

RH70

CONTROLLER DIAGNOSTIC
CERHADO

AH-8001D-MC

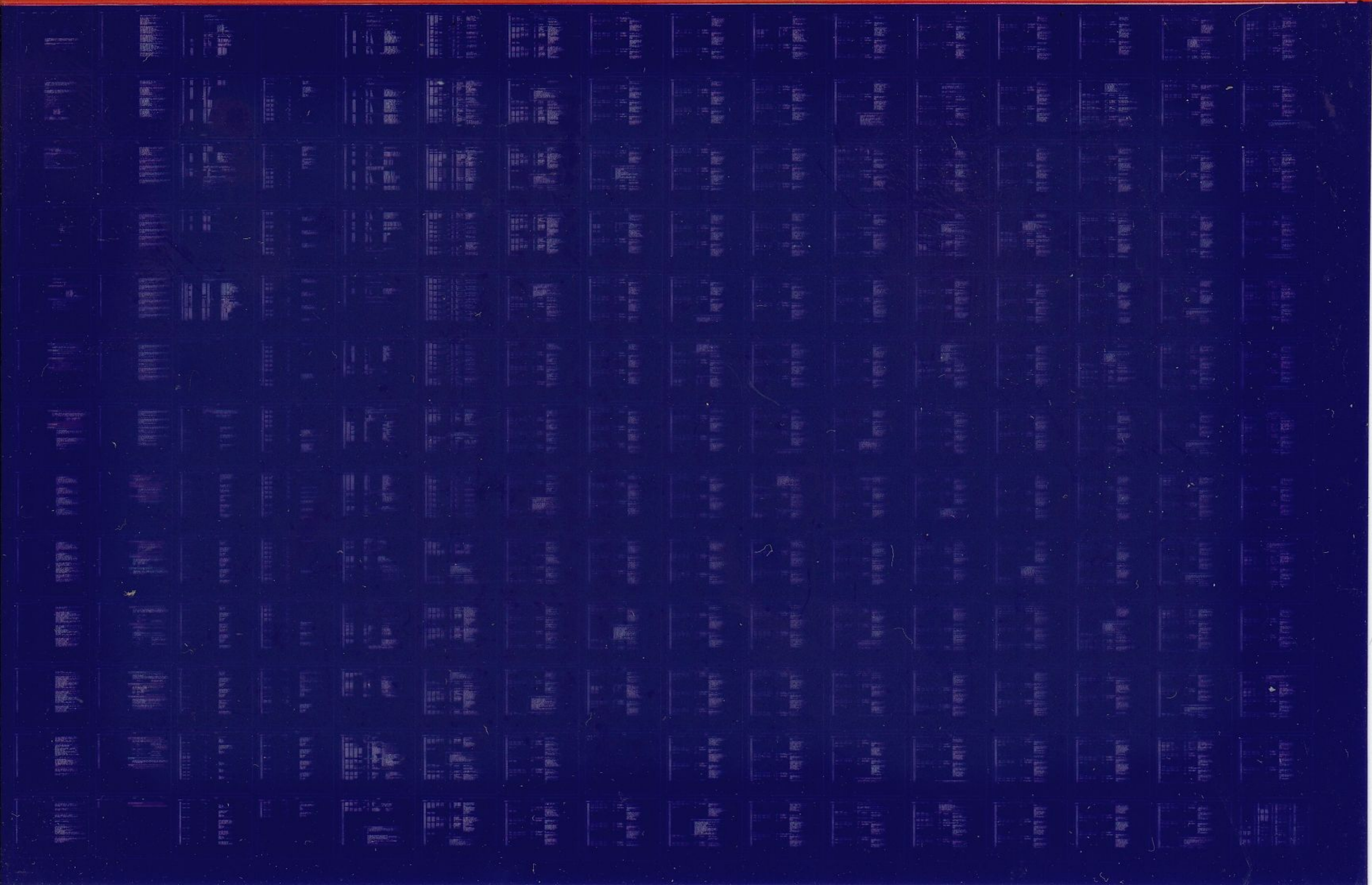
APR 1978

COPYRIGHT ©75-78

digital

FICHE 1 OF 2

MADE IN USA



RH70

CONTROLLER DIAGNOSTIC
CERHADO

AH-8001D-MC

COPYRIGHT ©75-78
FICHE 2 OF 2

APR 1978
digital
MADE IN USA

101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130

4.3 PROGRAM AND/OR OPERATOR ACTION

BEFORE STARTING THE PROGRAM, MAKE SURE THAT THE MAGNETIC MEDIUM IS PROPERLY POSITIONED:

- 1.) TU16 - MAGTAPE MUST BE AT LOAD POINT.
- 2.) RP - HEADS MUST BE IN HOME POSITION.

ON STARTING FROM 210 OPERATOR MUST CHOOSE BETWEEN THE RH70 PERIPHERALS AND TYPE THE BASE ADDRESS OF ANY OR ALL OF THE PERIPHERALS CONNECTED TO THE RH70.

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

SEE SECTION 4.1

5.2 SUB-ROUTINE ABSTRACTS

SEE SECTION 9 "SUBROUTINES"

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION

ON DETECTION OF AN ERROR THE PROGRAM WILL PRINT ALL RELEVANT INFORMATION AND THEN PROCEED ACCORDING TO THE SWITCH SETTINGS GIVEN IN 4.1. IF ALL SWITCHES ARE DOWN THE PROGRAM WILL CONTINUE.

XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX
XX

139 *****
TRAP CATCHER

142 ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

146 *****
STARTING ADDRESS(ES)

152 STARTING ADDRESS 200 FOR NORMAL STARTS
THIS WILL TEST ALL RPO4'S ON THE SYSTEM A SINGLE DRIVE AT A TIME
STARTING ADDRESS 210 WILL TEST ONLY ONE SPECIFIED DRIVE

160 *****
MEMORY MANAGEMENT DEFINITIONS

162 KT11 VECTOR ADDRESS
166 KT11 STATUS REGISTER ADDRESSES
173 KERNEL "I" PAGE DESCRIPTOR REGISTERS
184 KERNEL "I" PAGE ADDRESS REGISTERS

199 *****
COMMON TAGS

201 THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
USED IN THE PROGRAM.

CERHADO

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
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312

252

ERROR POINTER TABLE

254 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 :
NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERPPC).
NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS

260 EM ::POINTS TO THE ERROR MESSAGE
OH ::POINTS TO THE DATA HEADER
DT ::POINTS TO THE DATA
DF ::POINTS 0 THE DATA FORMAT

269 *****

1587

REGISTER ADDRESSES

1834

REGISTER TEST

1899 TEST 1 SIZE FOR RH DEVICES
THIS TEST DETERMINS WHICH RH DEVICE IS ON THE SYSTEM.
IF THE SYSTEM HAS MORE THAN ONE RH DEVICE THEN ONLY ONE
THE DEVICES WILL BE USED IN THE FOLLOWING TESTS.
THE ONE THAT WILL BE USED IS THE FIRST ONE THAT IS PRESE
IN THE FOLLOWING LIST
RS
RP
TM

THE WHY THE TEST WORKS IS AS FOLLOWS:-
A REFERENCE IS MADE TO THE CONTROL AND STATUS REGISTER 1
FOR THE RS BY A TST INSTRUCTION. IF IT RESPONDS THEN I
ASSUMED PRESENT AND THE LOCATIONS FOR THE I/O REGISTERS
FILLED WITH THE APPROPRIATE ADDRESSES.
THEN THE DRIVE TYPE REGISTER IS CHECKED TO HAVE GOOD
VALUES.
IF THE TST INSTRUCTION TRAPS TO A NON EXISTANT VECTOR
THEN A SIMILAR ATTEMPT IS MADE TO A RP CONTROL AND STATU
REGISTER 1.
THEN A TM IS TRYED.
THEN A MIXED SYSTEM WITH MORE THAN ONE RH DEVICES IS TRYE

2195 TEST 2

UNIT UNDER TEST
THIS TYPES THE UNIT TO BE TESTED
AND IS THE FIRST TEST AFTER THE END OF THE PROGRAM

014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068

CERHADO

2302 TEST 3 BIT BANG RHCS1

2304 TEST LOADING AND READING OF ALL POSSIBLE BITS IN RHCS1 REGISTER. USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC) AND WALKING 0'S (-2,-3,-5 ETC) IN THIS TEST SC!TR!MCP!DVA!BIT12!BIT10!RDY!GO BITS WILL ALWAYS BE SET AND RDY!DVA BITS WILL ALWAYS BE SET AND BIT10 BITS WILL ALWAYS BE CLEARED IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING THEN A RH CLEAR (RHCS2 BIT #5) WILL BE GIVEN TO AID SCOPE SYNC'S ON THE CLEAR SIGNAL.

2383 TEST 4 BIT BANG RHCS2

2385 TEST LOADING AND READING OF ALL POSSIBLE BITS IN RHCS2 REGISTER. USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC) AND WALKING 0'S (-2,-3,-5 ETC) IN THIS TEST 17740 BITS WILL NOT BE WRITTEN INTO AND IR BITS WILL ALWAYS BE SET IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING THEN A RH CLEAR (RHCS2 BIT #5) WILL BE GIVEN TO AID SCOPE SYNC'S ON THE CLEAR SIGNAL.

2460 TEST 5 BIT BANG RHCS3

2462 TEST LOADING AND READING OF ALL POSSIBLE BITS IN RHCS3 REGISTER. USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC) AND WALKING 0'S (-2,-3,-5 ETC) IN THIS TEST 17760 BITS WILL NOT BE WRITTEN INTO AND 0 BITS WILL ALWAYS BE SET AND BIT9!BIT8!BIT7!BITS!BIT4 BITS WILL ALWAYS BE CLEARED IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING THEN A RH CLEAR (RHCS2 BIT #5) WILL BE GIVEN TO AID SCOPE SYNC'S ON THE CLEAR SIGNAL.

2541 TEST 6 BIT BANG RHWC

2543 TEST LOADING AND READING OF ALL POSSIBLE BITS IN RHWC REGISTER. USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC) AND WALKING 0'S (-2,-3,-5 ETC) IN THIS TEST 0 BITS WILL NOT BE WRITTEN INTO AND 0 BITS WILL ALWAYS BE SET IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING THEN A RH CLEAR (RHCS2 BIT #5) WILL BE GIVEN TO AID SCOPE SYNC'S ON THE CLEAR SIGNAL.

CERHADO

2618
2620
2699
2701
2784
2973
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
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
2729
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
2779
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
2805
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
2835
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
2863
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
2889
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
2918
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
2954
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
2995
2996
2997
2998
2999
3000

2618 TEST 7 BIT BANG RHBA

2620 TEST LOADING AND READING OF ALL POSSIBLE BITS IN RHBA REGISTER.
USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC) AND WALKING 0'S (-2,-3,-5 ETC)
IN THIS TEST 0 BITS WILL NOT BE WRITTEN INTO AND 0 BITS WILL ALWAYS BE SET AND BIT00 BITS WILL ALWAYS BE CLEARED IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING THEN A RH CLEAR (RHCS2 BIT #5) WILL BE GIVEN TO AID SCOPE SYNC'S ON THE CLEAR SIGNAL.

2699 TEST 10 BIT BANG RHBAE

2701 TEST LOADING AND READING OF ALL POSSIBLE BITS IN RHBAE REGISTER.
USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC) AND WALKING 0'S (-2,-3,-5 ETC)
IN THIS TEST 177700 BITS WILL NOT BE WRITTEN INTO AND 0 BITS WILL ALWAYS BE SET AND 177700 BITS WILL ALWAYS BE CLEARED IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING THEN A RH CLEAR (RHCS2 BIT #5) WILL BE GIVEN TO AID SCOPE SYNC'S ON THE CLEAR SIGNAL.

2784 TEST 11 SILO TEST 1 (ONE WORD WRITE)
AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2) RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH TOGETHER WITH UNIT NUMBER
ONE WORD OF ALL ZEROS IS WRITTEN INTO RHDB BY "CLR"
"OR" (BIT #7) IS WAITED FOR BY A TRAP INSTRUCTION CALLED "WAT" (NO TIMING IS DONE)
HOWEVER IF "OR" DOES NOT SET WITHIN "WAT" COUNT DOWN AN ERROR IS REPORTED
RHDB IS READ AND CHECKED TO CONTAIN ZEROS
RHCS2 WILL BE CHECKED TO HAVE "IR" AND UNIT NUMBER
RHCS1, RHCS3, RHBA, RHBAE, RHWC, WILL BE CHECKED TO HAVE APPROPRIATE VALUES

2973 TEST 12 SILO TEST 2 (ONE WORD WRITE)
AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2) RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH TOGETHER WITH UNIT NUMBER
ONE WORD OF ALL ONES IS WRITTEN INTO RHDB BY "CLR"
"OR" (BIT #7) IS WAITED FOR BY A TRAP INSTRUCTION CALLED "WAT" (NO TIMING IS DONE)
HOWEVER IF "OR" DOES NOT SET WITHIN "WAT" COUNT DOWN AN ERROR IS REPORTED
RHDB IS READ AND CHECKED TO CONTAIN ALL ONES

CERHADO

RHCS2 WILL BE CHECKED TO HAVE "IR" AND UNIT NUMBER
RHCS1, RHCS3, RHBA, RHBAE, RHWC, WILL BE CHECKED TO HAVE
APPROPRIATE VALUES

3144 TEST 13 SILO TEST 3 (TWO WORD WRITE)

3146

AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
TOGETHER WITH UNIT NUMBER
ONE WORD = 52525 IS WRITTEN INTO RHDB
RHCS2 IS CHECKED TO HAVE "IR" AND UNIT NUMBER
A SECOND WORD = 12525 IS WRITTEN INTO RHDB
"OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
INSTRUCTION CALLED "WAT"
RHDB IS READ AND CHECKED TO CONTAIN 52525
RHCS2 IS CHECKED TO HAVE "IR", "OR" AND UNIT NUMBER
RHDB IS READ A SECOND TIME AND CHECKED TO
CONTAIN 12525
RHCS2 IS CHECKED TO HAVE "IR" AND UNIT NUMBER
THEN ALL REGISTERS RHCS1, RHCS3, RHBA, RHBAE, RHWC
ARE CHECKED TO HAVE APPROPRIATE VALUE

3416 TEST 14 SILO TEST 4 (COUNT PATTERN)

3418

AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH, TOGETHER
WITH UNIT NUMBER
EIGHT WORDS, A COUNT PATTERN 0 THRU 7 IS WRITTEN
INTO RHDB
"OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
INSTRUCTION CALLED "WAT"
THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
"OR" HIGH TOGETHER WITH THE UNIT NUMBER
THEN RHBS READ AND COMPARED TO HAVE THE RIGHT VALUE
EIGHT TIES
THEN RHCS2, RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
CHECKED TO HAVE THE APPROPRIATE VALUE

3808 TEST 15 SILO TEST 5 (FLOATING ONES)

3810

AFTER ARH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
TOGETHER WITH UNIT NUMBER
EIGHT WORDS, A PATTERN OF FLOATING ONES (1,2,4,10,20,40,
IS WRITTEN INTO RHDB
"OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
INSTRUCTION CALLED "WAT"
THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
"OR" HIGH TOGETHER WITH THE UNIT NUMBER
THEN RHDB IS READ AND COMPARED TO HAVE THE
RIGHT VALUE AFTER EACH OF THE EIGHT READS.
THEN RHCS2 IS CHECKED TO HAVE "IR"

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

CERHADO

480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
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

AND UNIT NUMBER
THEN RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
CHECKED TO HAVE THE APPROPRIATE VALUE

4227 TEST 16 SILO TEST 6 (FLOATING ONES IN UPPER BYTE)

4229 AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
TOGETHER WITH UNIT NUMBER
EIGHT WORDS, A PATTERN OF FLOATING ONES IN UPPER BYTE
400, 1000, 2000, 4000, 10000, 20000, 40000, 100000
IS WRITTEN INTO RHDB
"OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
INSTRUCTION CALLED "WAT"
THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
"OR" HIGH TOGETHER WITH THE UNIT NUMBER
THEN RHDB IS READ AND COMPARED TO HAVE THE
RIGHT VALUE AFTER EACH OF THE EIGHT READS.
THEN RHCS2 IS CHECKED TO HAVE "IR"
AND UNIT NUMBER
THEN RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
CHECKED TO HAVE THE APPROPRIATE VALUE

4649 TEST 17 SILO TEST 7 (FLOATING 0 IN LOWER BYTE)

4651 AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
TOGETHER WITH UNIT NUMBER
EIGHT WORDS OF FLOATING ZEROS 177776, 177775, 177773,
177767, 177757, 177737, 177677, 177577
IS WRITTEN INTO RHDB
"OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
INSTRUCTION CALLED "WAT"
THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
"OR" HIGH TOGETHER WITH THE UNIT NUMBER
THEN RHDB IS READ AND COMPARED TO HAVE THE
RIGHT VALUE AFTER EACH OF THE EIGHT READS.
THEN RHCS2 IS CHECKED TO HAVE "IR"
AND UNIT NUMBER
THEN RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
CHECKED TO HAVE THE APPROPRIATE VALUE

5070 TEST 20 SILO TEST 8 (FLOATING ZEROS IN UPPER BYTE)

5072 AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
TOGETHER WITH UNIT NUMBER
EIGHT WORDS, A PATTERN OF FLOATING ZEROS IN UPPER BYTE
177377, 177677, 175777, 173777, 167777, 157777, 137777.
IS WRITTEN INTO RHDB
"OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
INSTRUCTION CALLED "WAT"
THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND

CERHADO

535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
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

"OR" HIGH TOGETHER WITH THE UNIT NUMBER
THEN RHDB IS READ AND COMPARED TO HAVE THE
RIGHT VALUE AFTER EACH OF THE EIGHT READS.
THEN RHCS2 IS CHECKED TO HAVE "IR"
AND UNIT NUMBER
THEN RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
CHECKED TO HAVE THE APPROPRIATE VALUE

5508 TEST 21 RHCS1 - MCPE BIT #13 (PARITY LINE = 0)

5510 AN RH CLEAR (SET BIT #5 IN RHCS2) IS GIVEN TO
CLEAR ALL DEVICE REGISTERS
THEN RHER1 (ERROR REGISTER 1) IS CHECKED TO HAVE
ZEROS
SET "PAT" (BIT #4 IN RHCS2) TO INVERT PARITY CHECKING
READ ANY DEVICE REGISTER - HERE RHER1
READ AND CHECK RHCS1 TO CONTAIN SC (BIT #15)
AND MCPE (BIT #13)
WRITE "1" INTO TRE (BIT #14 IN RHCS1)
READ RHCS1 SC, MCPE, AND RDY SHOULD BE SET
GIVE AN RH CLEAR (CLR - BIT #5 IN RHCS2)
CHECK RHCS1 TO HAVE RDY
CHECK RHCS2 TO HAVE ONLY IR AND UNIT NUMBER
CHECK RHCS3, RHBA, RHBAE, RHWC TO HAVE APPROPRIATE
VALUES.

5785 TEST 22 RHCS1 - MCPE BIT #13 (PARITY LINE = 1)

5787 AN RH CLEAR (SET BIT #5 IN RHCS2) IS GIVEN TO CLEAR
ALL DEVICE REGISTERS
WRITE A ONE INTO DISK ADDRESS REGISTER (RHDA)
(IN TAPE DRIVES CALLED REGISTER)
SET "PAT" (RHCS2 BIT #4) THIS WILL INVERT THE PARITY CHE
READ RHDA AND COMPARE IT HAS ONE IN IT
THIS READING SHOULD SET SC, MCPE IN RHCS1
CHECK RHCS1 TO HAVE ONLY RDY SET
CHECK RHCS2, RHCS3, RHBA, RHBAE, RHWC TO HAVE APPROPRIATE
VALUES

5959 TEST 23 TEST DUPLICATED A16 (RHCS1 BIT #8)

5961 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
MOVE 400 (ONE INTO A16) IN RHCS1
READ RHCS1 TO CONTAIN 600 (BIT #8 AND RDY)
READ RHBAE TO CONTAIN "1" (BIT #0 HIGH)

MOVE 0 INTO RHBAE (WRITING 0 INTO RHBAE BIT #0)
READ RHBAE TO CONTAIN 0
READ RHCS1 TO CONTAIN ONLY RDY (BIT #8 IN RHCS1 IS ZERO)

NO1

CERHADD MACY11 30(1046) 21-DEC-77 13:06 PAGE 14
 CERHAD.P11 21-DEC-77 12:48

SEQ 0013

589	CERHADD	6079	TEST 24 TEST DUPLICATED A17 (RHCS1 BIT #9)
590		6081	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5, MOVE 400 (ONE INTO A17) IN RHCS1 READ RHCS1 TO CONTAIN 1200 (BIT #9 AND RDY) READ RHBAE TO CONTAIN "2" (BIT #1 HIGH)
591			
592			
593			
594			
595			
596			
597			
598			
599			
600			MOVE 0 INTO RHBAE (WRITING 0 INTO RHBAE BIT #1, READ RHBAE TO CONTAIN 2 READ RHCS1 TO CONTAIN ONLY RDY (BIT #9 IN RHCS1 IS ZERO)
601			
602			
603			
604		6199	TEST 25 TEST DUPLICATED IE (RHCS. BIT #6)
605		6201	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5) MOVE 100 (ONE INTO IE) IN RHCS1 READ RHCS1 TO CONTAIN 300 (BIT #6 AND RDY) READ RHCS3 TO CONTAIN "100" (BIT #6 HIGH)
606			
607			
608			
609			
610			
611			
612			
613			
614		6208	MOVE 0 INTO RHCS3 (WRITING 0 INTO RHCS3 BIT #6) READ RHCS3 TO CONTAIN 100 READ RHCS1 TO CONTAIN ONLY RDY (BIT #6 IN RHCS1 IS ZERO)
615			
616			
617			
618			
619		6319	TEST 26 RHCS1 PROGRAM ERROR (PGE BIT #10)
620		6321	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5, SET UP FOR A 10 WORD WRITE SET "GO" (RHCS1 BIT #0) TWICE IN TWO SUCCESSIVE INSTRUCTIONS THIS SHOULD SET SC AND TRE IN RHCS1 . AND PGE SHOULD BE SET IN RHCS2 RHCS3, ARE CHECKED THE NUMBER OF WORDS ARE LARGE ENOUGH SO THAT AFTER ONE "BIS" INSTRUCTION TO SET "GO" IN RHCS1 THERE IS SUFFICIENT TIME TO GIVE ANOTHER "BIS" INSTRUCTI TO SET "GO" BEFORE THE FIRST "GO" HAS TIME TO COMPLETE
621			
622			
623			
624			
625			
626			
627			
628			
629			
630			
631			
632			
633			
634		6415	TEST 27 RHCS2 - BUS ADDRESS INHIBIT BIT #3
635		6417	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #6) SET UP FOR A 2 WORD WRITE FROM A BUFFER TAGGED "WRFROM" SET BUS ADDRESS INHIBIT RHCS2 BIT #3 "BAI" AT THE END OF WRITE CHECK RHCS1 TO HAVE ONLY RDY AND COMMAND RHCS2 TO HAVE BAI RHCS3 TO HAVE 0
636			
637			
638			
639			
640			
641			
642			
643			

644 CERHADO
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699

6424 RHBA TO HAVE ADDRESS OF "WRFROM"
RHBAE AND RHWC TO HAVE ZEROS

NOW SET UP READ FOR THE SAME DATA WITH
BAI SET TO READ INTO A BUFFER TAGGED "REINTC"
AFTER READ CHECK
RHCS1 TO HAVE ONLY RDY AND COMMAND
RHCS2 TO HAVE BAI
RHCS3 TO HAVE 0
RHBA TO HAVE ADDRESS OF "REINTC"
RHBAE AND RHWC TO HAVE ZEROS
DATA IN REINTC BUFFER IS CHECKED WITH DATA IN
WRFROM BUFFER

6571 TEST 30 RHSC2 MDPE BIT #8 AND RHCS3 IPCK0 BIT #0
CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET IPCK0 (RHCS3 BIT #0)
MOVE ALL ZEROS INTO RHDB ONCE
THIS SHOULD SET TRE SC IN RHCS1
READ RHDB ONCE THIS SHOULD SET MDPE (RHCS2 BIT #8)
CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC (BIT #15)
"CLR" (RHCS2 BIT #5) IS GIVEN
ALL ERRORS ARE CLEARED
CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

6807 TEST 31 RHSC2 MDPE BIT #8 AND RHCS3 IPCK1 BIT #1
CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET IPCK1 (RHCS3 BIT #1)
MOVE ALL ZEROS INTO RHDB ONCE
THIS SHOULD SET TRE SC IN RHCS1
READ RHDB ONCE THIS SHOULD SET MDPE (RHCS2 BIT #8)
CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC (BIT #15)
"CLR" (RHCS2 BIT #5) IS GIVEN
ALL ERRORS ARE CLEARED
CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

7043 TEST 32 RHSC2 MDPE BIT #8 AND RHCS3 IPCK2 BIT #2
CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET IPCK2 (RHCS3 BIT #2)
MOVE ALL ZEROS INTO RHDB TWICE
THIS SHOULD NOT SET TRE SC IN RHCS1
READ RHDB ONCE THIS SHOULD SET MDPE (RHCS2 BIT #8)
CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC (BIT #15)
READ RHDB A SECOND TIME. CHECK RHCS1, RHCS2, RHCS3
"CLR" (RHCS2 BIT #5) IS GIVEN
ALL ERRORS ARE CLEARED
CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

7345 TEST 33 RHSC2 MDPE BIT #8 AND RHCS3 IPCK3 BIT #3
CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET IPCK3 (RHCS3 BIT #3)
MOVE ALL ZEROS INTO RHDB TWICE
THIS SHOULD NOT SET TRE SC IN RHCS1
READ RHDB ONCE THIS SHOULD SET MDPE (RHCS2 BIT #8)

CERHADO

700
701
702
703
704
705
706
707
708
709
710
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

CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC (BIT #15)
 READ RHBA A SECOND TIME. CHECK RHCS1, RHCS2, RHCS3
 "CLR" (RHCS2 BIT #5) IS GIVEN
 ALL ERRORS ARE CLEARED
 CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

7648 TEST 34 RHCS2-WCE BIT #14 AND RHCS3-WCE OW BIT #12

7650 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
 WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
 SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
 ODD WORD BOUNDARY
 DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
 ODD WORD BOUNDARY
 THIS SHOULD SET WCE OW (RHCS3 BIT #12)
 AND WCE (RHCS2 BIT #14)
 "CLR" (RHCS2 BIT #5) IS GIVEN
 ALL ERRORS ARE CLEARED
 CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

7881 TEST 35 RHCS2-WCE BIT #14 AND RHCS3-WCE OW BIT #12

7883 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
 WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
 SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
 ODD WORD BOUNDARY
 DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
 ODD WORD BOUNDARY
 THIS SHOULD SET WCE OW (RHCS3 BIT #12)
 AND WCE (RHCS2 BIT #14)
 "CLR" (RHCS2 BIT #5) IS GIVEN
 ALL ERRORS ARE CLEARED
 CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

8114 TEST 36 RHCS2-WCE BIT #14 AND RHCS3-WCE EW BIT #11

8116 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
 WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
 SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
 EVEN WORD BOUNDARY
 DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
 EVEN WORD BOUNDARY
 THIS SHOULD SET WCE EW (RHCS3 BIT #11)
 AND WCE (RHCS2 BIT #14)
 "CLR" (RHCS2 BIT #5) IS GIVEN
 ALL ERRORS ARE CLEARED
 CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

8347 TEST 37 RHCS2-WCE BIT #14 AND RHCS3-WCE EW BIT #11

795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
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
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
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
967
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

CERHADO

8349

CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
EVEN WORD BOUNDARY
DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
EVEN WORD BOUNDARY
THIS SHOULD SET WCE EW (RHCS3 BIT #11)
AND WCE (RHCS2 BIT #14)
"CLR" (RHCS2 BIT #5) IS GIVEN
ALL ERRORS ARE CLEARED
CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

8581

TEST 40 TEST DBL (RHCS3 BIT #10) TEST A

8583

CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)

8584

SET UP FOR A NINE WORD WRITE FROM AN EVEN WORD BOUNDARY
DO A NINE WORD WRITE FROM AN EVEN WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

8694

TEST 41 TEST DBL (RHCS3 BIT #10) TEST B

8696

CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY
DO A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY
THIS SHOULD SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

8808

TEST 42 TEST DBL (RHCS3 BIT #10) TEST C

8810

CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD WRITE FROM AN ODD WORD BOUNDARY
DO A TEN WORD WRITE FROM AN ODD WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

8921

TEST 43 TEST DBL (RHCS3 BIT #10) TEST D

8923

CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A ELEVEN WORD WRITE FROM AN EVEN WORD BOUNDARY
DO A ELEVEN WORD WRITE FROM AN EVEN WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
054
055
056
057
058
059
060
061
062
063
064
065
066
067
068
069
070
071
072
073
074
075
076
077
078
079
080
081
082
083
084
085
086
087
088
089
090
091
092
093
094
095
096
097
098
099
100

CERHADO

9034 TEST 44 TEST DBL (RHCS3 BIT #10) TEST E

9036 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY
WITH BAI IN RHCS2 BIT #3 SET
DO A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

9153 TEST 45 TEST DBL (RHCS3 BIT #10) TEST F

9155 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A ELEVEN WORD WRITE FROM AN ODD WORD BOUNDARY
DO A ELEVEN WORD WRITE FROM AN ODD WORD BOUNDARY
THIS SHOULD SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

9268 TEST 46 TEST DBL (RHCS3 BIT #10) TEST G

9270 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A NINE WORD READ FROM AN EVEN WORD BOUNDARY
DO A NINE WORD READ FROM AN EVEN WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

9381 TEST 47 TEST DBL (RHCS3 BIT #10) TEST H

9383 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD READ FROM AN EVEN WORD BOUNDARY
DO A TEN WORD READ FROM AN EVEN WORD BOUNDARY
THIS SHOULD SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

9495 TEST 50 TEST DBL (RHCS3 BIT #10) TEST I

9497 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD READ FROM AN ODD WORD BOUNDARY
DO A TEN WORD READ FROM AN ODD WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3

