R3,GOOD
12812	050116	000435		BR PERRAW	
12813					
12814	050120			\$PER21: LET ADDRESS := R1	
	050120	010137	002034		MOV R1,ADDRESS
12815	050124			LET BAD := R0	
	050124	010037	002052		MOV R0,BAD
12816	050130			LET GOOD := #177	
	050130	012737	000177 002044		MOV #177,GOOD
12817	050136	000477		BR PERRA7	
12818					
12819	050140			\$PER22: LET ADDRESS := R1	

12820	050140	010137	002034			MOV R1,ADDRESS
	050144			LET BAD := R2		
12821	050144	010237	002052			MOV R2,BAD
	050150			LET GOOD := R0		
12822	050150	010037	002044			MOV R0,GOOD
	050154	000416		BR PERRAW		
12823						
12824	050156			\$PER23: LET ADDRESS := R1		
	050156	010137	002034			MOV R1,ADDRESS
12825	050162			LET BAD := R0		
	050162	010037	002052			MOV R0,BAD
12826	050166			LET GOOD := R4		
	050166	010437	002044			MOV R4,GOOD
12827	050172	000407		BR PERRAW		
12828						
12829	050174			\$PER24: LET ADDRESS := R2		
	050174	010237	002034			MOV R2,ADDRESS
12830	050200			LET BAD := R0		
	050200	010037	002052			MOV R0,BAD
12831	050204			LET GOOD := R3		
	050204	010337	002044			MOV R3,GOOD
12832	050210	000400		BR PERRAW		

12834 050212

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

12835 050212 004737 050446

12836 050216 005737 002144

050216 005737 002144

050222 001402

050224 004737 045252

050230

12837 050230 005737 002022

050230 005737 002022

050234 001002

050236 004737 034152

050242

12838 050242 004737 050422

12839 050246 005737 002144

050246 005737 002144

050252 001002

12840 050254 104011

12841 050256 000401

050256 000401

050260

12842 050260 104012

12843 050262

050262

12844 050262 000002

12845

12846 050264

CALL PERBNK
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA

TST ABORTFLAG
BEQ L450
JSR PC,GETDATA
L450:;;;;;

IF BADPC EQ #0 THEN \$CALL BADSTACK

TST BADPC
BNE L451
JSR PC,BADSTACK
L451:;;;;;

CALL PERXOR
IF ABORTFLAG IS FALSE

TST ABORTFLAG
BNE L452

ERROR +11
ELSE

BR L453
L452:;;;;;

ERROR +12
END ;OF IF ABORTFLAG

L453:;;;;;

RTI

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

12847 050264 004737 050446

12848 050270 005737 002144

050270 005737 002144

050274 001402

050276 004737 045252

050302

12849 050302 005737 002022

050302 005737 002022

050306 001002

050310 004737 034152

050314

12850 050314 004737 050422

12851 050320 005737 002144

050320 005737 002144

050324 001002

12852 050326 104014

12853 050330 000401

050330 000401

050332

12854 050332 104015

12855 050334

050334

12856 050334 000002

CALL PERBNK
IF ABORTFLAG IS TRUE THEN \$CALL GETDATA

TST ABORTFLAG
BEQ L454
JSR PC,GETDATA
L454:;;;;;

IF BADPC EQ #0 THEN \$CALL BADSTACK

TST BADPC
BNE L455
JSR PC,BADSTACK
L455:;;;;;

CALL PERXOR
IF ABORTFLAG IS FALSE

TST ABORTFLAG
BNE L456

ERROR +14
ELSE

BR L457
L456:;;;;;

ERROR +15
END ;OF IF ABORTFLAG

L457:;;;;;

RTI

```

12859 050336          PERRA7: SUBTST <<DATA WAS A 7 BIT BYTE>>
;*****
;*SUBTEST          DATA WAS A 7 BIT BYTE
;*****
                    IF BADPC EQ #0 THEN $CALL BADSTACK
                                TST BADPC
                                BNE L460
                                JSR PC,BADSTACK
                                L460:::;:;:;

12860 050336          005737 002022
    050336          001002
    050342          004737 034152
    050344          004737 050422
    050350          004737 050446
12861 050350          004737 050422
12862 050354          004737 050446
12863 050360          104022
12864 050362          000002
12865
12866 050364          012737 100000 002046 $PER26: LET GOOD2 := #100000
    050364          012737 000100 002050 LET GOOD3 := #100
12867 050372          012737 000100 002050 JMP PERRA3
    050372          000137 045040
12868 050400          005037 002046 $PER27: CLR GOOD2
12869
12870 050404          005037 002046 LET GOOD3 := #077
12871 050410          012737 000077 002050 MOV #077,GOOD3
    050410          000137 045040 JMP PERRA3
12872 050416
12873
12874 050422          PERRA7: SUBTST <<DETERMINE XOR OF GOOD & BAD>>
;*****
;*SUBTEST          DETERMINE XOR OF GOOD & BAD
;*****
                    PUSH RO
                                MOV RO,-(SP)
12875 050422          010046
12876 050424          013700 002044 MOV GOOD,RO
12877 050430          013737 002052 002060 MOV BAD,BAD XOR
12878 050436          074037 002060 XOR RO,BAD XOR
12879 050442          012600 POP RO
12880 050444          000207 RETURN
                                MOV (SP)+,RO

```



```

12883 050446 PERBNK: SUBTST<<LOG ERROR ON BAD BANK>>
;*****
;*SUBTEST LOG ERROR ON BAD BANK
;*****
12884 ;WHILE WE'RE HERE LET'S MARK THE BAD BANK IN THE CONFIGURATION TABLE
12885 050446 PUSH RO,R1
050446 010046 MOV RO,-(SP)
050450 010146 MOV R1,-(SP)
12886 050452 013701 002102 MOV BANK,R1
12887 050456 006301 ASL R1
12888 050460 006301 ASL R1
12889 050462 052761 000001 002664 BIS #BIT0,CONFIG(R1)
12890 050470 105261 002666 INCB CONFIG+2(R1) ;BUMP BANK COUNTER
12891 050474 001002 BNE 12$ ;NO OVERFLOW - SKIP
12892 050476 105361 002666 DECB CONFIG+2(R1) ;SET BACK TO 255.
12893 050502 126137 002666 002554 12$: CMPB CONFIG+2(R1),ERRMAX ;IS IT PAST MAX?
12894 050510 101403 BLOS 11$ ;NO - SKIP
12895 050512 SET TOOMANY ;YES
050512 012737 177777 002406 MOV #-1,TOOMANY
12896 050520 11$: POP R1,R0
050520 012601 MOV (SP)+,R1
050522 012600 MOV (SP)+,R0
12897 050524 000207 RETURN
12898
12899 050526 010037 002052 PERECC: MOV RO,BAD
12900 050532 IF ADDRESS EQ TESTADD
050532 023737 002034 002412 CMP ADDRESS,TESTADD
050540 001004 BNE L461
12901 050542 013737 002246 002044 MOV TSTDAT,GOOD
12902 050550 ELSE
050550 000403 BR L462
050552 L461:;;;;;
12903 050552 013737 002250 002044 MOV TSTDAT+2,GOOD
12904 050560 END ;OF IF (R1)
050560 L462:;;;;;
12905 050560 004737 050422 CALL PERXOR
12906 050564 SET HEADER
050564 012737 177777 002612 MOV #-1,HEADER
12907 050572 000207 RETURN
12908
12909 050574 $PER31: IF REALPAT EQ #41
050574 023727 002300 000041 CMP REALPAT,#41
050602 001001 BNE L463
12910 050604 104023 ERROR +23
12911 050606 END
050606 L463:;;;;;
12912 050606 IF BADPC EQ #0 THEN $CALL BADSTACK
050606 005737 002022 TST BADPC
050612 001002 BNE L464
050614 004737 034152 JSR PC,BADSTACK
050620 L464:;;;;;
12913 050620 004737 050526 CALL PERECC
12914 050624 IF REALPAT EQ #11
050624 023727 002300 000011 CMP REALPAT,#11
050632 001001 BNE L465
12915 050634 104037 ERROR +37
12916 050636 END ;OF IF REALPAT

```

12917	050636				IF REALPAT EQ #15	L465:::~::~
	050636	023727	002300	000015		CMP REALPAT,#15
	050644	001001				BNE L466
12918	050646	104043			ERROR +43	
12919	050650				END ;OF IF REALPAT	
	050650				IF REALPAT EQ #16	L466:::~::~
12920	050650	023727	002300	000016		CMP REALPAT,#16
	050656	001001				BNE L467
12921	050660	104044			ERROR +44	
12922	050662				END ;OF IF REALPAT	
	050662				SET HEADER	L467:::~::~
12923	050662	012737	177777	002612		MOV #-1,HEADER
12924	050670	000002			RTI	

```

12927 050672          $PER32: IF BADPC EQ #0 THEN $CALL BADSTACK
      050672 005737 002022
      050676 001002
      050700 004737 034152
      050704
                                TST BADPC
                                BNE L470
                                JSR PC,BADSTACK
                                L470:;;;;;
12928 050704 010137 002034      MOV R1,ADDRESS
12929 050710 010037 002052      MOV RO,BAD
12930 050714 010237 002044      MOV R2,GOOD
12931 050720          SET HEADER
                                MOV #-1,HEADER
12932 050720 012737 177777 002612
12932 050726 104040      ERROR +40
12933 050730          SET HEADER
                                MOV #-1,HEADER
      050730 012737 177777 002612
12934 050736 000002      RTI
12935
12936 050740          $PER33: IF BADPC EQ #0 THEN $CALL BADSTACK
      050740 005737 002022
      050744 001002
      050746 004737 034152
      050752
                                TST BADPC
                                BNE L471
                                JSR PC,BADSTACK
                                L471:;;;;;
12937 050752 010137 002034      MOV R1,ADDRESS
12938 050756 010037 002052      MOV RO,BAD
12939 050762 105037 002053      CLRB BAD+1
12940 050766 012737 000377 002044      MOV #377,GOOD
12941 050774 004737 050422      CALL PERXOR
12942 051000          SET HEADER
                                MOV #-1,HEADER
      051000 012737 177777 002612
12943 051006 104041      ERROR +41
12944 051010          SET HEADER
                                MOV #-1,HEADER
      051010 012737 177777 002612
12945 051016 000002      RTI
12946
12947 051020          $PER34:IF BADPC EQ #0 THEN $CALL BADSTACK
      051020 005737 002022
      051024 001002
      051026 004737 034152
      051032
                                TST BADPC
                                BNE L472
                                JSR PC,BADSTACK
                                L472:;;;;;
12948 051032          IF #BIT15!BIT4 OFF.IN CSR
      051032 032737 100020 002150
      051040 001002
                                BIT #BIT15!BIT4,CSR
                                BNE L473
12949 051042 104016      ERROR +16          ;NO SBE OR DBE
12950 051044          ELSE
                                BR L474
      051044 000401
                                L473:;;;;;
12951 051046 104001      ERROR +1          ;EXPECTED SBE SO DBE MUST HAVE GOTTEN SET
12952 051050          END ;OF IF #BIT15!BIT4
                                L474:;;;;;
      051050 000002      RTI
12953 051050
12954
12955          ;DURING BRANCH GOBBLE THE CONDITION CODES WERE WRONG
12956 051052 004737 050446      $PER35: CALL PERBNK
12957 051056 004737 034152      CALL BADSTACK
12958 051062 013737 002032 002052      MOV BADPSW,BAD
12959 051070 012737 000012 002044      MOV #12,GOOD
12960 051076 104047      ERROR +47
12961 051100 062706 000004      ADD #4,SP
12962 051104 000207      RETURN          ;FIX STACK FROM TRAP
                                ;ABORTING TEST

```

12963								
12964	051106	010037	002044		\$PER36: MOV	R0,GOOD		
12965	051112	010137	002052		MOV	R1,BAD		
12966	051116				SET	HEADER		
	051116	012737	177777	002612			MOV	#-1,HEADER
12967	051124	104023			ERROR	+23		
12968	051126				SET	HEADER		
	051126	012737	177777	002612			MOV	#-1,HEADER
12969	051134	000002			RTI			
12970								
12971	051136	104053			\$PER37: ERROR	+53		:ILC;;REV B
12972	051140	000002			RTI			:ILC;;REV B
12973								
12974	051142	104054			\$PER40: ERROR	+54		:ILC;;REV B
12975	051144	000002			RTI			:ILC;;REV B


```

12978
12979
12980
12981
12982
12983
12984
12985
12986
12987 051146 005237 056656
12988 051152
      051152 002004
12989 051154 005037 056656
12990 051160 105237 056660
12991 051164
      051164
12992 051164 104410
12993 051166 005737 006216
12994 051172 001402
12995 051174 004737 055152
12996 051200
12997 051200 005737 052314
12998 051204 001410
12999 051206 013737 177766 052312
13000 051214 032737 000001 052312
13001 051222 001401
13002 051224 104177
13011 051226
      051226 005737 002420
      051232 001410
      051234 032777 000400 131374
      051242 001404
13012 051244 005037 002420
13013 051250 000137 040604
13014 051254
      051254
13015 051254
      051254 005737 002440
      051260 001401
13016 051262 000002
13017 051264
      051264
13018 051264
      051264 032777 040000 131344
      051272 001051
13019
13020 051274 000425
13021
13022 051276 013746 000004
13023 051302 012737 051322 000004
13024 051310 005737 177060
13025 051314 012637 000004
13026 051320 000430
13027 051322 062706 000004
13028 051326 022737 000005 004010
13029 051334 001002
13030 051336 005037 177766

```

```

.SBTTL ROUTINE SCOPE HANDLER
;*****
;THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;AND LOAD THE DISPLAY DATA INTO THE DISPLAY REGISTER
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1 LOOP ON TEST
;SW9=1 LOOP ON ERROR
;CALL
;*
* SCOPE          ;;SCOPE=IOT
$SCOPE: INC      $DEVCT      ;TELL API WE ARE ALIVE
      IF RESULT IS LT
                                BGE L475
                                CLR $DEVCT
                                INCB $UNIT
      END ;OF IF RESULT
                                L475:;;;;;
                                ;;TEST FOR CHANGE IN SOFT-SWR
                                CKSWR
                                TST TRACE
                                BEQ NOTRCE
                                CALL CONTT ;TRACE
                                NOTRCE:
                                TST CPERRF ;IS THERE A CPU ERROR REGISTER? ;R-C
                                BEQ SKJ ;BRANCH IF NOT ;R-C
                                MOV @#177766,CPSAVE ;GET CONTENTS OF ERROR REGISTER ;R-C
                                BIT #BIT0,CPSAVE ;IS THE POWER FAIL MONITOR BIT SET? ;R-C
                                BEQ SKJ ;BRANCH IF NOT ;R-C
                                ERROR +177 ;REPORT IF SO ;R-C
                                SKJ: IF STOPOK IS TRUE AND #SW8 SET.IN @SWR ;R-C
                                TST STOPOK
                                BEQ L476
                                BIT #SW8,@SWR
                                BEQ L476
                                CLR STOPOK
                                JMP EXIT
      END ;OF IF STOPOK
                                L476:;;;;;
                                IF NOSCOPE IS TRUE
                                TST NOSCOPE
                                BEQ L477
                                RTI
      END ;OF IF NOSCOPE
                                L477:;;;;;
      1$: IF #SW14 SET.IN @SWR THEN GOTO $OVER
                                BIT #SW14,@SWR
                                BNE $OVER
;####START OF CODE FOR THE XOR TESTER####
$XTSTR: BR 2$ ;IF RUNNING ON THE "XOR" TESTER CHANGE
;THIS INSTRUCTION TO A "NOP" (NOP=240)
;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV ERRVEC,-(SP) ;SET FOR TIMEOUT
MOV #1$,ERRVEC ;TIME OUT ON XOR?
TST 177060 ;RESTORE THE ERROR VECTOR
MOV (SP)+,ERRVEC ;GO TO THE NEXT TEST
BR $SVLAD ;FIX STACK FROM TRAP
;IS THIS AN 11/83 ?
1$: ADD #4,SP ;BRANCH IF NOT
CMP #5,PROTYP ;RESET CPU ERROR REGISTER
BNE 6$
CLR CPUERR

```

```

13031 051342 012637 000004      6$:   MOV    (SP)+,ERRVEC    ;;RESTORE THE ERROR VECTOR
13032 051346 000407              BR     4$                ;;LOOP ON THE PRESENT TEST
13033 051350              2$:;####END OF CODE FOR THE XOR TESTER####
13034 051350 105737 002014      3$:   TSTB   $ERFLG        ;;HAS AN ERROR OCCURRED?
13035 051354 001412              BEQ    $SVLAD            ;;BR IF NO
13036 051356 032777 001000 131252      BIT    #SW9,@SWR        ;;LOOP ON ERROR?
13037 051364 001404              BEQ    5$                ;;BR IF NO
13038 051366 013737 002624 002622 4$:   MOV    $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
13039 051374 000410              BR     $OVER            ;;
13040 051376 105037 002014      5$:   CLRB   $ERFLG          ;;ZERO THE ERROR FLAG
13041 051402 011637 002622  $SVLAD: MOV   (SP),$LPADR    ;;SAVE SCOPE LOOP ADDRESS
13042 051406 011637 002624      MOV   (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
13043 051412 005037 002362      CLR   $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
13044 051416 004737 051430  $OVER: CALL  GETDIS       ;;
13045 051422 013716 002622      MOV   $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
13046 051426 000002              RTI                    ;;FIXES PS

```

13048 051430

GETDIS: SUBTST <<SUBR DISPLAY>>

;SUBTEST SUBR DISPLAY

13049 051430 113737 002102 002013
13050 051436 113737 002300 002012
13051 051444
051444 010046
13052 051446 005737 002126
13053 051452 001403
13054 051454 052737 100000 002012
13055 051462
13059 051462 013777 002012 131150
13060 051470 013737 002012 000174
13061 051476
051476 012600
13062 051500 000207

MOV BANK,\$BANK
MOV REALPAT,\$PATMAR
PUSH R0
TST RLFLAG ;ARE WE RELOCATED? MOV R0,-(SP)
BEQ 1\$;NO - SKIP
BIS #BIT15,\$PATMAR ;YES - SET MSB
1\$: MOV \$PATMAR,@DISPLAY
MOV \$PATMAR,DISPREG ;SOFTWARE DISPLAY REGISTER
POP R0
RETURN MOV (SP)+,R0


```

13065
13066
13067
13068
13069
13070
13071
13072
13073
13074
13075
13076
13077
13078
13079 051502 105737 056666
13080 051506 001402
13081 051510 004737 000000
13082 051514 105037 052310
13083 051520
      051520 005737 002430
      051524 001027
13084 051526 104410
13085 051530
13086 051530 105237 002014
13087 051534 001775
13088 051536 004737 051430
13089 051542 013737 002012 056652
13090 051550 032777 002000 131060
13091 051556 001404
13092 051560
      051560 104401 002653
13093 051564
      051564 104401 066060
13094 051570 005237 002630
13095 051574
      051574 100003
13096 051576 012737 077777 002630
13097 051604
      051604
13098 051604
      051604
13099 051604 011637 002020
13100 051610 162737 000002 002020
13101 051616 010637 002024
13102 051622 016637 000002 002030
13103 051630 117737 130164 002015
13104
13105 051636 122737 000177 002015
13106 051644 001431
13107 051646 105737 052310
13108 051652 001024
13109 051654 005737 052314
13110 051660 001423
13111 051662 013737 177766 052312
13112 051670 032737 000001 052312

```

```

.SBTTL ROUTINE ERROR HANDLER
;*****
;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
;AND GO TO $ERRTYP ON ERROR
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW15=1 HALT ON ERROR
;SW13=1 INHIBIT ERROR TYPEOUTS
;SW10=1 BELL ON ERROR
;SW9=1 LOOP ON ERROR
;CALL
;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

.ENABL LSB
$ERROR: TSTB $ENV ;ARE WE RUNNING IN APT MODE ?
        BEQ NOAPT ;YES - DO UFD ABORT
        JSR PC,ABORT ;GOTO UFD ABORT
NOAPT: CLR B IBSAVE ;R-C
        IF NOERROR IS FALSE ;R-C
                                TST NOERROR
                                BNE L500
                                ;R-C
                                ;;TEST FOR CHANGE IN SOFT-SWR
BACK: CKSWR ;R-C
1$: INCB $ERFLG ;SET THE ERROR FLAG
    BEQ 1$ ;DON'T LET THE FLAG GO TO ZERO
    CALL GETDIS ;SETUP DISPLAY STUFF
    MOV $PATMAR,$TESTN ;FOR APT
    BIT #SW10,@SWR ;BELL ON ERROR?
    BEQ 2$ ;NO - SKIP
    TYPE $BELL ;RING BELL
    TYPEIT , $BELL
    .DSABL CRF
    TYPE MSG014 ;CONTROL Z
    TYPEIT ,MSG014
    .DSABL CRF
2$: INC $ERTTL ;COUNT THE NUMBER OF ERRORS
    IF RESULT IS MI ;R-C
                                BPL L501
                                MOV #77777,$ERTTL
                                END ;OF IF RESULT
                                L501:;;;;;
                                END ;OF IF NOERROR
                                L500:;;;;;
                                MOV (SP),ERRPC ;GET ADDRESS OF ERROR INSTRUCTION
                                SUB #2,ERRPC
                                MOV SP,ERRSP
                                MOV 2(SP),ERRPSW
                                MOV B @ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
                                CMPB #177,$ITEMB ;IS THIS THE POWER FAIL CALL? ;R-C
                                BEQ 1001$ ;BRANCH IF SO ;R-C
                                TSTB IBSAVE ;2ND ERROR CALL? ;R-C
                                BNE 1000$ ;BRANCH IF SO ;R-C
                                TST CPERRF ;IS THERE A CPU ERROR REGISTER? ;R-C
                                BEQ 1001$ ;BRANCH IF NOT ;R-C
                                MOV 177766,CPSAVE ;SAVE CONTENTS ;R-C
                                BIT #BIT0,CPSAVE ;POWER MONITOR BIT SET? ;R-C

```



```

13113 051676 001414          BEQ      1001$          ;BRANCH IF NOT          ;R-C
13114 051700 042737 000001 177766 BIC      #BIT0,177766 ;CLEAR THE BIT          ;R-C
13115 051706 112737 002015 052310 MOVB    #ITEMB,IBSAVE ;MAKE IBSAVE NON-ZERO FOR DUAL CALL ;R-C
13116 051714 112737 000177 002015 MOVB    #177,$ITEMB ;SET $ITEMB TO POWER FAIL POINTER ;R-C
13117 051722 000402          BR       1001$          ;R-C
13118 051724 105037 052310          CLRB    IBSAVE          ;R-C
13119 051730          1000$: ;R-C
13120 051730          1001$:
051730 005737 002430          IF NOERROR IS FALSE
051734 001043          TST NOERROR
13121 051736          IF BADPC NE #0          BNE L502
051736 005737 002022          TST BADPC
051742 001416          BEQ L503
13122 051744 013737 002022 002020 MOV BADPC,ERRPC
13123 051752 162737 000002 002020 SUB #2,ERRPC
13124 051760 013737 002026 002024 MOV BADSP,ERRSP
13125 051766 013737 002032 002030 MOV BADPSW,ERRPSW
13126 051774 005037 002022 CLR BADPC
13127 052000          END ;IF
052000          L503:;;;;;;
13128 052000 013737 002020 056650 MOV ERRPC,$FATAL ;FOR APT
13129 052006 004737 050446 CALL PERBNK ;LOG ERROR ON BANK
13130 052012          IF #SW13 SET.IN @SWR
052012 032777 020000 130616 BIT #SW13,@SWR
052020 001401          BEQ L504
13131 052022 000420          BR 3$
13132 052024          END ;OF IF #SW13
052024          L504:;;;;;;
13133 052024          IF #SW5 SET.IN @SWR AND TOOMANY IS TRUE
052024 032777 000040 130604 BIT #SW5,@SWR
052032 001404          BEQ L505
052034 005737 002406 TST TOOMANY
052040 001401          BEQ L505
13134 052042          GOTO 3$
052042 000410          BR 3$
13135 052044          END ;OF IF #SW5
052044          L505:;;;;;;
13136 052044          END ;OF IF NOERROR
052044          L502:;;;;;;
13137 052044 004737 052316 CALL $ERRTYP ;GO TO USER ERROR ROUTINE
13138 052050          IF MONFLG IS TRUE ;SHOULD WE RETURN TO XXDP MONITOR???
052050 005737 002276 TST MONFLG
052054 001403          BEQ L506
13139 052056 013706 002274 MOV SAVMON,SP ;GET MONITOR ADDRESS
13140 052062 000207 RTS PC ;GO TO MONITOR
13141 052064          END ;
052064          L506:;;;;;;