9502 CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

9608 TEST 51 TEST DBL (RHCS3 BIT #10) TEST J

851	CERHADO	9610	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5) SET UP FOR A ELEVEN WORD READ FROM AN EVEN WORD BOUNDARY DO A ELEVEN WORD READ FROM AN EVEN WORD BOUNDARY THIS SHOULD NOT SET RHCS3 BIT #10 DBL CHECK RHCS3 CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
852		9721	TEST 52 TEST DBL (RHCS3 BIT #10) TEST K
853		9723	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5) SET UP FOR A ELEVEN WORD READ FROM AN ODD WORD BOUNDARY DO A ELEVEN WORD READ FROM AN ODD WORD BOUNDARY THIS SHOULD SET RHCS3 BIT #10 DBL CHECK RHCS3 CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
854			
855			
856			
857			
858			
859			
860			
861			
862			
863			
864			
865			
866			
867			
868		9835	TEST 53 TEST DBL (RHCS3 BIT #10) TEST L
869		9837	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5) SET UP FOR A TEN WORD READ FROM AN EVEN WORD BOUNDARY WITH BAI IN RHCS2 BIT #3 SET DO A TEN WORD READ FROM AN EVEN WORD BOUNDARY THIS SHOULD NOT SET RHCS3 BIT #10 DBL CHECK RHCS3 CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
870			
871			
872			
873			
874			
875			
876			
877			
878		9955	TEST 54 TEST DBL (RHCS3 BIT #10) TEST M
879		9957	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5) SET UP FOR A ONE WORD WRITE REVERSE FROM AN EVEN WORD BO DO A ONE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY THIS SHOULD NOT SET RHCS3 BIT #10 DBL CHECK RHCS3 CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
880			
881			
882			
883			
884			
885			
886			
887		10068	TEST 55 TEST DBL (RHCS3 BIT #10) TEST N
888		10070	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5) SET UP FOR A TWO WORD WRITE REVERSE FROM AN EVEN WORD BO DO A TWO WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY THIS SHOULD NOT SET RHCS3 BIT #10 DBL CHECK RHCS3 CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
889			
890			
891			
892			
893			
894			
895			
896		10181	TEST 56 TEST DBL (RHCS3 BIT #10) TEST O
897		10183	CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5) SET UP FOR A TWO WORD WRITE REVERSE FROM AN ODD WORD BOUN DO A TWO WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY THIS SHOULD SET RHCS3 BIT #10 DBL CHECK RHCS3 CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
898			
899			
900			
901			
902			
903			
904			

G02

905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
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

CERHADO

10295 TEST 57 TEST DBL (RHCS3 BIT #10) TEST P

10297 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A THREE WORD WRITE REVERSE FROM AN EVEN WORD
DO A THREE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
THIS SHOULD SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

10409 TEST 60 TEST DBL (RHCS3 BIT #10) TEST Q

10411 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A THREE WORD WRITE REVERSE FROM AN ODD WORD B
DO A THREE WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

10522 TEST 61 TEST DBL (RHCS3 BIT #10) TEST R

10524 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TWO WORD WRITE REVERSE FROM AN EVEN WORD BC
WITH BAI IN RHCS2 BIT #3 SET
DO A TWO WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY

10528 THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

10642 TEST 62 TEST DBL (RHCS3 BIT #10) TEST S

10644 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A NINE WORD READ REVERSE FROM AN EVEN WORD BC
DO A NINE WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
THIS SHOULD SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE

10761 TEST 63 TEST DBL (RHCS3 BIT #10) TEST T

10763 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD READ REVERSE FROM AN EVEN WORD BOUND
DO A TEN WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE

H02

CERHADO

955
956
957
958
959
960
961
962
963
964
965
966
967
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

10879 TEST 64 TEST DBL (RHCS3 BIT #10) TEST U

10881 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD READ REVERSE FROM AN ODD WORD BOUN
DO A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
THIS SHOULD SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE

10998 TEST 65 TEST DBL (RHCS3 BIT #10) TEST V

11000 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A ELEVEN WORD READ REVERSE FROM AN EVEN WORD
DO A ELEVEN WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
THIS SHOULD SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE

11117 TEST 66 TEST DBL (RHCS3 BIT #10) TEST W

11119 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A ELEVEN WORD READ REVERSE FROM AN ODD WORD B
DO A ELEVEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY

11122 THIS SHOULD NOT SET RHCS3 BIT #10 DEL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE

11235 TEST 67 TEST DBL (RHCS3 BIT #10) TEST X

11237 CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
SET UP FOR A TEN WORD READ REVERSE FROM AN ODD WORD BOUN
WITH BAI IN RHCS2 BIT #3 SET
DO A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
THIS SHOULD NOT SET RHCS3 BIT #10 DBL
CHECK RHCS3
CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE

11362 TEST 70 END OF ONE RH
THIS IS THE END OF TEST FOR ONE RH
IF THERE ARE MORE RH THEN THE PROGRAM
JUMPS TO TEST 2 FOR NEXT RH TEST
END PASS IS REACHED ONLY AFTER ALL RH ARE COMPLETE

CERHADO

1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042

11400

END OF PASS ROUTINE

11402 INCREMENT THE PASS NUMBER (\$PASS)
TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
IF THERES A MONITOR GO TO IT
IF THERE ISN'T JUMP TO TST2

11439

SUBROUTINES

11655

RS DATA TRANSFER SUBROUTINE

11657 THIS SUBROUTINE WRITES OR READS OR DOES A
WRITE CHECK ON CYLINDER 0 TRACK 0 SECTOR 0
IT ASSUMES THE RH REGISTERS ARE APPROPRIATELY
FILLED
WHEN IT RETURNS FROM THIS SUBROUTINE TO THE
MAIN PROGRAM IT MEANS THE TRANSFER IS COMPLETE

THE CALL IS BY A JSR R0, @COMND COMAND

11704

RPO4 DATA TRANSFER SUBROUTINE

11706 THIS SUBROUTINE WRITE/READ/WRITE CHECK ON THE RPO4 CYLIN
TRACK 0, SECTOR 0.

11708 IT ASSUMES THE RH REGISTERS ARE APPROPRIATELY FILLED.
WHEN IT RETURNS FROM THIS SUBROUTINE TO THE MAIN
PROGRAM IT MEANS THE COMMAND IS COMPLETE

THE CALL IS BY A JSR R0, @COMND COMMAND.

1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091

CERHADO

11758

TMO2 DATA TRANSFER SUBROUTINE

11760 THIS SUBROUTINE WRITES/READS/WRITE CHECKS ON THE TMO2-7L
ON THE FIRST BLOCK FROM BEGINNING OF TAPE (BOT)

11762 IT ASSUMES THE RH REGISTERS ARE APPROPRIATELY FILLED.
WHEN IT RETURNS FROM THIS SUBROUTINE TO THE
MAIN PROGRAM IT MEANS THE COMMAND IS COMPLETE.

THE CALL IS
JSR RO, @COMND

11915 THIS ROUTINE WILL ALLOW THE CHANGE OF THE BASE
ADDRESS FROM 176700 TO ANY TYPED VALUE

11982 *****

11985

SCOPE HANDLER ROUTINE

11987 THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG. (DISPLA<7
AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY<15:08'
THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:

SW14=1 LOOP ON TEST
SW11=1 INHIBIT ITERATIONS
SW09=1 LOOP ON ERROR
SW08=1 LOOP ON TEST IN SWP<7:0'
CALL

SCOPE ;:SCOPE=IOT

12051

*****<*****
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

12053 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIG
SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER
NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE
BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS
REPLACED WITH SPACES.

CALL:
MOV NUM, -(SP) ;:PUT THE BINARY NUMBER ON THE 5
TYPDS ;:GO TO THE ROUTINE

1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144

CERHADO

12:19

TYPE ROUTINE

1212! ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 B
THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE
NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CH
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
OR
TYPE
MESADR
2) USING A JSR INSTRUCTION
MOV PS,-(SP) ;:PUSH PROCESSOR STATUS WORD ON
JSR PC,\$TYPE ;:CALL TYPE ROUTINE
MESADDR ;:FIRST ADDRESS OF MESSAGE

12192 *****
TTY INPUT ROUTINE

12201 TK INITIALIZE ROUTINE
THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
CALL:
JSR PC,\$TKINT
RETURN

12218 TK SERVICE ROUTINE
THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
IT IN THE QUEUE.
IF THE CHARACTER IS A "CONTROL-C" (^C) \$TKINT IS CALLED AND
UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (OPE

12245 THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
CALL:
RDCHR ;:GET A CHARACTER FROM THE QUEUE
RETURN HERE ;:CHARACTER IS ON THE STACK
;:WITH PARITY BIT STRIPPED OFF

12266 THIS ROUTINE WILL INPUT A STRING FROM THE TTY
CALL:
RDLIN ;:INPUT A STRING FROM THE TTY
RETURN HERE ;:ADDRESS OF FIRST CHARACTER WIL
;:TERMINATOR WILL BE A BYTE OF A

1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195

CERHADO

12303

READ AN OCTAL NUMBER FROM THE TTY

12305 THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
CHANGE IT TO BINARY.
THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPE
FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUS
THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RE
CALL:
RDOCT ;:READ AN OCTAL NUMBER
RETURN HERE ;:LOW ORDER BITS ARE ON TOP OF T
;:HIGH ORDER BITS ARE IN \$HI0C*

12357

ERROR HANDLER ROUTINE

12359 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
HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPE0L
SW13=1 INHIBIT ERROR TYPEOUTS
SW10=1 BELL ON ERROR
SW09=1 LOOP ON ERROR
CALL ERROR N ;:ERROR=EMT AND N=ERROR ITEM NUMBER

12401

ERROR MESSAGE TYPEOUT ROUTINE

12402 THIS ROUTINE USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE
ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE
AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
IT IS A COPY OF THE \$ERRTYP SUBROUTINE FROM SYSMAC.
WITH ONLY MINOR CHANGES
FIRST IF SWITCH 6 IS SET AND SWITCH 8 RESET THEN
ALL REGISTER CONTENTS WILL BE TYPED BEFOR REPORTING THE ERROR

12409 SECOND IF THE CURRENT ERROR HAS THE SAME ITEM NUMBER
AS THE PREVIOUS ERROR THEN ONLY THE DATA WILL BE TYPED
AND NOT THE ERROR MESSAGE AND HEADER.

12627

1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238

CERHADO

12631

BINARY TO OCTAL (ASCII) AND TYPE

12633 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIG
OCTAL (ASCII) NUMBER AND TYPE IT.
\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS
CALL:

```
MOV     NUM, -(SP)      ;; NUMBER TO BE TYPED
TYPOS   ;; CALL FOR TYPEOUT
.BYTE  N                ;; N=1 TO 6 FOR NUMBER OF DIGITS
.BYTE  M                ;; M=1 OR 0
                        ;; 1=TYPE LEADING ZEROS
                        ;; 0=SUPPRESS LEADING ZER
```

\$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE
\$TYPOS OR \$TYPOC

CALL: MOV NUM, -(SP) ;; NUMBER TO BE TYPED
TYPON ;; CALL FOR TYPEOUT

\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

CALL: MOV NUM, -(SP) ;; NUMBER TO BE TYPED
TYPOC ;; CALL FOR TYPEOUT

12709

TRAP DECODER

12711 THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTIO
AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDR
OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
GO TO THAT ROUTINE.

12724

TRAP TABLE

12726 THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLE
BY THE "TRAP" INSTRUCTION.

1239
1240
1241
1242
1243
1244
1245
1246
1247

CERHADO

12746

POWER DOWN AND UP ROUTINES

12786

1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303

001000

000011
000012
000015
000200
177776
177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

000000

```

.TITLE CERMADO
*COPYRIGHT (C) 1975-1977
*DIGITAL EQUIPMENT CORP.
*MAYNARD, MASS. 01754
*
*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
*PACKAGE (MAINDEC-11-DZQAC-C1), MAR 24, 1976.
*
.SBTTL OPERATIONAL SWITCH SETTINGS
*
*      SWITCH              USE
*      -----
*      15                  HALT ON ERROR
*      14                  LOOP ON TEST
*      13                  INHIBIT ERROR TYPEOUTS
*      11                  INHIBIT ITERATIONS
*      10                  BELL ON ERROR
*      9                   LOOP ON ERROR
*      8                   LOOP ON TEST IN SWR<7:0>
*      7                   STOP FURTHER COMPARES IF SW08 IS LOW
*      6                   TYPE ALL REG. WITH ERROR IF SW8 LOW
.SBTTL BASIC DEFINITIONS
*INITIAL ADDRESS OF THE STACK POINTER *** 1000 ***
STACK= 1000
.EQUIV EMT,ERROR          ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE         ;;BASIC DEFINITION OF SCOPE CALL
*
*MISCELLANEOUS DEFINITIONS
HT= 11                   ;;CODE FOR HORIZONTAL TAB
LF= 12                   ;;CODE FOR LINE FEED
CR= 15                   ;;CODE FOR CARRIAGE RETURN
CRLF= 200                ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776              ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774           ;;STACK LIMIT REGISTER
PIRQ= 177772             ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570            ;;HARDWARE SWITCH REGISTER
DDISP= 177570           ;;HARDWARE DISPLAY REGISTER
*
*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0                  ;;GENERAL REGISTER
R1= %1                  ;;GENERAL REGISTER
R2= %2                  ;;GENERAL REGISTER
R3= %3                  ;;GENERAL REGISTER
R4= %4                  ;;GENERAL REGISTER
R5= %5                  ;;GENERAL REGISTER
R6= %6                  ;;GENERAL REGISTER
R7= %7                  ;;GENERAL REGISTER
R6=SP                   ;;STACK POINTER
R7=PC                   ;;PROGRAM COUNTER
*
*PRIORITY LEVEL DEFINITIONS
PRO= 0                  ;;PRIORITY LEVEL 0
    
```

1304	000040	PR1=	40	:: PRIORITY LEVEL	1
1305	000100	PR2=	100	:: PRIORITY LEVEL	2
1306	000140	PR3=	140	:: PRIORITY LEVEL	3
1307	000200	PR4=	200	:: PRIORITY LEVEL	4
1308	000240	PR5=	240	:: PRIORITY LEVEL	5
1309	000300	PR6=	300	:: PRIORITY LEVEL	6
1310	000340	PR7=	340	:: PRIORITY LEVEL	7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

1312	100000	SW15=	100000
1313	040000	SW14=	40000
1314	020000	SW13=	20000
1315	010000	SW12=	10000
1316	004000	SW11=	4000
1317	002000	SW10=	2000
1318	001000	SW09=	1000
1319	000400	SW08=	400
1320	000200	SW07=	200
1321	000100	SW06=	100
1322	000040	SW05=	40
1323	000020	SW04=	20
1324	000010	SW03=	10
1325	000004	SW02=	4
1326	000002	SW01=	2
1327	000001	SW00=	1
1328		.EQUIV	SW09, SW9
1329		.EQUIV	SW08, SW8
1330		.EQUIV	SW07, SW7
1331		.EQUIV	SW06, SW6
1332		.EQUIV	SW05, SW5
1333		.EQUIV	SW04, SW4
1334		.EQUIV	SW03, SW3
1335		.EQUIV	SW02, SW2
1336		.EQUIV	SW01, SW1
1337		.EQUIV	SW00, SW0

.*DATA BIT DEFINITIONS (BIT00 TO BIT15

1340	100000	BIT15=	100000
1341	040000	BIT14=	40000
1342	020000	BIT13=	20000
1343	010000	BIT12=	10000
1344	004000	BIT11=	4000
1345	002000	BIT10=	2000
1346	001000	BIT09=	1000
1347	000400	BIT08=	400
1348	000200	BIT07=	200
1349	000100	BIT06=	100
1350	000040	BIT05=	40
1351	000020	BIT04=	20
1352	000010	BIT03=	10
1353	000004	BIT02=	4
1354	000002	BIT01=	2
1355	000001	BIT00=	1
1356		.EQUIV	BIT09, BIT9
1357		.EQUIV	BIT08, BIT8
1358		.EQUIV	BIT07, BIT7

```

1360 .EQUIV BIT06,BIT6
1361 .EQUIV BIT05,BIT5
1362 .EQUIV BIT04,BIT4
1363 .EQUIV BIT03,BIT3
1364 .EQUIV BIT02,BIT2
1365 .EQUIV BIT01,BIT1
1366 .EQUIV BIT00,BIT0
1368 .*BASIC "CPJ" TRAP VECTOR ADDRESSES
1369 ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
1370 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
1371 TBITVEC=14 ;: "T" BIT
1372 TRIVEC= 14 ;: TRACE TRAP
1373 BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
1374 IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
1375 PWRVEC= 24 ;: POWER FAIL
1376 EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**
1377 TRAPVEC=34 ;: "TRAP" TRAP
1378 TKVEC= 60 ;: TTY KEYBOARD VECTOR
1379 TPVEC= 64 ;: TTY PRINTER VECTOR
1380 PIRQVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR
1381
1382 .SBTTL TRAP CATCHER
1383
1384 .=0
1385 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
1386 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1387 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1388
1389 000174 000000 DISPREG: .WORD 0 ;: SOFTWARE DISPLAY REGISTER
1390 000176 000000 SWREG: .WORD 0 ;: SOFTWARE SWITCH REGISTER
1391
1392 000200 000137 005522 .SBTTL STARTING ADDRESS(ES)
1393 .JMP 2*BEGIN ;: JUMP TO STARTING ADDRESS OF PROGRAM
1394 000046 000046 .=46
1395 000052 000052 SENDAD
1396 000052 000000 .=52
1397 000210 000210 000000
1398 000210 000137 005532 .=210
1399 .JMP 2*BEGIN2 ;: JUMP SELECT TEST
1400 ;*STARTING ADDRESS 200 FOR NORMAL STARTS
1401 ;*THIS WILL TEST ALL RPO4'S ON THE SYSTEM A SINGLE DRIVE AT A TIME
1402 ;*
1403 ;*STARTING ADDRESS 210 WILL TEST ONLY ONE SPECIFIED DRIVE
1404 ;*
1405 ;*STARTING ADDRESS 220 WILL JUMP OVER THE TESTS REQUIRING AN OPERATOR
1406 ;*AT THE DRIVE
1407 .SBTTL MEMORY MANAGEMENT DEFINITIONS
1408
1409 ;*KT11 VECTOR ADDRESS
1410 MMVEC= 250
1411
1412 ;*KT11 STATUS REGISTER ADDRESSES
1413
1414 177572 SRO= 177572
1415 177574 SRI= 177574
    
```

177576
172516

SR2= 177576
SR3= 172516

;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

172300
172302
172304
172306
172310
172312
172314
172316

KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316

;*KERNEL "I" PAGE ADDRESS REGISTERS

172340
172342
172344
172346
172350
172352
172354
172356

KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356

301110

::*****
.=1110


```

1443 .SBTTL COMMON TAGS
1444
1445 ::*****
1446 ::*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
1447 ::*USED IN THE PROGRAM.
1448
1449 001100 . =1100
1450 001100 $CMTAG: .WORD 0 :: START OF COMMON TAGS
1451 001100 000000 $PASS: .WORD 0 :: CONTAINS PASS COUNT
1452 001102 000 .STSTNM: .BYTE 000 :: CONTAINS THE TEST NUMBER
1453 001103 000 .SERFLG: .BYTE 000 :: CONTAINS ERROR FLAG
1454 001104 000000 .STCNT: .WORD 000 :: CONTAINS SUBTEST ITERATION COUNT
1455 001106 000000 .SLPADR: .WORD 000 :: CONTAINS SCOPE LOOP ADDRESS
1456 001110 000000 .SLPERR: .WORD 000 :: CONTAINS SCOPE RETURN FOR ERRORS
1457 001112 000000 .SERTTL: .WORD 000 :: CONTAINS TOTAL ERRORS DETECTED
1458 001114 000 .SITEMB: .BYTE 000 :: CONTAINS ITEM CONTROL E TE
1459 001115 001 .SERMAX: .BYTE 1 :: CONTAINS MAX. ERRORS PER TEST
1460 001116 000000 .SERRPC: .WORD 000 :: CONTAINS PC OF LAST ERROR INSTRUCTION
1461 001120 000000 .SGDADR: .WORD 000 :: CONTAINS ADDRESS OF 'GOOD' DATA
1462 001122 000000 .SBDADR: .WORD 000 :: CONTAINS ADDRESS OF 'BAD' DATA
1463 001124 000000 .SGDDAT: .WORD 000 :: CONTAINS 'GOOD' DATA
1464 001126 000000 .SBDDAT: .WORD 000 :: CONTAINS 'BAD' DATA
1465 001130 000000 .WORD 000 :: RESERVED--NOT TO BE USED
1466 001132 000000 .WORD 000
1467 001134 000 .SAUTOB: .BYTE 000 :: AUTOMATIC MODE INDICATOR
1468 001135 000 .SINTAG: .BYTE 000 :: INTERRUPT MODE INDICATOR
1469 001136 000000 .WORD 0
1470 001140 177570 .SWR: .WORD DSWR :: ADDRESS OF SWITCH REGISTER
1471 001142 177570 .DISPLAY: .WORD DDISP :: ADDRESS OF DISPLAY REGISTER
1472 001144 177560 .STKS: 177560 :: TTY KBD STATUS
1473 001146 177562 .STKB: 177562 :: TTY KBD BUFFER
1474 001150 177564 .STPS: 177564 :: TTY PRINTER STATUS REG. ADDRESS
1475 001152 177566 .STPB: 177566 :: TTY PRINTER BUFFER REG. ADDRESS
1476 001154 000 .SNUL: .BYTE 0 :: CONTAINS NULL CHARACTER FOR FILLS
1477 001155 002 .SFILLS: .BYTE 2 :: CONTAINS # OF FILLER CHARACTERS REQUIRED
1478 001156 012 .SFILLC: .BYTE 12 :: INSERT FILL CHARS. AFTER A "LINE FEED"
1479 001157 000 .STPFLG: .BYTE 0 :: "TERMINAL AVAILABLE" FLAG (BIT 07 = 0 = ES
1480 001160 000000 .SREGAD: .WORD 0 :: CONTAINS THE ADDRESS FROM
1481 WHICH ($REGO) WAS OBTAINED
1482 001162 000000 .SREGO: .WORD 0 :: CONTAINS (($REGAD)+0)
1483 001164 000000 .SREG1: .WORD 000 :: CONTAINS (($REGAD)+2)
1484 001166 000000 .SREG2: .WORD 000 :: CONTAINS (($REGAD)+4)
1485 001170 000000 .SREG3: .WORD 000 :: CONTAINS (($REGAD)+6)
1486 001172 000000 .SREG4: .WORD 000 :: CONTAINS (($REGAD)+10)
1487 001174 000000 .SREG5: .WORD 000 :: CONTAINS (($REGAD)+12)
1488 001176 000000 .STMP0: .WORD 000 :: USER DEFINED
1489 001200 000000 .STMP1: .WORD 000 :: USER DEFINED
1490 001202 000000 .STMP2: .WORD 000 :: USER DEFINED
1491 001204 000000 .STMP3: .WORD 000 :: USER DEFINED
1492 001206 000000 .STMP4: .WORD 000 :: USER DEFINED
1493 001210 000000 .STMP5: .WORD 000 :: USER DEFINED
1494 001212 000000 .STIMES: 0 :: MAX. NUMBER OF ITERATIONS
1495 001214 000000 .SESCAPE: 0 :: ESCAPE ON ERROR ADDRESS
1496 001216 177607 000377 .SBELL: .ASCII <207 <377 <377> :: CODE FOR BELL
1497 001222 077 .SQUES: .ASCII ? :: QUESTION MARK
1498 001223 015 .SCRLF: .ASCII <15 :: CARRIAGE RETURN

```

G03

CERHADD MACY11 30:1046: 21-DEC-77 13:06 PAGE 33
CERHAD.P11 21-DEC-77 12:48 COMMON TAGS

SEQ 0032

1499 001224 000012
1500

\$LF: .ASCII 12 ::LINE FEED
:*****

ERROR POINTER TABLE

1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556

.SBTTL ERROR POINTER TABLE

:*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
:*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
:*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
:*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
:*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

:* EM ::POINTS TO THE ERROR MESSAGE
:* DH ::POINTS TO THE DATA HEADER
:* DT ::POINTS TO THE DATA
:* DF ::POINTS TO THE DATA FORMAT

001226

\$ERRTB:

:*****
:

001226 046676

:ITEM 1

EM1

: THE RH BASE ADDRESS WAS 772040
: INDICATING A R504 OR R503
: BUT THE DRIVE TYPE DID NOT
: CONTAIN 0,1,2,3 OR 4
: PC
: RMDT
: \$ERRPC,\$BDDAT,0
: 0

001230 065117

DH1

001232 066552

DT1

001234 066734

DF1

:ITEM 2

EM2

: THE RH BASE ADDRESS WAS 776700
: INDICATING AN RPO4, RPO5, OR RPO6
: BUT THE DRIVE TYPE DID NOT
: CONTAIN 20020 24020, 20021, 24021,
: 20022, OR 24022

001240 065117

DH1

001242 066552

DT1

001244 066734

DF1

: PC
: RMDT
: \$ERRPC,\$BDDAT,0
: 0

:ITEM 3

EM3

001246 047252

: THE RH BASE ADDRESS WAS 772440
: INDICATING A TMO2. BUT THE
: DRIVE TYPE DID NOT CONTAIN
: 142010 INDICATING NO
: TMO2
: PC
: RMDT
: \$ERRPC,\$BDDAT,0
: 0,0

001250 065117

DH1

001252 066552

DT1

001254 066734

DF1

:ITEM 4

1557	001256	047406	EM4		: THE RH BASE ADDRESS WAS 776300
1558					: INDICATING MORE THAN ONE
1559					: RH DEVICE. BUT THE DRIVE
1560					: TYPE DID NOT CONTAIN
1561					: 0, 1, 2, 3, 4, 20020, 24020, 20021, 24021,
1562					: 20022, 24022, OR 142010 INDICATING NO
1563					: RH DEVICE IS PRESENT
1564	001260	065117	DH1		: PC
1565					: RHDT
1566	001262	066552	DT1		: \$ERRPC, \$BDDAT, 0
1567	001264	066734	DF1		: 0, 0
1568					
1569				: ITEM 5	
1570	001266	047705	EM5		: THE UNIBUS TIMED OUT
1571					: FOR EACH OF THE FOLLOWING
1572					: ADDRESSES, INDICATING
1573					: NO RH ON THE SYSTEM
1574					
1575					: SO ABORT PROGRAM
1576	001270	065126	DH5		: PC
1577					: UNIBUS ADDRESSES ON WHICH
1578					: TIME OUT OCCURRED
1579	001272	066560	DT5		: \$ERRPC, RSCS1, RPCS1, TMC51, MIXCS1
1580	001274	066736	DF5		: 0, 0, 0, 0, 0
1581					
1582				: ITEM 6	
1583	001276	050060	EM6		: AFTER AN RH CLEAR (BIT #5
1584					: IN RHCS2) RHCS2 DOES NOT
1585					: HAVE ONLY IR AND UNIT
1586					: NUMBER
1587	001300	065202	DH6		: PC
1588					: TEST NO
1589					: RHCS2 GOOD
1590					: RHCS2 BAD
1591	001302	066574	DT6		: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1592	001304	066743	DF6		: 0, 0, 0, 0
1593					
1594				: ITEM 7	
1595	001306	050176	EM7		: AFTER CLEARING THE RH AND
1596					: WRITING ONE WORD INTO
1597					: RHDB AND READING IT
1598					: BACK CAUSED RHDB TO
1599					: HAVE WRONG VALUE GIVEN IN BAD RHDB
1600	001310	065245	DH7		: PC
1601					: TEST NO
1602					: RHDB GOOD
1603					: RHDB BAD
1604	001312	066574	DT6		: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1605	001314	066743	DF6		: 0, 0, 0, 0
1606					
1607				: ITEM 10	
1608	001316	050370	EM10		: AFTER CLEARING THE RH AND
1609					: WRITING ONE WORD INTO
1610					: RHDB AND READING IT
1611					: BACK, CAUSED RHCS2 TO
1612					: HAVE WRONG DATA

1613				: GIVEN IN BAD RHCS2
1614	001320	065202	DH6	: PC
1615				: TEST NUMBER
1616				: RHCS2 GOOD
1617				: RHCS2 BAD
1618	001322	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1619	001324	066743	DFE	: 0,0,0,0
1620				
1621			: ITEM 11	
1622	001326	050563	EM11	: AFTER CLEARING THE RH AND
1623				: WRITING ONE WORD INTO
1624				: RHDB AND READING IT
1625				: BACK, CAUSED RHCS1 TO
1626				: HAVE WRONG DATA
1627				: GIVEN IN BAD RHCS1
1628	001330	065305	DH11	: PC
1629				: TEST NUMBER
1630				: RHCS1 GOOD
1631				: RHCS1 BAD
1632	001332	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1633	001334	066743	DFE	: 0,0,0,0
1634				
1635			: ITEM 12	
1636	001336	050756	EM12	: AFTER CLEARING THE RH AND
1637				: WRITING ONE WORD INTO
1638				: RHDB AND READING IT
1639				: BACK, CAUSED RHCS3 TO
1640				: HAVE WRONG DATA
1641				: GIVEN IN BAD RHCS3
1642	001340	065347	DH12	: PC
1643				: TEST NUMBER
1644				: RHCS3 GOOD
1645				: RHCS3 BAD
1646	001342	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1647	001344	066743	DFE	: 0,0,0,0
1648				
1649			: ITEM 13	
1650	001346	051151	EM13	: AFTER CLEARING THE RH AND
1651				: WRITING ONE WORD INTO
1652				: RHDB AND READING IT
1653				: BACK, CAUSED RHBA TO
1654				: HAVE WRONG DATA
1655				: GIVEN IN BAD RHBA
1656	001350	065411	DH13	: PC
1657				: TEST NUMBER
1658				: RHBA GOOD
1659				: RHBA BAD
1660	001352	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1661	001354	066743	DFE	: 0,0,0,0
1662				
1663			: ITEM 14	
1664	001356	051343	EM14	: AFTER CLEARING THE RH AND
1665				: WRITING ONE WORD INTO
1666				: RHDB AND READING IT
1667				: BACK, CAUSED RHBAE TO
1668				: HAVE WRONG DATA