```

```

13143 052064          3$: IF NOERROR IS FALSE
      052064 005737 002430
      052070 001072
13144 052072          TST @SWR          ;;HALT ON ERROR
      005777 130540          BPL 7$          ;;SKIP IF CONTINUE
13145 052076          $HALT: HALT          ;;HALT ON ERROR!
      100002          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
13146 052100          7$: IF NOSCOPE IS FALSE AND #SW9 SET.IN @SWR
      000000
13147 052102          TST NOSCOPE
      104410          BNE L510
13148 052104          052104 005737 002440          BIT #SW9,@SWR
      052110 001006          BEQ L510
      052112 032777 001000 130516
      052120 001402
13149 052122          MOV $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
      013716 002624          END ;OF IF NOSCOPE
13150 052126          TST $ESCAPE          L510:;;;;;
      052126 005737 002362          BEQ 9$          ;;CHECK FOR AN ESCAPE ADDRESS
13151 052126          BEQ 9$          ;;BR IF NONE
13152 052132          MOV $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
      001402
13153 052134          9$: IF DETFLAG IS FALSE
      013716 002362          TST DETFLAG
13154 052140          052140 005737 002220          BNE L511
      052144 001043
13155 052146          CMP #5,PROTYP          ;IS THIS AN 11/83 ?
      022737 000005 004010          BNE 11$
13156 052154          CLR CPUERR
      001002
13157 052156          11$: IF ACTFLAG IS TRUE OR APTFLAG IS TRUE OR FATAL$ IS TRUE
      005037 177766          TST ACTFLAG
13158 052162          052162 005737 002350          BNE L512
      052166 001006          TST APTFLAG
      052170 005737 002352          BNE L512
      052174 001003          TST FATAL$
      052176 005737 002064          BEQ L513
      052202 001405          L512:;;;;;
      052204
13159 052204          MOV #1,$MSGTY          ;FOR APT
      012737 000001 056646          JMP EXIT
13160 052212          END ;OF IF ACTFLAG
      000137 040604          L513:;;;;;
13161 052216          IF XXDPCHAIN IS TRUE AND $ERTTL HI #20
      052216
13162 052216          052216 005737 002354          TST XXDPCHAIN
      052222 001414          BEQ L514
      052224 023727 002630 000020          CMP $ERTTL,#20
      052232 101410          BLOS L514
13163 052234          TYPE MSG066          ;ERROR COUNT EXCEEDED 20 - ABORTING FOR XXDP CHAIN
      052234 104401 070051          TYPEIT ,MSG066
      .DSABL CRF
13164 052240          MOV 42,R0
      013700 000042          CLR 42
13165 052244          JMP $ZAP42
      005037 000042          END ;OF IF XXDPCHAIN
13166 052250          END ;OF IF DETFLAG
      000137 014022          L514:;;;;;
13167 052254          L511:;;;;;
      052254          ELSE
13168 052254          BR L515
      052254 000403          L507:;;;;;
13169 052254          SET HEADER
      052256
13170 052256          012737 177777 002612          MOV #-1,HEADER
13171 052264          END ;OF IF NOERROR

```

```

13172 052264          10$: CLEAR TOOMANY,NOERROR          L515:;;;;;
      052264 005037 002406
      052270 005037 002430          CLR TOOMANY
13173 052274 105737 052310          CLR NOERROR
      052300 001402          TSTB IBSAVE          ;POWER FAIL ERROR CALL? ;R-C
13174 052300 001402          BEQ 213$          ;R-C
13175 052302 000137 051530          JMP BACK          ;JUMP IF SO          ;R-C
13176 052306 000002          213$: RTI          ;;RETURN          ;R-C
13177 052310 000000          IBSAVE: .WORD 0          ;R-C
13178 052312 000000          CPSAVE: .WORD 0          ;R-C
13179 052314 000000          CPERRF: .WORD 0          ;R-C
13180          .DSABL LSB

```


.SBTTL ROUTINE ERROR MESSAGE TYPEOUT

```

13183
13184
13185
13186
13187
13188
13189
13190 052316 104415
13191 052320 104401 002660
13192 052324 005000
13193 052326 153700 002015
13194 052332 001004
13195
13196 052334 013746 002020
052340 104402
13197 052342 000511
13198 052344 122700 000177
13199 052350 001003
13200 052352 012700 052626
13201 052356 000406
13202 052360 005300
13203 052362 006300
13204 052364 006300
13205 052366 006300
13206 052370 062700 057274
13207 052374 012037 052432
13208 052400 001417
13209 052402 005737 002430
13210 052406 001003
13211 052410 005737 002612
13212 052414 100011
13213 052416 005737 002064
13214 052422 001402
13215 052424 104401 070120
052424 104401
13216 052430 104401
052430 104401
13217 052432 000000
13218 052434 104401 002660
052434 104401
13219 052440 012037 052464
13220 052444 001412
13221 052446 005737 002430
13222 052452 001003
13223 052454 005737 002612
13224 052460 100004
13225 052462 104401
052462 104401

;*****
;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

$ERRTYP:SAVREG
TYPE $CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
TYPEIT , $CRLF
.DSABL CRF
CLR RO ;; PICKUP THE ITEM INDEX
BISB $ITEMB,RO
BNE 1$ ;; IF ITEM NUMBER IS ZERO, JUST
;; TYPE THE PC OF THE ERROR
TYPOCT ERRPC, <ERROR ADDRESS>
MOV ERRPC, -(SP) ;; SAVE ERRPC FOR TYPEOUT
;; ERROR ADDRESS
;; GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
BR 11$ ;; GET OUT
1$: CMPB #177,RO ;; POWER MONITOR CALL? ;R-C
BNE 100$ ;; BRANCH IF NOT ;R-C
MOV #PFECWS,RO ;; MOV ADDRESS OF PFE BIT ERROR TO RO ;R-C
BR 110$ ;; ;R-C
100$: DEC RO ;; ADJUST THE INDEX SO THAT IT WILL ;R-C
ASL RO ;; WORK FOR THE ERROR TABLE
ASL RO
ASL RO
ADD # $ERRTB,RO ;; FORM TABLE POINTER
110$: MOV (RO)+, 3$ ;; PICKUP "ERROR MESSAGE" POINTER ;R-C
BEQ 4$ ;; SKIP TYPEOUT IF NO POINTER
TST NOERROR ;; IS THIS REALLY AN ERROR?
BNE 12$ ;; YES - SKIP
TST HEADER ;; TYPE HEADER?
BPL 4$ ;; NO - SKIP
12$: TST FATAL$ ;; WAS IT A FATAL ERROR?
BEQ 2$ ;; NO - SKIP
TYPE MSG067 ;; FATAL
TYPEIT ,MSG067
.DSABL CRF
2$: TYPE ;; TYPE THE "ERROR MESSAGE"
TYPEIT
.DSABL CRF
3$: .WORD 0 ;; "ERROR MESSAGE" POINTER GOES HERE
TYPE $CRLF ;; "CARRIAGE RETURN" & "LINE FEED"
TYPEIT , $CRLF
.DSABL CRF
4$: MOV (RO)+, 5$ ;; PICKUP "DATA HEADER" POINTER
BEQ 6$ ;; SKIP TYPEOUT IF 0
TST NOERROR ;; IS THIS REALLY AN ERROR?
BNE 13$ ;; YES - SKIP
TST HEADER ;; TYPE HEADER?
BPL 6$ ;; NO - SKIP
13$: TYPE ;; TYPE THE "DATA HEADER"
TYPEIT
.DSABL CRF

```



```
13226 052464 000000      5$:      .WORD      0          ;; "DATA HEADER" POINTER GOES HERE
13227 052466          TYPE      $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
      052466 104401 002660      TYPEIT     , $CRLF
13228 052472 012001      6$:      .DSABL     CRF
13229 052474 001427      MOV       (R0)+,R1      ;; PICKUP "DATA TABLE" POINTER
13230 052476 012002      BEQ      10$          ;; BR IF NO DATA TO BE TYPED
      MOV       (R0)+,R2      ;; PICKUP "DATA FORMAT" POINTER
```

```

13233 052500 112203          7$:  MOVB  (R2)+,R3
13234 052502 006303          ASL   R3           ;MAKE IT A WORD ADDRESS
13235 052504 004773 052512    CALL  @8$(R3)
13236 052510 000412          BR    9$
13237 052512 052736          8$:  TAG70$
13238 052514 052746          TAG71$
13239 052516 052756          TAG72$
13240 052520 053026          TAG73$
13241 052522 053066          TAG74$
13242 052524 053100          TAG75$
13243 052526 053112          TAG76$
13244 052530 053156          TAG77$
13245 052532 053164          TAG78$
13246 052534 053244          TAG79$
13251 052536 062701 000002    9$:  ADD   #2,R1       ;UPDATE DATA TABLE POINTER
13252 052542 005711          TST   (R1)         ;;IS THERE ANOTHER NUMBER?
13253 052544 001403          BEQ   10$          ;;BR IF NO
13254 052546 052546 104401 066107    TYPE  MSG018       ;TYPE 2 SPACES
                                TYPEIT ,MSG018
                                .DSABL CRF
                                BR    7$           ;;LOOP
13255 052552 000752
13256
13257 052554 005737 002110    10$: TST   MUT        ;IS THERE A MEMORY UNDER TEST
13258 052560 001402          BEQ   11$          ;NO - SKIP
13259 052562 005237 002612    INC   HEADER       ;YES - BUMP HEADER FLAG
13260 052566 104416
13261 052570          11$: RESREG
                                IF #SW7 SET.IN @SWR AND DETFLAG IS FALSE AND NOERROR IS FALSE
                                BIT #SW7,@SWR
                                BEQ L516
                                TST DETFLAG
                                BNE L516
                                TST NOERROR
                                BNE L516
                                CALL  DETAIL
                                END ;OF IF #SW7
                                L516:::;
13262 052614 004737 053266
13263 052620
13264 052620          TYPE  MSG104       ;CONTROL Z
                                TYPEIT ,MSG104
                                .DSABL CRF
                                RETURN
                                .EVEN
13265 052624 000207
13266
13267 052626 052636 052672 052722 PFECWS: .WORD  PFECM,PFECDH,PFECDT,PFECDF ;R-C
                                052732
13268 052636 120 117 127 PFECM: .ASCIZ "POWER MONITOR BIT FOUND SET" ;R-C
                                052641 105 122 040
                                052644 115 117 116
                                052647 111 124 117
                                052652 122 040 102
                                052655 111 124 040
                                052660 106 117 125
                                052663 116 104 040
                                052666 123 105 124
                                052671 000
13269 052672 124 105 123 PFECDH: .ASCIZ "TESTNO ERR PC CPUERR" ;R-C
                                052675 124 116 117
                                052700 040 040 105
                                052703 122 122 040

```


13276
 13277
 13278
 13279 052736 017146 000000
 052736 017146 000000
 052742 104402

```

;*****
;*** OCTAL ***
;*****
TAG70$: TYPOCT @ (R1) ;:TYPE AN OCTAL NUMBER
        MOV @ (R1),-(SP) ;:SAVE @ (R1) FOR TYPEOUT
        TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
        .DSABL CRF
        RETURN
  
```

13280 052744 000207
 13281
 13282
 13283
 13284
 13285 052746 017146 000000
 052746 017146 000000
 052752 104405

```

;*****
;*** DECIMAL ***
;*****
TAG71$: TYPDEC @ (R1) ;:TYPE A DECIMAL NUMBER
        MOV @ (R1),-(SP) ;:SAVE @ (R1) FOR TYPEOUT
        TYPDS ;:GO TYPE--DECIMAL ASCII WITH SIGN
        .DSABL CRF
        RETURN
  
```

13286 052754 000207
 13287
 13288
 13289
 13290
 13291 052756 010146
 052756 010146
 052760 010546
 13292 052762 013701 002102
 13293 052766 070127 000004
 13294 052772 012737 177777 002372
 052772 012737 177777 002372
 13295 053000 104401 066060
 053000 104401 066060

```

;*****
;*** INTERLEAVE ***
;*****
TAG72$: PUSH R1,R5
;*****
        MOV R1,-(SP)
        MOV R5,-(SP)
        MOV BANK,R1
        MUL #4,R1
        SET NOTAB ;INDICATE NO TABLE TO BE PRINTED - NOW
        ;*****
        TYPE MSG014
        TYPEIT ,MSG014
        .DSABL CRF
        CALL TCFIG1
        CLR NOTAB
        POP R5,R1
        MOV (SP)+,R5
        MOV (SP)+,R1
        TYPE MSG014 ;1 SPACE
        TYPEIT ,MSG014
        .DSABL CRF
        RETURN
  
```

13296 053004 004737 033404
 13297 053010 005037 002372
 13298 053014 012605
 053014 012605
 053016 012601
 13299 053020 104401 066060
 053020 104401 066060

13300 053024 000207
 13301
 13302
 13303
 13304
 13305 053026 010146
 053026 010146
 053030 010546
 13306 053032 013701 002102
 13307 053036 070127 000004
 13308 053042 012737 177777 002372
 053042 012737 177777 002372
 13309 053050 004737 033600
 13310 053054 005037 002372
 13311 053060 012605
 053060 012605
 053062 012601
 13312 053064 000207

```

;*****
;*** CSR ***
;*****
TAG73$: PUSH R1,R5
;*****
        MOV R1,-(SP)
        MOV R5,-(SP)
        MOV BANK,R1
        MUL #4,R1
        SET NOTAB
        MOV #-1,NOTAB
        CALL TCFIG3
        CLR NOTAB
        POP R5,R1
        MOV (SP)+,R5
        MOV (SP)+,R1
        RETURN
  
```


13313
 13314
 13315
 13316
 13317 053066 013746 002300
 053066 053072 104403
 053074 002
 053075 001

```

;*****
;*** PATTERN ***
;*****
TAG74$: TYPOCS REALPAT,<TYPE (0-77)>,2,Z
        MOV     REALPAT,-(SP)      ;;SAVE REALPAT FOR TYPEOUT
                                           ;;TYPE (0-77)
                                           ;;GO TYPE--OCTAL ASCII
        TYPOS   2                   ;;TYPE 2 DIGIT(S)
        .BYTE  1                   ;;TYPE LEADING ZEROS
        .BYTE  1
        .DSABL CRF
        RETURN
  
```

13318 053076 000207
 13319
 13320
 13321
 13322
 13323 053100 013746 002102
 053100 053104 104403
 053106 003
 053107 000

```

;*****
;*** BANK ***
;*****
TAG75$: TYPOCS BANK,<TYPE (0-176)>,3
        MOV     BANK,-(SP)        ;;SAVE BANK FOR TYPEOUT
                                           ;;TYPE (0-176)
                                           ;;GO TYPE--OCTAL ASCII
        TYPOS   3                   ;;TYPE 3 DIGIT(S)
        .BYTE  0                   ;;SUPPRESS LEADING ZEROS
        .BYTE  0
        .DSABL CRF
        RETURN
  
```

13324 053110 000207

```

13326 ;*****
13327 ;*** MTYPE ***
13328 ;*****
13329 053112 TAG76$: PUSH R1,R5
      053112 010146 MOV R1,-(SP)
      053114 010546 MOV R5,-(SP)
13330 053116 013701 002102 MOV BANK,R1
13331 053122 070127 000004 MUL #4,R1
13332 053126 SET NOTAB
      053126 012737 177777 002372 MOV #-1,NOTAB
13333 053134 TYPE MSG019
      053134 104401 066112 TYPEIT ,MSG019
      .DSABL CRF
      CALL TCFIG2
      CLR NOTAB
      POP R5,R1
      MOV (SP)+,R5
      MOV (SP)+,R1
13334 053140 004737 033444
13335 053144 005037 002372
13336 053150
      053150 012605
      053152 012601
13337 053154 000207 RETURN
13338
13339 ;*****
13340 ;*** UNKNOWN DATA ***
13341 ;*****
13342 053156 TAG77$: TYPE MSG061
      053156 104401 067767 TYPEIT ,MSG061
      .DSABL CRF
      RETURN
13343 053162 000207
13344
13345 ;*****
13346 ;*** PHYSICAL ADDRESS ***
13347 ;*****
13348 053164 013737 002034 002040 TAG78$: MOV ADDRESS,PHYADD
13349 053172 162737 060000 002040 SUB #FIRST,PHYADD
13350 053200 013737 002102 002042 MOV BANK,PHYADD+2
13351 053206 006237 002042 ASR PHYADD+2
13352 053212 103003 BCC 1$
13353 053214 052737 100000 002040 BIS #BIT15,PHYADD
13354 053222 012746 002040 1$: MOV #PHYADD,-(SP) ;POINTER TO DOUBLE WORD ON STACK
13355 053226 004737 056526 CALL $DB20 ;CALL DOUBLE PRECISION CONVERSION ROUTINE
13356 053232 062706 000002 ADD #2,SP ;FIX STACK
13357 053236
      053236 104401 056634 TYPE $OCT8
      TYPEIT , $OCT8
      .DSABL CRF
      RETURN
13358 053242 000207
13359
13360 ;*****
13361 ;*** OCTAL BYTE ***
13362 ;*****
13363 053244 TAG79$: TYPE MSG018 ;2 SPACES
      053244 104401 066107 TYPEIT ,MSG018
      .DSABL CRF
      TYPOCS @ (R1), <TYPE BYTE>, 3, Z
      MOV @ (R1), -(SP) ;;SAVE @ (R1) FOR TYPEOUT
      ;;TYPE BYTE
      TYPOS ;;GO TYPE--OCTAL ASCII
      .BYTE 3 ;;TYPE 3 DIGIT(S)
      .BYTE 1 ;;TYPE LEADING ZEROS
13364 053250
      053250 017146 000000
      053254 104403
      053256 003
      053257 001

```

13365 053260
053260 104401 066060

13366 053264 000207

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

13410 053266

DETAIL: SUBTST <<SUBR DETAILED ERROR REPORT>>

;SUBTEST SUBR DETAILED ERROR REPORT

13411 053266 005237 002220
 13412 053272 022737 000003 002220
 13413 053300 101473
 13414 053302 022737 000002 002220
 13415 053310 001435
 13416 053312
 053312 013746 002612
 053316 013746 002110
 13417 053322
 053322 012737 177777 002612
 13418 053330 005037 002110
 13419 053334 010037 002200
 13420 053340 012700 002202
 13421 053344 010120
 13422 053346 010220
 13423 053350 010320
 13424 053352 010420
 13425 053354 010520
 13426 053356 013720 002024
 13427 053362 013720 002030
 13428 053366 013700 002200
 13429 053372
 053372 012737 177777 002430
 13430 053400 104013
 13431 053402 000423
 13432 053404
 053404 013746 002612
 053410 013746 002110
 13433 053414
 053414 012737 177777 002612
 13434 053422 005037 002110
 13435 053426
 053426 012737 177777 002430
 13436 053434 104031
 13437 053436 022737 000005 004010
 13438 053444 001002
 13439 053446 005037 177766
 13440 053452
 053452 012637 002110
 053456 012637 002612
 13441
 13442 053462 004737 053266
 13443 053466 000207

INC DETFLAG
 CMP #3,DETFLAG
 BLOS 4\$
 CMP #2,DETFLAG
 BEQ 2\$
 PUSH HEADER,MUT

 SET HEADER

 CLR MUT
 MOV R0,DETRO
 MOV #DETR1,R0
 MOV R1,(R0)+
 MOV R2,(R0)+
 MOV R3,(R0)+
 MOV R4,(R0)+
 MOV R5,(R0)+
 MOV ERRSP,(R0)+
 MOV ERRPSW,(R0)+
 MOV DETRO,R0
 SET NOERROR

 MOV #-1,NOERROR

 ERROR +13
 BR 1\$
 2\$: PUSH HEADER,MUT

 MOV HEADER,-(SP)
 MOV MUT,-(SP)

 SET HEADER

 CLR MUT
 SET NOERROR

 MOV #-1,NOERROR

 ERROR +31
 CMP #5,PROTYP ;IS THIS AN 11/83 ?
 BNE 1\$
 CLR CPUERR
 1\$: POP MUT,HEADER

 MOV (SP)+,MUT
 MOV (SP)+,HEADER

 ;WARNING RECURSIVE
 CALL DETAIL
 RETURN


```

13446
13447 053470 004737 055152      4:      ;SIMULATE CONTROL "T"
13448                                CALL      CONTT                ;DISPLAY "DISPLAY" INFO
13449                                ;TYPE CONTENTS OF ALL CSR'S
13450 053474                                PUSH     CSR,CSRNO,R1
                                MOV CSR,-(SP)
                                MOV CSRNO,-(SP)
                                MOV R1,-(SP)
                                053474 013746 002150
                                053500 013746 002152
                                053504 010146
13451 053506                                TYPE     MSG058
                                053506 104401 067745      TYPEIT   ,MSG058
                                .DSABL CRF
                                TYPE     $CRLF
                                TYPEIT   ,$CRLF
                                .DSABL CRF
13452 053512                                MOV     TOTCSRS,R1
                                053512 104401 002660      BEGIN  DUMPCSRLOOP
                                FOR CSRNO := #0 TO #36 BY #2
                                B103:;;;;;;;;
                                CLR CSRNO
                                B104:;;;;;;;;
                                ASL      R1
13456 053526 006301                                ON.ERROR
                                READCSR
13457 053530                                TYOCT   CSR
                                053530 103006      MOV     CSR,-(SP)      ;;SAVE CSR FOR TYPEOUT
                                053532 104426      TYOCT   CSR      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
13458 053532 104426      .DSABL CRF
13459 053534 013746 002150      .DSABL CRF
                                053534 104402      TYPE     MSG018      ;2 SPACES
                                053540 104402      TYPEIT   ,MSG018
                                .DSABL CRF
13460 053542 104401 066107      END ;OF ON.ERROR
                                L517:;;;;;;;;
13461 053546                                IF R1 EQ #0 THEN LEAVE DUMPCSRLOOP
13462 053546 005701                                TST R1
                                053546 001407      BEQ E103
13463 053552                                END ;OF FOR CSRNO
                                053552 062737 000002 002152      ADD #2,CSRNO
                                053560 023727 002152 000036      CMP CSRNO,#36
                                053566 003757      BLE B104
13464 053570                                E104:;;;;;;;;
                                053570                                E103:;;;;;;;;
13465 053570                                POP     R1,CSRNO,CSR
                                053570 012601      MOV (SP)+,R1
                                053572 012637 002152      MOV (SP)+,CSRNO
                                053576 012637 002150      MOV (SP)+,CSR
13466
13467                                ;TYPE STACKS
13468 053602                                PUSH     R0,R1
                                053602 010046
                                053604 010146
13469 053606                                TYPE     MSG088      ;KERNEL STACK
                                053606 104401 070336      TYPEIT   ,MSG088
                                .DSABL CRF
13470 053612 013701 002574      MOV     KSTACK,R1
13471 053616 162701 000002      SUB     #2,R1

```

```

13472 053622          FOR      RO := SP TO R1 BY #2
      053622 010600
      053624
13473 053624          TYPE    $CRLF
      053624 104401 002660  TYPEIT  , $CRLF
      .DSABL  CRF
      TYPOCT          RO
13474 053630          MOV      RO, -(SP)      ;;SAVE RO FOR TYPEOUT
      053630 010046          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      053632 104402          .DSABL  CRF
13475 053634          TYPE    MSG018          ;2 SPACES
      053634 104401 066107  TYPEIT  ,MSG018
      .DSABL  CRF
13476 053640          TYPOCT          (RO)
      053640 011046          MOV      (RO), -(SP)      ;;SAVE (RO) FOR TYPEOUT
      053642 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL  CRF
13477 053644          END ;OF FOR RO
      053644 062700 000002          ADD #2,RO
      053650 020001          CMP RO,R1
      053652 003764          BLE B105
      053654          E105:;;;;;

13478          ;SET PREVIOUS MODE TO SUPERVISOR
13479 053654 005737 002456  TST     NOSUPER
13480 053660 001036          BNE     DET1
13481 053662 042737 030000 177776  BIC     #BIT13!BIT12,PSW
13482 053670 052737 010000 177776  BIS     #BIT12,PSW
13483 053676 006506          MFPI   SSP
13484 053700          POP     R1,RO
      053700 012601          MOV (SP)+,R1
      053702 012600          MOV (SP)+,RO
13485 053704          TYPE    MSG089          ;SUPERVISOR STACK
      053704 104401 070354  TYPEIT  ,MSG089
      .DSABL  CRF
13486 053710          IF RO LT #SUPSTK
      053710 020027 000740          CMP RO,#SUPSTK
      053714 002016          BGE L520
13487 053716          FOR RO := RO TO #SUPSTK-2 BY #2
      053716          B106:;;;;;
13488 053716          TYPE    $CRLF
      053716 104401 002660  TYPEIT  , $CRLF
      .DSABL  CRF
      TYPOCT          RO
13489 053722          MOV      RO, -(SP)      ;;SAVE RO FOR TYPEOUT
      053722 010046          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      053724 104402          .DSABL  CRF
13490 053726          TYPE    MSG018          ;2 SPACES
      053726 104401 066107  TYPEIT  ,MSG018
      .DSABL  CRF
13491 053732          TYPOCT          (RO)
      053732 011046          MOV      (RO), -(SP)      ;;SAVE (RO) FOR TYPEOUT
      053734 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
      .DSABL  CRF
13492 053736          END ;OF FOR RO
      053736 062700 000002          ADD #2,RO
      053742 020027 000736          CMP RO,#SUPSTK-2
      053746 003763          BLE B106