1669				: GIVEN IN BAD RHBAE
1670	001360	065451	DH14	: PC
1671				: TEST NUMBER
1672				: RHBAE GOOD
1673				: RHBAE BAD
1674	001362	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1675	001364	066743	DF6	: 0,0,0,0
1676				
1677			: ITEM 15	
1678	001366	051536	EM15	: AFTER CLEARING THE RH AND
1679				: WRITING ONE WORD INTO
1680				: RHDB AND READING IT
1681				: BACK, CAUSED RHWC TO
1682				: HAVE WRONG DATA
1683				: GIVEN IN BAD RHWC
1684	001370	065513	DH15	: PC
1685				: TEST NUMBER
1686				: RHWC GOOD
1687				: RHWC BAD
1688	001372	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1689	001374	066743	DF6	: 0,0,0,0
1690				
1691			: ITEM 16	
1692	001376	051730	EM16	: AFTER CLEARING THE RH AND
1693				: WRITING ONE WORD INTO
1694				: RHDB
1695				: CAUSED RHCS2 TO
1696				: HAVE WRONG DATA
1697				: GIVEN IN BAD RHCS2
1698	001400	065202	DH6	: PC
1699				: TEST NUMBER
1700				: RHCS2 GOOD
1701				: RHCS2 BAD
1702	001402	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1703	001404	066743	DF6	: 0,0,0,0
1704				
1705			: ITEM 17	
1706	001406	052123	EM17	: AFTER CLEARING THE RH AND
1707				: WRITING TWO WORD INTO
1708				: RHDB AND READING IT
1709				: BACK, CAUSED RHDB TO
1710				: HAVE WRONG DATA
1711				: GIVEN IN BAD RHDB
1712	001410	065245	DH7	: PC
1713				: TEST NUMBER
1714				: RHDB GOOD
1715				: RHDB BAD
1716	001412	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1717	001414	066743	DF6	: 0,0,0,0
1718				
1719			: ITEM 20	
1720	001416	052315	EM20	: AFTER CLEARING THE RH AND
1721				: WRITING TWO WORD INTO
1722				: RHDB AND READING ONE
1723				: BACK, CAUSED RHCS2 TO
1724				: HAVE WRONG DATA

1725				: GIVEN IN BAD RHCS2
1726	001420	065202	DM6	: PC
1727				: TEST NUMBER
1728				: RHCS2 GOOD
1729				: RHCS2 BAD
1730	001422	066574	DT6	: \$ERRPC TSTNM, \$GDDAT, \$BDDAT
1731	001424	066743	DF6	: 0,0,0,0
1732				
1733			: ITEM 21	
1734	001426	052511	EM21	: AFTER CLEARING THE RH AND
1735				: WRITING TWO WORD INTO
1736				: RHDB AND READING IT
1737				: BACK TWICE, CAUSED RHDB TO
1738				: HAVE WRONG DATA
1739				: GIVEN IN BAD RHDB
1740	001430	065245	DM7	: PC
1741				: TEST NUMBER
1742				: RHDB GOOD
1743				: RHDB BAD
1744	001432	066574	DT6	: \$ERRPC TSTNM, \$GDDAT, \$BDDAT
1745	001434	066743	DF6	: 0,0,0,0
1746				
1747			: ITEM 22	
1748	001436	052711	EM22	: AFTER CLEARING THE RH AND
1749				: WRITING TWO WORD INTO
1750				: RHDB AND READING IT
1751				: BACK TWICE, CAUSED RHCS2 TO
1752				: HAVE WRONG DATA
1753				: GIVEN IN BAD RHCS2
1754	001440	065202	DM6	: PC
1755				: TEST NUMBER
1756				: RHCS2 GOOD
1757				: RHCS2 BAD
1758	001442	066574	DT6	: \$ERRPC TSTNM, \$GDDAT, \$BDDAT
1759	001444	066743	DF6	: 0,0,0,0
1760				
1761			: ITEM 23	
1762	001446	053112	EM23	: AFTER CLEARING THE RH AND
1763				: WRITING EIGHT WORD INTO
1764				: RHDB
1765				: CAUSED RHCS2 TO
1766				: HAVE WRONG DATA
1767				: GIVEN IN BAD RHCS2
1768	001450	065202	DM6	: PC
1769				: TEST NUMBER
1770				: RHCS2 GOOD
1771				: RHCS2 BAD
1772	001452	066574	DT6	: \$ERRPC TSTNM, \$GDDAT, \$BDDAT
1773	001454	066743	DF6	: 0,0,0,0
1774				
1775			: ITEM 24	
1776	001456	053307	EM24	: THE RH WAS CLEARED
1777				: AND A PATTERN OF 8 WORDS
1778				: WERE WRITTEN INTO RHDB
1779				: READING RHDB FOR THE
1780				: "N" TH. TIME GAVE WRONG

1781				: VALUE IN RHDB
1782				: N IS GIVEN IN "WORD NO"
1783	001460	065553	DH24	: PC
1784				: TEST NO
1785				: WORD NO
1786				: RHDB GOOD
1787				: RHDB BAD
1788	001462	066606	DT24	: \$ERRPC, TSTNM, SILJNM, \$GDDAT, \$BDDAT
1789	001464	066747	DF24	: 0,0,0,0,0
1790				
1791			: ITEM 25	
1792	001466	053537	EM25	: THE RH WAS CLEARED AND
1793				: A PATTERN OF 8 WORDS WERE
1794				: WRITTEN INTO RHDB
1795				: AFTER READING ALL 8 WORDS
1796				: FOLLOWING REGISTER CONTAINED WRONG
1797				: VALUE
1798	001470	065202	DH6	: PC
1799				: TEST NO
1800				: RHCS2 GOOD
1801				: RHCS2 BAD
1802	001472	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1803	001474	066743	DF6	: 0,0,0,0
1804				
1805			: ITEM 26	
1806	001476	053537	EM25	
1807	001500	065305	DH11	
1808	001502	066574	DT6	
1809	001504	066743	DF6	: 0,0
1810				
1811			: ITEM 27	
1812	001506	053537	EM25	
1813	001510	065347	DH12	: PC
1814				: TEST NO
1815				: RHCS3 GOOD
1816				: RHCS3 BAD
1817	001512	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1818	001514	066743	DF6	: 0,0,0,0
1819				
1820			: ITEM 30	
1821	001516	053537	EM25	: PC
1822	001520	065411	DH13	: TEST NO
1823				: RHBA GOOD
1824				: RHBA BAD
1825				: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1826	001522	066574	DT6	: 0,0,0,0
1827	001524	066743	DF6	
1828				
1829			: ITEM 31	
1830	001526	053537	EM25	: PC
1831	001530	065451	DH14	: TEST NO
1832				: RHBAE GOOD
1833				: RHBAE BAD
1834				: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1835	001532	066574	DT6	: 0,0,0,0
1836	001534	066743	DF6	

1837					
1838			: ITEM 32		
1839	001536	053537		EM25	
1840	001540	065513		DH15	: PC
1841					: TEST NO
1842					: RHWC GOOD
1843					: RHWC BAD
1844	001542	066574		DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1845	001544	066743		DF6	: 0,0,0,0
1846					
1847			: ITEM 33		
1848	001546	053766		EM33	: SETTING RH CLEAR BIT #5
1849					: IN RHCS2 CAUSED
1850					: ERROR REGISTER 1 TO HAVE
1851					: WRONG VALUE
1852	001550	065623		DH33	: PC
1853					: TEST NO
1854					: RHER1 GOOD
1855					: RHER1 BAD
1856	001552	066574		DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1857	001554	066743		DF6	
1858					
1859			: ITEM 34		
1860	001556	054102		EM34	: AN RH LCLEAR WAS GIVEN
1861					: RHER1 WAS CHECKED TO HAVE ZERO
1862					: PAT (BIT #4 RHCS2) WAS SET
1863					: TO INVERT PARITY CHECKING
1864					: RHER1 WAS READ BUT DID NOT
1865					: CONTAIN WHAT IS IN RHER1 GOOD
1866	001560	065623		DH33	: PC
1867					: TEST NO
1868					: RHER1 GOOD
1869					: RHER1 BAD
1870	001562	066574		DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1871	001564	066743		DF6	: 0,0,0,0
1872					
1873			: ITEM 35		
1874	001566	054344		EM35	: RH CLEAR WAS GIVEN
1875					: RHER1 WAS CHECKED
1876					: PAT (BIT #4 RHCS2) WAS SET
1877					: AND RHER1 WAS READ
1878					: RHCS1 SHOULD HAVE RDY
1879					: SC AND MCPE SET
1880	001570	065305		DH11	: PC
1881					: TEST NO
1882					: RHCS1 GOOD
1883					: RHCS1 BAD
1884	001572	066574		DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1885	001574	066743		DF6	: 0,0,0,0
1886					
1887			: ITEM 36		
1888	001576	054542		EM36	: RH CLEAR WAS GIVEN
1889					: RHER1 WAS CHECKED
1890					: PAT (RHCS2-BIT #4) WAS SET
1891					: RHER1 WAS READ AND CHECKED
1892					: "1" WAS WRITTEN INTO "TRF"-RHCS1

CERHADO MACY 11 30 1046, 21-DEC-77 13:06 PAGE 41
CERHAD.1 11 21-DEC-77 12:48 ERROR POINTER TABLE

SEE CC40

:893
:894
:895

:ON CHECKING RMCS!
:IT DID NOT CONTAIN SC,MCPE
:AND RD/

1896	001600	065305	DH11	: PC
1897				: TEST NUMBER
1898				: RHCS1 GOOD
1899				: RHCS1 BAD
1900	001602	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1901	001604	066743	DF6	: 0,0,0,0
1902				
1903			: ITEM 37	
1904	001606	055017	EM37	: RH WAS CLEARED
1905				: RHER1 WAS CHECKED
1906				: PAT (RHCS2-BIT #4) WAS SET
1907				: RHER1 WAS READ
1908				: "1" WAS WRITTEN IN "TRE" RHCS1
1909				: AN RH CLEAR WAS TO
1910				: GIVE WHAT IS IN "GOOD"
1911				: BUT GAVE WHAT IS IN "BAL"
1912	001610	065305	DH11	: PC
1913				: TEST NO
1914				: RHCS1 GOOD
1915				: RHCS1 BAD
1916	001612	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1917	001614	066743	DF6	: 0,0,0,0
1918				
1919			: ITEM 40	
1920	001616	055017	EM37	
1921	001620	065202	DH6	
1922	001622	066574	DT6	
1923	001624	066743	DF6	
1924				
1925			: ITEM 41	
1926	001626	055017	EM37	
1927	001630	065347	DH12	
1928	001632	066574	DT6	
1929	001634	066743	DF6	
1930				
1931			: ITEM 42	
1932	001636	055017	EM37	
1933	001640	065411	DH13	
1934	001642	066574	DT6	
1935	001644	066743	DF6	
1936				
1937			: ITEM 43	
1938	001646	055017	EM37	
1939	001650	065451	DH14	
1940	001652	066574	DT6	
1941	001654	066743	DF6	
1942				
1943			: ITEM 44	
1944	001656	055017	EM37	
1945	001660	065513	DH15	
1946	001662	066574	DT6	
1947	001664	066743	DF6	
1948				
1949			: ITEM 45	
1950	001666	055321	EM45	: AFTER AN RH CLEAR
1951				: "1" WAS WRITTEN IN THE DISK

1952				: ADDRESS REGISTER (IN TAPE
1953				: DRIVES CALLED FRAME COUN
1954				: PAT IN RHCSI WAS SET
1955				: ON READING RHDA (IN TAPE
1956				: DRIVES RHFC) IT DID NOT
1957				: CONTAIN "1"
1958				: RHDA=RHFC
1959	001670	065665	DH45	: PC
1960				: TEST NO
1961				: RHDA GOOD
1962				: RHDA BAD
1963	001672	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1964	001674	066743	DF6	: 0.0.0.0
1965				
1966			: ITEM 46	
1967	001676	055557	EM46	: AFTER AN RH CLEAR
1968				: "1" WAS WRITTEN IN THE
1969				: DISK ADDRESS REGISTER (IN
1970				: TAPE
1971				: PAT IN RHCSI WAS SET
1972				: ON READING RHDA (IN TAPE
1973				:) FOLLOWING
1974				: REGISTER DID NOT CONTAIN
1975				: WHAT IS IN "GOOD"
1976	001700	065305	DH11	: PC
1977				: TEST NO
1978				: RHCSI GOOD
1979				: RHCSI BAD
1980	001702	066574	DT6	: \$ERRPC, TSTNM, \$GDDAT, \$BDDAT
1981	001704	066743	DF6	
1982				
1983			: ITEM 47	
1984	001706	055557	EM46	
1985	001710	065202	DH6	
1986	001712	066574	DT6	
1987	001714	066743	DF6	
1988				
1989			: ITEM 50	
1990	001716	055557	EM46	
1991	001720	065347	DH12	
1992	001722	066574	DT6	
1993	001724	066743	DF6	
1994				
1995			: ITEM 51	
1996	001726	055557	EM46	
1997	001730	065411	DH13	
1998	001732	066574	DT6	
1999	001734	066743	DF6	
2000				
2001			: ITEM 52	
2002	001736	055557	EM46	
2003	001740	065451	DH14	
2004	001742	066574	DT6	
2005	001744	066743	DF6	
2006				
2007			: ITEM 53	

2008	001746	055557		EM46	
2009	001750	065513		DH15	
2010	001752	066574		DT6	
2011	001754	066743		DF6	
2012					
2013			: ITEM 54		
2014	001756	056060		EM54	: AN RH CLEAR (RHCS2 BIT #5)
2015					: WAS GIVEN
2016					: A16 (RHCS1 BIT #8) WAS SET
2017					: ON READING THE FOLLOWING
2018					: REGISTER IT DID NOT
2019					: CONTAIN WHAT IS IN "GOOD"
2020	001760	065305		DH11	
2021	001762	066574		DT6	
2022	001764	066743		DF6	
2023					
2024			: ITEM 55		
2025	001766	056060		EM54	
2026	001770	065451		DH14	
2027	001772	066574		DT6	
2028	001774	066743		DF6	
2029					
2030			: ITEM 56		
2031	001776	056271		EM56	: AN RH CLEAR (RHCS2 BIT #5)
2032					: WAS GIVEN
2033					: A16 WAS WRITTEN IN RHCS1
2034					: ALL ZEROS WERE WRITTEN IN RHBAE
2035					: THE FOLLOWING REGISTER DID
2036					: NOT CONTAIN WHAT IS IN "GOOD"
2037	002000	065451		DH14	
2038	002002	066574		DT6	
2039	002004	066743		DF6	
2040					
2041			: ITEM 57		
2042	002006	056271		EM56	
2043	002010	065305		DH11	
2044	002012	066574		DT6	
2045	002014	066743		DF6	
2046					
2047			: ITEM 60		
2048	002016	056527		EM60	: AN RH CLEAR (RHCS2 BIT #5)
2049					: WAS GIVEN
2050					: A17 (RHCS1 BIT #9) WAS SET
2051					: ON READING THE FOLLOWING
2052					: REGISTER IT DID NOT
2053					: CONTAIN WHAT IS IN "GOOD"
2054	002020	065305		DH11	
2055	002022	066574		DT6	
2056	002024	066743		DF6	
2057					
2058			: ITEM 61		
2059	002026	056527		EM60	
2060	002030	065451		DH14	
2061	002032	066574		DT6	
2062	002034	066743		DF6	
2063					

2064			: ITEM 62		
2065	002036	056740	EM62		: AN RH CLEAR (RHCS2 BIT #5)
2066					: WAS GIVEN
2067					: A17 WAS WRITTEN IN RHCS1
2068					: ALL ZEROS WERE WRITTEN IN RHBAE
2069					: THE FOLLOWING REGISTER DID
2070					: NOT CONTAIN WHAT IS IN "GOOD"
2071	002040	065451	DH14		
2072	002042	066574	DT6		
2073	002044	066743	DF6		
2074					
2075			: ITEM 63		
2076	002046	056740	EM62		
2077	002050	065305	DH11		
2078	002052	066574	DT6		
2079	002054	066743	DF6		
2080					
2081			: ITEM 64		
2082	002056	057175	EM64		: AN RH CLEAR (RHCS2 BIT #5)
2083					: WAS GIVEN
2084					: IE (RHCS1 BIT #6) WAS SET
2085					: ON READING THE FOLLOWING
2086					: REGISTER IT DID NOT
2087					: CONTAIN WHAT IS IN "GOOD"
2088	002060	065305	DH11		
2089	002062	066574	DT6		
2090	002064	066743	DF6		
2091					
2092			: ITEM 65		
2093	002066	057175	EM64		
2094	002070	065347	DH12		
2095	002072	066574	DT6		
2096	002074	066743	DF6		
2097					
2098			: ITEM 66		
2099	002076	057175	EM64		: AN RH CLEAR (RHCS2 BIT #5)
2100					: WAS GIVEN
2101					: A16 WAS WRITTEN IN RHCS1
2102					: ALL ZEROS WERE WRITTEN IN RHBAE
2103					: THE FOLLOWING REGISTER DID
2104					: NOT CONTAIN WHAT IS IN "GOOD"
2105	002100	065347	DH12		
2106	002102	066574	DT6		
2107	002104	066743	DF6		
2108					
2109			: ITEM 67		
2110	002106	057175	EM64		
2111	002110	065305	DH11		
2112	002112	066574	DT6		
2113	002114	066743	DF6		
2114					
2115			: ITEM 70		
2116	002116	057644	EM70		: RH CLEAR WAS GIVEN
2117					: TWO SUCCESSIVE "GO" (RHCS1 BIT #3)
2118					: WAS GIVEN WITHOUT GIVING
2119					: TIME FOR FIRST "GO" TO

1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175

002120 065305
002122 066574
002124 066743

002126 057644
002130 065202
002132 066574
002134 066743

002136 057644
002140 065347
002142 066574
002144 066743

002146 057644
002150 065411
002152 066574
002154 066743

002156 057644
002160 065451
002162 066574
002164 066743

002166 057644
002170 065513
002172 066574
002174 066743

002176 060120

002200 065305
002202 066574
002204 066743

002206 060120
002210 065202
002212 066574
002214 066743

DH11
DT6
DF6

: ITEM 71
EM70
DH6
DT6
DF6

: ITEM 72
EM70
DH12
DT6
DF6

: ITEM 73
EM70
DH13
DT6
DF6

: ITEM 74
EM70
DH14
DT6
DF6

: ITEM 75
EM70
DH15
DT6
DF6

: ITEM 76
EM76

: ITEM 77
EM76
DH6
DT6
DF6

: COMPLETE
: THE FOLLOWING REGISTER
: DID NOT CONTAIN WHAT IS
: IN "GOOD"

: RH CLEAR WAS GIVEN A 2 WORD
: WRITE WAS DONE FROM A
: LOCATION TAGED WRFROM
: AND WITH BAI BIT SET
: AT THE END OF THE WRITE
: THE FOLLOWING REGISTER DID
: NOT CONTAIN WHAT IS
: IN GOOD

2176					
2177			: ITEM 100		
2178	002216	060120	EM76		
2179	002220	065347	DH12		
2180	002222	066574	DT6		
2181	002224	066743	DF6		
2182					
2183			: ITEM 101		
2184	002226	060120	EM76		
2185	002230	065411	DH13		
2186	002232	066574	DT6		
2187	002234	066743	DF6		
2188					
2189			: ITEM 102		
2190	002236	060120	EM76		
2191	002240	065451	DH14		
2192	002242	066574	DT6		
2193	002244	066743	DF6		
2194					
2195			: ITEM 103		
2196	002246	060120	EM76		
2197	002250	065513	DH15		
2198	002252	066574	DT6		
2199	002254	066743	DF6		
2200					
2201			: ITEM 104		
2202	002256	060413	EM104		
2203				: RH CLEAR WAS GIVEN AN	
2204				: IPCK BIT SHOWN IN "IPCK" WAS SET	
2205				: ZEROS WERE MOVED INTO RHOB	
2206				: ON READING RHOB	
2207				: THE FOLLOWING REGISTER	
2208				: DID NOT CONTAIN WHAT	
2209				: IS IN GOOD	
2209	002260	065725	DH104	: PC TEST NO, IPCK, RHCS3 GOOD, RHCS3 BAD	
2210	002262	066622	DT104	: \$EARPC, TSTNM, IP, \$GDDAT, \$BDDAT	
2211	002264	066754	DF104	: 0,0,0,0,0	
2212					
2213			: ITEM 105		
2214	002266	060413	EM104		
2215	002270	065775	DH105	: PC TST NO, IPCK, RHCS2 GOOD, RHCS2 BAD	
2216	002272	066636	DT105	: \$EARPC, TSTNM, IP, \$GDDAT, \$BDDAT	
2217	002274	066761	DF105	: 0,0,0,0,0	
2218					
2219			: ITEM 106		
2220	002276	060651	EM106		
2221				: RH CLEAR WAS GIVEN AN	
2222				: IPCK BIT SHOWN IN "IPCK" WAS SET	
2223				: ZEROS WERE MOVED INTO	
2224				: RHOB FROM AN ODD WORD	
2225				: RHOB WAS READ	
2226				: AN RH CLEAR WAS GIVEN	
2227				: TO CLEAR ALL ERRORS	
2228				: THE FOLLOWING REGISTER	
2229				: DID NOT CONTAIN WHAT	
2230				: IS IN GOOD	
2230	002300	066045	DH106	: PC TSTNM, IPCK, RHCS1 GOOD, RHCS1 BAD	
2231	002302	066652	DT106	: \$EARPC, TSTNM, IP, \$GDDAT, \$BDDAT	

2232	002304	066766	DF106	:0,0,0,0,0
2233				
2234			: ITEM 107	
2235	002306	060651	EM106	
2236	002310	065775	DH105	:PC, TSTNM, IPCK, RHCS2 GOOD, RHCS2 BAD
2237	002312	066636	DT105	
2238	002314	066761	DF105	
2239				
2240			: ITEM 110	
2241	002316	060651	EM106	
2242	002320	065725	DH104	
2243	002322	066622	DT104	
2244	002324	066754	DF104	
2245				
2246			: ITEM 111	
2247	002326	060651	EM106	
2248	002330	066115	DH111	:PC, TSTNM, IPCK, RHBA GOOD, RHBA BAD
2249	002332	066622	DT104	:\$ERRPC, TSTNM, IP, \$GDDAT, \$BDDAT
2250	002334	066754	DF104	:0,0,0,0,0
2251				
2252			: ITEM 112	
2253	002336	060651	EM106	
2254	002340	066163	DH112	:PC, TSTNM, IPCK, RHBAE GOOD, RHBAE BAD
2255	002342	066622	DT104	:\$ERRPC, TSTNM, IP, \$GDDAT, \$BDDAT
2256	002344	066754	DF104	:0,0,0,0,0
2257				
2258			: ITEM 113	
2259	002346	060651	EM106	
2260	002350	066233	DH113	:PC, TSTNM, IPCK, RHWC GOOD, RHWC BAD
2261	002352	066622	DT104	:\$ERRPC, TSTNM, IP, \$GDDAT, \$BDDAT
2262	002354	066754	DF104	:0,0,0,0,0
2263				
2264			: ITEM 114	
2265	002356	061142	EM114	:RH CLEAR WAS GIVEN AN
2266				:A WRITE CHECK WAS DONE
2267				:THE FOLLOWING REGISTER
2268				:DID NOT CONTAIN WHAT IS
2269				:IN "GOOD"
2270	002360	065202	DH6	
2271	002362	066574	DT6	
2272	002364	066743	DF6	
2273				
2274			: ITEM 115	
2275	002366	061142	EM114	:RH CLEAR WAS GIVEN AN
2276				:A WRITE CHECK WAS DONE
2277				:THE FOLLOWING REGISTER
2278				:DID NOT CONTAIN WHAT IS
2279				:IN "GOOD"
2280	002370	065347	DH12	
2281	002372	066574	DT6	
2282	002374	066743	DF6	
2283				
2284			: ITEM 116	
2285	002376	061311	EM116	:RH CLEAR WAS GIVEN AN
2286				:A WRITE CHECK WAS DONE
2287				:THEN ANOTHER RH CLEAR WAS

2288			
2289			
2290			
2291	002400	065305	DH11
2292	002402	066574	DT6
2293	002404	066743	DF6
2294			
2295			: ITEM 117
2296	002406	061311	EM116
2297	002410	065202	DH6
2298	002412	066574	DT6
2299	002414	066743	DF6
2300			
2301			: ITEM 120
2302	002416	061311	EM116
2303	002420	065347	DH12
2304	002422	066574	DT6
2305	002424	066743	DF6
2306			
2307			: ITEM 121
2308	002426	061311	EM116
2309	002430	065411	DH13
2310	002432	066574	DT6
2311	002434	066743	DF6
2312			
2313			: ITEM 122
2314	002436	061311	EM116
2315	002440	065451	DH14
2316	002442	066574	DT6
2317	002444	066743	DF6
2318			
2319			: ITEM 123
2320	002446	061311	EM116
2321	002450	065513	DH15
2322	002452	066574	DT6
2323	002454	066743	DF6
2324			
2325			: ITEM 124
2326	002456	061521	EM124
2327			
2328			
2329			
2330			
2331	002460	065347	DH12
2332	002462	066574	DT6
2333	002464	066743	DF6
2334			
2335			: ITEM 125
2336	002466	061521	EM124
2337	002470	065305	DH11
2338	002472	066574	DT6
2339	002474	066743	DF6
2340			
2341			: ITEM 126
2342	002476	061521	EM124
2343	002500	065202	DH6

: GIVEN TO CLEAR ALL ERRORS
: THE FOLLOWING REGISTER DID
: NOT CONTAIN WHAT IS IN "GOOD"

: ON A SILO TEST TO
: TEST DBL RHCS3-BIT #10
: THE FOLLOWING REGISTER
: DID NOT CONTAIN WHAT
: IS IN "GOOD"

2344	002502	066574	DT6	
2345	002504	066743	DF6	
2346				
2347			: ITEM 127	
2348	002506	061521	EM124	
2349	002510	065411	DH13	
2350	002512	066574	DT6	
2351	002514	066743	DF6	
2352				
2353			TEM 130	
2354	002516	061521	EM124	
2355	002520	065451	DH14	
2356	002522	066574	DT6	
2357	002524	066743	DF6	
2358				
2359			: ITEM 131	
2360	002526	061521	EM124	
2361	002530	065513	DH15	
2362	002532	066574	DT6	
2363	002534	066743	DF6	
2364			: ITEM132	
2365	002536	061664	EM132	: BEFORE DATA TRANSFER
2366				: COMMAND WAS TO BE GIVEN
2367				: THE RP DRIVE
2368				: STATUS REGISTER DID NOT
2369				: CONTAIN WHAT IS IN GOOD
2370				
2371	002540	066301	DH132	: PC
2372				: TEST NO.
2373				: RHDS1 GOOD
2374				: RHDS1 BAD
2375	002542	066574	DT6	
2376	002544	066743	DF6	
2377			: ITEM133	
2378	002546	062042	EM133	: BEFORE DATA TRANSFER
2379				: COMMAND WAS TO BE GIVEN
2380				: THE RS DRIVE
2381				: STATUS REGISTER DID NOT
2382				: CONTAIN WHAT IS IN GOOD
2383				
2384	002550	066301	DH132	: PC
2385				: TEST NO.
2386				: RHDS1 GOOD
2387				: RHDS1 BAD
2388	002552	066574	DT6	
2389	002554	066743	DF6	
2390			: ITEM 134	
2391	002556	062220	EM134	: BEFORE DATA TRANSFER COMMAND
2392				: WAS TO BE GIVEN THE
2393				: MAG TAPE DRIVE STATUS
2394				: REGISTER DID NOT CONTAIN WHAT
2395				: IS IN GOOD
2396				
2397	002560	066301	DH132	
2398	002562	066574	DT6	
2399	002564	066743	DF6	

2400					
2401	002566	062410		EM135	: WAS WAITING FOR A BIT TO SET
2402					: BIT IN QUESTION IS IN "BIT WAITED FOR"
2403					: REGISTER IN QUESTION IN "REG ADDR"
2404	002570	066343		DH135	: PC
2405					: TEST NO
2406					: PC OF WAT
2407					: BIT
2408					: REG ADDR
2409	002572	066666		DT135	: \$ERRPC TSTNM, WATIPC, WAITBT, WAITRE
2410	002574	066773		DF135	: 0,0,0,0,0
2411					: ITEM136
2412	002576	062610		EM136	: WAS WAITING FOR A BIT TO RESET
2413					: BIT IN QUESTION IS IN "BIT WAITED FOR"
2414					: REGISTER IN QUESTION IN "REG ADDR"
2415	002600	066343		DH135	: PC
2416					: TEST NO
2417					: PC OF WAT
2418					: BIT
2419					: REG ADDR
2420	002602	066666		DT135	: \$ERRPC TSTNM, WATIPC, WAITBT, WAITRE
2421	002604	066773		DF135	: 0,0,0,0,0
2422					: ITEM 137
2423	002606	063012		EM137	: ON WRITING AND READING
2424					: THE REGISTER # IN "REG. ADDR"
2425					: IT DID NOT CONTAIN
2426					: EXPECTED VALUE.
2427	002610	066423		DH137	: PC
2428					: TEST NO
2429					: REG ADDR
2430					: GOOD DATA
2431					: BAD DATA
2432	002612	066702		DT137	: \$ERRPC TSTNM, \$BDDADR, \$GDDAT, \$BDDAT
2433	002614	067000		DF137	: 0,0,0,0,0
2434					
2435					
2436					: ITEM 140
2437	002616	063140		EM140	
2438	002620	066045		DH106	
2439	002622	066652		DT106	
2440	002624	066766		DF106	
2441					: ITEM 141
2442	002626	063372		EM141	: AFTER CLEARING THE RH AND
2443					: WRITING TWO WORD INTO
2444					: RHOB AND READING IT
2445					: BACK TWICE, CAUSED RHCS1 TO
2446					: HAVE WRONG DATA
2447					: GIVEN IN BAD RHCS1
2448	002630	065305		DH11	: PC
2449					: TEST NUMBER
2450					: RHCS1 GOOD
2451					: RHCS1 BAD
2452	002632	066574		DT6	: \$ERRPC TSTNM, \$GDDAT, \$BDDAT
2453	002634	066743		DF6	: 0,0,0,0
2454					
2455					

2456	002636	063573	: ITEM 142 EM142	: AFTER CLEARING THE RH AND : WRITING TWO WORD INTO : RHDB AND READING IT : BACK TWICE, CAUSED RHCS3 TO : HAVE WRONG DATA, : GIVEN IN BAD RHCS3 : PC : TEST NUMBER : RHCS3 GOOD : RHCS3 BAD : \$ERRPC, TSTNM, \$GDDAT, \$BDDAT : 0,0,0,0
2457				
2458				
2459				
2460				
2461				
2462	002640	065347	DH12	
2463				
2464				
2465				
2466				
2467	002642	066574	DT6	
2468	002644	066743	DF6	
2469				
2470				
2471				
2472				
2473				
2474				
2475				
2476				
2477				
2478	002650	065411	DH13	: AFTER CLEARING THE RH AND : WRITING TWO WORD INTO : RHDB AND READING IT : BACK TWICE, CAUSED RHBA TO : HAVE WRONG DATA, : GIVEN IN BAD RHBA : PC : TEST NUMBER : RHBA GOOD : RHBA BAD : \$ERRPC, TSTNM, \$GDDAT, \$BDDAT : 0,0,0,0
2479				
2480				
2481				
2482	002652	066574	DT6	
2483	002654	066743	DF6	
2484				
2485				
2486				
2487				
2488	002656	064174	: ITEM 144 EM144	: AFTER CLEARING THE RH AND : WRITING TWO WORD INTO : RHDB AND READING IT : BACK TWICE, CAUSED RHBAE TO : HAVE WRONG DATA, : GIVEN IN BAD RHBAE : PC : TEST NUMBER : RHBAE GOOD : RHBAE BAD : \$ERRPC, TSTNM, \$GDDAT, \$BDDAT : 0,0,0,0
2489				
2490				
2491				
2492				
2493	002660	065451	DH14	
2494				
2495				
2496				
2497	002662	066574	DT6	
2498	002664	066743	DF6	
2499				
2500				
2501				
2502				
2503	002666	064375	: ITEM 145 EM145	: AFTER CLEARING THE RH AND : WRITING TWO WORD INTO : RHDB AND READING IT : BACK TWICE, CAUSED RHWC TO : HAVE WRONG DATA, : GIVEN IN BAD RHWC : PC : TEST NUMBER : RHWC GOOD : RHWC BAD
2504				
2505				
2506				
2507				
2508	002670	065513	DH15	
2509				
2510				
2511				

2512	002672	066574	DT6	;\$ERRPC, TSTNM, \$GDDAT, \$BDDAT
2513	002674	066743	DF6	;\$0,0,0,0
2514			: ITEM146	
2515	002676	064575	EM146	;\$A DEVICE BASE ADDRESS DID NOT
2516				;\$TIME OUT BUT CORRESPONDING
2517				;\$VECTOR ADDRESS DID TIME OUT
2518	002700	066475	DH146	;\$PC
2519				;\$BASE
2520				;\$VECTOR
2521	002702	066716	DT146	;\$ERRPC, TESTDV, TESTVC, 0
2522	002704	067005	DF146	;\$0,0,0
2523				
2524			: ITEM147	
2525	002706	064767	EM147	;\$A DEVICE ADDRESS DID NOT
2526				;\$TIME OUT BUT NO UNITS HAD
2527				;\$APPROPRIATE DRIVE TYPE
2528	002710	066524	DH147	;\$PC
2529				;\$BASE ADDRESS
2530	002712	066726	DT147	;\$ERRPC, TESTDV
2531	002714	067010	DF147	
2532				

2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
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
2588

002716 000254

:RH70 REGISTERS

RPVEC: 254 ; RP VECTOR ADDRESS
:*****
:WORD COUNT REGISTER (RHWC)
:EACH BIT IS CALLED BY BIT NUMBER

:BUS ADDRESS REGISTER (RHBA)
:EACH BIT IS CALLED BY BIT NUMBER

:CONTROL AND STATUS REGISTER 2 (RHCS2)

000001	US1=	1	:UNIT SELECT (BIT #0)
000002	US2=	2	:UNIT SELECT (BIT #1)
000004	JS4=	4	:UNIT SELECT (BIT #2)
000010	BAI=	10	:BUS ADDRESS INCREMENT INHIBIT (BIT #3)
000020	PAT=	20	:INVERT PARITY
000040	CLR=	40	:CLEAR (BIT #5)
000100	IR=	100	:INPUT READY (BIT #6)
000200	OR=	200	:OUTPUT READY (BIT #7)
000400	MPE=	400	:MASS BUS PARITY ERROR (BIT #8)
000400	MDPE=	400	:MASS BUS PARITY ERROR (BIT #8)
001000	MXF=	1000	:MISSED TRANSFER ERROR (BIT #9)
002000	PGE=	2000	:PROGRAM ERROR (BIT #10)
004000	NEM=	4000	:NON EXISTANT MEMORY (BIT #11)
010000	NED=	10000	:NON EXISTANT DRIVE (BIT #12)
020000	PE=	20000	:UNIBUS PARITY ERROR (BIT #13)
040000	WCE=	40000	:WRITE CHECK ERROR (BIT #14)
100000	DLT=	100000	:DATA LATE (BIT #15)

:CONTROL AND STATUS REGISTER 3 (RHCS3)

000001	IPCK0=	1	:INVERT PARITY CHECK BIT 0
000002	IPCK1=	2	:INVERT PARITY CHECK BIT 1
000004	IPCK2=	4	:INVERT PARITY CHECK BIT 2
000010	IPCK3=	10	:INVERT PARITY CHECK BIT 3
000100	IE=	100	:INTERRUPT ENABLE (BIT #6)
002000	DBL=	2000	:DOUBLE WORD BOUNDARY (BIT #10)
004000	WCEEW=	4000	:WRITE CHECK EVEN WORD (BIT #11)
010000	WCEOW=	10000	:WRITE CHECK ODD WORD (BIT #12)
020000	DPEEW=	20000	:DATA PARITY ERROR EVEN WORD (BIT #13)
040000	DPEOW=	40000	:DATA PARITY ERROR ODD WORD (BIT #14)
100000	APE=	100000	:ADDRESS PARITY ERROR (BIT #15)

:DATA BUFFER REGISTER (RHDB)

:EACH BIT IS CALLED BY BIT NUMBER

:RPO4 REGISTERS

:CONTROL AND STATUS 1 REGISTER. (#00)

2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644

000001
000100
000200
000400
001000
002000
004000
020000
040000
100000

000001
000002
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
010000
020000
040000
100000

000001
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
010000

GO= 1
IE= 100
RDY= 200
A16= 400
A17= 1000
PSEL= 2000
DVA= 4000
MCPE= 20000
TRE= 40000
SC= 100000

DFF5= 1
BOT= 2
DFF20= 2
DIG8= 4
GRV= 10
DL64= 20
DE1= 40
VV= 100
DRY= 200
DPR= 400
PROG= 1000
LBT= 2000
WRL= 4000
MOL= 10000
PIP= 20000
ERR= 40000
ATA= 100000

:ERROR REGISTER #01 (RHEP: #02)
ILF= 1
ILR= 2
RMR= 4
PAR= 10
FER= 20
WCF= 40
ECH= 100
HCE= 200
HCRC= 400
AOE= 1000
IAE= 2000
WLE= 4000
DTE= 10000

:GO (BIT #0)
:INTERRUPT ENABLE (BIT #6)
:READY (BIT #7)
:HIGH ORDER UNIBUS BITS (BIT #8)
:HIGH ORDER UNIBUS BITS (BIT #9)
:PORT SELECT (BIT #10)
:DEVICE AVAILABLE (BIT #11)
:MASSBUSS PARITY ERROR (BIT #13)
:TRANSFER ERROR (BIT #14)
:SPECIAL CONDITION (BIT #15)

:DRIVE FORWARD 5"/SEC. (BIT #0)
:BEGINING OF TAPE (BIT #1)
:DRIVE FORWARD 20"/SEC. (BIT #1)
:DRIVE TO INNER GAVRD BAND (BIT #2)
:GO REVERSE (BIT #3)
:DIFFERENCE LESS THAN 64 (BIT #4)
:DIFFERENCE EQUALS 1 (BIT #5)
:VOLUME VALID (BIT #6)
:DRIVE READY (BIT #7)
:DRIVE PRESENT (BIT #8)
:PROGRAMABLE (BIT #9)
:LAST SECTOR TRANSFERRED (BIT #10)
:WRITE LOCK (BIT #11)
:MEDIUM ON-LINE (BIT #12)
:POSITIONING OPERATION IN PROGRESS BIT #13
:COMPOSIT ERROR. (BIT #14)
:ATTENTION ACTIVE (BIT #15)

:ILLEGAL FUNCTION (BIT #0)
:ILLEGAL REGISTER (BIT #1)
:REGISTER MODIFICATION REFUSED (BIT #2)
:PARITY ERROR (BIT #3)
:FORMAT ERROR (BIT #4)
:WRITE CLOCK FAIL (BIT #5)
:ECC HARD ERROR (BIT #6)
:HEADER COMPARE ERROR (BIT #7)
:HEADER CRC ERROR (BIT #8)
:ADDRESS OVERFLOW ERROR (BIT #9)
:INVALID ADDRESS ERROR (BIT #10)
:WRITE LOCK ERROR (BIT #11)
:DRIVE TIMING ERROR (BIT #12)

2645	020000	OPI= 20000	: OPERATION INCOMPLETE (BIT #13)
2646	040000	UNS= 40000	: DRIVE UNSAFE (BIT #14)
2647	100000	DCK= 100000	: DATA CHECK ERROR (BIT #15)
2648			
2649			
2650			
2651	000001	DMD= 1	: DIAGNOSTIC MODE (BIT #0)
2652	000002	MCLK= 2	: MAINTAINABILITY CLOCK (BIT #1)
2653	000004	MINX= 4	: MAINTAINABILITY INDEX (BIT #2)
2654	000010	MSTCK= 10	: MAINTAINABILITY SECTOR CLOCK (BIT #3)
2655	000020	MRD= 20	: MAINTAINABILITY READ (BIT #4)
2656	000040	MWR= 40	: MAINTAINABILITY WRITE (BIT #5)
2657	001000	DTSY= 1000	: MAINTAINABILITY SYNC DETECTED (BIT #9)
2658			
2659			
2660			
2661	000001	AT0= 1	: DEVICE 0 (BIT #0)
2662	000002	AT1= 2	: DEVICE 1 (BIT #1)
2663	000004	AT2= 4	: DEVICE 2 (BIT #2)
2664	000010	AT3= 10	: DEVICE 3 (BIT #3)
2665	000020	AT4= 20	: DEVICE 4 (BIT #4)
2666	000040	AT5= 40	: DEVICE 5 (BIT #5)
2667	000100	AT6= 100	: DEVICE 6 (BIT #6)
2668	000200	AT7= 200	: DEVICE 7 (BIT #7)
2669			
2670			
2671			
2672			
2673			
2674			
2675			
2676	000001	EXT1= 1	: EXTENSION 1 (BIT #0)
2677	000002	EXT2= 2	: EXTENSION 2 (BIT #1)
2678	000004	EXT4= 4	: EXTENSION 3 (BIT #2)
2679	000010	EXT10= 10	: EXTENSION 4 (BIT #3)
2680	000020	EXT20= 20	: EXTENSION 5 (BIT #4)
2681	000040	EXT40= 40	: EXTENSION 6 (BIT #5)
2682	000100	SC1= 100	: SECTOR COUNT FIELD 0 (BIT #6)
2683	000200	SC2= 200	: SECTOR COUNT FIELD 1 (BIT #7)
2684	000400	SC4= 400	: SECTOR COUNT FIELD 2 (BIT #8)
2685	001000	SC10= 1000	: SECTOR COUNT FIELD 3 (BIT #9)
2686	002000	SC20= 2000	: SECTOR COUNT FIELD 4 (BIT #10)
2687	004000	TRK1= 4000	: TRACK FIELD 1 (BIT #11)
2688	010000	TRK2= 10000	: TRACK FIELD 2 (BIT #12)
2689	020000	TRK4= 20000	: TRACK FIELD 3 (BIT #13)
2690	040000	TRK10= 40000	: TRACK FIELD 4 (BIT #14)
2691	100000	TRK20= 100000	: TRACK FIELD 5 (BIT #15)
2692			
2693			
2694			
2695	000001	WCU= 1	: WRITE CURRENT UNSAFE (BIT #0)
2696	000002	CSF= 2	: CURRENT SINK FAILURE (BIT #1)
2697	000004	WSU= 4	: WRITE SELECT UNSAFE (BIT #2)
2698	000010	CSU= 10	: CURRENT SWITCH UNSAFE (BIT #3)
2699	000020	MSE= 20	: MOTOR SEQUENCE ERROR (BIT #4)
2700	000040	TDF= 40	: TRANSITIONS DETECTOR FAILURE (BIT #F)

2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
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

000100
000200
000400
001000
002000
004000
010000
020000
100000

000001
000002
000004
000010
000020
000040

000200
002000
004000
010000

000001
000002
000004
000010
000020
000040
000100
000200
000400
001000
002000
004000
000300

002000
010000

TUF= 100
FEN= 200
WRU= 400
MHS= 1000
NHS= 2000
IXE= 4000
VU30= 10000
PLU= 20000
ACU= 100000

: OFFSET REGISTER (RHCF) (#11)

OF25= 1
OF50= 2
OF100= 4
OF200= 10
OF400= 20
OF800= 40

OFREV= 200
HCI= 2000
ECI= 4000
FMT22= 10000

: TAPE CONTROL REGISTER (RHTC)

S1= 1
S2= 2
S4= 4
EPAR= 10
FMT1= 20
FMT2= 40
FMT4= 100
FMT8= 200
DEN1= 400
DEN2= 1000
DEN4= 2000
SPI8= 1400
NML= 300

: DRIVE TYPE REGISTER

SLVPR= 2000
CH7= 10000

: TRANSITIONS UNSAFE (BIT #6)
: FAILSAFE ENABLED (BIT #7)
: WRITE READY UNSAFE (BIT #8)
: MULTIPLE HEAD SELECT (BIT #9)
: NO HEAD SELECTION (BIT #10)
: INDEX ERROR (BIT #11)
: 30VOLT UNSAFE (BIT #12)
: PLO UNSAFE (BIT #13)
: ACUNSAFE (BIT #15)

: OFFSET 25 MICRO INCHES (BIT #0)
: OFFSET 50 MICRO INCHES (BIT #1)
: OFFSET 100 MICRO INCHES (BIT #2)
: OFFSET 200 MICRO INCHES (BIT #3)
: OFFSET 400 MICRO INCHES (BIT #4)
: OFFSET 800 MICRO INCHES (BIT #5)

: OFFSET NEGATIVE (REVERSE) (BIT #5)
: HEADER COMPARE INHIBIT (BIT #10)
: ERROR CORRECTION CODE INHIBIT (BIT #11)
: FORMAT BIT (BIT #12)

: SLAVE NUMBER
: SLAVE NUMBER
: SLAVE NUMBER
: EVEN PARITY
: FORMAT
: FORMAT
: FORMAT
: FORMAT
: DENSITY
: DENSITY
: DENSITY
: 800 BPI
: NORMAL - 2 FRAMES PER WORD

: DESIRED CYLINDER ADDRESS (RHCA) (#12)
: EACH BIT IS CALLED BY BIT NUMBER.

: CURRENT CYLINDER ADDRESS (RHCC) (#13)

:EACH BIT IS CALLED BY BIT NUMBER

:SERIAL NUMBER REGISTER (RHSN) (#14)
:EACH IS CALLED BY BIT NUMBER

:ERROR REGISTER #03 (RHER3) (#15)

000001
000002
000010
000020
000040
000100
040000
100000

PSU= 1 :PACK SPEED UNSAFE (BIT #0)
VUF= 2 :VELOCITY UNSAFE (BIT #1)
UWR= 10 :ANY UNSAFE EXCEPT READ/LRITE (BIT #3)
PRE= 20 :DISK PACK ROTATION ERROR (BIT #4)
ACL= 40 :AC LOW (BIT #5)
DCL= 100 :DC LOW (BIT #6)
SKI= 40000 :SEEK INCOMPLETE (BIT #14)
OCYL= 100000 :OFF CYLINDER (BIT #15)

:ECC POSITION REGISTER (RHEC1) (#16)
:EACH BIT IS CALLED BY BIT NUMBER

:ECC PATTERN REGISTER (RHEC2) (#17)
:EACH BIT IS CALLED BY BIT NUMBER

:*****

000001
000002
000010
000020
000040
000100
040000
100000
000001
000002
000010
000020
000040
000100
040000
100000
000001
000002
000010
000020
000040
000100
040000
100000
000001
000002
000010
000020
000040
000100
040000
100000

28000
28001
28002
28003
28004
28005
28006
28007
28008
28009
28010
28011
28012
28013
28014
28015
28016
28017
28018
28019
28020
28021
28022
28023
28024
28025
28026
28027
28028
28029
28030
28031
28032
28033
28034
28035
28036
28037
28038

000000
000002
000004
000006
000006
000010
000012
000014
000016
000020
000022
000024
000026
000030
000032
000032
000034
000036
000040
000042
000044
000046

003000
003000
003000
003002
003002

004046 172040
004050 176700
004052 172440
004054 176300
004056 000000

CS1 = 0
WC = 4
BA = 4
FC = 6
DA = 6
CS2 = 10
DS = 12
ER1 = 14
AS = 16
LA = 20
DB = 22
MR = 24
DT = 26
SN = 30
OF = 32
TC = 32
DC = 34
CC = 36
ER2 = 40
ER3 = 42
EC1 = 44
EC2 = 46

; BUFFERS
= 3000
BUFEW:
BFEVEN: 0
BUFOW:
BFOOD: .BLKW 274.
; STARTING ADDRESS OF REGISTERS

RSCS1: 172040
RPCS1: 176700
TMCS1: 172440
MIXCS1: 176300
PRESENT: 0

: CONTROL STATUS 1
: WORD COUNT
: BUS ADDRESS
: FRAME COUNT
: DESIRED SECTOR/TRACK
: CONTROL STATUS 2
: DRIVE STATUS
: ERROR 1
: ATTENTION SUMMERY
: LOOK AHEAD
: DATA BUFFER
: MAINTENANCE REGISTER
: DRIVE TYPE
: SERIAL NUMBER
: OFFSET
: TAPE CONTROL
: DESIRED CYLINDER
: CURRENT CYLINDER
: ERROR 2
: ERROR 3
: ECC 1
: ECC 2

: BUFFER

: RS ALONG
: RP ALONE
: TM ALONE
: MIXED SYSTEM

.SBTTL REGISTER ADDRESSES

H05

```

2853          : THE FOLLOWING ARE THE I/O REGISTERS FOR THE DEVICE UNDER TEST
2854          : THEY WILL BE FILLED WITH ADDRESSES DEPENDING ON WHAT DEVICE IS ON
2855          : THE SYSTEM.
2856          : IN THE CASE OF A MIXED SYSTEM THAT IS WITH MORE THAN ONE RH DEVICE
2857          : THE FIRST ONE THAT EXISTS IN THE FOLLOWING LIST WILL BE PICKED
2858          : AND THE OTHERS WILL NOT BE USED.
2859          : LIST: -
2860          : 1)   RS03
2861          : 2)   RS04
2862          : 3)   RPO4,5,6
2863          : 4)   TMO2
2864          : THE CONTENTS OF RHCS1 WHICH IS THE BASE ADDRESS IS
2865          : 172040 FOR RS03/04
2866          : 176700 FOR RPO4,5,6
2867          : 172440 FOR TMO2
2868          : THE REST OF THE LOCATIONS WILL BE FILLED BY THE PROGRAM
2869          : RHCS1: 0           ; CONTROL AND STATUS 1
2870          : RHWC: 0           ; WORD COUNT
2871          : RHBA: 0           ; BUS ADDRESS
2872          : RHFC:           ; FRAME COUNT
2873          : RHDA:           ; DISK ADDRESS
2874          : RHDST: 0         ; DESIRED SECTOR/TRACK ADDRESS
2875          : RHCS2: 0         ; CONTROL AND STATUS 2
2876          : RHDS1: 0         ; DRIVE STATUS
2877          : RHER1: 0         ; ERROR #1
2878          : RHAS: 0         ; ATTENTION SUMMARY
2879          : RHCC:           ; CHECK CHARACTER
2880          : RHLA: 0         ; LOOK-AHEAD
2881          : RHOB: 0         ; DATA BUFFER
2882          : RHMR: 0         ; MAINTAINABILITY
2883          : RHDT: 0         ; DRIVE TYPE
2884          : RHSN: 0         ; SERIAL NUMBER
2885          : RHTC:           ; TAPE CONTROL
2886          : RHOF: 0         ; OFFSET
2887          : RHCA: 0         ; DESIRED CYLINDER ADDRESS
2888          : RHCCA: 0        ; CURRENT CYLINDER ADDRESS
2889          : RHER2: 0        ; ERROR #2
2890          : RHER3: 0        ; ERROR #3
2891          : RHEC1: 0        ; ECC POSITION
2892          : RHEC2: 0        ; ECC PATTERN
2893          : RHBAE: 0        ; BUS ADDRESS EXTENSION
2894          : RHCS3: 0        ; CONTROL AND STATUS 3
2895
2896          : FUNCTION EQUATES
2897
2898          : TABLE OF FUNCTIONS FOR RHCS1 THEN "GO" BIT HAS TO BE SET
2899          : FUTABL:
2900          : NOPEA: 0           ; NO OPERATION
  
```

2891	004136	000002	UNLOAD:	2	:UNLOAD (STAND BY)
2892	004140	000006	RECALI:	6	:RECALIBRATE
2893	004142	000010	DCLEAR:	10	:DRIVE CLEAR
2894	004144	000012	RELEAS:	12	:RELEASE (DUAL-PORT OPERATION)
2895	004146	000030	SEARCH:	30	:SEARCH COMMAND
2896	004150	000050	WRCHK:	50	:WRITE CHECK DATA
2897	004152	000052	WRCHDT:	52	:WRITE CHECK HEADER AND DATA
2898	004154	000060	WRIDAT:	60	:WRITE DATA
2899	004156	000062	WRIFOR:	62	:WRITE HEADER AND DATA (FORMAT)
2900	004160	000070	READAT:	70	:READ DATA
2901	004162	000072	REFOR:	72	:READ HEADER AND DATA
2902	004164	000004	SEECOM:	4	:SEEK COMMAND
2903	004166	000014	OFSETC:	14	:OFFSET COMMAND
2904	004170	000016	RETCL:	16	:RETURN TO CENTERLINE
2905	004172	000022	PKACK:	22	:PACK ACKNOWLEDGE
2906	004174	000020	READIN:	20	:READ IN
2907	004176	000006	REWIND:	6	:REWIND
2908	004200	000076	REVRED:	76	:REVERSE READ
2909	004202	000030	SPACFD:	30	:SPACE FORWARD
2910	004204	000066	REVWRT:	66	:REVERSE WRITE
2911	004206	000000	ILLEGL:	.WORD 0	:COMPUTED ILLEGAL FUNCTION
2912					
2913					
2914	004210	000106	WRFROM:	:DATA BUFFER FOR READ WRITE	:WRITE FROM THIS BUFFER
2915	004424	000106	REINTO:	.BLKW 70.	:READ INTO THIS BUFFER
2916	004640	000000	COMND:	0	:STORE COMMAND FOR COMMAND ROUTINE
2917	004642	000000	COMAND:	0	:STORE COMMAND
2918	004644	000000	NOGO:	0	:IF ZERO GO IS TO BE GIVEN IN COMMAND ROUTINE
2919					:IF ONES GO IS NOT TO BE GIVEN IN COMMAND ROUTINE
2920					:USED IN PGE TEST
2921	004646	000000	WATC:	0	:IF ZERO WAIT FOR BIT TO SET
2922					:IF ONES WAIT FOR BIT TO RESET
2923					:USED IN WAIT TRAP
2924	004650	000000	WRTBIT:	0	:BITS NOT WRITTEN INTO
2925	004652	000000	SETBIT:	0	:BITS ALWAYS SET
2926	004654	000000	CLRBIT:	0	:BITS ALWAYS CLEARED
2927					
2928					
2929					
2930					
2931	004656	000000	†STNM:	0	:TEST NUMBER
2932	004660	000000	SLAVE:	0	:KEEP SLAVE NO.
2933	004662	000000	IP:	0	:IPCK NUMBER FOR ERROR PRINTOUT
2934	004664	000000	SILONM:	0	:SILO WORD NUMBER FOR ERROR PRINT OUT
2935	004666	000000	WAITPC:	0	:WAIT TRAP PC
2936	004670	000000	WAITRE:	0	:WAITING FOR REGISTER ADDRESS IN WAIT TRAP
2937	004672	000000	WAITBT:	0	:WAITING FOR BIT IN WAIT TRAP
2938					
2939					
2940					
2941	004674	001			
2942	004677	010			
2943	004702	100			
2944					
2945					
2946					

: RESERVED LOCATIONS

: ATTENTION TABLE

ATABLE: .BYTE 1,2,4,10,20,40,100,200

:RESERVED LOCATIONS FOR UNIT SELECT
BASEAD: 172040 ;RS BASE ADDRESS
BASPAD: 176700 ;RP BASE ADDRESS
BASTAD: 172440 ;TU BASE ADDRESS
BASEMAD: 176300 ;MIXED SYSTEM BASE ADDRESS

BASEVC: 0 ;RS VECTOR
BASP: 254 ;RP VECTOR
BAST: 1 ;TU VECTOR
BASM: 2 ;MIXED VECTOR

:TABLE FOR GIVEN BASE ADDRESSES
BSGIVA: .BLKW 8.

:TABLE FOR GIVEN VECTOR
BSGIVV: .BLKW 8.

NMRHS: 4 ;NUMBER OF RH
BASINX: 0 ;INDEX FOR BASE
WORKBS: 0 ;WORKING BASE
WORKNM: 0 ;WORKING NUMBER FOR RH
WORKVC: 0 ;WORKING VECTOR FOR RH
FORBAE: 0 ;TOTAL NO. OF REG. FOR BAE CALCULATION
LOPCT: 0 ;LOOP COUNT FOR 200 START
LOPCT1: 0 ;LOOP COUNT FOR 210 START
USEVEC: 0 ;USE VECTOR

GIVE: 0 ;IF ONES OPERATOR WILL GIVE ADDRESSES
UNIT: 0 ;UNIT UNDER TEST
ST200: 0 ;ALL ONES INDICATE STARTING FROM 200
:TESTING TABLE
TSRANO: .BLKW 8. ;RH NO TO BE TESTED IN ORDER OF TEST
TSDEVICE: .BLKW 8. ;DEVICE TO BE TESTED IN ORDER OF TEST
:1=RS03,3/L,3/LA 2=RS04,4/L 4=RPO4,5,6 SINGLE PORT
:10=RPO4,5,6 DUAL PORT 20=TMO2

TSUNIT: .BLKW 8. ;UNIT NO. TO BE TESTED IN ORDER OF TEST

TSSLAV: .BLKW 8. ;SLAVE NO TO BE TESTED IN ORDER OF TEST

TSTOTL: 0 ;TOTAL NO. OF UNITS TO BE TESTED
TSINDX: 0 ;INDEX TO ONE UNDER TEST

TSVEC: .BLKW 8 ;VECTOR ADDRESS IN ORDER OF TEST

2947
2948 004704 172040
2949 004706 176700
2950 004710 172440
2951 004712 176300
2952
2953
2954 004714 000000
2955 004716 000254
2956 004720 000001
2957 004722 000002
2958
2959
2960 004724 000010
2961
2962
2963
2964 004744 000010
2965
2966
2967 004764 000004
2968 004766 000000
2969 004770 000000
2970 004772 000000
2971 004774 000000
2972 004776 000000
2973 005000 000000
2974 005002 000000
2975 005004 000000
2976
2977
2978
2979 005006 000000
2980 005010 000000
2981 005012 000000
2982
2983 005014 000010
2984
2985 005034 000010
2986
2987
2988
2989
2990 005054 000010
2991
2992
2993 005074 000010
2994
2995
2996 005114 000000
2997 005116 000000
2998
2999
3000 005120 000010
3001
3002

```

3003 005140 000010      TSBAE: .BLKW  8.      ;RHBAE ADDRESS IN ORDER OF TEST
3004
3005
3006 005160 000010      TSCS3: .BLKW  8.      ;RHCS3 ADDRESS IN ORDER OF TEST
3007
3008
3009 005200 000010      TSCOMD: .BLKW  8.     ;COMMAND ADDRESS IN ORDER OF TEST
3010
3011
3012 005220 000010      TSBASE: .BLKW  8.     ;BASE ADDRESS IN ORDER OF TEST
3013
3014
3015
3016
3017
3018
3019 005240 000000      ERFLGS: 0           ;ERROR FLAG
3020 005242 000000      FIRST:  0          ;IF ZERO WILL TYPE HEADER
3021                                     ;IF ONES WILL NOT TYPE HEADER
3022
3023
3024
3025 005244 000000      ATTENT: 0           ;ATTENTION BIT FOR PRESENT UNIT
3026 005246 000000      TOTALAT: 0         ;TATAL ATTENTION BITS
3027
3028 005250 000000      TMP0:  .WORD  0     ;TEMP STORAGE
3029 005252 000000      TMP1:  .WORD  0     ;TEMP STORAGE
3030 005254 000000      TMP4:  .WORD  0     ;TEMP STORAGE
3031                                     ;PERMANENT TABLE
3032 005256 172040      PERRS: 172040       ;RS BASE ADDRESS
3033 005260 176700      PERRP: 176700       ;RP BASE ADDRESS
3034 005262 172440      PERTU: 172440       ;TU BASE ADDRESS
3035 005264 176300      PERMX: 176300       ;MIXED BASE ADDRESS
3036
3037 005266 000204      PERRSV: 204         ;RS VECTOR ADDRESS
3038 005270 000254      PERRPV: 254         ;RP VECTOR ADDRESS
3039 005272 000224      PERTUV: 224         ;TU VECTOR ADDRESS
3040 005274 000000      PERMXV: 0           ;MIXED VECTOR ADDRESS
3041
3042 005276 000004      PERNUM: 4           ;NUMBER OF RH
3043
3044                                     ;WORKING TABLE
3045
3046 005300 000004      DEVPNT: .BLKW  4     ;BASE ADDRESS TO BE TESTED
3047 005310 000004      VECPNT: .BLKW  4     ;VECTOR ADDRESS TO BE TESTED
3048 005320 000000      LTRY:  0            ;NO. OF UNITS TO BE TESTED
3049 005322 000000      TFOUND: 0           ;TWICE NUMBER OF RH FOUND
3050 005324 000000      TESTDV: 0           ;STORES BASE ADDRESS OF TEST DEVICE
3051 005326 000000      TESTVC: 0           ;STORES VECTOR OF TEST DEVICE
3052                                     ;THE ABOVE TWO IS BEFORE ANY
3053                                     ;DEVICE IS FOUND.
3054                                     ;TABLE FOR DRIVE TYPES
3055                                     ;0=RS03, 1=RS03/L, 2=RS04, 3=RS04/L, 4=RS03/LA
3056                                     ;20020=SINGLE PORT RP04, 24020=DUAL PORT RP04
3057                                     ;20021=SINGLE PORT RP05, 24021=DUAL PORT RP05
3058                                     ;20022=SINGLE PORT RP06, 24022=DUAL PORT RP06