```

```

13493 053750 000402 ELSE
13494 053752 104401 070412 TYPE MSG091 ;IS EMPTY
TYPEIT ,MSG091
.DSABL CRF
END ;OF IF R0
13495 053756 ;SET PREVIOUS MODE TO USER
13497 053756 052737 030000 177776 DET1: BIS #BIT13!BIT12,PSW
13498 053764 006506 MFPI USP
13499 053766 012600 POP R0
13500 053770 104401 070376 TYPE MSG090 ;USER STACK
TYPEIT ,MSG090
.DSABL CRF
IF R0 LT #USESTK
13501 053774 020027 000700
13502 054002 002016 FOR R0 := R0 TO #USESTK-2 BY #2
13503 054002 104401 002660 TYPE $CRLF
TYPEIT , $CRLF
.DSABL CRF
TYPOCT R0
MOV R0, -(SP) ;;SAVE R0 FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
13504 054006 010046 104402 TYPE MSG018 ;2 SPACES
TYPEIT ,MSG018
.DSABL CRF
TYPOCT (R0)
MOV (R0), -(SP) ;;SAVE (R0) FOR TYPEOUT
TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
.DSABL CRF
13505 054012 104401 066107 END ;OF FOR R0
13506 054016 011046 104402
13507 054022 062700 000002 ADD #2,R0
054026 020027 000676 CMP R0,#USESTK-2
054032 003763 BLE B107
054034
13508 054034 000402 ELSE
054036
13509 054036 104401 070412 TYPE MSG091 ;IS EMPTY
TYPEIT ,MSG091
.DSABL CRF
END ;OF IF R0
13510 054042
13511 054042 104401 002660 TYPE $CRLF
TYPEIT , $CRLF
.DSABL CRF
13512 054046 005037 002220 CLR DETFLAG
13513 054052 012600 POP R0
13514 054054 000207 RETURN

```

```

E106:;;;;;;;;
BR L521
L520:;;;;;;;;
L521:;;;;;;;;
MOV (SP)+,R0
CMP R0,#USESTK
BGE L522
B107:;;;;;;;;
E107:;;;;;;;;
BR L523
L522:;;;;;;;;
L523:;;;;;;;;
MOV (SP)+,R0

```



```

13552          .SBTTL  ROUTINE BINARY TO OCTAL (ASCII) AND TYPE
13553
13554          ;*****
13555          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
13556          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
13557          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
13558          ;*CALL:
13559          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
13560          ;*      TYPOS          ;;CALL FOR TYPEOUT
13561          ;*      .BYTE  N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
13562          ;*      .BYTE  M              ;;M=1 OR 0
13563          ;*                                  ;;1=TYPE LEADING ZEROS
13564          ;*                                  ;;0=SUPPRESS LEADING ZEROS
13565          ;*
13566          ;*$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
13567          ;*$TYPOS OR $TYPOC
13568          ;*CALL:
13569          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
13570          ;*      TYPON          ;;CALL FOR TYPEOUT
13571          ;*
13572          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
13573          ;*CALL:
13574          ;*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
13575          ;*      TYPOC          ;;CALL FOR TYPEOUT
13576
13577 054056 017646 000000          $TYPOS: MOV      @(SP),-(SP)          ;;PICKUP THE MODE
13578 054062 116637 000001 054301  MOVB     1(SP), $OFILL          ;;LOAD ZERO FILL SWITCH
13579 054070 112637 054303          MOVB     (SP)+, $OMODE+1          ;;NUMBER OF DIGITS TO TYPE
13580 054074 062716 000002          ADD      #2,(SP)              ;;ADJUST RETURN ADDRESS
13581 054100 000406          BR      $TYPON
13582 054102 112737 000001 054301  $TYPOC: MOVB     #1, $OFILL          ;;SET THE ZERO FILL SWITCH
13583 054110 112737 000006 054303  MOVB     #6, $OMODE+1          ;;SET FOR SIX(6) DIGITS
13584 054116 112737 000005 054300  $TYPON: MOVB     #5, $OCNT          ;;SET THE ITERATION COUNT
13585 054124 010346          MOV      R3,-(SP)            ;;SAVE R3
13586 054126 010446          MOV      R4,-(SP)            ;;SAVE R4
13587 054130 010546          MOV      R5,-(SP)            ;;SAVE R5
13588 054132 113704 054303          MOVB     $OMODE+1,R4          ;;GET THE NUMBER OF DIGITS TO TYPE
13589 054136 005404          NEG      R4
13590 054140 062704 000006          ADD      #6,R4              ;;SUBTRACT IT FOR MAX. ALLOWED
13591 054144 110437 054302          MOVB     R4, $OMODE          ;;SAVE IT FOR USE
13592 054150 113704 054301          MOVB     $OFILL,R4          ;;GET THE ZERO FILL SWITCH
13593 054154 016605 000012          MOV      12(SP),R5          ;;PICKUP THE INPUT NUMBER
13594 054160 005003          CLR      R3                  ;;CLEAR THE OUTPUT WORD
13595 054162 006105          1$:    ROL      R5              ;;ROTATE MSB INTO "C"
13596 054164 000404          BR      3$                  ;;GO DO MSB
13597 054166 006105          2$:    ROL      R5              ;;FORM THIS DIGIT
13598 054170 006105          ROL      R5
13599 054172 006105          ROL      R5
13600 054174 010503          MOV      R5,R3
13601 054176 006103          3$:    ROL      R3              ;;GET LSB OF THIS DIGIT
13602 054200 105337 054302          DECB     $OMODE              ;;TYPE THIS DIGIT?
13603 054204 100016          BPL      6$                  ;;BR IF NO
13604 054206 042703 177770          BIC      #177770,R3          ;;GET RID OF JUNK
13605 054212 001002          BNE      4$                  ;;TEST FOR 0
13606 054214 005704          TST      R4                  ;;SUPPRESS THIS 0?
13607 054216 001403          BEQ      5$                  ;;BR IF YES
13608 054220 005204          4$:    INC      R4              ;;DON'T SUPPRESS ANYMORE 0'S

```



```

13630          .SBTTL  ROUTINE CONVERT BINARY TO DECIMAL AND TYPE
13631          ;*****
13632          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
13633          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
13634          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
13635          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
13636          ;*REPLACED WITH SPACES.
13637          ;*CALL:
13638          ;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
13639          ;*      TYPDS   R0,R1,R2,R3,R5 ;;GO TO THE ROUTINE
13640          $TYPDS: PUSH
13641          054304 010046          MOV      #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
13642          054306 010146          MOV      20(SP),R5      ;;GET THE INPUT NUMBER
13643          054310 010246          BPL     1$              ;;BR IF INPUT IS POS.
13644          054312 010346          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
13645          054314 010546          MOVB   #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
13646          054316 012746 020200 1$: CLR     R0              ;;ZERO THE CONSTANTS INDEX
13647          054322 016605 000020  MOV     #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
13648          054326 100004          MOVB   #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
13649          054330 005405          CLR     R2              ;;CLEAR THE BCD NUMBER
13650          054332 112766 000055 000001 2$: MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
13651          054334 016001 054510  SUB     R1,R5          ;;FORM THIS BCD DIGIT
13652          054336 002402          BLT     4$              ;;BR IF DONE
13653          054338 005202          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
13654          054340 000774          BR      3$              ;;BR IF DONE
13655          054342 012703 054520 4$: ADD     R1,R5          ;;ADD BACK THE CONSTANT
13656          054344 005000          TST     R2              ;;CHECK IF BCD DIGIT=0
13657          054346 112723 000040  BNE     5$              ;;FALL THROUGH IF 0
13658          054348 005002          TSTB   (SP)            ;;STILL DOING LEADING 0'S?
13659          054350 016001 054510  BMI     7$              ;;BR IF YES
13660          054352 112766 000055 000001 5$: ASLB   (SP)            ;;MSD?
13661          054354 016001 054510  BCC     6$              ;;BR IF NO
13662          054356 005000          MOVB   1(SP),-1(R3)    ;;YES--SET THE SIGN
13663          054358 005000          BIS     #'0,R2         ;;MAKE THE BCD DIGIT ASCII
13664          054360 005000          BIS     #' ,R2         ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
13665          054362 110223          MOVB   R2,(R3)+       ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
13666          054364 005720          TST    (R0)+          ;;JUST INCREMENTING
13667          054366 020027 000010  CMP     R0,#10        ;;CHECK THE TABLE INDEX
13668          054368 002746          BLT     2$              ;;GO DO THE NEXT DIGIT
13669          054370 003002          BGT     8$              ;;GO TO EXIT
13670          054372 010502          MOV     R5,R2         ;;GET THE LSD
13671          054374 000764          BR      6$              ;;GO CHANGE TO ASCII
13672          054376 105726          TSTB   (SP)+          ;;WAS THE LSD THE FIRST NON-ZERO?
13673          054378 100003          BPL     9$              ;;BR IF NO
13674          054380 116663 177777 177776 9$: MOVB   -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
13675          054382 105013          CLRB   (R3)           ;;SET THE TERMINATOR
13676          054460 012605          POP     R5,R3,R2,R1,R0
13677          054462 012603          MOV     (SP)+,R5
13678          054464 012602          MOV     (SP)+,R3
13679          054466 012601          MOV     (SP)+,R2
13680          054468 012600          MOV     (SP)+,R1
13681          054470 012600          MOV     (SP)+,R0

```

Address	Hex	Dec	Hex	Hex	Type	DBLK	Comments
13677	054472		054520		TYPE	\$DBLK	::NOW TYPE THE NUMBER
	054472	104401			TYPEIT	,\$DBLK	
					.DSABL	CRF	
13678	054476	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
13679	054504	012616			MOV	(SP)+,(SP)	
13680	054506	000002			RTI		::RETURN TO USER
13681	054510	023420			\$DTBL:	10000.	
13682	054512	001750				1000.	
13683	054514	000144				100.	
13684	054516	000012				10.	
13685	054520	000000	000000	000000	\$DBLK:	.WORD 0,0,0,0	
	054526	000000					


```

13687
13688
13689
13690
13691
13692
13693
13694 054530
13695 054530 005737 047246
13696 054534 001406
13697 054536 013746 047246
13698 054542 005037 047246
13699 054546 000137 054570
13700 054552 105777 126064
13701 054556 100136
13702 054560 117746 126060
13703 054564 042716 177600
13704 054570 022716 000006
13705 054574 001002
13706 054576 004737 041054
13707 054602 022716 000024
13708 054606 001002
13709 054610 004737 055152
13710 054614 022716 000003
13711 054620 001454
13712 054622 022716 000023
13713 054626 001002
13714 054630 004737 055226
13715 054634 022716 000013
13716 054640 001005
13717 054642
      054642 104401 055144

13718 054646 013706 002146
13719 054652 000207
13720 054654 022737 000176 002636 6$:
13721 054662 001075
13722 054664 022716 000007
13723 054670 001072
13724 054672 005737 002062
13725 054676 001067
13726 054700
      054700 104401 055750

13727 054704
      054704 104401 055755

13728 054710
      054710 017746 125722
      054714 104402

13729 054716
      054716 104401 055766

13730 054722 005046
13731 054724 005046
13732 054726 105777 125710

```

```

.SBTTL ROUTINE TTY INPUT
;*****
;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
;*WHEN OPERATING IN TTY FLAG MODE.
.ENABLE LSB
$CKSWR:
TST      XOCHAR      ;;SOMETHING THERE?
BEQ      NOCH        ;; GO ON IF NOT
MOV      XOCHAR, -(SP) ;; USE IT
CLR      XOCHAR
JMP      CONTS1
NOCH:    TSTB      @TKS      ;;CHAR THERE?
BPL      SWREND      ;;IF NO, DON'T WAIT AROUND
MOVB     @TKB, -(SP)  ;;SAVE THE CHAR
BIC      #C177, (SP)  ;;STRIP-OFF THE ASCII
CONTS1:  CMP      #6, (SP)  ;;IS IT CONTROL F?
BNE      1$          ;NO SKIP
CALL     FIELDSERVICE
1$:      CMP      #24, (SP)  ;;IS IT CONTROL T?
BNE      16$        ;NO - SKIP
CALL     CONTT       ;YES - CALL CONTROL T ROUTINE
16$:     CMP      #3, (SP)  ;;IS IT CONTROL C?
BEQ      5$          ;YES EXIT *****NOTE***** STACK IS SCREWED UP!
2$:      CMP      #23, (SP) ;;IS IT CONTROL S?
BNE      17$        ;NO - SKIP
CALL     CONTS       ;YES - CALL CONTROL S ROUTINE
17$:     CMP      #13, (SP) ;;IS IT CONTROL K?
BNE      6$          ;NO - SKIP
TYPE     $CNTLK      ;TYPE A +K
TYPEIT   , $CNTLK
.DSABL   CRF
MOV      CTLKVEC, SP  ;RESET KSP TO AFTER PATTERN EXEC ROUTINE
RETURN   ;RETURN TO PATTERN EXEC ROUTINE
6$:      CMP      #SWREG, SWR  ;;IS THE SOFT-SWR SELECTED?
BNE      CKEND       ;;BRANCH IF NO
CMP      #7, (SP)    ;;IS IT A CONTROL G?
BNE      CKEND       ;;NO, RETURN TO USER
TST      $AUTO       ;ARE WE RUNNING IN AUTO-MODE?
BNE      CKEND       ;BRANCH IF YES
TYPE     $CNTLG      ;;ECHO THE CONTROL-G (+G)
TYPEIT   , $CNTLG
.DSABL   CRF
$GTSWR:  TYPE     $MSWR      ;;TYPE CURRENT CONTENTS
TYPEIT   , $MSWR
.DSABL   CRF
TYPOCT   @SWR        ;;OF THE SWR
MOV      @SWR, -(SP)  ;;SAVE @SWR FOR TYPEOUT
TYPOC    .DSABL     CRF    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPE     $MNEW
TYPEIT   , $MNEW
.DSABL   CRF
3$:      CLR      -(SP)      ;;CLEAR COUNTER
CLR      -(SP)        ;;THE NEW SWR
4$:      TSTB      @TKS      ;;CHAR THERE?

```



```

13733 054732 100375          BPL      4$          ;;IF NOT TRY AGAIN
13734 054734 117746 125704  MOVB    @TKB,-(SP)  ;;PICK UP CHAR
13735 054740 042716 177600  BIC     #C177,(SP)  ;;MAKE IT 7-BIT ASCII
13736 054744 021627 000003  CMP     (SP),#3     ;;IS IT A CONTROL-C?
13737 054750 001006          BNE     7$          ;;BRANCH IF NOT
13738 054752          5$:  TYPE    $CNTLC      ;;YES, ECHO CONTROL-C (↑C)
      054752 104401 055736  TYPEIT  , $CNTLC
      .DSABL CRF
13739 054756 062706 000006  ADD     #6,SP       ;;CLEAN UP STACK
13740 054762 000137 040534  JMP     QUIT        ;;CONTROL-C HALT
13741 054766 021627 000025  7$:  CMP     (SP),#25  ;;IS IT A CONTROL-U?
13742 054772 001005          BNE     9$          ;;BRANCH IF NOT
13743 054774          TYPE    $CNTLU      ;;YES, ECHO CONTROL-U (↑U)
      054774 104401 055743  TYPEIT  , $CNTLU
      .DSABL CRF
13744 055000 062706 000006  8$:  ADD     #6,SP       ;;IGNORE PREVIOUS INPUT
13745 055004 000746          BR      3$          ;;LET'S TRY IT AGAIN
13746 055006 021627 000015  9$:  CMP     (SP),#15  ;;IS IT A <CR>?
13747 055012 001024          BNE     13$         ;;BRANCH IF NO
13748 055014 005766 000004  TST     4(SP)       ;;YES, IS IT THE FIRST CHAR?
13749 055020 001403          BEQ     10$        ;;BRANCH IF YES
13750 055022 016677 000002 125606 MOV     2(SP),@SWR  ;;SAVE NEW SWR
13751 055030          10$: IF SWRFLG IS TRUE
      055030 005737 002566          TST SWRFLG
      055034 001403          BEQ L524
13752 055036 062706 000006  ADD     #6,SP       ;;CLEAR UP STACK
13753 055042          ELSE
      055042 000402          BR L525
      055044          L524:;;;;;
13754 055044 062706 000010  ADD     #10,SP      ;;CLEAR UP STACK
13755 055050          END
      055050          L525:;;;;;
13756 055050          TYPE    $CRLF      ;;ECHO <CR> AND <LF>
      055050 104401 002660  TYPEIT  , $CRLF
      .DSABL CRF
13757 055054 000002  SWREND: RTI        ;;RETURN

```

13758	055056	062706	000002	CKEND:	ADD	#2,SP		;FIX STACK
13759	055062	000002			RTI			;RETURN
13760	055064	004737	047250	13\$:	CALL	\$TYPEC		::ECHO CHAR
13761	055070	021627	000060		CMP	(SP),#60		::CHAR < 0?
13762	055074	002420			BLT	15\$::BRANCH IF YES
13763	055076	021627	000067		CMP	(SP),#67		::CHAR > 7?
13764	055102	003015			BGT	15\$::BRANCH IF YES
13765	055104	042726	000060		BIC	#60,(SP)+		::STRIP-OFF ASCII
13766	055110	005766	000002		TST	2(SP)		::IS THIS THE FIRST CHAR
13767	055114	001403			BEQ	14\$::BRANCH IF YES
13768	055116	006316			ASL	(SP)		::NO, SHIFT PRESENT
13769	055120	006316			ASL	(SP)		:: CHAR OVER TO MAKE
13770	055122	006316			ASL	(SP)		:: ROOM FOR NEW ONE.
13771	055124	005266	000002	14\$:	INC	2(SP)		::KEEP COUNT OF CHAR
13772	055130	056616	177776		BIS	-2(SP),(SP)		::SET IN NEW CHAR
13773	055134	000674			BR	4\$::GET THE NEXT ONE
13774	055136			15\$:	TYPE	\$QUES		::TYPE ?<CR><LF>
	055136	104401	002657		TYPEIT	, \$QUES		
					.DSABL	CRF		
13775	055142	000716			BR	8\$::SIMULATE CONTROL-U
13776	055144	136	113	015	\$CNTLK:	.ASCIZ	/↑K/<15><12>	;CONTROL K ASCII STRING
	055147	012	000					
13777					.EVEN			
13778					.DSABL	LSB		

```

13781 055152          CONTT: SUBTST <<CONTROL T>>
;*****
;*SUBTEST          CONTROL T
;*****
13782 055152          PUSH      R0
055152 010046          MOV      RO,-(SP)
13783 055154          TYPE      $CRLF
055154 104401 002660  TYPEIT   , $CRLF
                                .DSABL   CRF
13793 055160          IF RLFLAG IS TRUE
055160 005737 002126          TST     RLFLAG
055164 001402          BEQ     L526
13794 055166          TYPE      MSG092          ;RELOCATED
055166 104401 070424  TYPEIT   ,MSG092
                                .DSABL   CRF
13795 055172          END ;OF IF RLFLAG
055172
13796 055172          TYPE      MSG093          ;BANK=          L526:;;;;;
055172 104401 070440  TYPEIT   ,MSG093
                                .DSABL   CRF
13797 055176          TYPOCS   BANK,,3          ;TYPE 3 DIGITS
055176 013746 002102  MOV      BANK,-(SP)          ;;SAVE BANK FOR TYPEOUT
055202 104403          TYPOS    3          ;;GO TYPE--OCTAL ASCII
055204 003          .BYTE    3          ;;TYPE 3 DIGIT(S)
055205 000          .BYTE    0          ;;SUPPRESS LEADING ZEROS
                                .DSABL   CRF
13798 055206          TYPE      MSG095          ;PAT=
055206 104401 070446  TYPEIT   ,MSG095
                                .DSABL   CRF
13799 055212          TYPOCS   REALPAT,,2          ;TYPE 2 DIGITS
055212 013746 002300  MOV      REALPAT,-(SP)          ;;SAVE REALPAT FOR TYPEOUT
055216 104403          TYPOS    2          ;;GO TYPE--OCTAL ASCII
055220 002          .BYTE    2          ;;TYPE 2 DIGIT(S)
055221 000          .BYTE    0          ;;SUPPRESS LEADING ZEROS
                                .DSABL   CRF
13803 055222          POP      R0
055222 012600          MOV     (SP)+,R0
13804 055224          RETURN
000207
13805

```

13806 055226

CONTS: SUBTST <<CONTROL S & CONTROL Q>>
;*****
;#SUBTEST CONTROL S & CONTROL Q
;*****

13807 055226

POP R0 ;GET RID OF RETURN ADDRESS FROM STACK
MOV (SP)+,R0

13808 055230 012600 105777 125406

CONTS2: TSTB @TKS ;WAIT FOR CHARACTER

13809 055234 100375

BPL CONTS2

13810 055236 117716 125402

MOVB @TKB,(SP) ;REPLACE OVER OLD CHARACTER ON STACK

13811 055242 042716 177600

BIC #C177,(SP) ;STRIP ALL BUT ASCII

13812 055246

IF (SP) EQ #21 ;IF IT IS A CONTROL Q

055246 021627 000021

CMP (SP),#21

055252 001003

BNE L527

13813 055254 000137 054570

JMP CONTS1

13814 055260

ELSE

BR L530

055260 000401

L527:::::

13815 055262 000762

BR CONTS2

13816 055264

END ;OF IF (SP)

L530:::::

055264


```

13818 ;*****
13819 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
13820 ;*CALL:
13821 ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
13822 ;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
13823 ;*                   ;;WITH PARITY BIT STRIPPED OFF
13824 ;
13825
13826 055264 011646 $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC
13827 055266 016666 000004 000002 MOV      4(SP),2(SP)      ;;SAVE THE PS
13828 055274 105777 125342 1$:      TSTB     @TKS          ;;WAIT FOR
13829 055300 100375 BPL      1$           ;;A CHARACTER
13830 055302 117766 125336 000004 MOVB     @TKB,4(SP)      ;;READ THE TTY
13831 055310 042766 177600 000004 BIC      #C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
13832 055316 026627 000004 000023 CMP      4(SP),#23     ;;IS IT A CONTROL-S?
13833 055324 001013 BNE      3$           ;;BRANCH IF NO
13834 055326 105777 125310 2$:      TSTB     @TKS          ;;WAIT FOR A CHARACTER
13835 055332 100375 BPL      2$           ;;LOOP UNTIL ITS THERE
13836 055334 117746 125304 MOVB     @TKB,-(SP)     ;;GET CHARACTER
13837 055340 042716 177600 BIC      #C177,(SP)   ;;MAKE IT 7-BIT ASCII
13838 055344 022627 000021 CMP      (SP)+,#21     ;;IS IT A CONTROL-Q?
13839 055350 001366 BNE      2$           ;;IF NOT DISCARD IT
13840 055352 000750 BR       1$           ;;YES, RESUME
13841 055354 026627 000004 000021 3$:      CMP      4(SP),#21     ;;IS IT A RANDOM CONTROL-Q?      ;R-C
13842 055362 001744 BEQ      1$           ;;BRANCH BACK IF SO          ;R-C
13843 055364 026627 000004 000140 CMP      4(SP),#140    ;;IS IT UPPER CASE?
13844 055372 002407 BLT      4$           ;;BRANCH IF YES
13845 055374 026627 000004 000175 CMP      4(SP),#175    ;;IS IT A SPECIAL CHAR?
13846 055402 003003 BGT      4$           ;;BRANCH IF YES
13847 055404 042766 000040 000004 BIC      #40,4(SP)    ;;MAKE IT UPPER CASE
13848 055412 000002 4$:      RTI          ;;GO BACK TO USER
13849 ;*****
13850 ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
13851 ;*CALL:
13852 ;*      RDLIN         ;;INPUT A STRING FROM THE TTY
13853 ;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
13854 ;*                   ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
13855 055414 010346 $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
13856 055416 005046 CLR      -(SP)          ;;CLEAR THE RUBOUT KEY
13857 055420 012703 055712 1$:      MOV      #TTYIN,R3      ;;GET ADDRESS
13858 055424 022703 055736 2$:      CMP      #TTYIN+20.,R3  ;;BUFFER FULL?
13859 055430 101477 BLOS     8$           ;;BR IF YES
13860 055432 104411 RDCHR    8$           ;;GO READ ONE CHARACTER FROM THE TTY
13861 055434 112613 MOVB     (SP)+,(R3)     ;;GET CHARACTER
13862 055436 122713 000003 CMPB     #3,(R3)        ;;IS IT A CONTROL-C?
13863 055442 001016 BNE      3$           ;;BRANCH IF NO
13864 055444 104401 055736 TYPE     $CNTLC        ;;TYPE A CONTROL-C (+C)
13865 055450 005726 TYPEIT  , $CNTLC
13866 055452 012603 .DSABL  CRF
13867 055454 032777 000400 125154 TST     (SP)+         ;;CLEAN RUBOUT KEY OFF OF THE STACK
13868 055462 001404 MOV      (SP)+,R3      ;;RESTORE R3
13869 055464 005037 002420 BIT      #BIT8,@SWR    ;;IS THERE A HALT FLAG SET IN THE SWR?
13870 055470 000137 040604 BEQ     11$          ;;BRANCH IF NOT TO HALT ROUTINE
13871 055474 000137 040534 CLR     STOPOK        ;;GET READY TO HALT PROGRAM
13872 055500 122713 000177 11$:      JMP     EXIT          ;;GO HALT PROGRAM
3$:      CMPB     #177,(R3)  ;;GOTO CONTROL-C HALT
3$:      CMPB     #177,(R3)  ;;IS IT A RUBOUT

```

13873	055504	001022			BNE	5\$::BR IF NO
13874	055506	005716			TST	(SP)	::IS THIS THE FIRST RUBOUT?
13875	055510	001007			BNE	4\$::BR IF NO
13876	055512	112737	000134	055710	MOVB	#'\,10\$::TYPE A BACK SLASH
13877	055520				TYPE	10\$	
	055520	104401	055710		TYPEIT	,10\$	
					.DSABL	CRF	
13878	055524	012716	177777		MOV	#-1,(SP)	::SET THE RUBOUT KEY
13879	055530	005303		4\$:	DEC	R3	::BACKUP BY ONE
13880	055532	020327	055712		CMP	R3,#\$TTYIN	::STACK EMPTY?
13881	055536	103434			BLO	8\$::BR IF YES
13882	055540	111337	055710		MOVB	(R3),10\$::SETUP TO TYPEOUT THE DELETED CHAR.
13883	055544				TYPE	10\$::GO TYPE
	055544	104401	055710		TYPEIT	,10\$	
					.DSABL	CRF	
13884	055550	000725			BR	2\$::GO READ ANOTHER CHAR.
13885	055552	005716		5\$:	TST	(SP)	::RUBOUT KEY SET?
13886	055554	001406			BEG	6\$::BR IF NO
13887	055556	112737	000134	055710	MOVB	#'\,10\$::TYPE A BACK SLASH
13888	055564				TYPE	10\$	
	055564	104401	055710		TYPEIT	,10\$	
					.DSABL	CRF	
13889	055570	005016			CLR	(SP)	::CLEAR THE RUBOUT KEY
13890	055572	122713	000025	6\$:	CMPB	#25,(R3)	::IS CHARACTER A CTRL U?
13891	055576	001003			BNE	7\$::BR IF NO
13892	055600				TYPE	,\$CNTLU	::TYPE A CONTROL "U"
	055600	104401	055743		TYPEIT	,,\$CNTLU	
					.DSABL	CRF	
13893	055604	000705			BR	1\$::GO START OVER
13894	055606	122713	000022	7\$:	CMPB	#22,(R3)	::IS CHARACTER A "r"?
13895	055612	001011			BNE	9\$::BRANCH IF NO
13896	055614	105013			CLRB	(R3)	::CLEAR THE CHARACTER
13897	055616				TYPE	,\$CRLF	::TYPE A "CR" & "LF"
	055616	104401	002660		TYPEIT	,,\$CRLF	
					.DSABL	CRF	
13898	055622				TYPE	,\$TTYIN	::TYPE THE INPUT STRING
	055622	104401	055712		TYPEIT	,,\$TTYIN	
					.DSABL	CRF	
13899	055626	000676			BR	2\$::GO PICKUP ANOTHER CHACTER
13900	055630			8\$:	TYPE	,\$QUES	::TYPE A '?'
	055630	104401	002657		TYPEIT	,,\$QUES	
					.DSABL	CRF	
13901	055634	000671			BR	1\$::CLEAR THE BUFFER AND LOOP
13902	055636	111337	055710	9\$:	MOVB	(R3),10\$::ECHO THE CHARACTER
13903	055642				TYPE	10\$	
	055642	104401	055710		TYPEIT	,10\$	
					.DSABL	CRF	
13904	055646	122723	000015		CMPB	#15,(R3)+	::CHECK FOR RETURN
13905	055652	001264			BNE	2\$::LOOP IF NOT RETURN
13906	055654	105063	177777		CLRB	-1(R3)	::CLEAR RETURN (THE 15)
13907	055660				TYPE	,\$LF	::TYPE A LINE FEED
	055660	104401	002661		TYPEIT	,,\$LF	
					.DSABL	CRF	
13908	055664	005726			TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK
13909	055666	012603			MOV	(SP)+,R3	::RESTORE R3
13910	055670	011646			MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE
13911	055672	016666	000004	000002	MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT

```

13912 055700 012766 055712 000004      MOV      #$TTYIN,4(SP)
13913 055706 000002                      RTI
13914 055710      000                    10$: .BYTE 0      ;;RETURN
13915 055711      000                    .BYTE 0      ;;STORAGE FOR ASCII CHAR. TO TYPE
13916 055712 000024      $TTYIN: .REPT 20.  ;;TERMINATOR
13919 055736      136      103      015 $CNTLC: .ASCIZ /+C/<15><12>  ;;RESERVE SIZE BYTES FOR TTY INPUT
      055741      012      000                    ;;CONTROL "C"
13920 055743      136      125      015 $CNTLU: .ASCIZ /+U/<15><12>  ;;CONTROL "U"
      055746      012      000
13921 055750      136      107      015 $CNTLG: .ASCIZ /+G/<15><12>  ;;CONTROL "G"
      055753      012      000
13922 055755      015      012      123 $MSWR: .ASCIZ <15><12>/SWR = /
      055760      127      122      040
      055763      075      040      000
13923 055766      040      040      116 $MNEW: .ASCIZ / NEW = /
      055771      105      127      040
      055774      075      040      000
13924                      .EVEN

```



```

13926 .SBTTL ROUTINE READ AN OCTAL NUMBER FROM THE TTY
13927 ;*****
13928 ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
13929 ;*CHANGE IT TO BINARY.
13930 ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
13931 ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
13932 ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
13933 ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
13934 ;*CALL:
13935 ;*      RDOCT          ;;READ AN OCTAL NUMBER
13936 ;*      RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
13937 ;*                  ;;HIGH ORDER BITS ARE IN $HIOCT
13938 056000 011646      $RDOCT: MOV      (SP),-(SP)   ;;PROVIDE SPACE FOR THE
13939 056002 016666 000004 000002  MOV      4(SP),2(SP)   ;;INPUT NUMBER
13940 056010          PUSH     R0,R1,R2
13941 056010 010046          MOV R0,-(SP)
13942 056012 010146          MOV R1,-(SP)
13943 056014 010246          MOV R2,-(SP)
13941 056016 104412      1$:  RDLIN          ;;READ AN ASCIZ LINE
13942 056020 012600      MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
13943 056022 010037 056126  MOV      R0,5$          ;;AND SAVE IT
13944 056026 005001      CLR      R1          ;;CLEAR DATA WORD
13945 056030 005002      CLR      R2
13946 056032 112046      2$:  MOV      (R0)+,-(SP)   ;;PICKUP THIS CHARACTER
13947 056034 001420      BEQ      3$          ;;IF ZERO GET OUT
13948 056036 122716 000060  CMPB   #'0,(SP)      ;;MAKE SURE THIS CHARACTER
13949 056042 003026      BGT      4$          ;;IS AN OCTAL DIGIT
13950 056044 122716 000067  CMPB   #'7,(SP)
13951 056050 002423      BLT      4$
13952 056052 006301      ASL      R1          ;;*2
13953 056054 006102      ROL      R2
13954 056056 006301      ASL      R1          ;;*4
13955 056060 006102      ROL      R2
13956 056062 006301      ASL      R1          ;;*8
13957 056064 006102      ROL      R2
13958 056066 042716 177770  BIC   #'C7,(SP)      ;;STRIP THE ASCII JUNK
13959 056072 062601      ADD   (SP)+,R1      ;;ADD IN THIS DIGIT
13960 056074 000756      BR      2$          ;;LOOP
13961 056076 005726      3$:  TST   (SP)+          ;;CLEAN TERMINATOR FROM STACK
13962 056100 010166 000012  MOV   R1,12(SP)     ;;SAVE THE RESULT
13963 056104 010237 056146  MOV   R2,$HIOCT
13964 056110      POP    R2,R1,R0
13964 056110 012602          MOV (SP)+,R2
13964 056112 012601          MOV (SP)+,R1
13964 056114 012600          MOV (SP)+,R0
13965 056116 000002      RTI          ;;RETURN
13966 056120 005726      4$:  TST   (SP)+          ;;CLEAN PARTIAL FROM STACK
13967 056122 105010      CLR   (R0)          ;;SET A TERMINATOR
13968 056124          TYPE          ;;TYPE UP THRU THE BAD CHAR.
13968 056124 104401      TYPEIT
13969 056126 000000      .DSABL CRF
13970 056130          .WORD 0
13970 056130 104401 067776  TYPE   MSG062          ;INPUT MUST BE A
13970 056130          TYPEIT ,MSG062
13971 056134          .DSABL CRF
13971 056134 104401 070016  TYPE   MSG063          ;N OCTAL
13971 056134          TYPEIT ,MSG063

```



```

13972 056140          .DSABL CRF
        056140 104401 070027 TYPE MSG064          ;NUMBER
                                TYPEIT ,MSG064
                                .DSABL CRF
13973 056144 000724 BR 1$          ;;TRY AGAIN
13974 056146 000000 $HIOCT: .WORD 0          ;;HIGH ORDER BITS GO HERE
13975          .SBTTL ROUTINE READ A DECIMAL NUMBER FROM THE TTY
13976
13977 ;*****
13978 ;*THIS ROUTINE WILL READ A DECIMAL (ASCII) NUMBER FROM THE TTY AND
13979 ;*CHANGE IT TO BINARY. IF TOO MANY CHARACTERS OR ANY ILLEGAL CHARACTERS
13980 ;*ARE READ A "?" FOLLOWED BY A CARRIAGE RETURN-LINE FEED WILL BE TYPED.
13981 ;*THE COMPLETE NUMBER MUST BE RETYPED. THE INPUT IS TERMINATED BY THE
13982 ;*USER TYPING A CARRIAGE RETURN. THE RANGE OF THE INPUT NUMBER IS
13983 ;*POSITIVE 32767 TO NEGATIVE 32768.
13984 ;*CALL:
13985 ;* RDDEC          ;;READ A DECIMAL NUMBER
13986 ;* RETURN HERE   ;;NUMBER IS ON TOP OF THE STACK
13987
13988
13989 056150 011646 $RDDEC: MOV (SP),-(SP)          ;;PROVIDE SPACE FOR
13990 056152 016666 000004 000002 MOV 4(SP),2(SP)          ;;THE INPUT NUMBER
13991 056160          PUSH RO,R1,R2
        056160 010046          MOV RO,-(SP)
        056162 010146          MOV R1,-(SP)
        056164 010246          MOV R2,-(SP)
13992 056166 104412 1$: RDLIN          ;;READ AN ASCII LINE
13993 056170 012600 MOV (SP)+,RO          ;;ADDRESS OF 1ST CHAR.
13994 056172 010037 056316 MOV RO,6$          ;;SAVE IN CASE OF BAD INPUT
13995 056176 005046 CLR -(SP)          ;;CLEAR DATA WORD
13996 056200 005002 CLR R2          ;;SIGN SET POSITIVE
13997 056202 122710 000055 CMPB #'-(,RO)          ;;SEE IF A MINUS SIGN WAS TYPED
13998 056206 001001 BNE 2$          ;;BR IF NO MINUS SIGN
13999 056210 112002 MOVB (RO)+,R2          ;;SAVE FOR LATER USE
14000 056212 112001 2$: MOVB (RO)+,R1          ;;PICKUP THIS CHARACTER
14001 056214 001424 BEQ 3$          ;;GET OUT IF ZERO
14002 056216 122701 000060 CMPB #'0,R1          ;;MAKE SURE THIS CHARACTER
14003 056222 003032 BGT 5$          ;;IS A DIGIT BETWEEN 0 & 9
14004 056224 122701 000071 CMPB #'9,R1
14005 056230 002427 BLT 5$
14006 056232 032716 170000 BIT #+C7777,(SP)          ;;DON'T LET NUMBER GET TO BIG
14007 056236 001024 BNE 5$          ;;BR IF NUMBER WOULD OVERFLOW
14008 056240 006316 ASL (SP)          ;;*2
14009 056242 011646 MOV (SP),-(SP)          ;;SAVE FOR LATER
14010 056244 006316 ASL (SP)          ;;*4
14011 056246 006316 ASL (SP)          ;;*8.
14012 056250 062616 ADD (SP)+,(SP)          ;;*10.
14013 056252 102416 BVS 5$          ;;OVERFLOW ISN'T ALLOWED
14014 056254 162701 000060 SUB #'0,R1          ;;STRIP AWAY THE ASCII JUNK
14015 056260 060116 ADD R1,(SP)          ;;ADD IN THIS DIGIT
14016 056262 102412 BVS 5$          ;;OVERFLOW ISN'T ALLOWED
14017 056264 000752 BR 2$          ;;LOOP
14018 056266 005702 3$: TST R2          ;;CHECK IF NUMBER IS NEG
14019 056270 001401 BEQ 4$          ;;BR IF NO
14020 056272 005416 NEG (SP)          ;;YES--NEGATE THE NUMBER
14021 056274 012666 000012 4$: MOV (SP)+,12(SP)          ;;SAVE THE RESULT
14022 056300 POP R2,R1,RO
    
```



```

14034                                     .SBTTL  ROUTINE SAVE AND RESTORE R0-R5
14035
14036                                     ;*****
14037                                     ;*SAVE R0-R5
14038                                     ;*CALL:
14039                                     ;*   SAVREG
14040                                     ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
14041                                     ;*
14042                                     ;*TOP---(+16)
14043                                     ;* +2---(+18)
14044                                     ;* +4---R5
14045                                     ;* +6---R4
14046                                     ;* +8---R3
14047                                     ;*+10---R2
14048                                     ;*+12---R1
14049                                     ;*+14---R0
14050
14051 056336                               $SAVREG:
14052 056336                               PUSH   R0,R1,R2,R3,R4,R5
14053 056336 010046                               MOV    R0,-(SP)
14054 056340 010146                               MOV    R1,-(SP)
14055 056342 010246                               MOV    R2,-(SP)
14056 056344 010346                               MOV    R3,-(SP)
14057 056346 010446                               MOV    R4,-(SP)
14058 056350 010546                               MOV    R5,-(SP)
14059 056352 016646 000022                       MOV    22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
14060 056356 016646 000022                       MOV    22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
14061 056362 016646 000022                       MOV    22(SP),-(SP)  ;;SAVE PS OF CALL
14062 056366 016646 000022                       MOV    22(SP),-(SP)  ;;SAVE PC OF CALL
14063 056372 000002                               RTI
14064
14065                                     ;*RESTORE R0-R5
14066                                     ;*CALL:
14067                                     ;*   RESREG
14068 056374                               $RESREG:
14069 056374 012666 000022                       MOV    (SP)+,22(SP)  ;;RESTORE PC OF CALL
14070 056400 012666 000022                       MOV    (SP)+,22(SP)  ;;RESTORE PS OF CALL
14071 056404 012666 000022                       MOV    (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
14072 056410 012666 000022                       MOV    (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
14073 056414                               POP    R5,R4,R3,R2,R1,R0
14074 056414 012605                               MOV    (SP)+,R5
14075 056416 012604                               MOV    (SP)+,R4
14076 056420 012603                               MOV    (SP)+,R3
14077 056422 012602                               MOV    (SP)+,R2
14078 056424 012601                               MOV    (SP)+,R1
14079 056426 012600                               MOV    (SP)+,R0
14080 056430 000002                               RTI

```

```

14070                                     .SBTTL  ROUTINE RANDOM NUMBER GENERATOR
14071
14072                                     ;*****
14073                                     ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
14074                                     ;*WITH A RANGE OF 0 TO 2**(+33)-1.
14075                                     ;*CALL:
14076                                     ;*   CALL   $RAND           ;;CALL THE ROUTINE
14077                                     ;*   RETURN          ;;RETURN HERE THE RANDOM
14078                                     ;*                                     ;;NUMBER WILL BE IN
14079                                     ;*                                     ;;$HINUM,$LONUM
14080
14081 056432 $RAND: PUSH   R0,R1,R2
14081 056432 010046
14081 056434 010146
14081 056436 010246
14082 056440 013700 002604
14083 056444 013701 002602
14084 056450 012702 000007
14085 056454 006300
14086 056456 006101
14087 056460 077203
14088 056462 063700 002604
14089 056466 005501
14090 056470 063701 002602
14091 056474 062700 001057
14092 056500 005501
14093 056502 062701 047401
14094 056506 010037 002604
14095 056512 010137 002602
14096 056516
14096 056516 012602
14096 056520 012601
14096 056522 012600
14097 056524 000207

                                     MOV   SEEDLO,R0           ;SET R0 WITH LOW
                                     MOV   SEEDHI,R1           ;SET R1 WITH HIGH
                                     MOV   #7,R2             ;SET SHIFT COUNT
1$: ASL   R0                ;;SHIFT R0 LEFT AND
    ROL   R1                ;;ROTATE CARRY INTO R1 AND
    SOB   R2,1$
    ADD   SEEDLO,R0         ;ADD NUMBER TO MAKE X 129
    ADC   R1                ;;PROPOGATE CARRY
    ADD   SEEDHI,R1        ;ADD NUMBER TO MAKE X 129
    ADD   #1057,R0         ;;ADD LOW CONSTANT
    ADC   R1                ;;PROPOGATE CARRY
    ADD   #47401,R1        ;;ADD HIGH CONSTANT
    MOV   R0,SEEDLO        ;SAVE R0
    MOV   R1,SEEDHI        ;SAVE R1
    POP   R2,R1,R0
                                     MOV (SP)+,R2
                                     MOV (SP)+,R1
                                     MOV (SP)+,R0

                                     RETURN
    
```



```

14100          .SBTTL  ROUTINE DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT
14101          ;*****
14102          ;*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
14103          ;*UNSIGNED OCTAL ASCII NUMBER.
14104          ;*CALL
14105          ;*   MOV     #PNTR,-(SP)      ;; POINTER TO LOW WORD OF BINARY NUMBER
14106          ;*   CALL   $DB20          ;; CALL THE ROUTINE
14107          ;*   RETURN                ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
14108
14109
14110 056526 104415          $DB20: SAVREG                ;; SAVE ALL REGISTERS
14111 056530 016601 000002  MOV     2(SP),R1          ;; PICKUP THE POINTER TO LOW WORD
14112 056534 012705 056645  MOV     #$OCTVL+13.,R5    ;; POINTER TO DATA TABLE
14113 056540 012704 000014  MOV     #12.,R4          ;; DO ELEVEN CHARACTERS
14114 056544 012703 177770  MOV     #+C7,R3         ;; MASK
14115 056550 012100  MOV     (R1)+,R0        ;; LOWER WORD
14116 056552 012101  MOV     (R1)+,R1        ;; HIGH WORD
14117 056554 005002  CLR     R2              ;; TERMINATOR
14118 056556 110245          1$:  MOVB   R2,-(R5)        ;; PUT CHARACTER IN DATA TABLE
14119 056560 010002  MOV     R0,R2           ;; GET THIS DIGIT
14120 056562 005304  DEC     R4              ;; COUNT THIS CHARACTER
14121 056564 003007  BGT    3$              ;; BR IF NOT THE LAST DIGIT
14122 056566 001405  BEQ    2$              ;; BR IF IT IS THE LAST DIGIT
14123 056570 005205  INC     R5              ;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
14124 056572 010566 000002  MOV     R5,2(SP)        ;; ASCII CHAR. & PUT IT ON THE STACK
14125 056576 104416  RESREG                ;; RESTORE ALL REGISTERS
14126 056600 000207  RETURN                ;; RETURN TO USER
14127 056602 006203          2$:  ASR    R3              ;; POSITION THE MASK FOR THE LAST DIGIT
14128 056604 006001          3$:  ROR    R1              ;; POSITION THE BINARY NUMBER FOR
14129 056606 006000  ROR    R0              ;; THE NEXT OCTAL DIGIT
14130 056610 006001  ROR    R1
14131 056612 006000  ROR    R0
14132 056614 006001  ROR    R1
14133 056616 006000  ROR    R0
14134 056620 040302  BIC    R3,R2           ;; MASK OUT ALL JUNK
14135 056622 062702 000060  ADD    #'0,R2         ;; MAKE THIS CHAR. ASCII
14136 056626 000753  BR     1$              ;; GO PUT IT IN THE DATA TABLE
14137 056630 000016          $OCTVL: .REPT 14.     ;; RESERVE DATA TABLE
14140          $OCT8=$OCTVL+4  ;; POINTER TO 11 DIGIT NUMBER
    
```

```

14142                                     .SBTTL TABLES
14143
14144                                     .SBTTL APT MAILBOX-ETABLE
14145 056646 $MAIL:
14146 056646 000000 $MSGTY: .WORD 0 ;;MESSAGE TYPE CODE
14147 056650 000000 $FATAL: .WORD 0 ;;FATAL ERROR NUMBER (ERROR PC)
14148 056652 000000 $TESTN: .WORD 0 ;;TEST PATTERN NUMBER
14149 056654 000000 $PASS: .WORD 0 ;;PASS COUNT
14150 056656 000000 $DEVCT: .WORD 0 ;;DEVICE COUNT
14151 056660 000000 $UNIT: .WORD 0 ;;I/O UNIT NUMBER
14152 056662 000000 $MSGAD: .WORD 0 ;;MESSAGE ADDRESS
14153 056664 000000 $MSGLG: .WORD 0 ;;MESSAGE LENGTH
14154 056666 $ETABLE: ;;APT ENVIRONMENT TABLE
14155 056666 000 $ENV: .BYTE 0 ;;ENVIRONMENT BYTE ;SET TO A 1 FOR APT AUTO MODE
14156 ;NOTE: IF BIT #7 IS SET IN $ENVM THE TABLE BELOW (BEGINNING AT $MAMS1 AND
14157 ; ENDING AT $MADR4) MUST BE FILLED IN TO INDICATE THE PROPER AMOUNT OF
14158 ; EACH TYPE OF MEMORY.
14159 056667 000 $ENVM: .BYTE 0 ;ENVIRONMENT MODE
14160 ;BIT7(200)=USE APT SIZE INFO ;BIT5(40)=NO CONSOLE
14161 056670 000101 $SWREG: .WORD 101 ;APT SWITCH REGISTER
14162 056672 000000 $USWR: .WORD 0 ;USED TO LIMIT THE NUMBER OF PASSES
14163 056674 000000 $CPUOP: .WORD 0 ;CPU TYPE,OPTIONS
14164 ;* BITS 15-11=CPU TYPE
14165 ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
14166 ;* 11/70=06,PDQ=07,Q=10
14167 ;* BIT 10=REAL TIME CLOCK
14168 ;* BIT 9=FLOATING POINT PROCESSOR
14169 ;* BIT 8=MEMORY MANAGEMENT
14170 056676 001 $MAMS1: .BYTE 1 ;HIGH ADDRESS,M.S. BYTE ;DEFAULT = 64K
14171 056677 004 $MTYP1: .BYTE 4 ;MEM. TYPE,BLK#1
14172 ;* MEM.TYPE BYTE -- (HIGH BYTE)
14173 ;* 900 NSEC CORE=001
14174 ;* 300 NSEC BIPOLAR=002
14175 ;* PARITY MOS=003
14176 ;* ERROR CORRECTING MOS=004
14177 056700 177776 $MADR1: .WORD 177776 ;HIGH ADDRESS,BLK#1
14178 ;* MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
14179 056702 000 $MAMS2: .BYTE 0 ;HIGH ADDRESS,M.S. BYTE
14180 056703 000 $MTYP2: .BYTE 0 ;MEM.TYPE,BLK#2
14181 056704 000000 $MADR2: .WORD 0 ;MEM.LAST ADDRESS,BLK#2
14182 056706 000 $MAMS3: .BYTE 0 ;HIGH ADDRESS,M.S.BYTE
14183 056707 000 $MTYP3: .BYTE 0 ;MEM.TYPE,BLK#3
14184 056710 000000 $MADR3: .WORD 0 ;MEM.LAST ADDRESS,BLK#3
14185 056712 000 $MAMS4: .BYTE 0 ;HIGH ADDRESS,M.S.BYTE
14186 056713 000 $MTYP4: .BYTE 0 ;MEM.TYPE,BLK#4
14187 056714 000000 $MADR4: .WORD 0 ;MEM.LAST ADDRESS,BLK#4
14188 056716 000000 $VECT1: .WORD 0 ;INTERRUPT VECTOR#1,BUS PRIORITY#1
14189 056720 000000 $VECT2: .WORD 0 ;INTERRUPT VECTOR#2BUS PRIORITY#2
14190 056722 000000 $BASE: .WORD 0 ;BASE ADDRESS OF EQUIPMENT UNDER TEST
14191 056724 000000 $DEVM: .WORD 0 ;DEVICE MAP
14192
14193 056726 000000 $CDW1: .WORD 0
14194 056730 000000 $CDW2: .WORD 0
14195 056732 000000 $DDW7: .WORD 0 ;UFD MODE FLAG 1=UFD MODE

```

```

14197 ;THE FOLLOWING LOCATIONS SPECIFY WHICH PATTERNS
14198 ;ARE TO BE RUN FOR PARTICULAR MEMORIES
14199 ;
14200 ;REFERENCE THE TABLE LISTED BELOW TO RELATE BITS TO PATTERNS.
14201 ;BITO SET WILL RUN THE FIRST ENTRY IN THE TABLE, BITO SET
14202 ;IN THE SECOND WORD WILL RUN THE 17TH ENTRY IN THE TABLE ...
14203 ;
14204 ;NOTE** NULL TESTS DO NOT TAKE ANY TIME
14205 ;
14206 056734 177777 ;FIELD SERVICE VALUE
14207 056736 177777 $DDW0: .WORD 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
14208 056740 177777 $DDW1: .WORD 177777 ;ECC CSR TESTS 177777 TABLE = MKCSRT:
14209 056742 177777 $DDW2: .WORD 177777 ;ECC PATTERNS 103777 TABLE = MKPAT:
14210 056744 177777 $DDW3: .WORD 177777 ;ECC PATTERNS 177777 TABLE = MKPAT:
14211 056746 177777 $DDW4: .WORD 177777 ;PARITY PATTERNS 003777 TABLE = MJPAT:
14215 056750 $DDW5: .WORD 177777 ;PARITY PATTERNS 177774 TABLE = MJPAT:
14216 ;ETEND:
14217 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
14218 ;INTERFACE SPEC.
14219 056750 $APTHD:
14220 056750 000000 $HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
14221 056752 056646 $MBADR: .WORD $MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
14222 056754 000043 $TSTM: .WORD 35. ;:RUN TIM OF LONGEST TEST
14223 056756 001274 $PASTM: .WORD 700. ;:RUN TIME IN SECS. OF 1ST PASS ON 128K (QUICK VERIFY)
14224 056760 000000 $UNITM: .WORD 0. ;:EXTRA RUN TIME OF A PASS FOR EACH ADDITIONAL 128K (QV)
14225 056762 000041 .WORD $ETEND-$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)