```



```

3059                                     ;142010=TU16
3060
3061 005330 000000 000001 000002 TYPNT: .WORD 0,1,2,3,4,20020,24020,20021,24021,20022,24022,142010
3062 005336 000003 000004 020020
3063 005344 024020 020021 024021
3064 005352 020022 024022 142010
3065 005360 000000 POINTT: 0 ;POITER TO ABOVE TABLE
3066 005362 000014 NTYPNT: 14 ;NUMBER OF ABOVE TYPES
3067
3068 005364 000000 TMSPLV: 0 ;TEMPORARY SLAVE COUNTER
3069
3070                                     ;TABLE OF DATA FOR DEVICES FOUND
3071 005366 000004 FDEVIC: .BLKW 4 ;DEVICE FOUND, 1=TU, 2=RP DUAL
3072                                     ;3=RP SINGLE, 4=RS04, 5=RS03
3073 005376 000004 FUNITN: .BLKW 4 ;UNIT NUMBER FOUND
3074 005406 000004 FTYPE: .BLKW 4 ;DRIVE TYPE FOUND
3075 005416 000004 FVECTR: .BLKW 4 ;VECTOR FOUND
3076 005426 000004 FBASEA: .BLKW 4 ;BASE ADDRESS FOUND
3077 005436 000004 FRHNM: .BLKW 4 ;RH NUMBER FOUND
3078 005446 000004 FSLAVE: .BLKW 4 ;TU16 SLAVE NUMBER FOUND
3079 005456 000004 FDRIVR: .BLKW 4 ;DRIVER ADDRESS FOUND
3080 005466 000004 FBAE: .BLKW 4 ;BAE ADDRESS FOUND
3081 005476 000004 FCS3: .BLKW 4 ;CS3 ADDRESS FOUND
3082
3083 005506 000000 TSTUNT: 0 ;TOTAL NUMBER OF RH FOUND HT
3084                                     ;END OF SIZE
3085 005510 000000 WORUNT: 0 ;SAME AS ABOVE BUT TO BE
3086                                     ;DECREASED AT END PROGRAM
3087
3088 005512 000000 VECTOR: 0 ;VECTOR UNDER TEST
3089 005514 177740 RADD: 177740 ;LOW ERROR ADDRESS REG.
3090 005516 177742 HERADD: 177742 ;HIGH ERROR ADDRESS REG.
3091 005520 177744 MEMERR: 177744 ;MEMORY SYSTEM ADDRESS REG.
3092 000114 PARE70=114 ;PARITY ERROR VECTOR FOR 70
    
```

M05

CERHADO MACY11 301046, 21-DEC-77 13:06 PAGE 65
 CERHAD.P11 21-DEC-77 12:48 REGISTER TEST

SEQ 0064

```

3093          005522 012737 177777 005012 BEGIN: SBTTL REGISTER TEST
3094          005530 000402          BR          #-1,ST200
3095          005532 005037 005012 BEGIN2: CLR          ST200
3096          005536          START:
3097          SBTTL INITIALIZE THE COMMON TAGS
3098          SBTTL INITIALIZE THE COMMON TAGS ($CMTAG) AREA
3099          SBTTL INITIALIZE THE COMMON TAGS ($CMTAG) AREA
3100          SBTTL INITIALIZE THE COMMON TAGS ($CMTAG) AREA
3101          005536 012706 001100          MOV          $CMTAG,R6          ;; FIRST LOCATION TO BE CLEARED
3102          005542 005026          CLR          (R6)+          ;; CLEAR MEMORY LOCATION
3103          005544 022706 001140          CMP          $SWR,R6          ;; DONE?
3104          005550 001374          BNE          #-6          ;; LOOP BACK IF NO
3105          005552 012706 001000          MOV          $STACK,SP          ;; SETUP THE STACK POINTER
3106          SBTTL INITIALIZE A FEW VECTORS
3107          005556 012737 043104 000020          MOV          $SCOPE,$IOTVEC          ;; IOT VECTOR FOR SCOPE ROUTINE
3108          005564 012737 000340 000022          MOV          $340,$IOTVEC+2          ;; LEVEL 7
3109          005572 012737 044730 000030          MOV          $ERROR,$EMTVEC          ;; EMT VECTOR FOR ERROR ROUTINE
3110          005600 012737 000340 000032          MOV          $340,$EMTVEC+2          ;; LEVEL 7
3111          005606 012737 046434 000034          MOV          $TRAP,$TRAPVEC          ;; TRAP VECTOR FOR TRAP CALLS
3112          005614 012737 000340 000036          MOV          $340,$TRAPVEC+2          ;; LEVEL 7
3113          005622 012737 046514 000024          MOV          $SPWRDN,$PWRVEC          ;; POWER FAILURE VECTOR
3114          005630 012737 000340 000026          MOV          $340,$PWRVEC+2          ;; LEVEL 7
3115          005636 005037 001212          CLR          $TIMES          ;; INITIALIZE NUMBER OF ITERATIONS
3116          005642 005037 001214          CLR          $ESCAPE          ;; CLEAR THE ESCAPE ON ERROR ADDRESS
3117          005646 112737 000001 001115          MOV          $1,$SERMAX          ;; ALLOW ONE ERROR PER TEST
3118          005654 012737 005654 001106          MOV          $,$SLPADR          ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
3119          005662 012737 005662 001110          MOV          $,$SLPERR          ;; SETUP THE ERROR LOOP ADDRESS
3120          SBTTL INITIALIZE A FEW VECTORS
3121          SBTTL INITIALIZE A FEW VECTORS
3122          005670 013746 000004          MOV          $,$ERRVEC          ;; SAVE ERROR VECTOR
3123          005674 012737 005730 000004          MOV          $64,$ERRVEC          ;; SET UP ERROR VECTOR
3124          005702 012737 177570 001140          MOV          $DSWR,$SWR          ;; SETUP FOR A HARDWARE SWICH REGISTER
3125          005710 012737 177570 001142          MOV          $DDISP,$DISPLAY          ;; AND A HARDWARE DISPLAY REGISTER
3126          005716 022777 177777 173214          CMP          #-1,$SWR          ;; TRY TO REFERENCE HARDWARE SWR
3127          005724 001012          BNE          $65          ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
3128          BR          $65          ;; AND THE HARDWARE SWR IS NOT = -1
3129          005726 000403          BR          $65          ;; BRANCH IF NO TIMEOUT
3130          005730 012716 005736          64$: MOV          $65,$(SP)          ;; SET UP FOR TRAP RETURN
3131          005734 000002          RTI
3132          005736 012737 000176 001140          65$: MOV          $SWREG,$SWR          ;; POINT TO SOFTWARE SWR
3133          005744 012737 000174 001142          MOV          $DISPREG,$DISPLAY
3134          005752 012637 000004          66$: MOV          (SP)+,$ERRVEC          ;; RESTORE ERROR VECTOR
3135
3136
3137
3138          005756 012737 000000 177776          MOV          $0,$PS          ;; SET PROCESSOR STATUS TO 0
3139          005764 012737 000200 000036          MOV          $200,$TRAPVEC+2          ;; TRAP PRIORITY = 4
3140          005772 013700 002716          MOV          $TRAPVEC,$RO          ;; GET RP VECTOR ADDRESS
3141          005776 012720 043042          MOV          $RPVEC,(RO)+          ;; THIS IS FOR UNTIMELY INTERRUPTS
3142          006002 012710 000340          MOV          $340,(RO)          ;; RPO4 INTERRUPT SERVICE ROUTINE
3143          PRIORITY = 7
3144          006006 004737 044050          JSR          PC,$STKINT          ;; INITILIZE THE TK
3145          006012 005037 045064          CLR          $PRITEM          ;; CLEAR FOR ERROR MESSAGE PRINTOUT
3146          006016 005737 005242          TST          $FIRST          ;; IS THIS FIRST TIME ROUND
3147          006022 001001          BNE          $1          ;; BRANCH IF NOT
3148          006024 000402          BR          $2          ;; BRANCH IF NOT
  
```

NOS

CERHADO MACY11 30(1046) 21-DEC-77 13:06 PAGE 66
CERHAD.P11 21-DEC-77 12:48

INITIALIZE THE COMMON TAGS

SEQ 0065

3149	006026	000137	006102		15:	TMP	2#SND1	
3150	006032	023737	000042	000046	25:	CMP	2#42,2#46	:ARE WE IN QV OR AUTO ACCEPT MODE?
3151	006040	001420				BEG	SND1	:IF YES, SKIP HEADER
3152	006042	104401	006050			TYPE	685	:TYPE ASCIZ STRING
3153	006046	000415				BR	675	:GET OVER THE ASCIZ
3154					685:	.ASCIZ	<15><12>/CERHADO	:RH70 CTRLR DIAG/
3155	006102				675:			
3156	006102	012737	177777	005242	SND1:	MOV	#-1,2#FIRST	:NEXT TIME DO NOT GIVE HEADEP
3157	006110	012737	037732	000114		MOV	#PARITY,2#PARE70	:PARITY VECTOR
3158	006116	012737	000340	000116		MOV	#340,2#PARE70+2	:PRIORITY 7
3159	006124	012737	037574	000004		MOV	#TIEOUT,2#ERRVEC	:TIMEOUT VECTOR
3160	006132	012737	000340	000006		MOV	#340,2#ERRVEC+2	:PRIORITY 7

3161
3162
3163
3164
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

```

*****
*TEST 1          SIZE FOR RH DEVICES
* THIS TEST DETERMINS WHICH RH DEVICE IS ON THE SYSTEM.
* IF THE SYSTEM HAS MORE THAN ONE RH DEVICE THEN ONLY ONE OF
* THE DEVICES WILL BE USED IN THE FOLLOWING TESTS.
* THE ONE THAT WILL BE USED IS THE FIRST ONE THAT IS PRESENT
* IN THE FOLLOWING LIST
* RS
* RP
* TM
*
* THE WAY THE TEST WORKS IS AS FOLLOWS:-
* A REFERENCE IS MADE TO THE CONTROL AND STATUS REGISTER 1
* FOR THE RS BY A TST INSTRUCTION. IF IT RESPONDS THEN IT IS
* ASSUMED PRESENT AND THE LOCATIONS FOR THE I/O REGISTERS ARE
* FILLED WITH THE APPROPRIATE ADDRESSES.
* THEN THE DRIVE TYPE REGISTER IS CHECKED TO HAVE GOOD
* VALUES.
* IF THE TST INSTRUCTION TRAPS TO A NON EXISTANT VECTOR
* THEN A SIMILAR ATTEMPT IS MADE TO A RP CONTROL AND STATUS
* REGISTER 1.
* THEN A TM IS TRYED.
* THEN A MIXED SYSTEM WITH MORE THAN ONE RH DEVICES IS TRYED.
*****

```

```

3205 006140 000004          TST1:  SCOPE
3206 006142 012737 000001 001212  MOV    #1,STIMES          ;;DO 1 ITERATION
3207 006150 005737 005012          TST    ST200             ;;IS IT A 200 START
3208 006154 001413          BEQ    2$               ;;BRANCH IF NOT 200 START
3209 006156 012700 005256          MOV    #PERRS,R0       ;; POINTER TO MOVE 9 WMRDS FROM
3210 006162 012701 000011          MOV    #9,R1           ;; NUMBER TO MOVE
3211 006166 012702 005300          MOV    #DEVPNT,R2      ;; POINTER TO MOVE 9 TORDS TO
3212 006172 012022          1$:   MOV    (R0)+,(R2)+   ;; FILL WORK TABLE
3213 006174 005301          DEC    R1
3214 006176 001375          BNE    1$              ;; BRANCH IF 9 NOT DONE
3215 006200 000137 006500          JMP    SETSIZ
3216 006204          2$:
3217 006204 104401 006212          TYPE   65$            ;; TYPE ASCIZ STRING
3218 006210 000417          BR     64$            ;; GET OVER THE ASCIZ
3219          ;;65$: .ASCIZ <15><12> HOW MANY RH TO BE TESTED
3220 006250          64$:
3221 006250 104401 006256          TYPE   67$            ;; TYPE ASCIZ STRING
3222 006254 000402          BR     66$            ;; GET OVER THE ASCIZ
3223          ;;67$: .ASCIZ <15><12>
3224 006262          66$:
3225 006262 104410          RDOCT
3226 006264 011637 005320          MOV    (SP),LTRY       ;; GET NUMBER OF RH TO BE TESTED
3227 006270 012637 005250          MOV    (SP)+,TMP0      ;; WORKING NUMBER OF RH
3228 006274 012737 000001 005252          MOV    #1,TMP1        ;; RH NUMBER TO BE TYPED
3229 006302 005000          CLR    R0              ;; INDEX TO WORKING TABLE
3230
3231 006304          3$:
3232 006304 104401 006312          TYPE   69$            ;; TYPE ASCIZ STRING
3233 006310 000417          BR     68$            ;; GET OVER THE ASCIZ
3234          ;;69$: .ASCIZ <15><12>/TYPE BASE ADDRESS OF RH NO/
3235 006350          68$:
3236 006350 013746 005252          MOV    TMP1,-(SP)
3237 006354 104405          TYPDS
3238 006356 104401 006364          TYPE   71$            ;; TYPE ASCIZ STRING
3239 006362 000402          BR     70$            ;; GET OVER THE ASCIZ
3240          ;;71$: .ASCIZ <15><12>/ /
3241 006370          70$:
3242 006370 104410          RDOCT
3243 006372 012660 005300          MOV    (SP)+,DEVPNT(R0) ;; FILL WORKING TABLE WITH BASE ADDRESS
3244 006376 104401 006404          TYPE   73$            ;; TYPE ASCIZ STRING
3245 006402 000420          BR     72$            ;; GET OVER THE ASCIZ
3246          ;;73$: .ASCIZ <15><12>/TYPE VECTOR ADDRESS OF RH NO/
3247          72$:
3248 006444          4$:
3249 006444 013746 005252          MOV    TMP1,-(SP)
3250 006450 104405          TYPDS
3251 006452 104401 000200          TYPE   ,CRLF
3252 006456 104410          RDOCT
3253 006460 012660 005310          MOV    (SP)+,VECPNT(R0) ;; FILL WORKING TABLE WITH VECTOR ADDRESS
3254 006464 005720          TST    (R0)+          ;; ADD 2 TO R0
3255 006466 005237 005252          INC    TMP1           ;; GET NEXT RH NUMBER
3256 006472 005337 005250          DEC    TMP0           ;; COUNT DOWN RH TO BE TESTED
3257 006476 001302          BNE    3$            ;; BRANCH IF ALL NOT DONE
3258
3259 006500 012700 005300          SETSIZ: ;R0 WILL HAVE DEVICE POINTER, R1 WILL HAVE VECTOR POINTER
3260 006504 012701 005310          MOV    #DEVPNT,R0     ;; DEVICE POINTER
          MOV    #VECPNT,R1 ;; VECTOR PMINTER
    
```

```

3261 006510 005037 005322 CLR TFOUND ;WILL BE USED AS POINTER
3262 006514 552: ;TO FOUND TABLE.
3263 006514 012737 010506 000004 15: MOV #554,ERRVEC ;TIME OUT VECTOR TO REDUCE L*P*
3264 006522 012037 005324 MOV (R0)+,TESTDV ;ADDRESS OF DEVICE TO BE TESTED
3265 006526 012137 005323 MOV (R1)+,TESTVC ;VECTOR TO BE TESTED
3266 006532 005777 176563 TST @TESTDV ;TRY TIME OUT ON DEVICE BASE
3267 ;IF NO TIME OUT OCCURS THEN DEVICE IN TESTDV IS PRESENT
3268 ;NOW FILL ALL REGISTER ADDRESS TABLE
3269 006536 012702 000026 MOV #22,R2 ;COUNT 22
3270 006542 012703 004060 MOV #RHCS1,R3 ;GET BASE LOCATION
3271 006546 013704 005324 MOV TESTDV,R4 ;GET BASE ADDRESS
3272 006552 010423 25: MOV R4,(R3)+ ;FILL PROPER ADDRESS
3273 006554 062704 000002 ADD #2,R4 ;INCREMENT ADDRESS BY 2
3274 006560 005302 DEC R2 ;COUNT DOWN
3275 006562 001373 BNE 25 ;BRANCH IF 22 NOT DONE.
3276 006564 005077 175300 CLR #RHCS2 ;SET UNIT NUMBER
3277 006570 556:
3278 006570 012737 005330 005360 35: MOV #TYPNT,POINTT ;POINTER TO DRIVE TYPE
3279 006576 012737 000014 005362 MOV #14,NTYPNT ;NUMBER OF DRIVE TYPES
3280
3281 006604 557:
3282 006604 022737 000001 005362 45: CMP #1,NTYPNT ;IS IT FOR TU
3283 006612 001424 BEQ 55 ;IF FOR TU BRANCH
3284 006614 017737 175266 001126 MOV #RHDT,$BDDAT ;READ DRIVE TYPE
3285 006622 012702 000014 MOV #14,R2 ;GET NUMBER FOR DT NOT DONE
3286 006626 163702 005362 SUB NTYPNT,R2 ;GET DT TO DO
3287 006632 006302 ASL R2 ;MULTIPLY R2 BY 2
3288 006634 026237 005330 001126 CMP TYPNT(R2),$BDDAT ;IS DRIVE TYPE FOUND
3289 006642 001402 BEQ 165 ;DRIVE TYPE FOUND SO BRANCH
3290 006644 000137 010532 JMP 558
3291 006650 032777 010000 175212 165: BIT #NED,#RHCS2 ;IS IT NON EXISTANT DRIVE
3292 006656 001437 BEQ 85 ;NED NOT SET, DRIVE TYPE FOUND SO BRANCH
3293 ;A DEVICE HAS NOT BEEN FOUND
3294 006660 000137 010524 JMP 555 ;A DEVICE HAS NOT BEEN FOUND SO BRANCH
3295
3296 ;TRYING THE TU16 DRIVE TYPE
3297 006664 005037 005364 55: CLR TMSLV ;SLAVE 0
3298 006670 013777 005364 175214 65: MOV TMSLV,#RHTC ;MOVE SLAVE NUMBER IN TAPE CONTROL
3299 006676 017737 175204 001126 MOV #RHDT,$BDDAT ;READ DRIVE TYPE
3300 006704 032737 002000 001126 BIT #SLVPA,$BDDAT ;IS SLAVE PRESENT
3301 006712 001014 BNE 75
3302 006714 022737 000007 005364 CMP #7,TMSLV ;IS 7 DONE
3303 006722 001002 BNE 175 ;BRANCH IF 7 DONE AND NONE FOUND
3304 006724 000137 010532 JMP 558
3305 006730 005237 005364 175: INC TMSLV ;GET NEXT SLAVE
3306 006734 012777 040000 175116 MOV #TRE,#RHCS1 ;CLEAR CONTROLLER ERRORS
3307 006742 000752 BR 65 ;TRY NEXT SLAVE
3308 ;A DEVICE HAS BEEN FOUND SAVE SLAVE
3309 006744 013702 005322 75: MOV TFOUND,R2 ;GET NO OF RH FOUND X2 FOR INDEX
3310 006750 013762 005364 005446 MOV TMSLV,FSLAVE(R2)
3311 ;A DEVICE HAS BEEN FOUND TEST VECTOR
3312 006756 012737 006772 000004 85: MOV #95,ERRVEC ;TIME OUT VECTOR TO REPORT ERROR
3313 006764 005777 176336 TST @TESTVC ;TRY TIME OUT ON VECTOR
3314 006770 000403 BR 105 ;A VECTOR EXISTS SO CONTINUE
3315 006772 104146 95: ERROR 146 ;A DEVICE BASE ADDRESS DID NOT
3316 ;TIME OUT BUT ITS CORRESPONDING

```

```

3317                                     ; VECTOR ADDRESS TIMED OUT
3318                                     ; HIT CONTINUE TO REPEAT THIS
3319                                     ; TEST
3320 006774 000000 HALT
3321 006776 000767 BR 8$ ; REPEAT THIS TEST
3322                                     ; A DEVICE HAS BEEN FOUND AND VECTOR RESPONDS SO STORE RESULTS
3323 007000 013702 005322 10$: MOV TFOUND,R2 ; GET NO OF RH FOUND X2 FOR INDEX
3324 007004 013762 005362 005366 MOV NTPNT,FDEVIC(R2) ; DEVICE 1=TU; 2,3,4,5,6,7=RP;
3325                                     ; 10,11,12,13,14=RS
3326 007012 017762 175052 005376 MOV DRHCS2,FUNITN(R2) ; GET RHCS2
3327 007020 042762 177770 005376 BIC #177770,FUNITN(R2) ; KEEP UNIT NUMBER
3328 007026 017762 175054 005406 MOV DRHDT,FTYPE(R2) ; DRIVE BYTE
3329 007034 013762 005326 005416 MOV TESTVC,FVECTA(R2) ; VECTOR
3330 007042 013762 005324 005426 MOV TESTDV,FBASEA(R2) ; BASE ADDRESS
3331 007050 013762 005322 005436 MOV TFOUND,FRHNM(R2) ; RH NUMBER X2 MINUS ONE
3332 007056 006262 005436 ASR FRHNM(R2) ; RH NUMBER MINUS ONE
3333 007062 005262 005436 INC FRHNM(R2) ; RH NUMBER
3334 007066 022762 000001 005366 CMP #1,FDEVIC(R2) ; IS IT TU
3335 007074 001016 BNE 11$ ; BRANCH IF NOT TU
3336 007076 012762 041674 005456 MOV #CMNDTM,FDRIVR(R2) ; DRIVER ADDRESS FOR TU
3337 007104 013746 005324 MOV TESTDV,-(SP) ; BASE ADDRESS
3338 007110 062716 000034 ADD #(<2*14>),(SP) ; 15TH ADDRESS IS BAE FOR TU
3339 007114 011662 005466 MOV (SP),FBAE(R2) ; SAVE BAE FOR TU
3340 007120 062716 000002 ADD #2,(SP) ; 16TH ADDRESS IS CS3 FOR TU
3341 007124 012662 005476 MOV (SP)+,FCS3(R2) ; SAVE CS3 FOR TU
3342
3343 007130 000511 BR SS1
3344 007132 022762 000002 005366 11$: CMP #2,FDEVIC(R2) ; IS IT RP DUAL PORT
3345 007140 001424 BEQ 12$ ; BRANCH IF RP FOUND
3346 007142 022762 000003 005366 CMP #3,FDEVIC(R2) ; IS IT RP
3347 007150 001420 BEQ 12$ ; BRANCH IF RP FOUND
3348 007152 022762 000004 005366 CMP #4,FDEVIC(R2) ; IS IT RP DUAL PORT
3349 007160 001414 BEQ 12$ ; BRANCH IF RP FOUND
3350 007162 022762 000005 005366 CMP #5,FDEVIC(R2) ; IS IT RP
3351 007170 001410 BEQ 12$ ; BRANCH IF RP FOUND
3352 007172 022762 000006 005366 CMP #6,FDEVIC(R2) ; IS IT RP DUAL PORT
3353 007200 001404 BEQ 12$ ; BRANCH IF RP FOUND
3354 007202 022762 000007 005366 CMP #7,FDEVIC(R2) ; IS IT RP
3355 007210 001016 BNE 13$ ; BRANCH IF NO RP
3356 007212 012762 041556 005456 12$: MOV #CMNDRP,FDRIVR(R2) ; DRIVER ADDRESS FOR RP
3357 007220 013746 005324 MOV TESTDV,-(SP) ; BASE ADDRESS
3358 007224 062716 000050 ADD #(<2*20>),(SP) ; 21ST ADDRESS IS BAE FOR RP
3359 007230 011662 005466 MOV (SP),FBAE(R2) ; SAVE BAE FOR RP
3360 007234 062716 000002 ADD #2,(SP) ; 22ND ADDRESS IS CS3 FOR RP
3361 007240 012662 005476 MOV (SP)+,FCS3(R2) ; SAVE CS3 FOR RP
3362
3363 007244 000443 BR SS1
3364 007246 022762 000010 005366 13$: CMP #10,FDEVIC(R2) ; IS IT RS03/LA
3365 007254 001420 BEQ 14$ ; BRANCH IF RS03 LA
3366 007256 022762 000011 005366 CMP #11,FDEVIC(R2) ; IS IT RS04/L
3367 007264 001414 BEQ 14$ ; BRANCH IF RS04/L
3368 007266 022762 000012 005366 CMP #12,FDEVIC(R2) ; IS IT RS04
3369 007274 001410 BEQ 14$ ; BRANCH IF RS04
3370 007276 022762 000013 005366 CMP #13,FDEVIC(R2) ; IS IT RS03/L
3371 007304 001404 BEQ 14$ ; BRANCH IF RS03 L
3372 007306 022762 000014 005366 CMP #14,FDEVIC(R2) ; IS IT RS03

```

```

3373 007314 001016          BNE      15$          ;BRANCH IF NOT RS03
3374 007316 012762 041466 C05456 14$: MOV      #CMNDRS,FDRIVR(R2) ;DRIVER ADDRESS FOR RS
3375 007324 013746 005324          MOV      TESTDV,-(SP)    ;BASE ADDRESS
3376 007330 062716 000030          ADD      #(<2*12>,(SP)  ;13TH ADDRESS IS BAE FOR RS
3377 007334 011662 005466          MOV      (SP),FBAE(R2)  ;SAVE BAE FOR RS
3378 007340 062716 000002          ADD      #2,(SP)       ;14TH ADDRESS IS CS3 FOR RS
3379 007344 012662 00547E          MOV      (SP)+,FCS3(R2);SAVE CS3 FOR RS
3380 007350 000401          BR       SS1          ;CONTINUE
3381 007352 000000          15$:  HALT          ;PROGRAM ERROR
3382
3383          ;NOW TYPE RESULT
3384
3385          SS1:
3386 007354          TYPE      65$          ;:TYPE ASCIZ STRING
3387 007354 104401 007362          BR       64$          ;:GET OVER THE ASCIZ
3388 007360 000406          ;:65$: .ASCIZ <15><12><ON RH NO/
3389 007376          64$:
3390 007376 016246 005436          MOV      FRHNM(R2),-(SP);TYPE RH NUMBER
3391 007402 104405          TYPDS
3392 007404 104401 007412          TYPE
3393 007410 000411          BR       67$          ;:TYPE ASCIZ STRING
3394          ;:67$: .ASCIZ /          ;:GET OVER THE ASCIZ
3395          66$:
3396 007434          MOV      FBASEA(R2),-(SP);TYPE BASE ADDRESS
3397 007434 016246 005426          TYPDC
3398 007440 104402          TYPE
3399 007442 104401 007450          BR       69$          ;:TYPE ASCIZ STRING
3400          ;:69$: .ASCIZ / FOUND/          ;:GET OVER THE ASCIZ
3401 007460          68$:
3402 007460 022762 000001 005366          CMP      #1,FDEVIC(R2)  ;IS IT TU
3403 007466 001020          BNE      1$          ;BRANCH IF NOT TU
3404 007470 104401 007476          TYPE      71$          ;:TYPE ASCIZ STRING
3405 007474 000410          BR       70$          ;:GET OVER THE ASCIZ
3406          ;:71$: .ASCIZ / TU16 AT SLAVE/
3407 007516          70$:
3408 007516 016246 005446          MOV      FSLAVE(R2),-(SP);TYPE SLAVE NUMBER
3409 007522 104405          TYPDS
3410 007524 000137 010324          JMP      6$
3411          ;THE FOLLOWING TYPES OUT
3412          ;THE DEVICE AND REDEFINES
3413          ;FDEVIC: 1=TU, 2=RP DUAL PORT
3414          ;3=RP SINGLE PORT
3415          ;4=RS04/L RS04
3416          ;5=RS03/LA,RS03/L,RS03
3417 007530 022762 000002 005366 1$: CMP      #2,FDEVIC(R2)  ;IS IT RPO6 DUAL PORT
3418 007536 001016          BNE      2$          ;BRANCH IF NOT RPO6 DUAL PORT
3419 007540 104401 007546          TYPE      73$          ;:TYPE ASCIZ STRING
3420 007544 000411          BR       72$          ;:GET OVER THE ASCIZ
3421          ;:73$: .ASCIZ / RPO6 DUAL PORT /
3422 007570          72$:
3423 007570 000137 010324          JMP      6$
3424 007574 022762 000003 005366 2$: CMP      #3,FDEVIC(R2)  ;IS IT RPO6 SINGLE PORT
3425 007602 001016          BNE      3$          ;BRANCH IF NOT RPO6 SINGLE PORT
3426 007604 104401 007612          TYPE      75$          ;:TYPE ASCIZ STRING
3427 007610 000411          BR       74$          ;:GET OVER THE ASCIZ
3428          ;:75$: .ASCIZ RPO6 SINGLE PORT /