```


.SBTTL ROUTINE TRAP DECODER

```

14227
14228
14229
14230
14231
14232
14233
14234
14235 056764 010046
14236 056766 016600 000002
14237 056772 005740
14238 056774 111000
14239 056776 006300
14240 057000 016000 057026
14241 057004 000200
14242
14243
14244
14245
14246 057006 011646
14247 057010 016666 000004 000002
14248 057016 000002
14249
14250 057020
      057020 104401 065507
14251 057024 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    $TRPAD(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

```



```

14254
14255
14256
14257
14258
14259
14260
14261 057026 057006
14262 057030 047122
14263 057032 054102
14264 057034 054056
14265 057036 057020
14266 057040 054304
14267 057042 057020
14268
14269 057044 054704
14270 057046 054530
14271
14272 057050 055264
14273 057052 055414
14274 057054 056000
14275 057056 056150
14276
14277 057060 056336
14278 057062 056374
14279
14280 057064 034202
14281 057066 034212
14282 057070 034222
14283
14284 057072 036310
14285
14286 057074 034232
14287 057076 034256
14288
14289 057100 034274
14290 057102 034370
14291
14292 057104 047456
14293 057106 047504
14294 057110 047532
14295 057112 047562
14296 057114 047644
14297 057116 047666
14298 057120 047716
14299 057122 047736
14300 057124 047760
14301 057126 050000
14302 057130 050022
14303 057132 050044
14304 057134 050064
14305 057136 050102
14306 057140 050120
14307 057142 050140
14308 057144 050156
14309 057146 050174
14310 057150 044756

.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

```

14311	057152	050364	\$PER26	;CALL=PERR26	TRAP+52(104452)	PROGRAM DETECTED ERROR
14312	057154	050404	\$PER27	;CALL=PERR27	TRAP+53(104453)	PROGRAM DETECTED ERROR
14313	057156	045204	\$PER30	;CALL=PERR30	TRAP+54(104454)	PROGRAM DETECTED ERROR
14314	057160	050574	\$PER31	;CALL=PERR31	TRAP+55(104455)	PROGRAM DETECTED ERROR
14315	057162	050672	\$PER32	;CALL=PERR32	TRAP+56(104456)	PROGRAM DETECTED ERROR
14316	057164	050740	\$PER33	;CALL=PERR33	TRAP+57(104457)	PROGRAM DETECTED ERROR
14317	057166	051020	\$PER34	;CALL=PERR34	TRAP+60(104460)	PROGRAM DETECTED ERROR
14318	057170	051052	\$PER35	;CALL=PERR35	TRAP+61(104461)	PROGRAM DETECTED ERROR
14319	057172	051106	\$PER36	;CALL=PERR36	TRAP+62(104462)	PROGRAM DETECTED ERROR
14320	057174	051136	\$PER37	;CALL=PERR37	TRAP+63(104463)	PROGRAM DETECTED ERROR
14321	057176	051142	\$PER40	;CALL=PERR40	TRAP+64(104464)	PROGRAM DETECTED ERROR
14322	057200	057020	\$NOTRAP	;CALL=PERR41	TRAP+65(104465)	PROGRAM DETECTED ERROR
14323	057202	057020	\$NOTRAP	;CALL=PERR42	TRAP+66(104466)	PROGRAM DETECTED ERROR
14324	057204	057020	\$NOTRAP	;CALL=PERR43	TRAP+67(104467)	PROGRAM DETECTED ERROR
14325						
14326	057206	034570	\$ECCDIS	;CALL=ECCDIS	TRAP+70(104470)	DISABLE ECC ON ALL CSR'S
14327	057210	034604	\$ECC1DIS	;CALL=ECC1DIS	TRAP+71(104471)	DISABLE ECC ON 1 SELECTED CSR
14328	057212	034616	\$ECCINIT	;CALL=ECCINIT	TRAP+72(104472)	INITIALIZE ALL MK11 CSR'S
14329	057214	034632	\$ECC1INIT	;CALL=ECC1INIT	TRAP+73(104473)	INITIALIZE 1 SELECTED MK11 CSR
14330	057216	034672	\$CBCSR	;CALL=CBCSR	TRAP+74(104474)	WRITE GENERATED CHECKBITS IN ALL CSR'S
14331	057220	034714	\$CB1CSR	;CALL=CB1CSR	TRAP+75(104475)	WRITE GENERATED CHECKBITS IN 1 SELECTED CSR
14332	057222	034734	\$WASSBE	;CALL=WASSBE	TRAP+76(104476)	WAS THERE A SBE ON ANY CSR?
14333	057224	035050	\$WAS1SBE	;CALL=WAS1SBE	TRAP+77(104477)	WAS THERE A SBE ON 1 SELECTED CSR?
14334	057226	035100	\$WASDBE	;CALL=WASDBE	TRAP+100(104500)	WAS THERE A DBE ON ANY CSR?
14335	057230	035214	\$WAS1DBE	;CALL=WAS1DBE	TRAP+101(104501)	WAS THERE A DBE ON 1 SELECTED CSR?
14336	057232	035244	\$CLRCSR	;CALL=CLRCSR	TRAP+102(104502)	CLEAR ALL CSR'S
14337	057234	035256	\$CLR1CSR	;CALL=CLR1CSR	TRAP+103(104503)	CLEAR 1 SELECTED CSR
14338	057236	035266	\$CHKDIS	;CALL=CHKDIS	TRAP+104(104504)	DISABLE ECC & WRITE CKBITS FROM ALL CSR'S
14339	057240	035302	\$CHK1DIS	;CALL=CHK1DIS	TRAP+105(104505)	DISABLE ECC & WRITE CKBITS FROM 1 CSR
14340	057242	034644	\$ENASBE	;CALL=ENASBE	TRAP+106(104506)	ENABLE TRAPS ON SBE'S FROM ALL CSR'S
14341	057244	034660	\$ENA1SBE	;CALL=ENA1SBE	TRAP+107(104507)	ENABLE TRAPS ON SBE'S FROM 1 SELECTED CSR
14342	057246	034410	\$TSTRD	;CALL=TSTREAD	TRAP+110(104510)	TEST LOC (R1) & TST FOR SBE (WITHOUT FETCHES)
14343	057250	035362	\$INVALID	;CALL=INVALID	TRAP+111(104511)	INVALIDATE BACKGROUND PATTERN ON BANK
14344	057252	035412	\$ERRGEN	;CALL=ERRGEN	TRAP+112(104512)	TEST ERROR ADDRESS
14345	057254	035662	\$CBREG	;CALL=CBREG	TRAP+113(104513)	ENABLE CHECK BIT REGISTER
14346	057256	035700	\$SYNREG	;CALL=SYNREG	TRAP+114(104514)	ENABLE SYNDROME BIT REGISTER
14347	057260	057020	\$NOTRAP			
14348	057262	057020	\$NOTRAP			
14349	057264	057020	\$NOTRAP			
14350	057266	057020	\$NOTRAP			
14351	057270	057020	\$NOTRAP			
14352	057272	057020	\$NOTRAP			

::ILC;REV B
::ILC;REV B

14355 177776 ST = 177776 ;STATUS REGISTER

14358
14359
14360
14361
14362
14363
14364
14365
14366
14367
14368
14369
14370
14371
14372 057274
14373 057274 061762
14374 057276 064172
14375 057300 060320
14376 057302 060701
14377
14378 057304 060747
14379 057306 063501
14380 057310 060144
14381 057312 060557
14382
14383 057314 061005
14384 057316 063561
14385 057320 060162
14386 057322 060674
14387
14388 057324 061037
14389 057326 063561
14390 057330 060172
14391 057332 060674
14392
14393 057334 061105
14394 057336 063615
14395 057340 060202
14396 057342 060557
14397
14398 057344 061162
14399 057346 063615
14400 057350 060202
14401 057352 060557
14402
14403 057354 061207
14404 057356 063615
14405 057360 060202
14406 057362 060557
14407
14408 057364 063173
14409 057366 064714
14410 057370 060476
14411 057372 060557

.SBTTL TABLE ERROR POINTER

;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (ERRPC).
;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

;* EM ;;POINTS TO THE ERROR MESSAGE
;* DH ;;POINTS TO THE DATA HEADER
;* DT ;;POINTS TO THE DATA
;* DF ;;POINTS TO THE DATA FORMAT

\$ERRTB: ;ERROR 1
EM24
DH13
DT13
DF11
;ERROR 2
EM2
DH1
DT1
DF2
;ERROR 3
EM3
DH3
DT3
DF9
;ERROR 4
EM4
DH3
DT4
DF9
;ERROR 5
EM5
DH5
DT5
DF2
;ERROR 6
EM6
DH5
DT5
DF2
;ERROR 7
EM7
DH5
DT5
DF2
;ERROR 10
EM53
DH25
DT25
DF2

14414			;ERROR	11
14415	057374	061247	EM11	
14416	057376	063741	DH7	
14417	057400	060234	DT7	
14418	057402	060603	DF3	
14419			;ERROR	12
14420	057404	061247	EM11	
14421	057406	063741	DH7	
14422	057410	060234	DT7	
14423	057412	060616	DF4	
14424			;ERROR	13
14425	057414	061271	EM12	
14426	057416	064051	DH10	
14427	057420	060264	DT10	
14428	057422	060557	DF2	
14429			;ERROR	14
14430	057424	061247	EM11	
14431	057426	063741	DH7	
14432	057430	060234	DT7	
14433	057432	060631	DF5	
14434			;ERROR	15
14435	057434	061247	EM11	
14436	057436	063741	DH7	
14437	057440	060234	DT7	
14438	057442	060644	DF6	
14439			;ERROR	16
14440	057444	061315	EM13	
14441	057446	064172	DH13	
14442	057450	060320	DT13	
14443	057452	060701	DF11	
14444			;ERROR	17
14445	057454	061347	EM14	
14446	057456	064172	DH13	
14447	057460	060320	DT13	
14448	057462	060701	DF11	
14449			;ERROR	20
14450	057464	061413	EM15	
14451	057466	064172	DH13	
14452	057470	060320	DT13	
14453	057472	060701	DF11	
14454			;ERROR	21
14455	057474	063222	EM55	
14456	057476	064752	DH26	
14457	057500	060510	DT26	
14458	057502	060557	DF2	
14459			;ERROR	22
14460	057504	061461	EM17	
14461	057506	063741	DH7	
14462	057510	060234	DT7	
14463	057512	060631	DF5	
14464			;ERROR	23
14465	057514	063042	EM50	
14466	057516	064566	DH23	
14467	057520	060434	DT23	
14468	057522	060712	DF13	

14471			;ERROR	24	
14472	057524	061521	EM19		
14473	057526	064172	DH13		
14474	057530	060320	DT13		
14475	057532	060701	DF11		
14476			;ERROR	25	
14477	057534	061573	EM20		
14478	057536	064172	DH13		
14479	057540	060320	DT13		
14480	057542	060701	DF11		
14481			;ERROR	26	
14482	057544	000000	0		;NO MESSAGE
14483	057546	064165	DH12		
14484	057550	060314	DT12		
14485	057552	060557	DF2		
14486			;ERROR	27	
14487	057554	061652	EM21		
14488	057556	064147	DH11		
14489	057560	060306	DT11		
14490	057562	060557	DF2		
14491			;ERROR	30	
14492	057564	061706	EM22		
14493	057566	064172	DH13		
14494	057570	060320	DT13		
14495	057572	060701	DF11		
14496			;ERROR	31	
14497	057574	000000	0		;NO MESSAGE
14498	057576	064267	DH14		
14499	057600	060342	DT14		
14500	057602	060557	DF2		
14501			;ERROR	32	
14502	057604	061733	EM23		
14503	057606	063615	DH5		
14504	057610	060202	DT5		
14505	057612	060557	DF2		
14506			;ERROR	33	
14507	057614	062041	EM25		
14508	057616	064346	DH15		
14509	057620	060360	DT16		
14510	057622	060657	DF7		
14511			;ERROR	34	
14512	057624	062066	EM26		
14513	057626	064465	DH16		
14514	057630	060410	DT17		
14515	057632	060603	DF3		

14518			;ERROR	35
14519	057634	063146	EM52	
14520	057636	064714	DH25	
14521	057640	060476	DT25	
14522	057642	060557	DF2	
14523			;ERROR	36
14524	057644	062137	EM27	
14525	057646	064465	DH16	
14526	057650	060410	DT17	
14527	057652	060672	DF8	
14528			;ERROR	37
14529	057654	062634	EM35	
14530	057656	063741	DH7	
14531	057660	060234	DT7	
14532	057662	060603	DF3	
14533			;ERROR	40
14534	057664	062227	EM29	
14535	057666	063741	DH7	
14536	057670	060234	DT7	
14537	057672	060603	DF3	
14538			;ERROR	41
14539	057674	062311	EM30	
14540	057676	063741	DH7	
14541	057700	060234	DT7	
14542	057702	060631	DF5	
14543			;ERROR	42
14544	057704	063343	EM60	
14545	057706	064507	DH20	
14546	057710	060434	DT23	
14547	057712	060712	DF13	
14548			;ERROR	43
14549	057714	062421	EM32	
14550	057716	063741	DH7	
14551	057720	060234	DT7	
14552	057722	060603	DF3	
14553			;ERROR	44
14554	057724	062526	EM33	
14555	057726	063741	DH7	
14556	057730	060234	DT7	
14557	057732	060603	DF3	
14558			;ERROR	45
14559	057734	063076	EM51	
14560	057736	064645	DH24	
14561	057740	060456	DT24	
14562	057742	060722	DF14	
14563			;ERROR	46
14564	057744	062721	EM36	
14565	057746	063674	DH6	
14566	057750	060220	DT6	
14567	057752	060557	DF2	

14570			;ERROR	47
14571	057754	062770	EM40	
14572	057756	063536	DH2	
14573	057760	060416	DT20	
14574	057762	060557	DF2	
14575			;ERROR	50
14576	057764	063243	EM56	
14577	057766	064770	DH27	
14578	057770	060516	DT27	
14579	057772	060556	DF1	
14580			;ERROR	51
14581	057774	063405	EM61	
14582	057776	064645	DH24	
14583	060000	060456	DT24	
14584	060002	060722	DF14	
14585			;ERROR	52
14586	060004	062227	EM29	
14587	060006	064172	DH13	
14588	060010	060320	DT13	
14589	060012	060701	DF11	
14590			;ERROR	53
14591	060014	062041	EM25	
14592	060016	065044	DH30	
14593	060020	060536	DT30	
14594	060022	060740	DF16	
14595			;ERROR	54
14596	060024	063275	EM57	
14597	060026	065044	DH30	
14598	060030	060536	DT30	
14599	060032	060740	DF16	
14600			;ERROR	55
14601	060034	061573	EM20	
14602	060036	064645	DH24	
14603	060040	060456	DT24	
14604	060042	060722	DF14	
14605			;ERROR	56
14606	060044	061573	EM20	
14607	060046	065044	DH30	
14608	060050	060536	DT30	
14609	060052	060740	DF16	
14610			;ERROR	57
14611	060054	061521	EM19	
14612	060056	064645	DH24	
14613	060060	060456	DT24	
14614	060062	060722	DF14	
14615			;ERROR	60
14616	060064	061315	EM13	
14617	060066	064507	DH20	
14618	060070	060434	DT23	
14619	060072	060712	DF13	
14620			;ERROR	61
14621	060074	062311	EM30	
14622	060076	064507	DH20	
14623	060100	060434	DT23	
14624	060102	060712	DF13	
14625			;ERROR	62
14626	060104	061573	EM20	

14627	060106	064645	DH24
14628	060110	060456	DT24
14629	060112	060722	DF14
14630			;ERROR 63
14631	060114	061706	EM22
14632	060116	064645	DH24
14633	060120	060456	DT24
14634	060122	060722	DF14
14635			;ERROR 64
14636	060124	063426	EM62
14637	060126	063615	DH5
14638	060130	060202	DT5
14639	060132	060557	DF2
14640			;ERROR 65
14641	060134	061706	EM22
14642	060136	064507	DH20
14643	060140	060434	DT23
14644	060142	060712	DF13

14646						.SBTTL	ERROR DATA TAGS (DT)
14647	060144	002020	002034	002044	DT1:	.WORD	ERRPC, ADDRESS, GOOD, BAD, 0
	060152	002052	000000				
14648	060156	002020	000000		DT2:	.WORD	ERRPC, 0
14649	060162	002020	002036	002072	DT3:	.WORD	ERRPC, PADDRESS, PARCNT, 0
	060170	000000					
14650	060172	002020	002034	002070	DT4:	.WORD	ERRPC, ADDRESS, NEMCNT, 0
	060200	000000					
14651	060202	002020	177572	177574	DT5:	.WORD	ERRPC, MMRO, MMR1, MMR2, MMR3, CPUERR, 0
	060210	177576	172516	177766			
	060216	000000					
14652	060220	002020	002422	002400	DT6:	.WORD	ERRPC, APTPAR, LSIZE, APTECC, MSIZE, 0
	060226	002424	002402	000000			
14653	060234	002020	002176	002034	DT7:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, GOOD, BAD, BADXOR
	060242	002176	002044	002052			
	060250	002060					
14654	060252	002176	002176	002176		.WORD	DUMMY, DUMMY, DUMMY, DUMMY, 0
	060260	002176	000000				
14655	060264	002200	002202	002204	DT10:	.WORD	DETRO, DETR1, DETR2, DETR3, DETR4, DETR5, DETSP, DETPSW, 0
	060272	002206	002210	002212			
	060300	002214	002216	000000			
14656	060306	002020	002150	000000	DT11:	.WORD	ERRPC, CSR, 0
14657	060314	002150	000000		DT12:	.WORD	CSR, 0
14658	060320	002020	002176	002034	DT13:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, TSTDAT, TSTDAT+2, CHECK, CSR, 0
	060326	002176	002246	002250			
	060334	002314	002150	000000			
14659	060342	177746	177572	177574	DT14:	.WORD	CONTRL, MMRO, MMR1, MMR2, MMR3, CPUERR, 0
	060350	177576	172516	177766			
	060356	000000					
14660	060360	002020	002176	002176	DT16:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, GOOD2, GOOD3
	060366	002044	002046	002050			
14661	060374	002052	002054	002056		.WORD	BAD, BAD2, BAD3, DUMMY, DUMMY, 0
	060402	002176	002176	000000			
14662	060410	002020	002176	000000	DT17:	.WORD	ERRPC, DUMMY, 0
14663	060416	002020	002044	002052	DT20:	.WORD	ERRPC, GOOD, BAD, 0
	060424	000000					
14664	060426	002020	002176	000000	DT22:	.WORD	ERRPC, DUMMY, 0
14665	060434	002020	002176	002044	DT23:	.WORD	ERRPC, DUMMY, GOOD, BAD, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060442	002052	002176	002176			
	060450	002176	002176	000000			
14666	060456	002020	002176	002150	DT24:	.WORD	ERRPC, DUMMY, CSR, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060464	002176	002176	002176			
	060472	002176	000000				
14667	060476	002020	002044	002150	DT25:	.WORD	ERRPC, GOOD, CSR, CSRNO, 0
	060504	002152	000000				
14668	060510	002020	002052	000000	DT26:	.WORD	ERRPC, BAD, 0
14669	060516	002020	002176	002034	DT27:	.WORD	ERRPC, DUMMY, ADDRESS, DUMMY, DUMMY, DUMMY, DUMMY, 0
	060524	002176	002176	002176			
	060532	002176	000000				
14670	060536	002020	002176	002176	DT30:	.WORD	ERRPC, DUMMY, DUMMY, GOOD, BAD, CSR, DUMMY, 0
	060544	002044	002052	002150			
	060552	002176	000000				

Address	Hex	Hex	Hex	Hex	DF	Format	Value
14673						.SBTTL	ERROR DATA FORMATS (DF)
14674	060556	000			DF1:	.BYTE	0
14675	060557	000	000	000	DF2:	.BYTE	0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
	060562	000	000	000			
	060565	000	000	000			
	060570	000	000	000			
	060573	000	000	000			
	060576	000	000	000			
	060601	000	000	000			
14676	060603	000	005	000	DF3:	.BYTE	0,5,0,8.,0,0,0,3,6,2,4
	060606	010	000	000			
	060611	000	003	006			
	060614	002	004	006			
14677	060616	000	005	000	DF4:	.BYTE	0,5,0,8.,0,8.,8.,3,6,2,4
	060621	010	000	010			
	060624	010	003	006			
	060627	002	004	006			
14678	060631	000	005	000	DF5:	.BYTE	0,5,0,8.,9.,9.,9.,3,6,2,4
	060634	010	011	011			
	060637	011	003	006			
	060642	002	004	006			
14679	060644	000	005	000	DF6:	.BYTE	0,5,0,8.,9.,8.,8.,3,6,2,4
	060647	010	011	010			
	060652	010	003	006			
	060655	002	004	006			
14680	060657	000	005	010	DF7:	.BYTE	0,5,8.,0,0,9.,0,0,9.,2,4
	060662	000	000	011			
	060665	000	000	011			
	060670	002	004	006			
14681	060672	000	005	000	DF8:	.BYTE	0,5
14682	060674	000	001	001	DF9:	.BYTE	0,1,1,1,1
	060677	001	001	001			
14683	060701	000	005	000	DF11:	.BYTE	0,5,0,8.,0,0,0,0,0
	060704	010	000	000			
	060707	000	000	000			
14684	060712	000	005	000	DF13:	.BYTE	0,5,0,0,3,6,2,4
	060715	000	003	006			
	060720	002	004	006			
14685	060722	000	005	000	DF14:	.BYTE	0,5,0,3,6,2,4
	060725	003	006	002			
	060730	004	006	002			
14686	060731	000	005	000	DF15:	.BYTE	0,5,0,8.,3,6,4
	060734	010	003	006			
	060737	004	006	002			
14687	060740	000	005	010	DF16:	.BYTE	0,5,8.,0,0,3,4
	060743	000	000	003			
	060746	004	006	002			

	061215	040	115	101	
	061220	116	101	107	
	061223	105	115	105	
	061226	116	124	040	
	061231	050	124	122	
	061234	101	120	040	
	061237	124	117	040	
	061242	062	065	060	
	061245	051	000		
14702	061247	115	105	115	EM11: .ASCIZ /MEMORY DATA ERROR/
	061252	117	122	131	
	061255	040	104	101	
	061260	124	101	040	
	061263	105	122	122	
	061266	117	122	000	
14703	061271	104	105	124	EM12: .ASCIZ /DETAILED ERROR DUMP/
	061274	101	111	114	
	061277	105	104	040	
	061302	105	122	122	
	061305	117	122	040	
	061310	104	125	115	
	061313	120	000		
14704	061315	115	111	123	EM13: .ASCIZ /MISSING EXPECTED SBE FLAG/
	061320	123	111	116	
	061323	107	040	105	
	061326	130	120	105	
	061331	103	124	105	
	061334	104	040	123	
	061337	102	105	040	
	061342	106	114	101	
	061345	107	000		
14705	061347	127	122	111	EM14: .ASCIZ /WRITE BYTE FAILED TO CLEAR SBE FLAG/
	061352	124	105	040	
	061355	102	131	124	
	061360	105	040	106	
	061363	101	111	114	
	061366	105	104	040	
	061371	124	117	040	
	061374	103	114	105	
	061377	101	122	040	
	061402	123	102	105	
	061405	040	106	114	
	061410	101	107	000	
14706	061413	106	101	111	EM15: .ASCIZ /FAILED TO GET INTERRUPT WITH DBE FLAG/
	061416	114	105	104	
	061421	040	124	117	
	061424	040	107	105	
	061427	124	040	111	
	061432	116	124	105	
	061435	122	122	125	
	061440	120	124	040	
	061443	127	111	124	
	061446	110	040	104	
	061451	102	105	040	
	061454	106	114	101	
	061457	107	000		
14707	061461	115	105	115	EM17: .ASCIZ /MEMORY DATA ERROR IN CHECK BITS/