```

F06

CERHADO MACY11 301046, 21-DEC-77 13:06 PAGE 71
 CERHAD.P11 21-DEC-77 12:48

SIZE FOR RH DEVICES

SEG 0070

```

3429 007634          74$:
3430 007634 000137 010324      JMP      6$
3431 007640 022762 000004 005366 3$:  CMP      #4,FDEVIC(R2)  ;IS IT RPO5 DUAL PORT
3432 007646 001020          BNE      4$            ;BRANCH IF NOT RPO5 DUAL PORT
3433 007650 012762 000002 005366      MOV      #2,FDEVIC(R2)
3434 007656 104401 007664      TYPE    77$          ;;TYPE ASCIZ STRING
3435 007662 000410          BR       76$          ;;GET OVER THE ASCIZ
3436          ;;77$: .ASCIZ / RPO5 DUAL PORT/
3437 007704          76$:
3438 007704 000137 010324      JMP      6$
3439 007710 022762 000005 005366 4$:  CMP      #5,FDEVIC(R2)  ;IS IT RPO5 SINGLE PORT
3440 007716 001020          BNE      8$            ;BRANCH IF NOT RPO5 SINGLE PORT
3441 007720 012762 000003 005366      MOV      #3,FDEVIC(R2)
3442 007726 104401 007734      TYPE    79$          ;;TYPE ASCIZ STRING
3443 007732 000411          BR       78$          ;;GET OVER THE ASCIZ
3444          ;;79$: .ASCIZ / RPO5 SINGLE PORT/
3445 007756          78$:
3446 007756 000562          BR       6$
3447 007760 022762 000006 005366 8$:  CMP      #6,FDEVIC(R2)  ;IS IT RPO4 DUAL PORT
3448 007766 001017          BNE      9$            ;BRANCH IF NOT RPO4 DUAL PORT
3449 007770 012762 000002 005366      MOV      #2,FDEVIC(R2)
3450 007776 104401 010004      TYPE    81$          ;;TYPE ASCIZ STRING
3451 010002 000410          BR       80$          ;;GET OVER THE ASCIZ
3452          ;;81$: .ASCIZ / RPO4 DUAL PORT/
3453 010024          80$:
3454 010024 000537          BR       6$
3455 010026 022762 000007 005366 9$:  CMP      #7,FDEVIC(R2)  ;IS IT RPO4 SINGLE PORT
3456 010034 001020          BNE     10$           ;BRANCH IF NOT RPO4 SINGLE PORT
3457 010036 012762 000003 005366      MOV      #3,FDEVIC(R2)
3458 010044 104401 010052      TYPE    83$          ;;TYPE ASCIZ STRING
3459 010050 000411          BR       82$          ;;GET OVER THE ASCIZ
3460          ;;83$: .ASCIZ / RPO4 SINGLE PORT/
3461 010074          82$:
3462 010074 000513          BR       6$
3463 010076 022762 000010 005366 10$: CMP      #10,FDEVIC(R2) ;IS IT RS03-LA
3464 010104 001014          BNE     11$           ;BRANCH IF NOT RS03-LA
3465 010106 012762 000005 005366      MOV      #5,FDEVIC(R2)
3466 010114 104401 010122      TYPE    85$          ;;TYPE ASCIZ STRING
3467 010120 000405          BR       84$          ;;GET OVER THE ASCIZ
3468          ;;85$: .ASCIZ / RS03-LA/
3469 010134          84$:
3470 010134 000473          BR       6$
3471 010136 022762 000011 005366 11$: CMP      #11,FDEVIC(R2) ;IS IT RS04-L
3472 010144 001013          BNE     12$           ;BRANCH IF NOT RS04-L
3473 010146 012762 000004 005366      MOV      #4,FDEVIC(R2)
3474 010154 104401 010162      TYPE    87$          ;;TYPE ASCIZ STRING
3475 010160 000404          BR       86$          ;;GET OVER THE ASCIZ
3476          ;;87$: .ASCIZ / RS04-L/
3477 010172          86$:
3478 010172 000454          BR       6$
3479 010174 022762 000012 005366 12$: CMP      #12,FDEVIC(R2) ;IS IT RS04
3480 010202 001012          BNE     13$           ;BRANCH IF NOT RS04
3481 010214 012762 000004 005366      MOV      #4,FDEVIC(R2)
3482 010212 104401 010220      TYPE    89$          ;;TYPE ASCIZ STRING
3483 010216 000403          BR       88$          ;;GET OVER THE ASCIZ
3484          ;;89$: .ASCIZ / RS04/
    
```


G06

CERHADO MACY11 3011046 21-DEC-77 13:06 PAGE 72
 CERHAD.P11 21-DEC-77 12:48 T1

SIZE FOR RH DEVICES

SEE C071

```

3485 010226          88$:
3486 010226 000436
3487 010230 022762 000013 005366 13$: BR 6$
3488 010236 001013          CMP 13,FDEVIC(R2) ;IS IT RS03-L
3489 010240 012762 000005 005366      BNE 14$ ;BRANCH IF NOT RS03-L
3490 010246 104401 010254      MOV 15,FDEVIC(R2)
3491 010252 000404          TYPE 91$ ;:TYPE ASCIZ STRING
3492          BR 90$ ;:GET OVER THE ASCIZ
3493          :.91$: .ASCIZ / RS03-L/
3494 010264          90$:
3495 010266 000417          BR 6$
3496 010274 022762 000014 005366 14$: CMP 14,FDEVIC(R2) ;IS IT RS03
3497 010276 001012          BNE 5$ ;BRANCH IF NOT RS03
3498 010276 012762 000005 005366      MOV 15,FDEVIC(R2)
3499 010304 104401 010312      TYPE 93$ ;:TYPE ASCIZ STRING
3500 010310 000403          BR 92$ ;:GET OVER THE ASCIZ
3501          :.93$: .ASCIZ / RS03/
3502 010320          92$:
3503 010322 000000          BR 6$
3504 010324          5$: HALT ;PROGRAM ERROR
3505 010324 104401 010332      6$: TYPE 95$ ;:TYPE ASCIZ STRING
3506 010330 000411          BR 94$ ;:GET OVER THE ASCIZ
3507          :.95$: .ASCIZ <15><12> AT UNIT NUMBER/
3508 010354          94$:
3509 010354 016246 005376      MOV FUNITN(R2),-(SP) ;TYPE UNIT NUMBER
3510 010360 104405          TYPDS
3511 010362 104401 010370      TYPE 97$ ;:TYPE ASCIZ STRING
3512 010366 000411          BR 96$ ;:GET OVER THE ASCIZ
3513          :.97$: .ASCIZ / VECTOR /
3514 010412          96$:
3515 010412 016246 005416      MOV FVECTR(R2),-(SP) ;TYPE VECTOR
3516 010416 104402          TYPOC
3517 010420 104401 010426      TYPE 99$ ;:TYPE ASCIZ STRING
3518 010424 000402          BR 98$ ;:GET OVER THE ASCIZ
3519          :.99$: .ASCIZ <15><12>/ /
3520 010432          98$:
3521 010432 104401 010440      TYPE 101$ ;:TYPE ASCIZ STRING
3522 010436 000402          BR 100$ ;:GET OVER THE ASCIZ
3523          :.101$: .ASCIZ <15><12>/ /
3524 010444          100$:
3525
3526          :UPDATE TFOUND
3527 010444 062737 000002 005322      ADD #2,TFOUND ;GET READY FOR NEXT DEVICE
3528
3529          :ALL UNITS DONE? IF LTRY BECOMES ZERO YES ALL DONE
3530 010452 005337 005320      DEC LTRY ;REDUCE DEVICES LEFT TO TRY
3531 010456 001402          BEQ 7$ ;BRANCH IF ALL COMPLETE
3532 010460 000137 006514          JMP 52$ ;ALL NOT DONE
3533 010464          7$:
3534 010464 013746 005322      52$: MOV TFOUND, -(SP) ;GET TWICE NUMBER OF RH FOUND
3535 010470 006216          ASR (SP) ;DIVIDE BY 2
3536 010472 011637 005506      MOV (SP),TSTUNT
3537 010476 012637 005510      MOV (SP)+,WORUNT
3538 010502 000137 010610      JMP TST2 ;JUMP TO NEXT TEST
3539
3540          ;A RH TIMED OUT SO TRY NEXT UNIT OR END TRYING
  
```

```

3541 010506 005337 005320 554: DEC LTR1 :RH TIMES OUT SO DECREASE RH TYPES
3542 010512 001402 BEQ 1$ :BRANCH IF ALL DONE
3543 010514 000137 006514 1$: JMP 552 :ALL NOT DONE SO JUMP
3544 010520 000137 010464 1$: JMP 553 :ALL DONE SO STORE NUMBER OF RH FOUND
3545
3546 ;A DRIVE TYPE DID NOT MATCH SO TRY NEXT
3547 010524 012777 040000 173326 555: MOV #TIRE,DRHCS1 :CLEAR NED ERROR
3548 010532 005337 005362 558: DEC NTPNT :DECREASE NUMBER OF DRIVE TYPES TRIED
3549 010536 001402 1$ BEQ 1$ :BRANCH IF ALL DONE
3550 010540 000137 006604 1$: JMP 557 :ALL NOT DONE
3551 010544 005277 173320 1$: INC DRHCS2 :GET NEXT UNIT NO.
3552 010550 012777 040000 173302 555: MOV #TIRE,DRHCS1 :CLEAR NED ERROR
3553 010556 017746 173306 558: MOV DRHCS2, -(SP) :GET RHCS2
3554 010562 042716 177770 558: BIC #177770, SP :GET UNIT NO
3555 010566 022726 000010 2$ CMP #8, (SP) :ARE 7 DONE
3556 010572 001402 2$ BEQ 2$ :BRANCH IF 7 DONE
3557 010574 000137 006570 556: JMP 556 :NOT DONE SO TRY NEXT UNIT NO
3558
3559 010600 104147 2$: ERROR 147 :DEVICE ADDRESS DID NOT TIME
3560 010602 000000 HALT :OUT BUT NO UNITS HAD
3561 :APPROPRIATE DRIVE TYPE.
3562 010604 000137 010506 JMP 554
3563
3564
3565
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575 *****
3576 *TEST 2 UNIT UNDER TEST
3577 * THIS TYPES THE UNIT TO BE TESTED
3578 * AND IS THE FIRST TEST AFTER THE END OF THE PROGRAM
3579 *****
3580 ST2: SCOPE
3581 MOV #1, $TIMES ;:DC 1 ITERATION
3582 MOV #TIMEOUT, @ERRVEC ;:TIMEOUT VECTOR
3583 MOV #340, @ERRVEC+2 ;:PRIORITY 7
3584 MOV WORUNT, -(SP) ;:GET WORKING RH NUMBER LEFT
3585 MOV TSTUNT, R2 ;:GET TOTAL NO OF RH FOUND
3586 SUB (SP)+, R2 ;:NO OF RH TO BE TESTED
3587 ASL R2 ;:INDEX FOR RH TO BE TESTED.
3588 TYPE 65$ ;:TYPE ASCIZ STRING
3589 BR 64$ ;:GET OVER THE ASCIZ
3590 ;:65$: .ASCIZ <15><12>
3591 ;:64$:
3592 TYPE 67$ ;:TYPE ASCIZ STRING
3593 BR 66$ ;:GET OVER THE ASCIZ
3594 ;:67$: .ASCIZ <15><12> TESTING RH NO/
3595 ;:66$:
3596 MOV FRHNM(R2), -(SP) ;:TYPE RH NO.
3597 TYPDS
3598 TYPE 69$ ;:TYPE ASCIZ STRING
3599 BR 68$ ;:GET OVER THE ASCIZ

```

```

3597          ::69$: .ASCIZ / USING
3598 010744    69$:
3599 010744    104401 010752    TYPE 71$      ;;TYPE ASCIZ STRING
3600 010750    000404          BR 70$      ;;GET OVER THE ASCIZ
3601          ::71$: .ASCIZ / BASE
3602 010762    70$:
3603 010762    016246 005426    MOV FBASEA(R2),-(SP) ;TYPE BASE ADDRESS
3604 010766    104402          TYPDC
3605 010770    022762 000001 005366    CMP #1,FDEVIC(R2)  ;IS IT TU
3606 010776    001017          BNE 1$      ;BRANCH IF NOT TU
3607 011000    104401 011006    TYPE 73$      ;;TYPE ASCIZ STRING
3608 011004    000410          BR 72$      ;;GET OVER THE ASCIZ
3609          ::73$: .ASCIZ / TU16 AT SLAVE
3610 011026    72$:
3611 011026    016246 005446    MOV FSLAVE(R2),-(SP) ;TYPE SLAVE NUMBER
3612 011032    104405          TYPDS
3613 011034    000467          BR 6$
3614 011036    022762 000002 005366    1$: CMP #2,FDEVIC(R2)  ;IS IT RP DUAL PORT
3615 011044    001014          BNE 2$      ;BRANCH IF NOT RP DUAL PORT
3616 011046    104401 011054    TYPE 75$      ;;TYPE ASCIZ STRING
3617 011052    000410          BR 74$      ;;GET OVER THE ASCIZ
3618          ::75$: .ASCIZ / RP DUAL PORT /
3619 011074    74$:
3620 011074    000447          BR 6$
3621 011076    022762 000003 005366    2$: CMP #3,FDEVIC(R2)  ;IS IT RP SINGLE PORT
3622 011104    001014          BNE 3$      ;BRANCH IF NOT RP
3623 011106    104401 011114    TYPE 77$      ;;TYPE ASCIZ STRING
3624 011112    000410          BR 76$      ;;GET OVER THE ASCIZ
3625          ::77$: .ASCIZ / RP SINGLE PORT /
3626 011134    76$:
3627 011134    000427          BR 6$
3628 011136    022762 000004 005366    3$: CMP #4,FDEVIC(R2)  ;IS IT RS04
3629 011144    001007          BNE 4$      ;BRANCH IF NOT RS04
3630 011146    104401 011154    TYPE 79$      ;;TYPE ASCIZ STRING
3631 011152    000403          BR 78$      ;;GET OVER THE ASCIZ
3632          ::79$: .ASCIZ / RS04 /
3633 011162    78$:
3634 011162    000414          BR 6$
3635 011164    022762 000005 005366    4$: CMP #5,FDEVIC(R2)  ;IS IT RS03
3636 011172    001007          BNE 5$      ;BRANCH IF NOT RS03
3637 011174    104401 011202    TYPE 81$      ;;TYPE ASCIZ STRING
3638 011200    000403          BR 80$      ;;GET OVER THE ASCIZ
3639          ::81$: .ASCIZ / RS03 /
3640 011210    80$:
3641 011210    000401          BR 6$
3642 011212    000000          5$: HALT ;PROGRAM ERROR
3643          6$:
3644 011214          BR 6$
3645 011214    104401 011222    TYPE 83$      ;;TYPE ASCIZ STRING
3646 011220    000411          BR 82$      ;;GET OVER THE ASCIZ
3647          ::83$: .ASCIZ <15, 12> AT UNIT NUMBER /
3648 011244    82$:
3649 011244    016246 005376    MOV FUNITN(R2),-(SP) ;TYPE UNIT NUMBER
3650 011250    104405          TYPDS
3651 011252    104401 011260    TYPE 85$      ;;TYPE ASCIZ STRING
3652 011256    000411          BR 84$      ;;GET OVER THE ASCIZ
    
```

```

3653      011302      016246      005416      005010      :;85$: .ASCIZ      VECTOR
3654      011302      016246      005416      005010      :84$:      MOV      FVECTR(R2),- SP) ;TYPE VECTOR
3655      011302      016246      005416      005010      TYP0C
3656      011306      104402
3657      011310      012703      000024      MOV      #20, R3      ;COUNT 20
3658      011314      012704      004060      MOV      #RHCS1, R4    ;GET BASE LOCATION
3659      011320      016205      005426      MOV      FBASEA(R2), R5 ;GET BASE ADDRESS
3660      011324      010524      000002      7$:      MOV      R5, (R4)+     ;FILL PROPER ADDRESS
3661      011326      062705      000002      ADD      #2, R5      ;INCREMENT BY 2
3662      011332      005303      DEC      R3          ;COUNT DOWN
3663      011334      001373      BNE     7$          ;BRANCH IF 20 NOT DONE
3664
3665      011336      016237      005376      005010      MOV      FJNITN(R2), UNIT ;UNIT NUMBER
3666      011344      016237      005416      005512      MOV      FVECTR(R2), VECTOR ;VECTOR
3667      011352      016237      005446      004660      MOV      FSLAVE(R2), SLAVE ;SLAVE NO
3668      011360      016237      005456      004640      MOV      FDRIVR(R2), COMND ;DRIVER ADDRESS
3669      011366      016237      005466      004130      MOV      FBAE(R2), RHBAE ;BAE ADDRESS
3670      011374      016237      005476      004132      MOV      FCS3 R2), RHCS3 ;CS3 ADDRESS
3671
3672      011402      005037      045064      CLR     PRITEM
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682      :;*****
3683      :*TEST 3      B) BANG RHCS1
3684
3685      :*
3686      :*      TEST LOADING AND READING OF ALL POSSIBLE BITS
3687      :*      IN RHCS1 REGISTER.
3688      :*      USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC)
3689      :*      AND WALKING 0'S (-2 -3 -5 ETC)
3690      :*      IN THIS TEST SC!TRE!MCPE!DVA!BIT12!BIT10!RDY!GO BITS WILL NOT BE WRITTEN INTO
3691      :*      AND RDY!DVA BITS WILL ALWAYS BE SET
3692      :*      AND BIT10 BITS WILL ALWAYS BE CLEARED
3693      :*      IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING
3694      :*      THEN A RH CLEAR (RHCS2 BIT #5) WILL BE
3695      :*      GIVEN TO AID SCOPE SYNC'S ON THE CLEAR
3696      :*      SIGNAL.
3697
3698      011406      000004      :;*****
3699      011410      012706      001000      †ST3:      SCOPE
3700      011414      012737      000003      004656      MOV      #STACK, SP      ;RESET STACK
3701
3702      011422      004737      040306      MOV      #3, @#TSTNM     ;SAVE TEST NUMBER
3703
3704
3705
3706      011426      012737      176201      004650      JSR     PC, @#CLDISK    ;GIVE RH INITIALIZE
3707      011434      012737      004200      004652      ;SETUP UNIT NUBER
3708      011442      012737      002000      004654      ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
3709
3710      011426      012737      176201      004650      MOV      #SC!TRE!MCPE!DVA!BIT12!BIT10!RDY!GO, WRTBIT ;SC!TRE!MCPE!DVA!BIT12!B
3711      011434      012737      004200      004652      MOV      #RDY!DVA, SETBIT ;RDY!DVA ARE BITS ALWAYS SET
3712      011442      012737      002000      004654      MOV      #BIT10, CLRBIT  ;BIT10 ARE BITS ALWAYS CLEARED

```

```

3709
3710
3711 011450 012737 011474 001110
3712 011456 012705 000001
3713 011462 010537 001124 1$:
3714 011466 042737 176201 001124
3715 011474 032737 041400 001140 2$:
3716 011502 001403
3717 011504 052777 000040 172356
3718
3719 011512 013777 001124 172340 3$:
3720 011520 017737 172334 001126
3721 011526 043737 004654 001124
3722 011534 053737 004652 001124
3723 011542 023737 001124 001126
3724 011550 001404
3725 011552 013737 004060 001122
3726 011560 104137
3727
3728
3729
3730
3731 011562 000241 4$:
3732 011564 006305
3733 011566 103335
3734
3735
3736
3737 011570 012737 011614 001110
3738 011576 012705 177776
3739 011602 010537 001124 5$:
3740 011606 042737 176201 001124
3741 011614 032737 041400 001140 6$:
3742 011622 001403
3743 011624 012777 000040 172236
3744
3745 011632 013777 001124 172220 7$:
3746 011640 017737 172214 001126
3747 011646 053737 004652 001124
3748 011654 043737 004654 001124
3749 011662 023737 001124 001126
3750 011670 001404
3751 011672 013737 004060 001122
3752 011700 104137
3753
3754
3755
3756
3757
3758 011702 000261 8$:
3759 011704 006105
3760 011706 103735
3761
3762
3763
3764

```

```

;FLOAT 1'S THRU THE RHCS1 REGISTER
MOV #25,$LPERR ;SET LOOP ON ERROR ADDRESS
MOV #1,R5 ;GETTING READY TO FLOAT A ONE
MOV R5,$GDDAT ;START WITH DATA
BIC #SC!TRE!MCPE!DVA!BIT12!BIT10!RDY!GO,$GDDAT ;CLEAR BITS NOT WRITTEN
BIT #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
BEQ 3$ ;BRANCH IF ANY ONE SET
BIS #CLR,$RHCS2 ;GIVE CLEAR FOR SCOPE
;SYNC IF IN DEBUG:
MOV $GDDAT,$RHCS1 ;WRITE RHCS1 REGISTER
MOV $RHCS1,$BDDAT ;READ RHCS1 REGISTER
BIC CLRBIT,$GDDAT ;CLEAR ALWAYS CLEARED BITS
BIS SETBIT,$GDDAT ;SET ALWAYS SET BITS
CMP $GDDAT,$BDDAT ;TEST
BEQ 4$ ;BRANCH IF GOOD
MOV RHCS1,$BDADR ;GET REGISTER ADDRESS
ERROR 137 ;ONE WAS BEING FLOATED
;THRU RHCS1 REGISTER
;ON READING RHCS1 BACK
;IT DID NOT CONTAIN WHAT
;WAS EXPECTED.
CLC ;CLEAR CARRY
ASL R5 ;GET 1 ONE LEFT
BCC 1$ ;BRANCH IF 16 NOT DONE

;FLOAT A ZERO THRU RHCS1 REGISTER.
MOV #65,$LPERR ;SET LOOP BACK POINT.
MOV #177776,R5 ;GET READY TO FLOAT A ZERO
MOV R5,$GDDAT ;GET READY TO WRITE DATA
BIC #SC!TRE!MCPE!DVA!BIT12!BIT10!RDY!GO,$GDDAT ;CLEAR BITS NOT WRITTEN
BIT #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
BEQ 7$ ;BRANCH IF ANY OF THE ABOVE SET
MOV #CLR,$RHCS2 ;CLEAR FOR SCOPE AID TO
;SYNC IF IN DEBUG
MOV $GDDAT,$RHCS1 ;WRITE INTO RHCS1.
MOV $RHCS1,$BDDAT ;READ RHCS1
BIS SETBIT,$GDDAT ;SET BITS ALWAYS SET
BIC CLRBIT,$GDDAT ;CLEAR BITS ALWAYS 0
CMP $GDDAT,$BDDAT ;TEST
BEQ 8$ ;BRANCH IF GOOD
MOV RHCS1,$BDADR ;GET REGISTER ADDRESS
ERROR 137 ;ZERO WAS BEING FLOATED
;THRU RHCS1 REGISTER
;ON READING IT BACK IT
;DID NOT CONTAIN WHAT
;WAS EXPECTED
SEC
ROL R5
BCS 5$

```

```

*****
; *TEST 4 BIT BANG RHCS2

```

```

3765
3766      :*      TEST LOADING AND READING OF ALL POSSIBLE BITS
3767      :*      IN RHCS2 REGISTER.
3768      :*      USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC)
3769      :*      AND WALKING 0'S (-2,-3,-5 ETC)
3770      :*      IN THIS TEST 177740 BITS WILL NOT BE WRITTEN INTO
3771      :*      AND IR BITS WILL ALWAYS BE SET
3772      :*      IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING
3773      :*      THEN A RM CLEAR (RHCS2 BIT #5) WILL BE
3774      :*      GIVEN TO AID SCOPE SYNC'S ON THE CLEAR
3775      :*      SIGNAL.
3776
3777      :*****
3778      ST4:  SCOPE
3779      MOV      #STACK,SP      ;RESET STACK
3780      MOV      #4,#TSTNM      ;SAVE TEST NUMBER
3781
3782      JSR      PC,#CLDISK     ;GIVE RM INITIALIZE
3783      ;SETUP UNIT NUMBER
3784      ;CLEAR RHC AND FUNCTION BITS IN RHCS1
3785
3786      MOV      #177740,WRTBIT ;177740 ARE BITS NOT WRITTEN INTO
3787      MOV      #IR,SETBIT     ;IR ARE BITS ALWAYS SET
3788
3789      ;FLOAT 1'S THRU THE RHCS2 REGISTER
3790      MOV      #25,$LPERR     ;SET LOOP ON ERROR ADDRESS
3791      MOV      #1,R5         ;GETTING READY TO FLOAT A ONE
3792      MOV      R5,$GDDAT     ;START WITH DATA
3793      BIC      #177740,$GDDAT ;CLEAR BITS NOT WRITTEN INTO
3794      BIT      #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
3795      BEQ      3$           ;BRANCH IF ANY ONE SET
3796      BIS      #CLR,#RHCS2   ;GIVE CLEAR FOR SCOPE
3797      ;SYNC IF IN DEBUG:
3798      MOV      $GDDAT,#RHCS2 ;WRITE RHCS2 REGISTER
3799      MOV      #RHCS2,$BDDAT  ;READ RHCS2 REGISTER
3800      BIS      SETBIT,$GDDAT  ;SET ALWAYS SET BITS
3801      CMP      $GDDAT,$BDDAT  ;TEST
3802      BEQ      4$           ;BRANCH IF GOOD
3803      MOV      RHCS2,$BDADR   ;GET REGISTER ADDRESS
3804      ERROR   137          ;ONE WAS BEING FLOATED
3805      ;THRU RHCS2 REGISTER
3806      ;ON READING RHCS2 BACK
3807      ;IT DID NOT CONTAIN WHAT
3808      ;WAS EXPECTED.
3809      CLC                ;CLEAR CARRY
3810      ASL      R5          ;GET 1 ONE LEFT
3811      BCC      1$         ;BRANCH IF 16 NOT DONE
3812
3813      ;FLOAT A ZERO THRU RHCS2 REGISTER.
3814
3815      MOV      #65,$LPERR     ;SET LOOP BACK POINT.
3816      MOV      #177776,R5    ;GET READY TO FLOAT A ZERO
3817      MOV      R5,$GDDAT     ;GET READY TO WRITE DATA
3818      BIC      #177740,$GDDAT ;CLEAR BITS NOT WRITTEN INTO
3819      BIT      #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
3820      BEQ      7$           ;BRANCH IF ANY OF THE ABOVE SE

```


BI* BANG RHCS3

```

3877 012274 013777 001124 171630 3$: MOV $GDDAT,2RHCS3 ;WRITE RHCS3 REGISTER
3878 012302 017737 171624 001126 MOV 2RHCS3,$BDDAT ;READ RHCS3 REGISTER
3879 012310 043737 004654 001124 BIC CLRBIT,$GDDAT ;CLEAR ALWAYS CLEARED BITS
3880 012316 053737 004652 001124 BIS SETBIT,$GDDAT ;SET ALWAYS SET BITS
3881 012324 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST
3882 012332 001404 BEQ 4$ ;BRANCH IF GOOD
3883 012334 013737 004132 001122 MOV RHCS3,$BODADR ;GET REGISTER ADDRESS
3884 012342 104137 ERROR 137 ;ZERO WAS BEING FLOATED
3885 ;THRU RHCS3 REGISTER
3886 ;ON READING RHCS3 BACK
3887 ;IT DID NOT CONTAIN WHAT
3888 ;WAS EXPECTED.
3889 012344 000241 4$: CLC ;CLEAR CARRY
3890 012346 006305 ASL R5 ;GET 1 ONE LEFT
3891 012350 103335 BCC 1$ ;BRANCH IF 16 NOT DONE
3892 ;
3893 ;FLOAT A ZERO THRU RHCS3 REGISTER.
3894 ;
3895 012352 012737 012376 001110 MOV #6,$SLPERR ;SET LOOP BACK POINT.
3896 012360 012705 177776 MOV #177776,R5 ;GET READY TO FLOAT A ZERO
3897 012364 010537 001124 5$: MOV R5,$GDDAT ;GET READY TO WRITE DATA
3898 012370 042737 177660 001124 BIC #177660,$GDDAT ;CLEAR BITS NOT WRITTEN INTO
3899 012376 032737 041400 001140 6$: BIT #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
3900 012404 001403 BEQ 7$ ;BRANCH IF ANY OF THE ABOVE SET
3901 012406 012777 000040 171454 MOV #CLR,2RHCS2 ;CLEAR FOR SCOPE AID TO
3902 ;SYNC IF IN DEBUG
3903 012414 013777 001124 171510 7$: MOV $GDDAT,2RHCS3 ;WRITE INTO RH'XA.
3904 012422 017737 171504 001126 MOV 2RHCS3,$BDDAT ;READ RHCS3
3905 012430 053737 004652 001124 BIS SETBIT,$GDDAT ;SET BITS ALWAYS SET
3906 012436 043737 004654 001124 BIC CLRBIT,$GDDAT ;CLEAR BITS ALWAYS 0
3907 012444 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST
3908 012452 001404 BEQ 8$ ;BRANCH IF GOOD
3909 012454 013737 004132 001122 MOV RHCS3,$BODADR ;GET REGISTER ADDRESS
3910 012462 104137 ERROR 137 ;ZERO WAS BEING FLOATED
3911 ;THRU RHCS3 REGISTER
3912 ;ON READING IT BACK IT
3913 ;DID NOT CONTAIN WHAT
3914 ;WAS EXPECTED
3915 ;
3916 012464 000261 8$: SEC
3917 012466 006105 ROL R5
3918 012470 103735 BCS 5$
3919 ;
3920 ;
3921 ;*****
3922 ;*TEST 6 BIT BANG RHWC
3923 ;
3924 ;* TEST LOADING AND READING OF ALL POSSIBLE BITS
3925 ;* IN RHWC REGISTER.
3926 ;* USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC)
3927 ;* AND WALKING 0'S (-2,-3,-5 ETC)
3928 ;* IN THIS TEST 0 BITS WILL NOT BE WRITTEN INTO
3929 ;* AND 0 BITS WILL ALWAYS BE SET
3930 ;* IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING
3931 ;* THEN A RH CLEAR (RHCS2 BIT #5) WILL BE
3932 ;* GIVEN TO AID SCOPE SYNCs ON THE CLEAR

```



```

3933 ;* SIGNAL.
3934
3935 ;*****
3936 012472 050004 001000 004656 15: SCOPE
3937 012474 012706 001000 004656 MOV #STACK,SP ;RESET STACK
3938 012500 012737 000006 004656 MOV #6,#TSTNM ;SAVE TEST NUMBER
3939
3940 012506 004737 040306 JSR PC,#CLDISK ;GIVE RM INITIALIZE
3941 ;SETUP UNIT NUMBER
3942 ;CLEAR RHWG AND FUNCTION BITS IN RHCS1
3943
3944 012512 012737 000000 004650 MOV #0,WRTBIT ;0 ARE BITS NOT WRITTEN INTO
3945 012520 012737 000000 004652 MOV #0,SETBIT ;0 ARE BITS ALWAYS SET
3946
3947 ;FLOAT 1'S THRU THE RHWG REGISTER
3948 012526 012737 012552 001110 MOV #25,SLPERR ;SET LOOP ON ERROR ADDRESS
3949 012534 012705 000001 001110 MOV #1,R5 ;GETTING READY TO FLOAT A ONE
3950 012540 010537 001124 001110 15: MOV R5,$GDDAT ;START WITH DATA
3951 012544 042737 000000 001124 BIC #0,$GDDAT ;CLEAR BITS NOT WRITTEN INTO
3952 012552 032737 041400 001140 25: BIT #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
3953 012560 001403 000040 171300 BEQ 35 ;BRANCH IF ANY ONE SET
3954 012562 052777 000040 171300 BIS #CLR,#RHCS2 ;GIVE CLEAR FOR SCOPE
3955 ;SYNC IF IN DEBUG:
3956 012570 013777 001124 171264 35: MOV $GDDAT,#RHWC ;WRITE RHWG REGISTER
3957 012576 017737 171260 001126 MOV #RHWC,$BDDAT ;READ RHWG REGISTER
3958 012604 053737 004652 001124 BIS SETBIT,$GDDAT ;SET ALWAYS SET BITS
3959 012612 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST
3960 012620 001404 004062 001122 BEQ 45 ;BRANCH IF GOOD
3961 012622 013737 004062 001122 MOV RHWG,$BDADR ;GET REGISTER ADDRESS
3962 012630 104137 004062 001122 ERROR 137 ;ONE WAS BEING FLOATED
3963 ;THRU RHWG REGISTER
3964 ;ON READING RHWG BACK
3965 ;IT DID NOT CONTAIN WHAT
3966 ;WAS EXPECTED.
3967 012632 000241 000241 45: CLC ;CLEAR CARRY
3968 012634 006305 006305 ASL R5 ;GET 1 ONE LEFT
3969 012636 103340 103340 BCC 15 ;BRANCH IF 16 NOT DONE
3970
3971 ;FLOAT A ZERO THRU RHWG REGISTER.
3972
3973 012640 012737 012664 001110 MOV #65,SLPERR ;SET LOOP BACK POINT.
3974 012646 012705 177776 001110 MOV #177776,R5 ;GET READY TO FLOAT A ZERO
3975 012652 010537 001124 001110 55: MOV R5,$GDDAT ;GET READY TO WRITE DATA
3976 012656 042737 000000 001124 BIC #0,$GDDAT ;CLEAR BITS NOT WRITTEN INTO
3977 012664 032737 041400 001140 65: BIT #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
3978 012672 001403 000040 171166 BEQ 75 ;BRANCH IF ANY OF THE ABOVE SET
3979 012674 012777 000040 171166 MOV #CLR,#RHCS2 ;CLEAR FOR SCOPE AID TO
3980 ;SYNC IF IN DEBUG
3981 012702 013777 001124 171152 75: MOV $GDDAT,#RHWC ;WRITE INTO RH'XA.
3982 012710 017737 171146 001126 MOV #RHWC,$BDDAT ;READ RHWG
3983 012716 053737 004652 001124 BIS SETBIT,$GDDAT ;SET BITS ALWAYS SET
3984 012724 023737 001124 001126 CMP $GDDAT,$BDDAT ;TEST
3985 012732 001404 004062 001122 BEQ 85 ;BRANCH IF GOOD
3986 012734 013737 004062 001122 MOV RHWG,$BDADR ;GET REGISTER ADDRESS
3987 012742 104137 004062 001122 ERROR 137 ;ZERO WAS BEING FLOATED
3988 ;THRU RHWG REGISTER

```

:ON READING IT BACK IT
:DID NOT CONTAIN WHAT
:WAS EXPECTED

3989
3990
3991
3992
3993 012744 000261
3994 012746 006105
3995 012750 103740
3996
3997
3998
3999
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
4010
4011
4012
4013
4014 012752 000004
4015 012754 012706 001000
4016 012760 012737 000007 004656
4017
4018 012766 004737 040306
4019
4020
4021
4022 012772 012737 000000 004650
4023 013000 012737 000000 004652
4024 013006 012737 000001 004654
4025
4026
4027 013014 012737 013040 001110
4028 013022 012705 000001
4029 013026 010537 001124
4030 013032 042737 000000 001124
4031 013040 032737 041400 001140
4032 013046 001403
4033 013050 052777 000040 171012
4034
4035 013056 013777 001124 171000
4036 013064 017737 170774 001126
4037 013072 043737 004654 001124
4038 013100 053737 004652 001124
4039 013106 023737 001124 001126
4040 013114 001404
4041 013116 013737 004064 001122
4042 013124 104137
4043
4044

8\$: SEC
ROL P5
BCS 53

:TEST 7 BIT BANG RHBA

:* TEST LOADING AND READING OF ALL POSSIBLE BITS
:* IN RHBA REGISTER.
:* USE A PATTERN OF WALKING 1'S (1,2,4,10 ETC)
:* AND WALKING 0'S (-2,-3,-5 ETC)
:* IN THIS TEST 0 BITS WILL NOT BE WRITTEN INTO
:* AND 0 BITS WILL ALWAYS BE SET
:* AND BIT00 BITS WILL ALWAYS BE CLEARED
:* IF SWITCH 14 OR 9 OR 8 ARE SET FOR DEBUGGING
:* THEN A RH CLEAR (RHCS2 BIT #5) WILL BE
:* GIVEN TO AID SCOPE SYNC'S ON THE CLEAR
:* SIGNAL.

:ST7:

SCOPE ;RESET STACK
MOV #STACK, SP ;SAVE TEST NUMBER
MOV #7, #STSTNM ;GIVE RH INITIALIZE
JSR PC, #CLDISK ;SETUP UNIT NUBER
;CLEAR RHWC AND FUNCTION BITS IN RHCS1
MOV #0, WRTBIT ;0 ARE BITS NOT WRITTEN INTO
MOV #0, SETBIT ;0 ARE BITS ALWAYS SET
MOV #BIT00, CLRBIT ;BIT00 ARE BITS ALWAYS CLEARED
;FLOAT 1'S THRU THE RHBA REGISTER
MOV #2\$, \$LPERR ;SET LOOP ON ERROR ADDRESS
MOV #1, R5 ;GETTING READY TO FLOAT A ONE
1\$: MOV R5, \$GDDAT ;START WITH DATA
BIC #0, \$GDDAT ;CLEAR BITS NOT WRITTEN INTO
2\$: BIT #BIT14!BIT9!BIT8, SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
BEQ 3\$;BRANCH IF ANY ONE SET
BIS #CLR, #RHCS2 ;GIVE CLEAR FOR SCOPE
;SYNC IF IN DEBUG:
3\$: MOV \$GDDAT, #RHBA ;WRITE RHBA REGISTER
MOV #RHBA, \$BDDAT ;READ RHBA REGISTER
BIC CLRBIT, \$GDDAT ;CLEAR ALWAYS CLEARED BITS
BIS SETBIT, \$GDDAT ;SET ALWAYS SET BITS
CMP \$GDDAT, \$BDDAT ;TEST
BEQ 4\$;BRANCH IF GOOD
MOV RHBA, \$BDADR ;GET REGISTER ADDRESS
ERROR 137 ;ONE WAS BEING FLOATED
;THRU RHBA REGISTER
;ON READING RHBA BACK


```

4101                                     ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
4102
4103 013274 012737 177700 004650      MOV    #177700,WRTBIT ;177700 ARE BITS NOT WRITTEN INTO
4104 013302 012737 000000 004652      MOV    #0,SETBIT     ;0 ARE BITS ALWAYS SET
4105 013310 012737 177700 004654      MOV    #177700,CLRBIT ;177700 ARE BITS ALWAYS CLEARED
4106
4107                                     ;FLOAT 1'S THRU THE RHBAE REGISTER
4108 013316 012737 013342 001110      MOV    #25,SLPERR    ;SET LOOP ON ERROR ADDRESS
4109 013324 012705 000001                MOV    #1,R5         ;GETTING READY TO FLOAT A ONE
4110 013330 010537 001124                MOV    R5,$GDDAT     ;START WITH DATA
4111 013334 042737 177700 001124      1$:   BIC    #177700,$GDDAT ;CLEAR BITS NOT WRITTEN INTO
4112 013342 032737 041400 001140      2$:   BIT    #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
4113 013350 001403                BEQ    3$           ;BRANCH IF ANY ONE SET
4114 013352 052777 000040 170510      BIS    #CLR,DRHCS2  ;GIVE CLEAR FOR SCOPE
4115                                     ;SYNC IF IN DEBUG:
4116 013360 013777 001124 170542      3$:   MOV    $GDDAT,DRHBAE ;WRITE RHBAE REGISTER
4117 013366 017737 170536 001126      MOV    DRHBAE,$BDDAT ;READ RHBAE REGISTER
4118 013374 043737 004654 001124      BIC    CLRBIT,$GDDAT ;CLEAR ALWAYS CLEARED BITS
4119 013402 053737 004652 001124      BIS    SETBIT,$GDDAT ;SET ALWAYS SET BITS
4120 013410 023737 001124 001126      CMP    $GDDAT,$BDDAT ;TEST
4121 013416 001404                BEQ    4$           ;BRANCH IF GOOD
4122 013420 013737 004130 001122      MOV    RHBAE,$BDADR  ;GET REGISTER ADDRESS
4123 013426 104137                ERROR  137         ;ONE WAS BEING FLOATED
4124                                     ;THRU RHBAE REGISTER
4125                                     ;ON READING RHBAE BACK
4126                                     ;IT DID NOT CONTAIN WHAT
4127                                     ;WAS EXPECTED.
4128 013430 000241                4$:   CLC                    ;CLEAR CARRY
4129 013432 006305                ASL    R5           ;GET 1 ONE LEFT
4130 013434 103335                BCC    1$          ;BRANCH IF 16 NOT DONE
4131
4132                                     ;FLOAT A ZERO THRU RHBAE REGISTER.
4133
4134 013436 012737 013462 001110      MOV    #65,SLPERR    ;SET LOOP BACK POINT.
4135 013444 012705 177776                MOV    #177776,R5   ;GET READY TO FLOAT A ZERO
4136 013450 010537 001124                MOV    R5,$GDDAT     ;GET READY TO WRITE DATA
4137 013454 042737 177700 001124      5$:   BIC    #177700,$GDDAT ;CLEAR BITS NOT WRITTEN INTO
4138 013462 032737 041400 001140      6$:   BIT    #BIT14!BIT9!BIT8,SWR ;ARE SWITCHES 14 OR 9 OR 8 SET
4139 013470 001403                BEQ    7$           ;BRANCH IF ANY OF THE ABOVE SET
4140 013472 012777 000040 170370      MOV    #CLR,DRHCS2  ;CLEAR FOR SCOPE AID TO
4141                                     ;SYNC IF IN DEBUG
4142 013500 013777 001124 170422      7$:   MOV    $GDDAT,DRHBAE ;WRITE INTO RH'XA.
4143 013506 017737 170416 001126      MOV    DRHBAE,$BDDAT ;READ RHBAE
4144 013514 053737 004652 001124      BIS    SETBIT,$GDDAT ;SET BITS ALWAYS SET
4145 013522 043737 004654 001124      BIC    CLRBIT,$GDDAT ;CLEAR BITS ALWAYS 0
4146 013530 023737 001124 001126      CMP    $GDDAT,$BDDAT ;TEST
4147 013536 001404                BEQ    8$           ;BRANCH IF GOOD
4148 013540 013737 004130 001122      MOV    RHBAE,$BDADR  ;GET REGISTER ADDRESS
4149 013546 104137                ERROR  137         ;ZERO WAS BEING FLOATED
4150                                     ;THRU RHBAE REGISTER
4151                                     ;ON READING IT BACK IT
4152                                     ;DID NOT CONTAIN WHAT
4153                                     ;WAS EXPECTED
4154
4155 013550 000261                8$:   SEC                    ;
4156 013552 006105                ROL    R5
    
```

4157 013554 103735
 4158
 4159
 4160
 4161
 4162
 4163
 4164
 4165
 4166
 4167
 4168
 4169
 4170
 4171
 4172
 4173
 4174
 4175
 4176
 4177
 4178
 4179
 4180 013556 000004
 4181 013560 012706 001000
 4182 013564 012737 000011 004656
 4183
 4184 013572 004737 040306
 4185
 4186
 4187
 4188
 4189 013576 012737 000100 001124
 4190 013604 053737 005010 001124
 4191
 4192 013612 017737 170252 001126
 4193 013620 023737 001124 001126
 4194
 4195
 4196 013626 001401
 4197 013630 104006
 4198
 4199
 4200
 4201
 4202
 4203
 4204
 4205
 4206 013632
 4207
 4208
 4209 013632 005077 170244
 4210
 4211
 4212 013636 104411

```

*****
*TEST 11      SILO TEST 1 (ONE WORD WRITE)
*      AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2,
*      RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH
*      TOGETHER WITH UNIT NUMBER
*      ONE WORD OF ALL ZEROS IS WRITTEN INTO RHDB
*      BY "CLR"
*      "OR" (BIT #7) IS WAITED FOR BY A TRAP INSTRUCTION
*      CALLED "WAT" (NO TIMING IS DONE)
*      HOWEVER IF "OR" DOES NOT SET WITHIN "WAT" COUNT
*      DOWN AN ERROR IS REPORTED
*      RHDB IS READ AND CHECKED TO CONTAIN ZEROS
*      RHCS2 WILL BE CHECKED TO HAVE "IR" AND UNIT NUMBER
*      RHCS1, RHCS3, RHBA, RHBAE, RHWC, WILL BE CHECKED TO HAVE
*      APPROPRIATE VALUES
*****
↑ST11:  SCOPE
        MOV      #STACK.SP      ;RESET STACK
        MOV      #11, @↑STNM    ;SAVE TEST NUMBER
        JSR     PC, @↑CLDISK    ;GIVE RH INITIALIZE
                                   ;SETUP UNIT NUBER
                                   ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
        ;CHECK THAT RHCS2 HAS IR
        MOV      #IR, $GDDAT    ;GET GOOD = 100
        BIS      UNIT, $GDDAT    ;INCLUDE UNIT NUMBER
        MOV      @RHCS2, $BDDAT ;READ RHCS2 FOR COMPARISON
        CMP     $GDDAT, $BDDAT  ;COMPARE EXPECTED
                                   ;DATA WITH DATA READ FROM
                                   ;RHCS2
        BEQ     65$             ;BRANCH IF GOOD
        ERROR   6
        ;AFTER SETTING CLR BIT #5
        ;IN RHCS2 TO INIT THE RH
        ;AND HAVING DONE NOTHING ELSE
        ;RHCS2 SHOULD HAVE IR
        ;=100
        ;TOGETHER WITH UNIT NUMBER
        ;BUT CONTAINED WHAT IS
        ;GIVEN IN BAD RHCS2
65$:
        ;WRITE ONE WORD OF ALL ZEROS INTO RHDB
        CLR     @RHDB           ;WRITE 0 IN RHDB
        ;WAIT FOR OR BIT IN RHCS2 REGISTER TO SET
        WAT                    ;TRAP TO WAIT.T SUBROUTINE
    
```

```

43 013640 004070 RHCS2 ;AND WAIT FOR OR BIT IN
44 013642 000200 OR ;RHCS2 REGISTER
45 ;IF ERROR OCCURS HERE
46 ;IT MEANS "OR" DID NOT
47 ;SET FOR THE FULL COUNT
48 ;DOWN OF THE WAIT.T SUBROUTINE
49 ;THIS TIME IS APPROXIMATELY
50 ;LARGER THAN 500 MILLISECONDS
51
52 ;CHECK THAT RHDB HAS 0
53 MOV #0,$GDDAT ;GET GOOD =
54
55 013644 012737 000000 001124
56 013652 017737 170224 001126 MOV JRHDB,$BDDAT ;READ RHDB FOR COMPARISON
57 013660 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
58 ;DATA WITH DATA READ FROM
59 ;RHDB
60 BEQ 67$ ;BRANCH IF GOOD
61 ERROR 7
62 ;AFTER CLEARING THE RH
63 ;AND WRITING ALL ZEROS
64 ;INTO RHDB
65 ;THEN READING IT BACK
66 ;RHDB SHOULD HAVE 0
67 =
68 ;BUT CONTAINED WHAT IS
69 ;GIVEN IN BAD RHDB
70
71 013672 67$:
72 ;CHECK THAT RHCS2 HAS IR
73 MOV #IR,$GDDAT ;GET GOOD = 100
74 013672 012737 000100 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
75 013700 053737 005010 001124
76 013706 017737 170156 001126 MOV JRHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
77 013714 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
78 ;DATA WITH DATA READ FROM
79 ;RHCS2
80 BEQ 69$ ;BRANCH IF GOOD
81 ERROR 10
82 ;AFTER CLEARING THE RH
83 ;AND WRITING ALL ZEROS
84 ;INTO RHDB
85 ;THEN READING IT BACK
86 ;RHCS2 SHOULD HAVE IR
87 =100
88 ;TOGETHER WITH UNIT NUMBER
89 ;BUT CONTAINED WHAT IS
90 ;GIVEN IN BAD RHCS2
91
92 013726 69$:
93 ;CHECK THAT RHCS1 HAS DVA!RDY
94 MOV #DVA!RDY,$GDDAT ;GET GOOD = 4200
95 013726 012737 004200 001124
96 013734 017737 170120 001126 MOV JRHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
97 013742 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
98 ;DATA WITH DATA READ FROM
99 ;RHCS1
100 BEQ 71$ ;BRANCH IF GOOD
101 ERROR

```



```

4325                                     ; INTO RHDB
4326                                     ; THEN READING IT BACK
4327                                     ; RHBAE SHOULD HAVE 0
4328                                     ; =
4329                                     ; BUT CONTAINED WHAT IS
4330                                     ; GIVEN IN BAD RHBAE
4331 014056                                77$:
4332                                     ; CHECK THAT RHWC HAS 0
4333 014056 012737 000000 001124          MOV      #0,SGDDAT      ; GET GOOD =
4334
4335 014064 017737 167772 001126          MOV      @RHWC,$BDDAT  ; READ RHWC FOR COMPARISON
4336 014072 023737 001124 001126          CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
4337                                     ; DATA WITH DATA READ FROM
4338                                     ; RHWC
4339 014100 001401                          BEQ      79$           ; BRANCH IF GOOD
4340 014102 104015                          ERROR    15
4341                                     ; AFTER CLEARING THE RH
4342                                     ; AND WRITING ALL ZEROS
4343                                     ; INTO RHDB
4344                                     ; THEN READING IT BACK
4345                                     ; RHWC SHOULD HAVE 0
4346                                     ; =
4347                                     ; BUT CONTAINED WHAT IS
4348                                     ; GIVEN IN BAD RHWC
4349 014104                                79$:
4350
4351
4352
4353
4354
4355 *****
4356 *TEST 12      SILO TEST 2 (ONE WORD WRITE)
4357 *      AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
4358 *      RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH
4359 *      TOGETHER WITH UNIT NUMBER
4360 *      ONE WORD OF ALL ONES IS WRITTEN INTO RHDB
4361 *      BY "CLR"
4362 *      "OR" (BIT #7) IS WAITED FOR BY A TRAP INSTRUCTION
4363 *      CALLED "WAT" (NO TIMING IS DONE)
4364 *      HOWEVER IF "OR" DOES NOT SET WITHIN "WAT" COUNT
4365 *      DOWN AN ERROR IS REPORTED
4366 *      RHDB IS READ AND CHECKED TO CONTAIN ALL ONES
4367 *      RHCS2 WILL BE CHECKED TO HAVE "IR" AND UNIT NUMBER
4368 *      RHCS1,RHCS3,RHBA,RHBAE,RHWC, WILL BE CHECKED TO HAVE
4369 *      APPROPRIATE VALUES
4370 * *****
4371 *ST12: SCOPE
4372 MOV      #STACK,SP      ; RESET STACK
4373 MOV      #12,@#1STNM    ; SAVE TEST NUMBER
4374 JSR      PC,@#CLDISK    ; GIVE RH INITIALIZE
4375                                     ; SETUP UNIT NUMBER
4376                                     ; CLEAR RHWC AND FUNCTION BITS IN RHCS1
4377
4378                                     ; CHECK THAT RHCS2 HAS IR
4379 014124 012737 000100 001124          MOV      #IR,$GDDAT   ; GET GOOD = 100
4380 014132 053737 005010 001124          BIS      UNIT,$GDDAT  ; INCLUDE UNIT NUMBER

```



```

4493 014354 001401      BEQ      75$      ; BRANCH IF GOOD
4494 014356 104014      ERROR    14
4495                                     ; AFTER CLEARING THE RH
4496                                     ; AND WRITING ALL ONES
4497                                     ; INTO RHDB
4498                                     ; THEN READING IT BACK
4499                                     ; RHBAE SHOULD HAVE 0
4500                                     ; =
4501                                     ; BUT CONTAINED WHAT IS
4502                                     ; GIVEN IN BAD RHBAE
4503 014360      75$:
4504                                     ; CHECK THAT RHWC HAS 0
4505 014360 012737 000000 001124  MOV      #0,$GDDAT ; GET GOOD =
4506                                     ;
4507 014366 017737 167470 001126  MOV      @RHWC,$BDDAT ; READ RHWC FOR COMPARISON
4508 014374 023737 001124 001126  CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
4509                                     ; DATA WITH DATA READ FROM
4510                                     ; RHWC
4511 014402 001401      BEQ      77$      ; BRANCH IF GOOD
4512 014404 104015      ERROR    15
4513                                     ; AFTER CLEARING THE RH
4514                                     ; AND WRITING ALL ONES
4515                                     ; INTO RHDB
4516                                     ; THEN READING IT BACK
4517                                     ; RHWC SHOULD HAVE 0
4518                                     ; =
4519                                     ; BUT CONTAINED WHAT IS
4520                                     ; GIVEN IN BAD RHWC
4521 014406      77$:
4522                                     ;
4523                                     ; *****
4524                                     ; *TEST 13      SILO TEST 3 (TWO WORD WRITE)
4525                                     ;
4526                                     ; *
4527                                     ; * AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
4528                                     ; * RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
4529                                     ; * TOGETHER WITH UNIT NUMBER
4530                                     ; * ONE WORD = 52525 IS WRITTEN INTO RHDB
4531                                     ; * RHCS2 IS CHECKED TO HAVE "IR" AND UNIT NUMBER
4532                                     ; * A SECOND WORD = 12525 IS WRITTEN INTO RHDB
4533                                     ; * "OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
4534                                     ; * INSTRUCTION CALLED "WAT"
4535                                     ; * RHDB IS READ AND CHECKED TO CONTAIN 52525
4536                                     ; * RHCS2 IS CHECKED TO HAVE "IR", "OR" AND UNIT NUMBER
4537                                     ; * RHDB IS READ A SECOND TIME AND CHECKED TO
4538                                     ; * CONTAIN 125252
4539                                     ; * RHCS2 IS CHECKED TO HAVE "IR" AND UNIT NUMBER
4540                                     ; * THEN ALL REGISTERS RHCS1, RHCS3, RHBA, RHBAE, RHWC
4541                                     ; * ARE CHECKED TO HAVE APPROPRIATE VALUE
4542                                     ; *
4543                                     ; *
4544                                     ; * *****
4545 014406 000004      ST13: SCOPE
4546 014410 012706 001000  MOV      #STACK,SP      ; RESET STACK
4547 014414 012737 000013 004656  MOV      #13,@STNM      ; SAVE TEST NUMBER
4548

```

```

4549 014422 004737 040306 JSR PC, @CLDISK ;GIVE RH INITIALIZE
4550 ;SETUP UNIT NUMBER
4551 ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
4552
4553 ;CHECK THAT RHCS2 HAS IR
4554 014426 012737 000100 001124 MOV #IR, $GDDAT ;GET GOOD = 100
4555 014434 053737 005010 001124 BIS UNIT, $GDDAT ;INCLUDE UNIT NUMBER
4556
4557 014442 017737 167422 001126 MOV @RHCS2, $BDDAT ;READ RHCS2 FOR COMPARISON
4558 014450 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
4559 ;DATA WITH DATA READ FROM
4560 ;RHCS2
4561 014456 001401 BEQ ERROR 65$ ;BRANCH IF GOOD
4562 014460 104006
4563
4564 ;AFTER SETTING "CLR" BIT #5
4565 ;IN RHCS2 TO INIT THE RH
4566 ;AND HAVING DONE NOTHING ELSE
4567
4568 ;RHCS2 SHOULD HAVE IR
4569 ;=100
4570 ;TOGETHER WITH UNIT NUMBER
4571 ;BUT CONTAINED WHAT IS
4572 014462 65$: ;GIVEN IN BAD RHCS2
4573
4574 ;WRITE ONE WORD = 52525 INTO RHDB
4575 014462 012777 052525 167412 MOV #52525, @RHDB ;WRITE IN RHDB
4576
4577 ;WAIT FOR OR BIT IN RHCS2 REGISTER TO SET
4578 014470 104411 WAT ;TRAP TO WAIT.T SUBROUTINE
4579 014472 004070 RHCS2 ;AND WAIT FOR OR BIT IN
4580 014474 000200 OR ;RHCS2 REGISTER
4581 ;IF ERROR OCCURS HERE
4582 ;IT MEANS "OR" DID NOT
4583 ;SET FOR THE FULL COUNT
4584 ;DOWN OF THE WAIT.T SUBROUTINE
4585 ;THIS TIME IS APPROXIMATELY
4586 ;LARGER THAN 500 MILLISECONDS
4587
4588
4589 ;CHECK THAT RHCS2 HAS IR!OR
4590 014476 012737 000300 001124 MOV #IR!OR, $GDDAT ;GET GOOD = 300
4591 014504 053737 005010 001124 BIS UNIT, $GDDAT ;INCLUDE UNIT NUMBER
4592
4593 014512 017737 167352 001126 MOV @RHCS2, $BDDAT ;READ RHCS2 FOR COMPARISON
4594 014520 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
4595 ;DATA WITH DATA READ FROM
4596 ;RHCS2
4597 014526 001401 BEQ ERROR 67$ ;BRANCH IF GOOD
4598 014530 104016
4599
4600 ;AFTER SETTING "CLR" BIT #5
4601 ;IN RHCS2 TO INIT THE RH
4602 ;THEN WRITING 52525 INTO
4603 ;RHDB
4604 ;RHCS2 SHOULD HAVE IR!OR
;=300
    
```

```

4605 ; TOGETHER WITH UNIT NUMBER
4606 ; BUT CONTAINED WHAT IS
4607 ; GIVEN IN BAD RHCS2
4608 014532 67$:
4609
4610 ; WRITE A SECOND WORD = 125252 INTO RHDB
4611 014532 012777 125252 167342 MOV #125252, @RHDB ; WRITE 125252 INTO RHDB
4612
4613 ; WAIT FOR OR BIT IN RHCS2 REGISTER TO SET
4614 014540 104411 WAT ; TRAP TO WAIT.T SUBROUTINE
4615 014542 004070 RHCS2 ; AND WAIT FOR OR BIT IN
4616 014544 000200 OR ; RHCS2 REGISTER
4617 ; IF ERROR OCCURS HERE
4618 ; IT MEANS "OR" DID NOT
4619 ; SET FOR THE FULL COUNT
4620 ; DOWN OF THE WAIT.T SUBROUTINE
4621 ; THIS TIME IS APPROXIMATELY
4622 ; LARGER THAN 500 MILLISECONDS
4623
4624 ; CHECK THAT RHDB HAS 52525
4625 014546 012737 052525 001124 MOV #52525, $GDDAT ; GET GOOD = 0
4626
4627 014554 017737 167322 001126 MOV @RHDB, $BDDAT ; READ RHDB FOR COMPARISON
4628 014562 023737 001124 001126 CMP $GDDAT, $BDDAT ; COMPARE EXPECTED
4629 ; DATA WITH DATA READ FROM
4630 ; RHDB
4631 014570 001401 BEQ 69$ ; BRANCH IF GOOD
4632 014572 104017 ERROR 17
4633
4634 ; AFTERCLEARING THE RH THEN
4635 ; WRITING TWO WORDS INTO
4636 ; RHDB 52525 ND 125252
4637 ; THEN READING RHDB ONCE
4638 ; RHDB SHOULD HAVE 52525
4639 ; BUT CONTAINED WHAT IS
4640 ; GIVEN IN BAD RHDB
4640 014574 69$:
4641
4642 014574 012737 000300 001124 ; CHECK THAT RHCS2 HAS IR!OR
4643 014602 053737 005010 001124 MOV #IR!OR, $GDDAT ; GET GOOD = 300
4644 BIS UNIT, $GDDAT ; INCLUDE UNIT NUMBER
4645 014610 017737 167254 001126 MOV @RHCS2, $BDDAT ; READ RHCS2 FOR COMPARISON
4646 014616 023737 001124 001126 CMP $GDDAT, $BDDAT ; COMPARE EXPECTED
4647 ; DATA WITH DATA READ FROM
4648 ; RHCS2
4649 014624 001401 BEQ 71$ ; BRANCH IF GOOD
4650 014626 104020 ERROR 20
4651
4652 ; AFTERCLEARING THE RH THEN
4653 ; WRITING TWO WORDS INTO
4654 ; RHDB 52525 ND 125252
4655 ; THEN READING RHDB ONCE
4656 ; RHCS2 SHOULD HAVE IR!OR
4657 ; =300
4658 ; TOGETHER WITH UNIT NUMBER
4659 ; BUT CONTAINED WHAT IS
4660 014630 71$:
4660 ; GIVEN IN BAD RHCS2
    
```

```

4661
4662
4663 ;READ RHDB A SECOND TIME
4664 ;CHECK THAT RHDB HAS 125252
4665 014630 012737 125252 001124 MOV #125252,$GDDAT ;GET GOOD = 0
4666
4667 014636 017737 167240 001126 MOV @RHDB,$BDDAT ;READ RHDB FOR COMPARISON
4668 014644 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
4669 ;DATA WITH DATA READ FROM
4670 ;RHDB
4671 014652 001401 BEQ 73$ ;BRANCH IF GOOD
4672 014654 104021 ERROR 21
4673 ;AFTER CLEARING RH THEN
4674 ;WRITING TWO WORDS INTO
4675 ;RHDB 52525 AND 125252
4676 ;THEN READING RHDB THE
4677 ;SECOND TIME
4678 ;RHDB SHOULD HAVE 125252
4679 ;BUT CONTAINED WHAT IS
4680 ;GIVEN IN BAD RHDB
4681 014656 73$:
4682 ;CHECK THAT RHCS2 HAS IR
4683 014656 012737 000100 001124 MOV #IR,$GDDAT ;GET GOOD = 100
4684 014664 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
4685
4686 014672 017737 167172 001126 MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
4687 014700 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
4688 ;DATA WITH DATA READ FROM
4689 ;RHCS2
4690 014706 001401 BEQ 75$ ;BRANCH IF GOOD
4691 014710 104022 ERROR 22
4692 ;AFTER CLEARING RH THEN
4693 ;WRITING TWO WORDS INTO
4694 ;RHDB 52525 AND 125252
4695 ;THEN READING RHDB THE
4696 ;SECOND TIME
4697 ;RHCS2 SHOULD HAVE IR
4698 ;=100
4699 ;TOGETHER WITH UNIT NUMBER
4700 ;BUT CONTAINED WHAT IS
4701 ;GIVEN IN BAD RHCS2
4702 014712 75$:
4703 ;CHECK THAT RHCS1 HAS DVA!RDY
4704 014712 012737 004200 001124 MOV #DVA!RDY,$GDDAT ;GET GOOD = 4200
4705
4706 014720 017737 167134 001126 MOV @RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
4707 014726 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
4708 ;DATA WITH DATA READ FROM
4709 ;RHCS1
4710 014734 001401 BEQ 77$ ;BRANCH IF GOOD
4711 014736 104141 ERROR 141
4712 ;AFTER CLEARING RH THEN
4713 ;WRITING TWO WORDS INTO
4714 ;RHDB 52525 AND 125252
4715 ;THEN READING RHDB THE
4716 ;SECOND TIME
    
```

```

4717 ;RHCS1 SHOULD HAVE DVA!RDY
4718 ;=4200
4719 ;BUT CONTAINED WHAT IS
4720 ;GIVEN IN BAD RHCS1
4721 014740 77$:
4722 ;CHECK THAT RHCS3 HAS 0
4723 014740 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
4724
4725 014746 017737 167160 001126 MOV 2RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
4726 014754 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
4727 ;DATA WITH DATA READ FROM
4728 ;RF 33
4729 014762 001401 BEQ 79$ ;BRANCH IF GOOD
4730 014764 104142 ERROR 142
4731
4732 ;AFTER CLEARING RH THEN
4733 ;WRITING TWO WORDS INTO
4734 ;RHDB 52525 AND 125252
4735 ;THEN READING RHDB THE
4736 ;SECOND TIME
4737 ;RHCS3 SHOULD HAVE 0
4738 ;BUT CONTAINED WHAT IS
4739 ;GIVEN IN BAD RHCS3
4740 014766 79$:
4741 014766 012737 000000 001124 ;CHECK THAT RHBA HAS 0
4742 MOV #0,$GDDAT ;GET GOOD = 0
4743
4744 014774 017737 167064 001126 MOV 2RHBA,$BDDAT ;READ RHBA FOR COMPARISON
4745 015002 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
4746 ;DATA WITH DATA READ FROM
4747 ;RHBA
4748 015010 001401 BEQ 81$ ;BRANCH IF GOOD
4749 015012 104143 ERROR 143
4750
4751 ;AFTER CLEARING RH THEN
4752 ;WRITING TWO WORDS INTO
4753 ;RHDB 52525 AND 125252
4754 ;THEN READING RHDB THE
4755 ;SECOND TIME
4756 ;RHBA SHOULD HAVE 0
4757 ;BUT CONTAINED WHAT IS
4758 ;GIVEN IN BAD RHBA
4759 015014 81$:
4760 015014 012737 000000 001124 ;CHECK THAT RHBAE HAS 0
4761 MOV #0,$GDDAT ;GET GOOD = 0
4762
4763 015022 017737 167102 001126 MOV 2RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
4764 015030 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
4765 ;DATA WITH DATA READ FROM
4766 ;RHBAE
4767 015036 001401 BEQ 83$ ;BRANCH IF GOOD
4768 015040 104144 ERROR 144
4769
4770 ;AFTER CLEARING RH THEN
4771 ;WRITING TWO WORDS INTO
4772 ;RHDB 52525 AND 125252
4773 ;THEN READING RHDB THE
4774 ;SECOND TIME
4775 ;RHBAE SHOULD HAVE 0