	061464	117	122	131	
	061467	040	104	101	
	061472	124	101	040	
	061475	105	122	122	
	061500	117	122	040	
	061503	111	116	040	
	061506	103	110	105	
	061511	103	113	040	
	061514	102	111	124	
	061517	123	000		
14708	061521	123	102	105	EM19: .ASCIZ /SBE-DBE CAUSED PARITY TRAP WHEN INHIBITED/
	061524	055	104	102	
	061527	105	040	103	
	061532	101	125	123	
	061535	105	104	040	
	061540	120	101	122	
	061543	111	124	131	
	061546	040	124	122	
	061551	101	120	040	
	061554	127	110	105	
	061557	116	040	111	
	061562	116	110	111	
	061565	102	111	124	
14709	061570	105	104	000	
	061573	123	102	105	EM20: .ASCIZ /SBE-DBE DID NOT CAUSE PARITY TRAP WHEN ENABLED/
	061576	055	104	102	
	061601	105	040	104	
	061604	111	104	040	
	061607	116	117	124	
	061612	040	103	101	
	061615	125	123	105	
	061620	040	120	101	
	061623	122	111	124	
	061626	131	040	124	
	061631	122	101	120	
	061634	040	127	110	
	061637	105	116	040	
	061642	105	116	101	
	061645	102	114	105	
	061650	104	000		
14710	061652	123	102	105	EM21: .ASCIZ /SBE-DBE ON MASTER TEST WORD/
	061655	055	104	102	
	061660	105	040	117	
	061663	116	040	115	
	061666	101	123	124	
	061671	105	122	040	
	061674	124	105	123	
	061677	124	040	127	
	061702	117	122	104	
	061705	000			
14711	061706	115	111	123	EM22: .ASCIZ /MISSING EXPECTED DBE/
	061711	123	111	116	
	061714	107	040	105	
	061717	130	120	105	
	061722	103	124	105	
	061725	104	040	104	
	061730	102	105	000	

14712	061733	125	116	105	EM23: .ASCIZ /UNEXPECTED PARITY TRAP/
	061736	130	120	105	
	061741	103	124	105	
	061744	104	040	120	
	061747	101	122	111	
	061752	124	131	040	
	061755	124	122	101	
	061760	120	000		
14713	061762	122	105	103	EM24: .ASCIZ /RECEIVED DBE FLAG WHEN EXPECTING ONLY SBE FLAG/
	061765	105	111	126	
	061770	105	104	040	
	061773	104	102	105	
	061776	040	106	114	
	062001	101	107	040	
	062004	127	110	105	
	062007	116	040	105	
	062012	130	120	105	
	062015	103	124	111	
	062020	116	107	040	
	062023	117	116	114	
	062026	131	040	123	
	062031	102	105	040	
	062034	106	114	101	
	062037	107	000		
14714	062041	103	110	105	EM25: .ASCIZ /CHECK BIT DATA ERROR/
	062044	103	113	040	
	062047	102	111	124	
	062052	040	104	101	
	062055	124	101	040	
	062060	105	122	122	
	062063	117	122	000	
14715	062066	101	104	104	EM26: .ASCIZ /ADDRESS PARITY ERROR DID NOT CAUSE ABORT/
	062071	122	105	123	
	062074	123	040	120	
	062077	101	122	111	
	062102	124	131	040	
	062105	105	122	122	
	062110	117	122	040	
	062113	104	111	104	
	062116	040	116	117	
	062121	124	040	103	
	062124	101	125	123	
	062127	105	040	101	
	062132	102	117	122	
	062135	124	000		
14716	062137	105	103	103	EM27: .ASCIZ /ECC INHIBIT MODE POINTER FAILURE - DID NOT PROTECT BANK/
	062142	040	111	116	
	062145	110	111	102	
	062150	111	124	040	
	062153	115	117	104	
	062156	105	040	120	
	062161	117	111	116	
	062164	124	105	122	
	062167	040	106	101	
	062172	111	114	125	
	062175	122	105	040	
	062200	055	040	104	

	062203	111	104	040	
	062206	116	117	124	
	062211	040	120	122	
	062214	117	124	105	
	062217	103	124	040	
	062222	102	101	116	
	062225	113	000		
14717	062227	103	117	122	EM29: .ASCIZ /CORRECTION FAILURE WITH ECC ENABLED ON FORCED SBE/
	062232	122	105	103	
	062235	124	111	117	
	062240	116	040	106	
	062243	101	111	114	
	062246	125	122	105	
	062251	040	127	111	
	062254	124	110	040	
	062257	105	103	103	
	062262	040	105	116	
	062265	101	102	114	
	062270	105	104	040	
	062273	117	116	040	
	062276	106	117	122	
	062301	103	105	104	
	062304	040	123	102	
	062307	105	000		
14718	062311	127	122	111	EM30: .ASCII /WRITE BYTE WITH ECC ENABLED FAILED TO CLEAR DATA AT/<CRLF>
	062314	124	105	040	
	062317	102	131	124	
	062322	105	040	127	
	062325	111	124	110	
	062330	040	105	103	
	062333	103	040	105	
	062336	116	101	102	
	062341	114	105	104	
	062344	040	106	101	
	062347	111	114	105	
	062352	104	040	124	
	062355	117	040	103	
	062360	114	105	101	
	062363	122	040	104	
	062366	101	124	101	
	062371	040	101	124	
	062374	200			
14719	062375	106	117	122	.ASCIZ /FORCED SBE LOCATION/
	062400	103	105	104	
	062403	040	123	102	
	062406	105	040	114	
	062411	117	103	101	
	062414	124	111	117	
	062417	116	000		
14720	062421	115	117	126	EM32: .ASCIZ /MOVB #360,(R2)+ WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	062424	102	040	043	
	062427	063	066	060	
	062432	054	050	122	
	062435	062	051	053	
	062440	040	127	111	
	062443	124	110	040	
	062446	105	103	103	

	062451	040	105	116	
	062454	101	102	114	
	062457	105	104	040	
	062462	103	110	101	
	062465	116	107	105	
	062470	104	040	104	
	062473	101	124	101	
	062476	040	101	124	
	062501	040	106	117	
	062504	122	103	105	
	062507	104	040	104	
	062512	102	105	040	
	062515	114	117	103	
	062520	101	124	111	
	062523	117	116	000	
14721	062526	115	117	126	EM33: .ASCIZ /MOV #177400,(R1) WITH ECC ENABLED CHANGED DATA AT FORCED DBE LOCATION/
	062531	040	043	061	
	062534	067	067	064	
	062537	060	060	054	
	062542	050	122	061	
	062545	051	040	127	
	062550	111	124	110	
	062553	040	105	103	
	062556	103	040	105	
	062561	116	101	102	
	062564	114	105	104	
	062567	040	103	110	
	062572	101	116	107	
	062575	105	104	040	
	062600	104	101	124	
	062603	101	040	101	
	062606	124	040	106	
	062611	117	122	103	
	062614	105	104	040	
	062617	104	102	105	
	062622	040	114	117	
	062625	103	101	124	
	062630	111	117	116	
	062633	000			
14722	062634	125	116	105	EM35: .ASCIZ /UNEXPECTED CORRECTION WITH ECC DISABLE ON FORCED SBE/
	062637	130	120	105	
	062642	103	124	105	
	062645	104	040	103	
	062650	117	122	122	
	062653	105	103	124	
	062656	111	117	116	
	062661	040	127	111	
	062664	124	110	040	
	062667	105	103	103	
	062672	040	104	111	
	062675	123	101	102	
	062700	114	105	040	
	062703	117	116	040	
	062706	106	117	122	
	062711	103	105	104	
	062714	040	123	102	
	062717	105	000		

14723	062721	101	120	124	EM36:	.ASCIZ /APT SIZE DISAGREES WITH PROGRAM SIZING/
	062724	040	123	111		
	062727	132	105	040		
	062732	104	111	123		
	062735	101	107	122		
	062740	105	105	123		
	062743	040	127	111		
	062746	124	110	040		
	062751	120	122	117		
	062754	107	122	101		
	062757	115	040	123		
	062762	111	132	111		
	062765	116	107	000		
14724	062770	102	122	101	EM40:	.ASCIZ /BRANCH GOBBLE FAILED CONDITION CODES TEST/
	062773	116	103	110		
	062776	040	107	117		
	063001	102	102	114		
	063004	105	040	106		
	063007	101	111	114		
	063012	105	104	040		
	063015	103	117	116		
	063020	104	111	124		
	063023	111	117	116		
	063026	040	103	117		
	063031	104	105	123		
	063034	040	124	105		
	063037	123	124	000		
14725	063042	102	101	104	EM50:	.ASCIZ /BAD ERROR ADDRESS GENERATED/
	063045	040	105	122		
	063050	122	117	122		
	063053	040	101	104		
	063056	104	122	105		
	063061	123	123	040		
	063064	107	105	116		
	063067	105	122	101		
	063072	124	105	104		
	063075	000				
14726	063076	106	114	101	EM51:	.ASCIZ /FLAGS NOT SET ON FORCED UNCORRECTED SBE/
	063101	107	123	040		
	063104	116	117	124		
	063107	040	123	105		
	063112	124	040	117		
	063115	116	040	106		
	063120	117	122	103		
	063123	105	104	040		
	063126	125	116	103		
	063131	117	122	122		
	063134	105	103	124		
	063137	105	104	040		
	063142	123	102	105		
	063145	000				
14727	063146	102	111	124	EM52:	.ASCIZ /BIT SET ERROR IN CSR/
	063151	040	123	105		
	063154	124	040	105		
	063157	122	122	117		
	063162	122	040	111		
	063165	116	040	103		

	063170	123	122	000		
14728	063173	102	111	124	EM53:	.ASCIZ /BIT CLEAR ERROR IN CSR/
	063176	040	103	114		
	063201	105	101	122		
	063204	040	105	122		
	063207	122	117	122		
	063212	040	111	116		
	063215	040	103	123		
	063220	122	000			
14729	063222	111	114	114	EM55:	.ASCIZ /ILLEGAL CSR TYPE/
	063225	105	107	101		
	063230	114	040	103		
	063233	123	122	040		
	063236	124	131	120		
	063241	105	000			
14730	063243	102	101	104	EM56:	.ASCIZ /BAD PARITY TRAP GENERATED/
	063246	040	120	101		
	063251	122	111	124		
	063254	131	040	124		
	063257	122	101	120		
	063262	040	107	105		
	063265	116	105	122		
	063270	101	124	105		
	063273	104	000			
14731	063275	127	122	117	EM57:	.ASCIZ /WRONG CHECK BIT READ BACK FROM MEMORY/
	063300	116	107	040		
	063303	103	110	105		
	063306	103	113	040		
	063311	102	111	124		
	063314	040	122	105		
	063317	101	104	040		
	063322	102	101	103		
	063325	113	040	106		
	063330	122	117	115		
	063333	040	115	105		
	063336	115	117	122		
	063341	131	000			
14732	063343	127	122	117	EM60:	.ASCIZ /WRONG SYNDROME BITS READ INTO CSR/
	063346	116	107	040		
	063351	123	131	116		
	063354	104	122	117		
	063357	115	105	040		
	063362	102	111	124		
	063365	123	040	122		
	063370	105	101	104		
	063373	040	111	116		
	063376	124	117	040		
	063401	103	123	122		
	063404	000				
14733	063405	103	123	122	EM61:	.ASCIZ /CSR UPDATE ERROR/
	063410	040	125	120		
	063413	104	101	124		
	063416	105	040	105		
	063421	122	122	117		
	063424	122	000			
14734	063426	120	122	117	EM62:	.ASCIZ /PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC/
	063431	103	105	123		

063434	123	117	122
063437	040	116	117
063442	124	040	123
063445	125	120	120
063450	117	122	124
063453	105	104	040
063456	102	131	040
063461	124	110	111
063464	123	040	104
063467	111	101	107
063472	116	117	123
063475	124	111	103
063500	000		

Address	PC	DEV	ADD	GOOD	BAD/
14737					
14738	063501	040	040	120	DH1: .SBTTL ERROR DATA HEADERS (DH)
	063504	103	040	040	.ASCIZ / PC DEV ADD GOOD BAD/
	063507	040	104	105	
	063512	126	040	101	
	063515	104	104	040	
	063520	040	040	107	
	063523	117	117	104	
	063526	040	040	040	
	063531	040	102	101	
	063534	104	000		
14739	063536	040	040	120	DH2: .ASCIZ / PC GD-CC BD-CC/
	063541	103	040	040	
	063544	040	107	104	
	063547	055	103	103	
	063552	040	102	104	
	063555	055	103	103	
	063560	000			
14740	063561	040	040	120	DH3: .ASCIZ / PC 1ST ADD # OF ERRORS/
	063564	103	040	040	
	063567	040	061	123	
	063572	124	040	101	
	063575	104	104	040	
	063600	040	043	040	
	063603	117	106	040	
	063606	105	122	122	
	063611	117	122	123	
	063614	000			
14741	063615	040	040	120	DH5: .ASCIZ / PC MMR0 MMR1 MMR2 MMR3 CPUERR/
	063620	103	040	040	
	063623	040	040	040	
	063626	115	115	122	
	063631	060	040	040	
	063634	040	040	115	
	063637	115	122	061	
	063642	040	040	040	
	063645	040	115	115	
	063650	122	062	040	
	063653	040	040	040	
	063656	115	115	122	
	063661	063	040	040	
	063664	040	103	120	
	063667	125	105	122	
	063672	122	000		
14742	063674	040	040	120	DH6: .ASCIZ / PC APTPAR LSIZE APTECC MSIZE/
	063677	103	040	040	
	063702	040	101	120	
	063705	124	120	101	
	063710	122	040	040	
	063713	040	114	123	
	063716	111	132	105	
	063721	040	040	101	
	063724	120	124	105	
	063727	103	103	040	
	063732	040	115	123	
	063735	111	132	105	
	063740	000			

MESSAGES

Address	Hex	Hex	Hex	Hex	Text	Text
14764					.SBTTL	MESSAGES
14765	065123	200	040	040	MSG001: .ASCIZ	<CRLF>/
	065126	040	040	040		
	065131	040	040	040		
	065134	040	040	040		
	065137	040	040	040		
	065142	040	040	040		
	065145	040	040	040		
	065150	040	040	040		
	065153	040	115	105		
	065156	115	117	122		
	065161	131	040	103		
	065164	117	116	106		
	065167	111	107	125		
	065172	122	101	124		
	065175	111	117	116		
	065200	040	115	101		
	065203	120	000			
14766	065205	200	040	040	MSG002: .ASCIZ	<CRLF>/
	065210	040	040	040		
	065213	040	040	040		
	065216	040	040	040		
	065221	040	040	040		
	065224	040	040	040		
	065227	040	040	040		
	065232	040	040	040		
	065235	040	040	040		
	065240	040	040	040		
	065243	061	066	113		
	065246	040	127	117		
	065251	122	104	040		
	065254	102	101	116		
	065257	113	123	000		
14767	065262	200	040	040	MSG003: .ASCII	<CRLF>/
	065265	040	040	040		
	065270	040	040	040		
	065273	040	040	040		
	065276	040	040	040		
	065301	040	040	061		
	065304	040	040	040		
	065307	040	040	040		
	065312	040	062	040		
	065315	040	040	040		
	065320	040	040	040		
	065323	063				
14768	065324	040	040	040	.ASCIZ /	4 5 6 7 /
	065327	040	040	040		
	065332	040	064	040		
	065335	040	040	040		
	065340	040	040	040		
	065343	065	040	040		
	065346	040	040	040		
	065351	040	040	066		
	065354	040	040	040		
	065357	040	040	040		
	065362	040	067	040		
	065365	040	000			

MEMORY CONFIGURATION MAP/

16K WORD BANKS/

14769	065367	200	040	040	MSG004: .ASCII <CRLF>/	012345670123456701234567/
	065372	040	040	040		
	065375	040	040	040		
	065400	060	061	062		
	065403	063	064	065		
	065406	066	067	060		
	065411	061	062	063		
	065414	064	065	066		
	065417	067	060	061		
	065422	062	063	064		
	065425	065	066	067		
14770	065430	060	061	062	.ASCIZ /012345670123456701234567012345670123/	
	065433	063	064	065		
	065436	066	067	060		
	065441	061	062	063		
	065444	064	065	066		
	065447	067	060	061		
	065452	062	063	064		
	065455	065	066	067		
	065460	060	061	062		
	065463	063	064	065		
	065466	066	067	060		
	065471	061	062	063		
	065474	000				
14771	065475	200	105	122	MSG005: .ASCIZ <CRLF>/ERRORS /	
	065500	122	117	122		
	065503	123	040	040		
	065506	000				
14772	065507	200	125	116	MSG006: .ASCIZ <CRLF>/UNDEFINED TRAP INSTRUCTION/<32>	
	065512	104	105	106		
	065515	111	116	105		
	065520	104	040	124		
	065523	122	101	120		
	065526	040	111	116		
	065531	123	124	122		
	065534	125	103	124		
	065537	111	117	116		
	065542	032	000			
14773	065544	200	115	105	MSG009: .ASCIZ <CRLF>/MEMTYPE /	;MEMORY TYPE
	065547	115	124	131		
	065552	120	105	040		
	065555	000				
14774	065556	200	120	122	MSG010: .ASCIZ <CRLF>/PROTECT /	;MEMORY PROTECTED
	065561	117	124	105		
	065564	103	124	040		
	065567	000				
14775	065570	040	040	040	MSG11A: .ASCIZ /	1 1 1 1 1 1 1/
	065573	040	061	040		
	065576	040	040	040		
	065601	040	040	040		
	065604	061	040	040		
	065607	040	040	040		
	065612	040	040	061		
	065615	040	040	040		
	065620	040	040	040		
	065623	040	061	040		
	065626	040	040	040		

```

065631 040 040 040
065634 061 040 040
065637 040 040 040
065642 040 040 061
065645 040 040 040
065650 040 040 040
065653 040 061 000
14776 065656 040 040 040
065661 040 060 040
065664 040 040 040
065667 040 040 040
065672 061 040 040
065675 040 040 040
065700 040 040 062
065703 040 040 040
065706 040 040 040
065711 040 063 040
065714 040 040 040
065717 040 040 040
065722 064 040 040
065725 040 040 040
065730 040 040 065
065733 040 040 040
065736 040 040 040
065741 040 066 000
14777 065744 064 065 066
065747 067 060 061
065752 062 063 064
065755 065 066 067
065760 060 061 062
065763 063 064 065
065766 066 067 060
065771 061 062 063
065774 064 065 066
065777 067 060 061
066002 062 063 064
066005 065 066 067
066010 060 061 062
066013 063 064 065
066016 066 067 060
066021 061 062 063
066024 064 065 066
066027 067 060 061
066032 062 063 064
066035 065 066 067
066040 000
14778 066041 061 000
14779 066043 067 000
14780 066045 060 061 062
066050 063 064 065
066053 066 067 000
14781 066056 130 000
14782 066060 040 000
14783 066062 000 000
14784 066064 200 103 123
066067 122 040 040
066072 040 040 040

```

MSG011: .ASCIZ / 0 1 2 3 4 5 6/

MSG012: .ASCIZ /4567012345670123456701234567012345670123456701234567/

```

MSG11B: .ASCIZ /1/
MSG11C: .ASCIZ /7/
MSG11D: .ASCIZ /01234567/

```

```

MSG013: .ASCIZ /X/
MSG014: .ASCIZ / /
MSG015: .BYTE 0,0
MSG016: .ASCIZ <CRLF>/CSR /
;SPACE
;FOR SINGLE ASCII CHARACTERS & TERMINATOR

```


14785	066075	000						
	066076	040	040	040	MSG017: .ASCIZ / /			:8 SPACES
	066101	040	040	040				
	066104	040	040	000				
14786	066107	040	040	000	MSG018: .ASCIZ / /			:2 SPACES
14787	066112	040	040	040	MSG019: .ASCIZ / /			:3 SPACES
	066115	000						
14788	066116	200	106	123	MSG020: .ASCIZ <CRLF>/FS COMMAND MODE/			
	066121	040	103	117				
	066124	115	115	101				
	066127	116	104	040				
	066132	115	117	104				
	066135	105	000					
14789	066137	200	103	117	MSG021: .ASCII <CRLF>/COMMANDS AVAILABLE:/			
	066142	115	115	101				
	066145	116	104	123				
	066150	040	101	126				
	066153	101	111	114				
	066156	101	102	114				
	066161	105	072					
14790	066163	200	060	040	.ASCII <CRLF>/0 = EXIT/			
	066166	075	040	105				
	066171	130	111	124				
14791	066174	200	061	040	.ASCII <CRLF>/1 = READ CSR/			
	066177	075	040	122				
	066202	105	101	104				
	066205	040	103	123				
	066210	122						
14792	066211	200	062	040	.ASCII <CRLF>/2 = LOAD CSR/			
	066214	075	040	114				
	066217	117	101	104				
	066222	040	103	123				
	066225	122						
14793	066226	200	063	040	.ASCII <CRLF>/3 = EXAMINE MEMORY/			
	066231	075	040	105				
	066234	130	101	115				
	066237	111	116	105				
	066242	040	115	105				
	066245	115	117	122				
	066250	131						
14794	066251	200	064	040	.ASCII <CRLF>/4 = MODIFY MEMORY/			
	066254	075	040	115				
	066257	117	104	111				
	066262	106	131	040				
	066265	115	105	115				
	066270	117	122	131				
14795	066273	200	065	040	.ASCII <CRLF>/5 = SELECT BANK & TEST/			
	066276	075	040	123				
	066301	105	114	105				
	066304	103	124	040				
	066307	102	101	116				
	066312	113	040	046				
	066315	040	124	105				
	066320	123	124					
14796	066322	200	066	040	.ASCII <CRLF>/6 = TYPE CONFIG MAP/			
	066325	075	040	124				
	066330	131	120	105				

	066333	040	103	117	
	066336	116	106	111	
	066341	107	040	115	
	066344	101	120		
14797	066346	200	067	040	.ASCII <CRLF>/7 = SOB-A-LONG TEST/
	066351	075	040	123	
	066354	117	102	055	
	066357	101	055	114	
	066362	117	116	107	
	066365	040	124	105	
	066370	123	124		
14798	066372	200	070	040	.ASCII <CRLF>/8 = ERROR SUMMARY/
	066375	075	040	105	
	066400	122	122	117	
	066403	122	040	123	
	066406	125	115	115	
14799	066411	101	122	131	.ASCII <CRLF>/9= REFRESH TEST/
	066414	200	071	075	
	066417	040	040	122	
	066422	105	106	122	
	066425	105	123	110	
	066430	040	124	105	
	066433	123	124		
14800	066435	200	061	060	.ASCII <CRLF>/10= SET FILL COUNT/
	066440	075	040	123	
	066443	105	124	040	
	066446	106	111	114	
	066451	114	040	103	
	066454	117	125	116	
	066457	124			
14801	066460	200	061	061	.ASCII <CRLF>/11= ENTER KAMIKAZE MODE/
	066463	075	040	105	
	066466	116	124	105	
	066471	122	040	113	
	066474	101	115	111	
	066477	113	101	132	
	066502	105	040	115	
	066505	117	104	105	
14802	066510	200	061	062	.ASCII <CRLF>/12= EXIT KAMIKAZE MODE/
	066513	075	040	105	
	066516	130	111	124	
	066521	040	113	101	
	066524	115	111	113	
	066527	101	132	105	
	066532	040	115	117	
	066535	104	105		
14803	066537	200	061	063	.ASCII <CRLF>/13= TURN CACHE OFF/
	066542	075	040	124	
	066545	125	122	116	
	066550	040	103	101	
	066553	103	110	105	
	066556	040	117	106	
	066561	106			
14804	066562	200	061	064	.ASCII <CRLF>/14= TURN CACHE ON/
	066565	075	040	124	
	066570	125	122	116	
	066573	040	103	101	

	066576	103	110	105	
	066601	040	117	116	
14805	066604	200	061	065	.ASCII <CRLF>/15= TEST SELECTED BANKS/
	066607	075	040	124	
	066612	105	123	124	
	066615	040	123	105	
	066620	114	105	103	
	066623	124	105	104	
	066626	040	102	101	
	066631	116	113	123	
14806	066634	200	061	066	.ASCII <CRLF>/16= TEST ALL BANKS/
	066637	075	040	124	
	066642	105	123	124	
	066645	040	101	114	
	066650	114	040	102	
	066653	101	116	113	
	066656	123			
14807	066657	200	061	067	.ASCII <CRLF>/17= ENABLE TRACE/
	066662	075	040	105	
	066665	116	101	102	
	066670	114	105	040	
	066673	124	122	101	
	066676	103	105		
14808	066700	200	061	070	.ASCII <CRLF>/18= DISABLE TRACE/
	066703	075	040	104	
	066706	111	123	101	
	066711	102	114	105	
	066714	040	124	122	
	066717	101	103	105	
14809	066722	015	012	000	.BYTE 15,12,0
14810	066725	200	127	110	MSG022: .ASCIZ <CRLF>/WHICH CSR(0-F)? /
	066730	111	103	110	
	066733	040	103	123	
	066736	122	050	060	
	066741	055	106	051	
	066744	077	040	000	
14811	066747	200	103	123	MSG023: .ASCIZ <CRLF>/CSR WORD? /
	066752	122	040	127	
	066755	117	122	104	
	066760	077	040	000	
14812	066763	200	103	123	MSG025: .ASCIZ <CRLF>/CSR DOES NOT EXIST/
	066766	122	040	104	
	066771	117	105	123	
	066774	040	116	117	
	066777	124	040	105	
	067002	130	111	123	
	067005	124	000		
14813	067007	200	103	117	MSG026: .ASCIZ <CRLF>/COMMAND:/
	067012	115	115	101	
	067015	116	104	072	
	067020	000			
14814	067021	200	117	114	MSG027: .ASCIZ <CRLF>/OLD CSR WAS/
	067024	104	040	103	
	067027	123	122	040	
	067032	127	101	123	
	067035	000			
14815	067036	200	103	123	MSG028: .ASCIZ <CRLF>/CSR IS NOW/

MESSAGES

	067041	122	040	111	
	067044	123	040	116	
	067047	117	127	000	
14816	067052	200	105	130	MSG029: .ASCIZ <CRLF>/EXAMINE MEMORY/
	067055	101	115	111	
	067060	116	105	040	
	067063	115	105	115	
	067066	117	122	131	
	067071	000			
14817	067072	200	102	101	MSG030: .ASCIZ <CRLF>/BANK(0-177)? /
	067075	116	113	050	
	067100	060	055	061	
	067103	067	067	051	
	067106	077	040	000	
14818	067111	200	120	110	MSG031: .ASCIZ <CRLF>/PHYSICAL ADDRESS(0-17757776)? /
	067114	131	123	111	
	067117	103	101	114	
	067122	040	101	104	
	067125	104	122	105	
	067130	123	123	050	
	067133	060	055	061	
	067136	067	067	065	
	067141	067	067	067	
	067144	066	051	077	
	067147	040	000		
14819	067151	200	120	101	MSG032: .ASCIZ <CRLF>/PARITY ABORT/<32>
	067154	122	111	124	
	067157	131	040	101	
	067162	102	117	122	
	067165	124	032	000	
14820	067170	200	124	111	MSG033: .ASCIZ <CRLF>/TIMEOUT TRAP/<32>
	067173	115	105	117	
	067176	125	124	040	
	067201	124	122	101	
	067204	120	032	000	
14821	067207	200	102	131	MSG034: .ASCIZ <CRLF>/BYPASSING ECC TESTS ON BANK /
	067212	120	101	123	
	067215	123	111	116	
	067220	107	040	105	
	067223	103	103	040	
	067226	124	105	123	
	067231	124	123	040	
	067234	117	116	040	
	067237	102	101	116	
	067242	113	040	000	
14822	067245	040	104	125	MSG034: .ASCIZ / DUE TO SBE LOCATIONS/
	067250	105	040	124	
	067253	117	040	123	
	067256	102	105	040	
	067261	114	117	103	
	067264	101	124	111	
	067267	117	116	123	
	067272	000			
14823	067273	121	126	000	MSG035: .ASCIZ /QV/
14824	067276	200	115	117	MSG036: .ASCIZ <CRLF>/MODIFY MEMORY/
	067301	104	111	106	
	067304	131	040	115	

	067307	105	115	117	
	067312	122	131	000	
14825	067315	200	117	114	MSG037: .ASCIZ <CRLF>/OLD DATA WAS /
	067320	104	040	104	
	067323	101	124	101	
	067326	040	127	101	
	067331	123	040	000	
14826	067334	200	104	101	MSG038: .ASCIZ <CRLF>/DATA IS NOW /
	067337	124	101	040	
	067342	111	123	040	
	067345	116	117	127	
	067350	040	000		
14827	067352	200	111	116	MSG039: .ASCIZ <CRLF>/INPUT NEW DATA? /
	067355	120	125	124	
	067360	040	116	105	
	067363	127	040	104	
	067366	101	124	101	
	067371	077	040	000	
14828	067374	200	123	105	MSG040: .ASCIZ <CRLF>/SELECT BANK & TEST/
	067377	114	105	103	
	067402	124	040	102	
	067405	101	116	113	
	067410	040	046	040	
	067413	124	105	123	
	067416	124	000		
14829	067420	200	102	101	MSG041: .ASCIZ <CRLF>/BANK NOT ACCESSABLE/
	067423	116	113	040	
	067426	116	117	124	
	067431	040	101	103	
	067434	103	105	123	
	067437	123	101	102	
	067442	114	105	000	
14830	067445	200	124	105	MSG042: .ASCIZ <CRLF>/TEST(0-47)? /
	067450	123	124	050	
	067453	060	055	064	
	067456	067	051	077	
	067461	040	000		
14831	067463	200	124	105	MSG043: .ASCIZ <CRLF>/TEST 0 DATA IS? /
	067466	123	124	040	
	067471	060	040	104	
	067474	101	124	101	
	067477	040	111	123	
	067502	077	040	000	
14832	067505	200	124	117	MSG046: .ASCIZ <CRLF>/TO ESCAPE TYPE ANY KEY/<CRLF><12><12>
	067510	040	105	123	
	067513	103	101	120	
	067516	105	040	124	
	067521	131	120	105	
	067524	040	101	116	
	067527	131	040	113	
	067532	105	131	200	
	067535	012	012	000	
14833	067540	200	124	105	MSG047: .ASCIZ <CRLF>/TEST COMPLETE/
	067543	123	124	040	
	067546	103	117	115	
	067551	120	114	105	
	067554	124	105	000	

14834	067557	040	116	117	MSG048: .ASCIZ / NOT AVAILABLE NOW - TRY LATER!/ /
	067562	124	040	101	
	067565	126	101	111	
	067570	114	101	102	
	067573	114	105	040	
	067576	116	117	127	
	067601	040	055	040	
	067604	124	122	131	
	067607	040	114	101	
	067612	124	105	122	
	067615	041	000		
14835	067617	200	102	101	MSG049: .ASCIZ <CRLF>/BANK REQUIRES RELOCATION/ /
	067622	116	113	040	
	067625	122	105	121	
	067630	125	111	122	
	067633	105	123	040	
	067636	122	105	114	
	067641	117	103	101	
	067644	124	111	117	
	067647	116	000		
14836					.EVEN
14837	067652	120	117	127	MSG051: .ASCIZ /POWER RECOVERY/ /
	067655	105	122	040	
	067660	122	105	103	
	067663	117	126	105	
	067666	122	131	000	
14838	067671	200	123	117	MSG055: .ASCIZ <CRLF>/SOB-A-LONG TEST/ /
	067674	102	055	101	
	067677	055	114	117	
	067702	116	107	040	
	067705	124	105	123	
	067710	124	000		
14839	067712	200	102	105	MSG056: .ASCIZ <CRLF>/BELL = EACH PASS COMPLETE/ /
	067715	114	114	040	
	067720	075	040	105	
	067723	101	103	110	
	067726	040	120	101	
	067731	123	123	040	
	067734	103	117	115	
	067737	120	114	105	
	067742	124	105	000	
14840	067745	200	040	040	MSG058: .ASCIZ <CRLF>/ CSR CSR .../ /
	067750	103	123	122	
	067753	040	040	040	
	067756	040	103	123	
	067761	122	040	056	
	067764	056	056	000	
14841	067767	077	077	077	MSG061: .ASCIZ /??????/ /
	067772	077	077	077	
	067775	000			
14842	067776	111	116	120	MSG062: .ASCIZ /INPUT MUST BE A/ /
	070001	125	124	040	
	070004	115	125	123	
	070007	124	040	102	
	070012	105	040	101	
	070015	000			
14843	070016	116	040	117	MSG063: .ASCIZ /N OCTAL / /

	070021	103	124	101	
	070024	114	040	000	
14844	070027	116	125	115	MSG064: .ASCIZ /NUMBER/<CRLF>
	070032	102	105	122	
	070035	200	000		
14845	070037	040	104	105	MSG065: .ASCIZ / DECIMAL /
	070042	103	111	115	
	070045	101	114	040	
	070050	000			
14846	070051	200	105	122	MSG066: .ASCIZ <CRLF>/ERRORS > 20 - ABORTING FOR XXDP CHAIN/
	070054	122	117	122	
	070057	123	040	076	
	070062	040	062	060	
	070065	040	055	040	
	070070	101	102	117	
	070073	122	124	111	
	070076	116	107	040	
	070101	106	117	122	
	070104	040	130	130	
	070107	104	120	040	
	070112	103	110	101	
	070115	111	116	000	
14847	070120	106	101	124	MSG067: .ASCIZ /FATAL /
	070123	101	114	040	
	070126	000			
14848	070127	113	040	127	MSG070: .ASCIZ /K WORDS OF MEMORY TOTAL/<CRLF>
	070132	117	122	104	
	070135	123	040	117	
	070140	106	040	115	
	070143	105	115	117	
	070146	122	131	040	
	070151	124	117	124	
	070154	101	114	200	
	070157	000			
14849	070160	200	122	105	MSG073: .ASCIZ <CRLF>/REFRESH TEST/
	070163	106	122	105	
	070166	123	110	040	
	070171	124	105	123	
	070174	124	000		
14850	070176	200	122	105	MSG075: .ASCIZ <CRLF>/RELOCATION NOT POSSIBLE/<32>
	070201	114	117	103	
	070204	101	124	111	
	070207	117	116	040	
	070212	116	117	124	
	070215	040	120	117	
	070220	123	123	111	
	070223	102	114	105	
	070226	032	000		
14851	070230	200	040	040	MSG076: .ASCIZ <CRLF>/ BANK ERRORS/<CRLF>
	070233	102	101	116	
	070236	113	040	040	
	070241	105	122	122	
	070244	117	122	123	
	070247	200	000		
14852	070251	200	105	116	MSG077: .ASCIZ <CRLF>/END PASS #/
	070254	104	040	120	
	070257	101	123	123	

14853	070262	040	043	000	
	070265	040	105	122	MSG079: .ASCIZ / ERROR(S) DETECTED/<CRLF>
	070270	122	117	122	
	070273	050	123	051	
	070276	040	104	105	
	070301	124	105	103	
	070304	124	105	104	
	070307	200	000		
14854	070311	200	106	111	MSG085: .ASCIZ <CRLF>/FILL COUNT(OCTAL)? /
	070314	114	114	040	
	070317	103	117	125	
	070322	116	124	050	
	070325	117	103	124	
	070330	101	114	051	
	070333	077	040	000	
14855	070336	200	113	105	MSG088: .ASCIZ <CRLF>/KERNEL STACK/
	070341	122	116	105	
	070344	114	040	123	
	070347	124	101	103	
	070352	113	000		
14856	070354	200	123	125	MSG089: .ASCIZ <CRLF>/SUPERVISOR STACK/
	070357	120	105	122	
	070362	126	111	123	
	070365	117	122	040	
	070370	123	124	101	
	070373	103	113	000	
14857	070376	200	125	123	MSG090: .ASCIZ <CRLF>/USER STACK/
	070401	105	122	040	
	070404	123	124	101	
	070407	103	113	000	
14858	070412	040	111	123	MSG091: .ASCIZ / IS EMPTY/
	070415	040	105	115	
	070420	120	124	131	
	070423	000			
14859	070424	122	105	114	MSG092: .ASCIZ /RELOCATED /
	070427	117	103	101	
	070432	124	105	104	
	070435	040	040	000	
14860	070440	102	101	116	MSG093: .ASCIZ /BANK=/
	070443	113	075	000	
14861	070446	040	040	124	MSG095: .ASCIZ / TEST=/
	070451	105	123	124	
	070454	075	000		
14862	070456	200	105	116	MSG101: .ASCIZ <CRLF>/ENTERING KAMIKAZE MODE/
	070461	124	105	122	
	070464	111	116	107	
	070467	040	113	101	
	070472	115	111	113	
	070475	101	132	105	
	070500	040	115	117	
	070503	104	105	000	
14863	070506	200	114	105	MSG102: .ASCIZ <CRLF>/LEAVING KAMIKAZE MODE/
	070511	101	126	111	
	070514	116	107	040	
	070517	113	101	115	
	070522	111	113	101	
	070525	132	105	040	

	070530	115	117	104	
	070533	105	000		
14864	070535	200	114	105	MSG103: .ASCIZ <CRLF>/LEAVING FS MODE/<CRLF>
	070540	101	126	111	
	070543	116	107	040	
	070546	106	123	040	
	070551	115	117	104	
	070554	105	200	000	
14865	070557	032	000		MSG104: .BYTE 32,0 ;CONTROL Z
14866	070561	200	105	116	MSG105: .ASCIZ <CRLF>/ENTER BANKS - USE NUMBER 200 TO TERMINATE/
	070564	124	105	122	
	070567	040	102	101	
	070572	116	113	123	
	070575	040	055	040	
	070600	125	123	105	
	070603	040	116	125	
	070606	115	102	105	
	070611	122	040	062	
	070614	060	060	040	
	070617	124	117	040	
	070622	124	105	122	
	070625	115	111	116	
	070630	101	124	105	
	070633	000			
14867	070634	200	103	101	MSG106: .ASCIZ <CRLF>/CACHE IS OFF/
	070637	103	110	105	
	070642	040	111	123	
	070645	040	117	106	
	070650	106	000		
14868	070652	200	103	101	MSG107: .ASCIZ <CRLF>/CACHE IS ON (EXCEPT DURING ACTUAL PATTERNS)/
	070655	103	110	105	
	070660	040	111	123	
	070663	040	117	116	
	070666	040	050	105	
	070671	130	103	105	
	070674	120	124	040	
	070677	104	125	122	
	070702	111	116	107	
	070705	040	101	103	
	070710	124	125	101	
	070713	114	040	120	
	070716	101	124	124	
	070721	105	122	116	
	070724	123	051	000	
14869	070727	200	117	116	MSG110: .ASCIZ <CRLF>/ONLY SELECTED BANKS WILL BE TESTED/
	070732	114	131	040	
	070735	123	105	114	
	070740	105	103	124	
	070743	105	104	040	
	070746	102	101	116	
	070751	113	123	040	
	070754	127	111	114	
	070757	114	040	102	
	070762	105	040	124	
	070765	105	123	124	
	070770	105	104	000	
14870	070773	200	101	114	MSG111: .ASCIZ <CRLF>/ALL BANKS WILL BE TESTED/

	070776	114	040	102	
	071001	101	116	113	
	071004	123	040	127	
	071007	111	114	114	
	071012	040	102	105	
	071015	040	124	105	
	071020	123	124	105	
	071023	104	000		
14871	071025	113	040	117	MSG112: .ASCIZ /K OF Q-BUS PARITY MEMORY/<CRLF>
	071030	106	040	121	
	071033	055	102	125	
	071036	123	040	120	
	071041	101	122	111	
	071044	124	131	040	
	071047	115	105	115	
	071052	117	122	131	
	071055	200	000		
14872	071057	113	040	117	MSG113: .ASCIZ /K OF ECC MEMORY/<CRLF>
	071062	106	040	105	
	071065	103	103	040	
	071070	115	105	115	
	071073	117	122	131	
	071076	200	000		
14873	071100	200	040	040	M1184: .ASCIZ <CRLF>" 11/84"
	071103	040	061	061	
	071106	057	070	064	
	071111	000			
14874	071112	200	040	040	MSG117: .ASCIZ <CRLF>" 11/83"
	071115	040	061	061	
	071120	057	070	063	
	071123	000			
14875	071124	200	040	040	MSG119: .ASCIZ <CRLF>/ NO/
	071127	040	116	117	
	071132	000			
14876	071133	040	103	101	MSG120: .ASCIZ / CACHE AVAILABLE/
	071136	103	110	105	
	071141	040	101	126	
	071144	101	111	114	
	071147	101	102	114	
	071152	105	000		
14877	071154	040	103	101	MSG121: .ASCIZ / CACHE BYPASSED/
	071157	103	110	105	
	071162	040	102	131	
	071165	120	101	123	
	071170	123	105	104	
	071173	000			
14878	071174	200	103	123	MSG122: .ASCII <CRLF>/CSR NUMBER /
	071177	122	040	116	
	071202	125	115	102	
	071205	105	122	040	
14879	071210	000			MSG122: .BYTE 0
14880	071211	040	103	117	.ASCIZ / CONTROLS TOO MANY BANKS/
	071214	116	124	122	
	071217	117	114	123	
	071222	040	124	117	
	071225	117	040	115	
	071230	101	116	131	

	071233	040	102	101	
	071236	116	113	123	
	071241	000			
14881	071242	040	120	101	MSG125: .ASCIZ / PASSES COMPLETED/
	071245	123	123	105	
	071250	123	040	103	
	071253	117	115	120	
	071256	114	105	124	
	071261	105	104	000	
14882	071264	200	120	122	MSG126: .ASCIZ <CRLF>/PROGRAM CSR COULD NOT BE DETERMINED/
	071267	117	107	122	
	071272	101	115	040	
	071275	103	123	122	
	071300	040	103	117	
	071303	125	114	104	
	071306	040	116	117	
	071311	124	040	102	
	071314	105	040	104	
	071317	105	124	105	
	071322	122	115	111	
	071325	116	105	104	
	071330	000			
14883	071331	200	124	122	MSG127: .ASCIZ <CRLF>/TRACE ENABLED/
	071334	101	103	105	
	071337	040	105	116	
	071342	101	102	114	
	071345	105	104	000	
14884	071350	200	124	122	MSG128: .ASCIZ <CRLF>/TRACE DISABLED/
	071353	101	103	105	
	071356	040	104	111	
	071361	123	101	102	
	071364	114	105	104	
	071367	000			
14885	071370	200	200	040	MSG008: .ASCIZ <CRLF><CRLF>/ CSR MAP/<CRLF>
	071373	040	040	040	
	071376	040	040	040	
	071401	040	040	040	
	071404	040	040	040	
	071407	040	040	103	
	071412	123	122	040	
	071415	115	101	120	
	071420	200	000		
14886	071422	200	040	103	MSG000: .ASCIZ <CRLF>" CVMJAA0 ECC/PARITY MEMORY DIAGNOSTIC"
	071425	126	115	112	
	071430	101	060	040	
	071433	040	105	103	
	071436	103	057	120	
	071441	101	122	111	
	071444	124	131	040	
	071447	115	105	115	
	071452	117	122	131	
	071455	040	104	111	
	071460	101	107	116	
	071463	117	123	124	
	071466	111	103	000	
14887	071471	200	200	000	MSG129: .ASCIZ <CRLF><CRLF>
14888	071474	120	122	117	MSG130: .ASCIZ /PROCESSOR NOT SUPPORTED BY THIS DIAGNOSTIC/

	071477	103	105	123
	071502	123	117	122
	071505	040	116	117
	071510	124	040	123
	071513	125	120	120
	071516	117	122	124
	071521	105	104	040
	071524	102	131	040
	071527	124	110	111
	071532	123	040	104
	071535	111	101	107
	071540	116	117	123
	071543	124	111	103
	071546	000		
14889	071547	200	125	116
	071552	111	102	125
	071555	123	040	115
	071560	105	115	117
	071563	122	131	040
	071566	127	111	114
	071571	114	040	116
	071574	117	124	040
	071577	102	105	040
	071602	124	105	123
	071605	124	105	104
	071610	200	000	

NOUBMT: .ASCIZ <CRLF>/UNIBUS MEMORY WILL NOT BE TESTED/<CRLF>

14890
14896 071612
14897 071612
14898 071612 010322
14902 000200

END: .EVEN
\$\$END
.PRINT 60000-SUPLIMIT ;SUPERVISOR ADDRESSES LEFT
.END START3

ABORT	000000	B107	054002	B61	035112	CPUERR=	177766	DH13	064172
ABORTF	002144	B108	053522	B62	035116	CR	= 000015	DH14	064267
ABORT1	000032	B109	053526	B63	035322	CRLF	= 000200	DH15	064346
ABORT2	000046	B110	053716	B64	035326	CSR	002150	DH16	064465
ACFLAG	002116	B111	054002	B65	036444	CSRADD=	172100	DH19	064502
ACTFLA	002350	B112	013010	B66	036452	CSRBMP	006024	DH2	063536
ADDRES	002034	B113	013060	B67	036602	CSRCAS	017004	DH20	064507
ANA2	007372	B114	013122	B7	012220	CSRFBFA	002232	DH23	064566
APTDOW	040720	B115	013122	B70	036616	CSRFBIR	002226	DH24	064645
APTECC	002424	B116	013400	B71	043510	CSRHOL	002526	DH25	064714
APTFLA	002352	B117	015610	B72	043746	CSRINC	002330	DH26	064752
APTHAN	014126	B2	006050	B73	044230	CSRINF	002462	DH27	064770
APTHLT	040766	B20	016066	B74	044524	CSRINT	002236	DH3	063561
APTPAR	002422	B21	016104	B75	044524	CSRLAS	002230	DH30	065044
APTSIZ	002444	B22	016146	B76	044622	CSRLBA	002234	DH5	063615
BACK	051530	B23	016220	B77	045562	CSRL00	002332	DH6	063674
BACKGN	032630	B24	021642	CACHKF	002550	CSRMAP	006034	DH7	063741
BAD	002052	B25	021646	CACHKN	002544	CSRNO	002152	DIAGFL	002004
BADPC	002022	B26	022026	CACHOF=	104424	CSR0UT	035314	DISPLA	002640
BADPSW	002032	B27	022034	CACHON=	104423	CSRSTU=	000021	DISPRE=	000174
BADSP	002026	B3	006120	CACHVE=	000114	CSR1S	002322	DISPTB	013600
BADSTA	034152	B30	022264	CBCSR =	104474	CTEST	006714	DOBACK	014162
BADTOR	002060	B31	025750	CBITS	002316	CTLKVE	002146	DONE	006700
BAD2	002054	B32	026140	CBREG =	104513	DATARG=	177754	DSWR	= 177570
BAD3	002056	B33	026176	CB1CSR=	104475	DATBUF	002242	DT1	060144
BAFPAF	014262	B34	026620	CHECK	002314	DBEMSK	002256	DT10	060264
BAFPAR	014370	B35	026656	CHKDIS=	104504	DDISP =	177570	DT11	060306
BAKPAT	002632	B36	026672	CHKTRP	032220	DEENER=	104421	DT12	060314
BANK	002102	B37	027012	CHK1DI=	104505	DETAIL	053266	DT13	060320
BANKIN	002104	B4	006400	CKEND	055056	DETFLA	002220	DT14	060342
BANKMO	037442	B40	027172	CKSWR =	104410	DETPSW	002216	DT16	060360
BANKOK	040456	B41	027214	CLRCR=	104502	DETRO	002200	DT17	060410
BAWPAF	014476	B42	027742	CLREX	007342	DETR0	002202	DT2	060156
BAWPAR	014626	B43	030102	CLRMEM	007232	DETR1	002202	DT20	060416
BGTEST	027610	B44	030140	CLR1CS=	104503	DETR2	002204	DT22	060426
BITNO	002324	B45	030432	CMD16A	044610	DETR3	002206	DT23	060434
BIT0 =	000001	B46	030646	CMD16L=	000074	DETR4	002210	DT24	060456
BIT1 =	000002	B47	031050	CMD5B	042752	DETR5	002212	DT25	060476
BIT10 =	002000	B5	006434	CMD5C	043246	DET1	053756	DT26	060510
BIT11 =	004000	B50	031224	CMD7B	043504	DF1	060556	DT27	060516
BIT12 =	010000	B51	031244	CMD7C	043560	DF11	060701	DT3	060162
BIT13 =	020000	B52	031254	CMD9B	044224	DF13	060712	DT30	060536
BIT14 =	040000	B53	031572	CMD9C	044300	DF14	060722	DT4	060172
BIT15 =	100000	B54	031742	CONFGE	002450	DF15	060731	DT5	060202
BIT2 =	000004	B55	031756	CONFIE	003670	DF16	060740	DT6	060220
BIT3 =	000010	B56	032022	CONFIG	002664	DF2	060557	DT7	060234
BIT4 =	000020	B57	034746	CONTFI	002222	DF3	060603	DUMMY	002176
BIT5 =	000040	B6	011544	CONTRL=	177746	DF4	060616	DUMPCS=	000103
BIT6 =	000100	B60	034752	CONTS	055226	DF5	060631	ECCDIS=	104470
BIT7 =	000200			CONTS1	054570	DF6	060644	ECCINI=	104472
BIT8 =	000400			CONTS2	055230	DF7	060657	ECC1DI=	104471
BIT9 =	001000			CONTS3	047340	DF8	060672	ECC1IN=	104473
BLOCK1	040770			CONTT	055152	DF9	060674	EMTSV	003670
BLOCK2	041010			COUNT	002370	DH1	063501	EMTVEC=	000030
BLOCK3	041024			CPERRF	052314	DH10	064051	EM11	061247
BMFLAG	002130			CPSAVE	052312	DH11	064147	EM12	061271
BRGOBB	027612			CPUBIT	002106	DH12	064165	EM13	061315

EM14	061347	E0	004700	E62	035164	GTSWR =	104407	LOADBA	002432
EM15	061413	E1	005776	E63	035356	HEADER	002612	LOADCS=	104425
EM17	061461	E10	012346	E64	035356	HIPAT	040510	LOADER=	000065
EM19	061521	E100	045622	E65	036602	HOLDLO=	000016	LOADHO	002576
EM2	060747	E101	046102	E66	036562	HT	= 000011	LOOP	013552
EM20	061573	E102	046102	E67	037016	I	002452	LSIZE	002400
EM21	061652	E103	053570	E7	012362	IBSAVE	052310	LST\$\$ =	000000
EM22	061706	E104	053570	E70	036730	IIII =	177777	LWDBE =	000061
EM23	061733	E105	053654	E71	043552	ILLCSR	012724	LWSBE =	000057
EM24	061762	E106	053750	E72	044050	IMPTES	011750	L0	004662
EM25	062041	E107	054034	E73	044272	INCBNK	040520	L1	004724
EM26	062066	E11	013052	E74	044564	INCPAT	040474	L10	005222
EM27	062137	E12	013120	E75	044564	INCRPT	040474	L100	012334
EM29	062227	E13	013150	E76	044640	INHBAN	002540	L101	012340
EM3	061005	E14	013520	E77	045622	INHECC	002536	L102	013042
EM30	062311	E15	014260	FASTCI=	177640	INTFLA	002136	L103	013036
EM32	062421	E16	015630	FATAL\$	002064	INT64K	002140	L104	013042
EM33	062526	E17	016414	FCMD10	044364	INVALI=	104511	L105	013136
EM35	062634	E2	006114	FCMD11	044412	IOTVEC=	000020	L106	013206
EM36	062721	E20	016374	FCMD12	044434	JMPRL1	037076	L107	013352
EM4	061037	E21	016324	FCMD13	044454	KAMIKA	002006	L11	005230
EM40	062770	E22	016324	FCMD14	044476	KAMITE	024234	L110	013552
EM5	061105	E23	016274	FCMD15	044514	KDIAG =	000010	L111	013510
EM50	063042	E24	022004	FCMD16	044600	KDPAR0=	172360	L112	013426
EM51	063076	E25	021770	FCMD17	044642	KDPAR6=	172374	L113	013430
EM52	063146	E26	022172	FCMD18	044656	KDPAR7=	172376	L114	013470
EM53	063173	E27	022156	FIELDS	041054	KERNEL =	104417	L115	013504
EM55	063222	E3	006210	FINDBA=	000067	KERSTK=	002000	L116	013550
EM56	063243	E30	022342	FINT\$	007214	KFLAG	002530	L117	013552
EM57	063275	E31	020034	FIRST =	060000	KIPAR0=	172340	L12	005300
EM6	061162	E32	026414	FLIPLO	002616	KIPAR4=	172350	L120	013666
EM60	063343	E33	026402	FLIPWA	032500	KIPAR5=	172352	L121	013776
EM61	063405	E34	027164	FLUSH	013672	KIPAR6=	172354	L122	014006
EM62	063426	E35	027150	FSCMD0	041252	KIPDR0=	172300	L123	014156
EM7	061207	E36	026764	FSCMD1	041354	KMAP =	104422	L124	014156
ENASBE=	104506	E37	027134	FSCMD2	041464	KPFLAG	002114	L125	014244
ENA1SB=	104507	E4	006622	FSCMD3	041632	KSIZE	002376	L126	014356
END	071612	E40	027210	FSCMD4	042106	KSTACK	002574	L127	014464
ENDADD	002562	E41	027234	FSCMD5	042426	LAST =	157776	L13	005274
ENDFLG	002564	E42	030064	FSCMD6	043344	LASTBA	002556	L130	014614
ENERGI=	104420	E43	030270	FSCMD7	043352	LASTBL	002560	L131	014744
ENEXBK	040446	E44	030256	FSCMD8	043644	LASTER	002016	L132	015074
EQFLAG	002132	E45	030536	FSCMD9	044072	LBLS0 =	000530	L133	015246
ERRADD	002460	E46	031022	FSINFL	002442	LBLS1 =	000107	L134	015376
ERRGEN=	104512	E47	031220	FSPAT	043126	LBLS2 =	000520	L135	015550
ERRMAX	002554	E5	006612	FSSTAC	002306	LBLS3 =	000513	L136	015630
ERROR =	104000	E50	031542	FS1	041140	LBLS4 =	000366	L137	015666
ERRPC	002020	E51	031524	FS7FLA	002446	LBLS5 =	000370	L14	005300
ERRPSW	002030	E52	031510	FULLRE	002542	LBLS6 =	000023	L140	015676
ERRSP	002024	E53	031726	GBLENG=	000076	LCSR0U=	000063	L141	016324
ERRVEC=	000004	E54	032212	GETCSR	032674	LCSRRE=	000101	L142	016306
EVEN	002364	E55	032202	GETDAT	045252	LCSRSA=	000077	L143	016356
EXBANK	040052	E56	032170	GETDA1	045350	LEGALC=	000010	L144	016626
EXCMD3	042040	E57	035020	GETDIS	051430	LF =	000012	L145	016736
EXCMD4	042360	E6	011676	GOOD	002044	LINK1	002522	L146	017074
EXIT	040604	E60	035020	GOOD2	002046	LINK2	002524	L147	017122
EXIT2	040610	E61	035164	GOOD3	002050	LKS =	177546	L15	005610

L150	017126	L237	026764	L334	033434	L416	043044	L501	051604
L151	017130	L24	006156	L335	033530	L417	043060	L502	052044
L152	017174	L240	026714	L336	033520	L42	006552	L503	052000
L153	017230	L241	026726	L337	033526	L420	043072	L504	052024
L154	017234	L242	026744	L34	006374	L421	043422	L505	052044
L155	017236	L243	026752	L340	033536	L422	043424	L506	052064
L156	017404	L244	026776	L341	033562	L423	043536	L507	052256
L157	017764	L245	027134	L342	033664	L424	044050	L51	006700
L16	005610	L246	027034	L343	034532	L425	044034	L510	052126
L160	020370	L247	027046	L344	034540	L426	044004	L511	052254
L161	020400	L25	006146	L345	034776	L427	044142	L512	052204
L162	020400	L250	027100	L346	034776	L43	006554	L513	052216
L163	020472	L251	027064	L347	035040	L430	044144	L514	052254
L164	020502	L252	027076	L35	006430	L431	044256	L515	052264
L165	020502	L253	027122	L350	035046	L432	044546	L516	052620
L166	021036	L254	027110	L351	035142	L433	044550	L517	053546
L167	021070	L255	027122	L352	035142	L434	045014	L52	010636
L17	005724	L256	027234	L353	035204	L435	045030	L520	053752
L170	021134	L257	030134	L354	035212	L436	045034	L521	053756
L171	021754	L26	006154	L355	035334	L437	045052	L522	054036
L172	021732	L260	030140	L356	036416	L44	006576	L523	054042
L173	021754	L261	030224	L357	036432	L440	045200	L524	055044
L174	022020	L262	030242	L36	006434	L441	045202	L525	055050
L175	022142	L263	030246	L360	036444	L442	045246	L526	055172
L176	022120	L266	030364	L361	036444	L443	045472	L527	055262
L177	022142	L267	030522	L362	036546	L444	045600	L53	010642
L2	005032	L27	006164	L363	036576	L445	046060	L530	055264
L20	005752	L271	030570	L364	036720	L446	047376	L54	010644
L200	022326	L272	030576	L365	036720	L447	047400	L55	011374
L201	022326	L273	030722	L366	036720	L45	006602	L56	011410
L202	022434	L274	030772	L367	036700	L450	050230	L57	011512
L203	022406	L276	031132	L37	006464	L451	050242	L6	005230
L204	022464	L277	031202	L370	036720	L452	050260	L60	011516
L205	022654	L3	005054	L371	036762	L453	050262	L61	011662
L206	023170	L301	031354	L372	036762	L454	050302	L62	011662
L207	023424	L302	031430	L373	037012	L455	050314	L63	011662
L21	005770	L303	031446	L374	040204	L456	050332	L64	011662
L210	023444	L304	031474	L375	040374	L457	050334	L65	011656
L211	023544	L31	006262	L376	040556	L460	050350	L66	011734
L212	023724	L310	031636	L377	040624	L461	050552	L67	011740
L213	023732	L311	031700	L4	005072	L462	050560	L7	005230
L214	024070	L313	032016	L40	006510	L463	050606	L70	011742
L215	024076	L314	032022	L400	040630	L464	050620	L71	012124
L216	024256	L315	032074	L401	040636	L465	050636	L72	012124
L217	024264	L316	032076	L402	040652	L466	050650	L73	012124
L220	024270	L317	032134	L403	040664	L467	050662	L74	012130
L221	025704	L32	006266	L404	041076	L470	050704	L75	012346
L222	025742	L320	032154	L405	041106	L471	050752	L76	012346
L223	026022	L321	032160	L406	041242	L472	051032	L77	012336
L225	026172	L325	032252	L407	041302	L473	051046	MAINT =	177750
L226	026176	L326	032250	L41	006562	L474	051050	MAPHO =	170202
L227	026272	L327	032272	L410	041306	L475	051164	MAPKER	037740
L230	026350	L33	006332	L411	041336	L476	051254	MAPLO =	170200
L231	026366	L330	032272	L412	041350	L477	051264	MAPL1 =	170204
L232	026372	L331	032410	L413	042536	L5	005150	MAPPER	035716
L235	026642	L332	032456	L414	042576	L50	006636	MASK	002320
L236	026652	L333	033034	L415	042636	L500	051604	MBERR	012602

MEMDON	013620	MSG037	067315	MSG126	071264	MT0005	020140	OLDCSR	002156
MFPT =	000007	MSG038	067334	MSG127	071331	MT0006	020222	ONES	002614
MJPAT	017430	MSG039	067352	MSG128	071350	MT0007	020256	PADDRE	002036
MJTEST	017324	MSG040	067374	MSG129	071471	MT0010	020320	PAFBAF	014756
MKCNT	016476	MSG041	067420	MSG130	071474	MT0014	020354	PAFBAW	015106
MKCONT	015656	MSG042	067445	MSIZE	002402	MT0017	020434	PARBAF	015260
MKCSRT	017014	MSG043	067463	MTA030	022212	MT0020	020456	PARBAW	015410
MKFLAG	002120	MSG046	067505	MTEST	015562	MT0021	020536	PARCNT	002072
MKLOOP	016040	MSG047	067540	MTPA03	024750	MT0022	021026	PARITY	033744
MKPAT	017244	MSG048	067557	MTPA04	025106	MT0023	021060	PARTHE	002304
MKTEST	017104	MSG049	067617	MTPA21	026442	MT0024	021124	PARVEC =	000114
MMRO =	177572	MSG051	067652	MTPA24	027266	MT0026	021332	PASCNT	002570
MMR1 =	177574	MSG055	067671	MTPA26	027360	MT0027	021610	PASFLG	002264
MMR2 =	177576	MSG056	067712	MTPB03	025010	MT0030	022200	PASSNO	002266
MMR3 =	172516	MSG058	067745	MTPB04	025142	MT0031	022454	PATERR	002074
MMTRAP	034126	MSG061	067767	MTPB21	026472	MT0032	022644	PATPLU	004644
MMVEC =	000250	MSG062	067776	MTPB24	027326	MT0033	023160	PATTER	002112
MONFLG	002276	MSG063	070016	MTPB26	027374	MT0034	023346	PCBUMP	002326
MSEEDH	002606	MSG064	070027	MTPC03	025050	MT0035	023446	PCONFI	032722
MSEEDL	002610	MSG065	070037	MTPC21	026526	MT0036	023560	PCONFS	033334
MSG012	071210	MSG066	070051	MTPC24	027342	MT0037	023622	PCONF1	033242
MSG034	067207	MSG067	070120	MTPC26	027430	MT0041	023660	PCONF2	033302
MSG034	067245	MSG070	070127	MTPD03	025066	MT0042	023740	PDP110	034140
MSG000	071422	MSG073	070160	MTPD21	026562	MT0043	023774	PD1	045472
MSG001	065123	MSG075	070176	MTPD26	027450	MT0044	024024	PERA05	047614
MSG002	065205	MSG076	070230	MTP000	024640	MT0045	024104	PERBNK	050446
MSG003	065262	MSG077	070251	MTP001	024664	MT0046	024134	PERECC	050526
MSG004	065367	MSG079	070265	MTP002	024716	MT0047	024164	PERRAB	050264
MSG005	065475	MSG085	070311	MTP005	025162	MT0999	024220	PERRAW	050212
MSG006	065507	MSG088	070336	MTP006	025216	MT1	015636	PERRA3	045040
MSG008	071370	MSG089	070354	MTP007	025416	MT2	015642	PERRA7	050336
MSG009	065544	MSG090	070376	MTP010	025516	MUT	002110	PERR01 =	104427
MSG010	065556	MSG091	070412	MTP014	025624	M1184	071100	PERR02 =	104430
MSG011	065656	MSG092	070424	MTP017	026050	NC	047400	PERR03 =	104431
MSG012	065744	MSG093	070440	MTP020	026126	NEMCNT	002070	PERR04 =	104432
MSG013	066056	MSG095	070446	MTP022	026612	NEWBAN	002310	PERR05	047610
MSG014	066060	MSG101	070456	MTP030	027466	NEWKER	037672	PERR06	047636
MSG015	066062	MSG102	070506	MTP031	027476	NEWLOA	037774	PERR07 =	104433
MSG016	066064	MSG103	070535	MTP032	027554	NOAPT	051514	PERR10 =	104434
MSG017	066076	MSG104	070557	MTP033	027606	NOCH	054552	PERR11 =	104435
MSG018	066107	MSG105	070561	MTP034	027704	NOERRO	002430	PERR12 =	104436
MSG019	066112	MSG106	070634	MTP035	027730	NOFSMO	002426	PERR13 =	104437
MSG020	066116	MSG107	070652	MTP036	030072	NONEM	002100	PERR14 =	104440
MSG021	066137	MSG11A	065570	MTP037	030316	NONEXI	034050	PERR15 =	104441
MSG022	066725	MSG11B	066041	MTP041	030370	NOOJ	033034	PERR16 =	104442
MSG023	066747	MSG11C	066043	MTP042	030542	NOOJ1	033144	PERR17 =	104443
MSG025	066763	MSG11D	066045	MTP043	031030	NOPAR	002076	PERR20 =	104444
MSG026	067007	MSG110	070727	MTP044	031224	NORES	003722	PERR21 =	104445
MSG027	067021	MSG111	070773	MTP045	031546	NOSCOPI	002440	PERR22 =	104446
MSG028	067036	MSG112	071025	MTP046	031734	NOSUPE	002456	PERR23 =	104447
MSG029	067052	MSG113	071057	MTP047	032274	NOTAB	002372	PERR24 =	104450
MSG030	067072	MSG117	071112	MTST3	011414	NOTRCE	051200	PERR25 =	104451
MSG031	067111	MSG119	071124	MT0000	017510	NOUBMT	071547	PERR26 =	104452
MSG032	067151	MSG120	071133	MT0001	017544	NO22BI	002454	PERR27 =	104453
MSG033	067170	MSG121	071154	MT0002	017640	NULLFL	002344	PERR30 =	104454
MSG035	067273	MSG122	071174	MT0003	017754	NXTCSR	005752	PERR31 =	104455
MSG036	067276	MSG125	071242	MT0004	020070	OLDCAC	002302	PERR32 =	104456

Symbol table

PERR33=	104457	SAV30	003706	SUPDR0	002160	TMFLAG	002134	WOOPSA	046704
PERR34=	104460	SAV32	003710	SUPDR1	002162	TOOMAN	002406	WOOPUP	046472
PERR35=	104461	SAV4	002270	SUPDR2	002164	TJTCSR	002224	WORST	002600
PERR36=	104462	SBEMSK	002252	SUPDR3	002166	TRACE	006216	XOCHAR	047246
PERR37=	104463	SBENT	016756	SUPDR4	002170	TRAPVE=	000034	XXDPCH	002354
PERR40=	104464	SBESYN	026422	SUPDR5	002172	TSTBAN	011302	ZEROS	002336
PERR41=	104465	SBETES	016500	SUPDR6	002174	TSTDAT	002246	\$APTHD	056750
PERR42=	104466	SCOPE =	000004	SUPLIM	047456	TSTRD1	034542	\$AUTO	002062
PERR43=	104467	SDPAR0=	172260	SUPSTK=	000740	TSTREA=	104510	\$BANK	002013
PERXOR	050422	SDPAR5=	172272	SWAPAT	002634	TST1	005630	\$BASE	056722
PFECDF	052732	SDPAR6=	172274	SWR	002636	TST2	010460	\$BELL	002653
PFECDH	052672	SDPAR7=	172276	SWREG =	000176	TST3	010644	\$CACHF	034256
PFECDT	052722	SEEDHI	002602	SWREND	055054	TST4	011526	\$CACHN	034232
PFECFM	052636	SEEDLO	002604	SWRFLG	002566	TST5	013552	\$CBCSR	034672
PFLAG	002122	SELONL	002002	SWO =	000001	TST6	013624	\$CBREG	035662
PGMCSR	002532	SETPAT	040510	SW1 =	000002	TYPDS =	104405	\$CB1CS	034714
PHEBE	012604	SHADL1	011444	SW10 =	002000	TYPEIT=	104401	\$CDW1	056726
PHYADD	002040	SHUTUP	040636	SW11 =	004000	TYPOC =	104402	\$CDW2	056730
PMEMFL	002142	SIPAR0=	172240	SW12 =	010000	TYPOS =	104403	\$CHARC	047434
PROTYP	004010	SIPAR3=	172246	SW13 =	020000	TYPS0 =	000000	\$CHKDI	035266
PSIZE	002404	SIPAR5=	172252	SW14 =	040000	TYPS1 =	000002	\$CHK1D	035302
PSW =	177776	SIPAR6=	172254	SW15 =	100000	TYPS2 =	000000	\$CKSWR	054530
PTABLE	030276	SIPDR0=	172200	SW2 =	000004	TYPS3 =	000000	\$CLRCS	035244
PWRVEC=	000024	SIZE =	040000	SW3 =	000010	TYPS4 =	000000	\$CLR1C	035256
QUICK	002436	SKIPKA	002010	SW4 =	000020	TYPS5 =	000000	\$CMTAG	002000
QUIT	040534	SKIPMK	002342	SW5 =	000040	TYPS6 =	000002	\$CMTGE	002544
QUIT1	040600	SKJ	051226	SW6 =	000100	UDPAR0=	177660	\$CNTLC	055736
QVFLAG	002346	SKPERR	002066	SW7 =	000200	UDPAR7=	177676	\$CNTLG	055750
RANODD	027410	SKUB	037012	SW8 =	000400	UFDSET	002000	\$CNTLK	055144
RDCHR =	104411	SKUJ	012606	SW9 =	001000	UIPAR0=	177640	\$CNTLU	055743
RDDEC =	104414	SOBK	002572	SYNREG=	104514	UIPAR1=	177642	\$CPUOP	056674
RDLIN =	104412	SOBLEN=	000056	SYSSIZ	004012	UIPAR2=	177644	\$CRLF	002660
RDOCT =	104413	SOFTPA	002620	TAG2\$	011046	UIPAR3=	177646	\$DBLK	054520
READCS=	104426	SOURCE	002312	TAG3\$	011102	UIPAR4=	177650	\$DB20	056526
READON	002410	SPLTCS	002240	TAG4\$	024364	UIPAR5=	177652	\$DDW0	056734
REALPA	002300	SSP =	%000006	TAG70\$	052736	UIPAR6=	177654	\$DDW1	056736
REFRES	027166	ST =	177776	TAG71\$	052746	UIPDRO=	177600	\$DDW2	056740
REFSUB	027236	STACK =	002000	TAG72\$	052756	UNITOP	002416	\$DDW3	056742
REGCOP	032470	START	003670	TAG73\$	053026	UNMAP	040026	\$DDW4	056744
RELENT	037022	START1	000300	TAG74\$	053066	UNRELO	037214	\$DDW5	056746
RELOCA	036402	START2	000310	TAG75\$	053100	UPPFLG	002265	\$DDW7	056732
RELOC1	037036	START3	000200	TAG76\$	053112	USERMA	037610	\$DEENE	034222
RESREG=	104416	STAR27	021634	TAG77\$	053156	USESTK=	000700	\$DEVCT	056656
RESTAR	002626	STOPOK	002420	TAG78\$	053164	USP =	%000006	\$DEVM	056724
RESVEC=	000010	STRIPE	002366	TAG79\$	053244	VMKOR	003712	\$DIDDO=	000000
RESO	041350	SUBAAA	004702	TAG9\$	010710	WARN1	010776	\$DOAGA	014156
RES1	041430	SUBAAB	005032	TBG4\$	024542	WARN2	024762	\$DOAGN	014052
RES2	041576	SUBAAI	011440	TCFIG1	033404	WARN3	024776	\$DOWN	045722
RLFLAG	002126	SUBAAP	012766	TCFIG2	033444	WARN4	025022	\$DTBL	054510
RRFLAG	002124	SUBAAR	012172	TCFIG3	033600	WARN5	025036	\$ECCDI	034570
RTNVAL =	%000000	SUBAAS	010454	TCONFI	033336	WASDBE=	104500	\$ECCIN	034616
RWCSR	006220	SUCCES	002334	TEMP	002434	WASSBE=	104476	\$ECC1D	034604
SAVCSR	002154	SUPDOA	002262	TESTAD	002412	WAS1DB=	104501	\$ECC1I	034632
SAVMON	002274	SUPD01	024270	TESTMO	002552	WAS1SB=	104477	\$ENASB	034644
SAVPAR	002272	SUPD02	024304	TIME	002340	WHICHC	044670	\$ENA1S	034660
SAVREG=	104415	SUPD03	024446	TIMEOU	034114	WOOPEN	046654	\$ENDAD	014042
		SUPD04	024462	TKVEC =	000060	WOOPS	046306	\$ENERG	034212

Symbol table

\$ENV	056666	\$LL	= 000106	\$0\$	= 000000	\$PER37	051136	\$TRAP2	057006
\$ENVM	056667	\$LOADC	034274	\$PASS	056654	\$PER40	051142	\$TRPAD	057026
\$EOP	013676	\$LPADR	002622	\$PASTM	056756	\$PWRDN	045352	\$TSTM	056754
\$ERFLG	002014	\$LPERR	002624	\$PATMA	002012	\$PWRUP	045726	\$TSTRD	034410
\$ERRGE	035412	\$L\$	= 000000	\$PER01	047456	\$QUES	002657	\$TTYIN	055712
\$ERROR	051502	\$MADR1	056700	\$PER02	047504	\$R	= 177777	\$TYPDS	054304
\$ERRTB	057274	\$MADR2	056704	\$PER03	047532	\$RAND	056432	\$TYPE	047122
\$ERRTY	052316	\$MADR3	056710	\$PER04	047562	\$RDCHR	055264	\$TYPEC	047250
\$ERTTL	002630	\$MADR4	056714	\$PER07	047644	\$RDDEC	056150	\$TYPEX	047436
\$ESCAP	002362	\$MAIL	056646	\$PER10	047666	\$RDLIN	055414	\$TYPC	054102
\$ETABL	056666	\$MAMS1	056676	\$PER11	047716	\$RDOCT	056000	\$TYPON	054116
\$ETEND	056750	\$MAMS2	056702	\$PER12	047736	\$READC	034370	\$TYPOS	054056
\$EXHAL	040630	\$MAMS3	056706	\$PER13	047760	\$RESRE	056374	\$T1	= 000000
\$E\$	= 000001	\$MAMS4	056712	\$PER14	050000	\$SAVRE	056336	\$T2	= 000530
\$FATAL	056650	\$MBADR	056752	\$PER15	050022	\$SAVR6	046304	\$UNIT	056660
\$FILLC	002652	\$MNEW	055766	\$PER16	050044	\$SCOPE	051146	\$UNITM	056760
\$FILLS	002357	\$MSGAD	056662	\$PER17	050064	\$STN	= 000001	\$USWR	056672
\$F\$	= 000000	\$MSGLG	056664	\$PER20	050102	\$SVLAD	051402	\$VECT1	056716
\$GTSWR	054704	\$MSGTY	056646	\$PER21	050120	\$SV\$	= 000000	\$VECT2	056720
\$HALT	052100	\$MSWR	055755	\$PER22	050140	\$SWR	= 163000	\$WASDB	035100
\$HALT2	057024	\$MTYP1	056677	\$PER23	050156	\$SWREG	056670	\$WASSB	034734
\$HIBTS	056750	\$MTYP2	056703	\$PER24	050174	\$SYNRE	035700	\$WAS1D	035214
\$HIOCT	056146	\$MTYP3	056707	\$PER25	044756	\$T	= 000531	\$WAS1S	035050
\$ILLUP	046300	\$MTYP4	056713	\$PER26	050364	\$TESTN	056652	\$XTSTR	051274
\$INVAL	035362	\$NOTRA	057020	\$PER27	050404	\$TKB	002644	\$Y\$	= 000000
\$ITEMB	002015	\$NULL	002356	\$PER30	045204	\$TKS	002642	\$ZAP42	014022
\$I\$	= 000001	\$NWTST	= 000001	\$PER31	050574	\$TN	= 000007	\$Z\$	= 000000
\$KERNE	034202	\$OCNT	054300	\$PER32	050672	\$TPB	002650	\$S\$	= 000000
\$KMAP	036310	\$OCTVL	056630	\$PER33	050740	\$TPFLG	002360	\$T\$	= 000512
\$K\$	= 000103	\$OCT8	= 056634	\$PER34	051020	\$TPS	002646	\$TT	= 000520
\$L	= 000110	\$OMODE	054302	\$PER35	051052	\$TRAP	056764	\$OFILL	054301
\$LF	002661	\$OVER	051416	\$PER36	051106				

. ABS. 071612 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 1286
 Work file writes: 949
 Size of work file: 25487 Words (100 Pages)
 Size of core pool: 17688 Words (68 Pages)
 Operating system: RSX-11M/PLUS

Elapsed time: 00:32:57.57
 CVMJAO.BIN,CVMJAO/NL:TOC=CVMJAO/ML,CVMJAO.MAC