```

```

4773                                     :BUT CONTAINED WHAT IS
4774                                     :GIVEN IN BAD RHBAE
4775 015042                               83$:
4776                                     :CHECK THAT RHWC HAS 0
4777 015042 012737 000000 001124         MOV    #0,$GDDAT    :GET GOOD = 0
4778                                     :READ RHWC FOR COMPARISON
4779 015050 017737 167006 001126         MOV    @RHWC,$BDDAT
4780 015056 023737 001124 001126         CMP    $GDDAT,$BDDAT :COMPARE EXPECTED
4781                                     :DATA WITH DATA READ FROM
4782                                     :RHWC
4783 015064 001401                         BEQ    85$          :BRANCH IF GOOD
4784 015066 104145                         ERROR  145
4785                                     :AFTER CLEARING RH THEN
4786                                     :WRITING TWO WORDS INTO
4787                                     :RHDB 52525 AND 125252
4788                                     :THEN READING RHDB THE
4789                                     :SECOND TIME
4790                                     :RHWC SHOULD HAVE 0
4791                                     :BUT CONTAINED WHAT IS
4792                                     :GIVEN IN BAD RHWC
4793 015070                               85$:
4794
4795 :*****
4796 :*TEST 14      SILO TEST 4 (COUNT PATTERN)
4797
4798 :*
4799 :*   AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
4800 :*   RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH, TOGETHER
4801 :*   WITH UNIT NUMBER
4802 :*   EIGHT WORDS, A COUNT PATTERN 0 THRU 7 IS WRITTEN
4803 :*   INTO RHDB
4804 :*   "OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
4805 :*   INSTRUCTION CALLED "WAT"
4806 :*   THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
4807 :*   "OR" HIGH TOGETHER WITH THE UNIT NUMBER
4808 :*   THEN RHDB IS READ AND COMPARED TO HAVE THE RIGHT VALUE
4809 :*   EIGHT TIMES
4810 :*   THEN RHCS2, RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
4811 :*   CHECKED TO HAVE THE APPROPRIATE VALUE
4812 :*****
4813 015070 000004                         †ST14: SCOPE
4814 015072 012706 001000                 MOV    #STACK,SP    :RESET STACK
4815 015076 012737 000014 004656         MOV    #14,@†STNM   :SAVE TEST NUMBER
4816
4817 015104 004737 040306                 JSR    PC,@#CLDISK  :GIVE RH INITIALIZE
4818                                     :SETUP UNIT NUMBER
4819                                     :CLEAR RHWC AND FUNCTION BITS IN RHCS:
4820
4821                                     :CHECK THAT RHCS2 HAS IR
4822 015110 012737 000100 001124         MOV    #IR,$GDDAT   :GET GOOD = 100
4823 015116 053737 005010 001124         BIS    UNIT,$GDDAT  :INCLUDE UNIT NUMBER
4824
4825 015124 017737 166740 001126         MOV    @RHCS2,$BDDAT :READ RHCS2 FOR COMPARISON
4826 015132 023737 001124 001126         CMP    $GDDAT,$BDDAT :COMPARE EXPECTED
4827                                     :DATA WITH DATA READ FROM
4828                                     :RHCS2

```


E08

```

4829 015140 001401      BEQ      65$      ;BRANCH IF GOOD
4830 015142 104006      ERROR    6        ;
4831                                     ;AFTER SETTING "CLR" BIT #5
4832                                     ;IN RHCS2 TO INIT THE RH
4833                                     ;AND HAVING DONE NOTHING ELSE
4834                                     ;RHCS2 SHOULD HAVE IR
4835                                     ;=100
4836                                     ;TOGETHER WITH UNIT NUMBER
4837                                     ;BUT CONTAINED WHAT IS
4838                                     ;GIVEN IN BAD RHCS2
4839 015144                65$:
4840
4841                                     ;WRITE EIGHT WORDS INTO RHDB TO FILL IT
4842                                     ;THIS IS A COUNT PATTERN 0 THRU 7
4843 015144 012702 000010  MOV      #8,R2      ;8 NUMBERS TO BE WRITTEN
4844 015150 005001      CLR      R1          ;FIRST WORD TO BE WRITTEN = 0
4845 015152 010177 166724 1$:  MOV      R1,RHDB    ;WRITE INTO RHDB - A COUNT
4846 015156 005201      INC      R1          ;GET NEXT WORD
4847 015160 005302      DEC      R2          ;COUNT TO 8
4848 015162 001373      BNE     1$          ;BRANCH IF 8 NOT DONE
4849
4850                                     ;WAIT FOR OR BIT IN RHCS2 REGISTER TO SET
4851 015164 104411      WAT      ;TRAP TO WAIT.T SUBROUTINE
4852 015166 004070      RHCS2   ;AND WAIT FOR OR BIT IN
4853 015170 000200      OR      ;RHCS2 REGISTER
4854                                     ;IF ERROR OCCURS HERE
4855                                     ;IT MEANS "OR" DID NOT
4856                                     ;SET FOR THE FULL COUNT
4857                                     ;DOWN OF THE WAIT.T SUBROUTINE
4858                                     ;THIS TIME IS APPROXIMATELY
4859                                     ;LARGER THAN 500 MILLISECONDS
4860
4861                                     ;CHECK THAT RHCS2 HAS OR
4862 015172 012737 000200 001124  MOV      #OR,$GDDAT ;GET GOOD = 200
4863 015200 053737 005010 0C1124  BIS      UNIT,$GDDAT ;INCLUDE UNIT NUMBER
4864
4865 015206 017737 166656 001126  MOV      @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
4866 015214 023737 001124 001126  CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
4867                                     ;DATA WITH DATA READ FROM
4868                                     ;RHCS2
4869 015222 001401      BEQ      67$      ;BRANCH IF GOOD
4870 015224 104023      ERROR    23        ;
4871                                     ;AFTER SETTING "CLR" BIT #5
4872                                     ;IN RHCS2 TO INIT THE RH
4873                                     ;AND HAVING WRITTEN A
4874                                     ;COUNT PATTERN 0 THRU 7
4875                                     ;INTO RHDB TO FILL IT
4876                                     ;RHCS2 SHOULD HAVE OR
4877                                     ;=200
4878                                     ;TOGETHER WITH UNIT NUMBER
4879                                     ;BUT CONTAINED WHAT IS
4880                                     ;GIVEN IN BAD RHCS2
4881 015226                67$:
4882
4883
4884

```

F08

CERHAD MACY11 30(1046) 21-DEC-77 13:06 PAGE 97
 CERHAD.P11 21-DEC-77 12:48 714

SILO TEST 4 (COUNT PATTERN)

SE2 CC96

4885	015226	012737	000001	004664	MOV #1,SILONM	;FIRST WORD TO BE READ
4886					;CHECK THAT RHDB HAS 0	
4887	015234	012737	000000	001124	MOV #0,\$GDDAT	;GET GOOD = 0
4888						
4889	015242	017737	166634	001126	MOV @RHDB,\$BDDAT	;READ RHDB FOR COMPARISON
4890	015250	023737	001124	001126	CMP \$GDDAT,\$BDDAT	;COMPARE EXPECTED
4891						;DATA WITH DATA READ FROM
4892						;RHDB
4893	015256	001401			BEO 69\$;BRANCH IF GOOD
4894	015260	104024			ERROR 24	
4895						;AFTER SETTING "CLR" BIT #5
4896						;IN RHCS2 TO INIT THE RH
4897						;AND WRITING A
4898						;COUNT PATTERN 0 THRU 7
4899						;INTO RHDB TO FILL IT
4900						;THEN ON READING RHDB THE
4901						;FIRST TIME
4902						;RHDB SHOULD HAVE 0
4903						;BUT CONTAINED WHAT IS
4904						;GIVEN IN BAD RHDB
4905	015262					69\$:
4906						
4907	015262	012737	000002	004664	MOV #2,SILONM	;SECOND WORD TO BE READ
4908					;CHECK THAT RHDB HAS 1	
4909	015270	012737	000001	001124	MOV #1,\$GDDAT	;GET GOOD = 0
4910						
4911	015276	017737	166600	001126	MOV @RHDB,\$BDDAT	;READ RHDB FOR COMPARISON
4912	015304	023737	001124	001126	CMP \$GDDAT,\$BDDAT	;COMPARE EXPECTED
4913						;DATA WITH DATA READ FROM
4914						;RHDB
4915	015312	001401			BEO 71\$;BRANCH IF GOOD
4916	015314	104024			ERROR 24	
4917						;AFTER SETTING "CLR" BIT #5
4918						;IN RHCS2 TO INIT THE RH
4919						;AND WRITING A
4920						;COUNT PATTERN 0 THRU 7
4921						;INTO RHDB TO FILL IT
4922						;THEN ON READING RHDB THE
4923						;SECOND TIME
4924						;RHDB SHOULD HAVE 1
4925						;BUT CONTAINED WHAT IS
4926						;GIVEN IN BAD RHDB
4927	015316					71\$:
4928						
4929	015316	012737	000003	004664	MOV #3,SILONM	;THIRD WORD TO BE READ
4930					;CHECK THAT RHDB HAS 2	
4931	015324	012737	000002	001124	MOV #2,\$GDDAT	;GET GOOD = 0
4932						
4933	015332	017737	166544	001126	MOV @RHDB,\$BDDAT	;READ RHDB FOR COMPARISON
4934	015340	023737	001124	001126	CMP \$GDDAT,\$BDDAT	;COMPARE EXPECTED
4935						;DATA WITH DATA READ FROM
4936						;RHDB
4937	015346	001401			BEO 73\$;BRANCH IF GOOD
4938	015350	104024			ERROR 24	
4939						;AFTER SETTING "CLR" BIT #5
4940						;IN RHCS2 TO INIT THE RH


```

S053                                     ; INTO RHDB TO FILL IT
S054                                     ; THEN ON READING RHDB THE
S055                                     ; EIGHTH TIME
S056                                     ; RHDB SHOULD HAVE 7
S057                                     ; BUT CONTAINED WHAT IS
S058                                     ; GIVEN IN BAD RHDB
S059 015566                               83$:
S060
S061
S062                                     ; CHECK THAT RHCS2 HAS IR
S063 015566 012737 000100 001124      MOV #IR,$GDDAT ; GET GOOD = 100
S064 015574 053737 005010 001124      BIS UNIT,$GDDAT ; INCLUDE UNIT NUMBER
S065
S066 015602 017737 166262 001126      MOV @RHCS2,$BDDAT ; READ RHCS2 FOR COMPARISON
S067 015610 023737 001124 001126      CMP $GDDAT,$BDDAT ; COMPARE EXPECTED
S068                                     ; DATA WITH DATA READ FROM
S069                                     ; RHCS2
S070 015616 001401                       BEQ 85$ ; BRANCH IF GOOD
S071 015620 104025                       ERROR 25
S072
S073                                     ; AFTER SETTING "CLR" BIT #5
S074                                     ; IN RHCS2 TO INIT THE RH
S075                                     ; AND WRITING A
S076                                     ; COUNT PATTERN 0 THRU 7
S077                                     ; INTO RHDB TO FILL IT
S078                                     ; THEN ON READING RHDB THE
S079                                     ; TOTAL 8 TIMES
S080                                     ; RHCS2 SHOULD HAVE IR
S081                                     ; =100
S082                                     ; TOGETHER WITH UNIT NUMBER
S083                                     ; BUT CONTAINED WHAT IS
S084 015622                               85$:
S085                                     ; CHECK THAT RHCS1 HAS DVA!RDY
S086 015622 012737 004200 001124      MOV #DVA!RDY,$GDDAT ; GET GOOD = 4200
S087
S088 015630 017737 166224 001126      MOV @RHCS1,$BDDAT ; READ RHCS1 FOR COMPARISON
S089 015636 023737 001124 001126      CMP $GDDAT,$BDDAT ; COMPARE EXPECTED
S090                                     ; DATA WITH DATA READ FROM
S091                                     ; RHCS1
S092 015644 001401                       BEQ 87$ ; BRANCH IF GOOD
S093 015646 104026                       ERROR 26
S094
S095                                     ; AFTER SETTING "CLR" BIT #5
S096                                     ; IN RHCS2 TO INIT THE RH
S097                                     ; AND WRITING A
S098                                     ; COUNT PATTERN 0 THRU 7
S099                                     ; INTO RHDB TO FILL IT
S100                                     ; THEN ON READING RHDB THE
S101                                     ; TOTAL 8 TIMES
S102                                     ; RHCS1 SHOULD HAVE DVA!RDY
S103                                     ; =4200
S104                                     ; BUT CONTAINED WHAT IS
S105 015650                               87$:
S106                                     ; CHECK THAT RHCS3 HAS 0
S107 015650 012737 000000 001124      MOV #0,$GDDAT ; GET GOOD = 0
S108

```



```

S165 015752          93$:
S166          :CHECK THAT RHWC HAS 0
S167 015752 012737 000000 001124  MOV      #0,$GDDAT      ;GET GOOD = 0
S168
S169 015760 017737 166076 001126  MOV      @RHWC,$BDDAT    ;READ RHWC FOR COMPARISON
S170 015766 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE EXPECTED
S171          :DATA WITH DATA READ FROM
S172          :RHWC
S173 015774 001401          BEQ      95$            ;BRANCH IF GOOD
S174 015776 104032          ERROR    32
S175
S176          :AFTER SETTING "CLR" BIT #5
S177          :IN RHCS2 TO INIT THE RH
S178          :AND WRITING A
S179          :COUNT PATTERN 0 THRU 7
S180          :INTO RHDB TO FILL IT
S181          :THEN ON READING RHDB THE
S182          :TOTAL 8 TIMES
S183          :RHWC SHOULD HAVE 0
S184          :BUT CONTAINED WHAT IS
S185          :GIVEN IN BAD RHWC
S185 016000          95$:
S186
S187
S188          :*****
S189          :*TEST 15          SILO TEST 5 (FLOATING ONES)
S190
S191          :*
S192          :* AFTER ARH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
S193          :* RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
S194          :* TOGETHER WITH UNIT NUMBER
S195          :* EIGHT WORDS, A PATTERN OF FLOATING ONES (1,2,4,10,20,40,100,200)
S196          :* IS WRITTEN INTO RHDB
S197          :* "OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
S198          :* INSTRUCTION CALLED "WAT"
S199          :* THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
S200          :* "OR" HIGH TOGETHER WITH THE UNIT NUMBER
S201          :* THEN RHDB IS READ AND COMPARED TO HAVE THE
S202          :* RIGHT VALUE AFTER EACH OF THE EIGHT READS.
S203          :* THEN RHCS2 IS CHECKED TO HAVE "IR"
S204          :* AND UNIT NUMBER
S205          :* THEN RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
S206          :* CHECKED TO HAVE THE APPROPRIATE VALUE
S206          :*****
S207 016000 000004          †ST15: SCOPE
S208 016002 012706 001000  MOV      #STACK,SP      ;RESET STACK
S209 016006 012737 000015 004656  MOV      #15,@†STNM    ;SAVE TEST NUMBER
S210
S211 016014 004737 040306  JSR      PC,@#CLDISK   ;GIVE RH INITIALIZE
S212          :SETUP UNIT NUBER
S213          :CLEAR RHWC AND FUNCTION BITS IN RHCS1
S214
S215
S216          :CHECK THAT RHCS2 HAS IR
S217 016020 012737 000100 001124  MOV      #IR,$GDDAT     ;GET GOOD = 100
S218 016026 053737 005010 001124  BIS      UNIT,$GDDAT    ;INCLUDE UNIT NUMBER
S219
S220 016034 017737 166030 001126  MOV      @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON

```


M08

CERHADO MACY11 30(1046) 21-DEC-77 13:06 PAGE 104
CERHAD.P11 21-DEC-77 12:48 T15

SILO TEST 5 (FLOATING ONES)

SEQ 0103

5277
5278 016140
5279
5280

675:

;GIVEN IN BAD RHCS2

```

5281
5282 016140 012737 000001 004664      MOV      #1,SILONM      ;FIRST WORD TO BE READ
5283      ;CHECK THAT RHDB HAS 1
5284 016146 012737 000001 001124      MOV      #1,$GDDAT      ;GET GOOD = 0
5285
5286 016154 017737 165722 001126      MOV      @RHDB,$BDDAT    ;READ RHDB FOR COMPARISON
5287 016162 023737 001124 001126      CMP      $GDDAT,$BDDAT   ;COMPARE EXPECTED
5288      ;DATA WITH DATA READ FROM
5289      ;RHDB
5290 016170 001401      BEQ      69$             ;BRANCH IF GOOD
5291 016172 104024      ERROR    24
5292
5293      ;AFTER SETTING "CLR" BIT #5
5294      ;IN RHCS2 TO INIT THE RH
5295      ;AND WRITING A PATTERN OF
5296      ;FLOATING ONES (WHICH IS
5297      ;1,2,4,10,20,40,100,200)
5298      ;INTO RHDB TO FILL IT
5299      ;THEN ON READING RHDB THE
5300      ;FIRST TIME
5301      ;RHDB SHOULD HAVE 1
5302      ;BUT CONTAINED WHAT IS
5303      ;GIVEN IN BAD RHDB
5303 016174      69$:
5304
5305
5306 016174 012737 000002 004664      MOV      #2,SILONM      ;SECOND WORD TO BE READ
5307      ;CHECK THAT RHDB HAS 2
5308 016202 012737 000002 001124      MOV      #2,$GDDAT      ;GET GOOD = 0
5309
5310 016210 017737 165666 001126      MOV      @RHDB,$BDDAT    ;READ RHDB FOR COMPARISON
5311 016216 023737 001124 001126      CMP      $GDDAT,$BDDAT   ;COMPARE EXPECTED
5312      ;DATA WITH DATA READ FROM
5313      ;RHDB
5314 016224 001401      BEQ      71$             ;BRANCH IF GOOD
5315 016226 104024      ERROR    24
5316
5317      ;AFTER SETTING "CLR" BIT #5
5318      ;IN RHCS2 TO INIT THE RH
5319      ;AND WRITING A PATTERN OF
5320      ;FLOATING ONES (WHICH IS
5321      ;1,2,4,10,20,40,100,200)
5322      ;INTO RHDB TO FILL IT
5323      ;THEN ON READING RHDB THE
5324      ;SECOND TIME
5325      ;RHDB SHOULD HAVE 2
5326      ;BUT CONTAINED WHAT IS
5327      ;GIVEN IN BAD RHDB
5327 016230      71$:
5328
5329
5330 016230 012737 000003 004664      MOV      #3,SILONM      ;THIRD WORD TO BE READ
5331      ;CHECK THAT RHDB HAS 4
5332 016236 012737 000004 001124      MOV      #4,$GDDAT      ;GET GOOD = 0
5333
5334 016244 017737 165632 001126      MOV      @RHDB,$BDDAT    ;READ RHDB FOR COMPARISON
5335 016252 023737 001124 001126      CMP      $GDDAT,$BDDAT   ;COMPARE EXPECTED
5336      ;DATA WITH DATA READ FROM
    
```

```

5337
5338 016260 001401 BEQ 73$ ;RHDB
5339 016262 104024 ERROR 24 ;BRANCH IF GOOD
5340
5341 ;AFTER SETTING "CLR" BIT #5
5342 ;IN RHCS2 TO INIT THE RH
5343 ;AND WRITING A PATTERN OF
5344 ;FLOATING ONES (WHICH IS
5345 ;1 2 4 10 20 40 100 200)
5346 ;INTO RHDB TO FILL IT
5347 ;THEN ON READING RHDB THE
5348 ;THIRD TIME
5349 ;RHDB SHOULD HAVE 4
5350 ;BUT CONTAINED WHAT IS
5351 ;GIVEN IN BAD RHDB
5352
5353 016264 73$:
5354
5355 016264 012737 000004 004664 MOV #4,SILONM ;FOURTH WORD TO BE READ
5356 ;CHECK THAT RHDB HAS 10
5357 016272 012737 000010 001124 MOV #10,$GDDAT ;GET GOOD = 0
5358
5359 016300 017737 165576 001126 MOV @RHDB,$BDDAT ;READ RHDB FOR COMPARISON
5360 016305 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
5361 ;DATA WITH DATA READ FROM
5362 ;RHDB
5363 016314 001401 BEQ 75$ ;BRANCH IF GOOD
5364 016316 104024 ERROR 24
5365
5366 ;AFTER SETTING "CLR" BIT #5
5367 ;IN RHCS2 TO INIT THE RH
5368 ;AND WRITING A PATTERN OF
5369 ;FLOATING ONES (WHICH IS
5370 ;1 2 4 10 20 40 100 200)
5371 ;INTO RHDB TO FILL IT
5372 ;THEN ON READING RHDB THE
5373 ;FOURTH TIME
5374 ;RHDB SHOULD HAVE 10
5375 ;BUT CONTAINED WHAT IS
5376 ;GIVEN IN BAD RHDB
5377
5378 016320 75$:
5379
5380 016320 012737 000005 004664 MOV #5,SILONM ;FIFTH WORD TO BE READ
5381 ;CHECK THAT RHDB HAS 20
5382 016326 012737 000020 001124 MOV #20,$GDDAT ;GET GOOD = 0
5383
5384 016334 017737 165542 001126 MOV @RHDB,$BDDAT ;READ RHDB FOR COMPARISON
5385 016342 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
5386 ;DATA WITH DATA READ FROM
5387 ;RHDB
5388 016350 001401 BEQ 77$ ;BRANCH IF GOOD
5389 016352 104024 ERROR 24
5390
5391 ;AFTER SETTING "CLR" BIT #5
5392 ;IN RHCS2 TO INIT THE RH
5393 ;AND WRITING A PATTERN OF
5394 ;FLOATING ONES (WHICH IS
5395 ;1.2.4.10.20.40.100.200.

```

5393
5394
5395
5396
5397
5398
5399
4000
4001
4002
4003
4004
4005
4006
4007
4008
4009
410
411
412
413
414
415
416
417
418
419
420
421
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

016354 77\$:

016354 012737 000006 004664

MOV #6,SILONM
;CHECK THAT RHDB HAS 40
MOV #40,\$GDDAT

; INTO RHDB TO FILL IT
; THEN ON READING RHDB THE
; FIFTH TIME
; RHDB SHOULD HAVE 20
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHDB

; SIXTH WORD TO BE READ

016362 012737 000040 001124

MOV #40,\$GDDAT

; GET GOOD = 0

016370 017737 165506 001126

MOV JRHDB,\$BDDAT
CMP \$GDDAT,\$BDDAT

; READ RHDB FOR COMPARISON
; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHDB

016376 023737 001124 001126

BEQ 79\$
ERROR 24

; BRANCH IF GOOD

016404 001401
016406 104024

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; AND WRITING A PATTERN OF
; FLOATING ONES (WHICH IS
; 1 2 4 10 20 40 100 200)
; INTO RHDB TO FILL IT
; THEN ON READING RHDB THE
; SIXTH TIME
; RHDB SHOULD HAVE 40
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHDB

016410 79\$:

016410 012737 000007 004664

MOV #7,SILONM
;CHECK THAT RHDB HAS 100
MOV #100,\$GDDAT

; SEVENTH WORD TO BE READ

016416 012737 000100 001124

MOV #100,\$GDDAT

; GET GOOD = 0

016424 017737 165452 001126

MOV JRHDB,\$BDDAT
CMP \$GDDAT,\$BDDAT

; READ RHDB FOR COMPARISON
; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHDB

016432 023737 001124 001126

BEQ 81\$
ERROR 24

; BRANCH IF GOOD

016440 001401
016442 104024

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; AND WRITING A PATTERN OF
; FLOATING ONES (WHICH IS
; 1 2 4 10 20 40 100 200)
; INTO RHDB TO FILL IT
; THEN ON READING RHDB THE
; SEVENTH TIME
; RHDB SHOULD HAVE 100
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHDB

016444 81\$:

```

55001 016444 012737 000010 004664      MOV   #8,SILONM      ;EIGHTH WORD TO BE READ
55002 016452 012737 000200 001124      ;CHECK THAT RHDE HAS 200
55003 016452 012737 000200 001124      MOV   #200,$GDDAT    ;GET GOOD = 0
55004 016460 017737 165416 001126      MOV   @RHDB,$BDDAT   ;READ RHDB FOR COMPARISON
55005 016466 023737 001124 001126      CMP   $GDDAT,$BDDAT  ;COMPARE EXPECTED
55006 016474 001401 001124 001126      ;DATA WITH DATA READ FROM
55007 016475 104024 001124 001126      ;RHDB
55008 BEQ   83$                ;BRANCH IF GOOD
55009 016475 104024 001124 001126      ;AFTER SETTING "CLR" BIT #5
55010 016500 001124 001124 001126      ;IN RHCS2 TO INIT THE RH
55011 016500 001124 001124 001126      ;AND WRITING A PATTERN OF
55012 016500 001124 001124 001126      ;FLOATING ONES (WHICH IS
55013 016500 001124 001124 001126      ;1 2 4 10 20 40 100 200)
55014 016500 001124 001124 001126      ;INTO RHDB TO FILL IT
55015 016500 001124 001124 001126      ;THEN ON READING RHDB THE
55016 016500 001124 001124 001126      ;EIGHTH TIME
55017 016500 001124 001124 001126      ;RHDB SHOULD HAVE 200
55018 016500 001124 001124 001126      ;BUT CONTAINED WHAT IS
55019 016500 001124 001124 001126      ;GIVEN IN BAD RHDB
55020 016500 001124 001124 001126      83$:
55021 016500 012737 000100 001124      ;CHECK THAT RHCS2 HAS IR
55022 016506 053737 005010 001124      MOV   #IR,$GDDAT    ;GET GOOD = 100
55023 016514 017737 165350 001126      BIS   UNIT,$GDDAT   ;INCLUDE UNIT NUMBER
55024 016522 023737 001124 001126      MOV   @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
55025 016522 023737 001124 001126      CMP   $GDDAT,$BDDAT ;COMPARE EXPECTED
55026 016530 001401 001124 001126      ;DATA WITH DATA READ FROM
55027 016532 104025 001124 001126      ;RHCS2
55028 BEQ   85$                ;BRANCH IF GOOD
55029 016534 001124 001124 001126      ;AFTER SETTING "CLR" BIT #5
55030 016534 001124 001124 001126      ;IN RHCS2 TO INIT THE RH
55031 016534 001124 001124 001126      ;AND WRITING A PATTERN OF
55032 016534 001124 001124 001126      ;FLOATING ONES (WHICH IS
55033 016534 001124 001124 001126      ;1 2 4 10 20 40 100 200)
55034 016534 001124 001124 001126      ;INTO RHDB TO FILL IT
55035 016534 001124 001124 001126      ;THEN ON READING RHDB THE
55036 016534 001124 001124 001126      ;TOTAL 8 TIMES
55037 016534 001124 001124 001126      ;RHCS2 SHOULD HAVE IR
55038 016534 001124 001124 001126      ;=100
55039 016534 001124 001124 001126      ;TOGETHER WITH UNIT NUMBER
55040 016534 001124 001124 001126      ;BUT CONTAINED WHAT IS
55041 016534 001124 001124 001126      ;GIVEN IN BAD RHCS2
55042 016534 001124 001124 001126      85$:
55043 016534 012737 004200 001124      ;CHECK THAT RHCS1 HAS DVA!RDY
55044 016542 017737 165312 001126      MOV   #DVA!RDY,$GDDAT ;GET GOOD = 4200
55045 016550 023737 001124 001126      MOV   @RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
55046 016550 023737 001124 001126      CMP   $GDDAT,$BDDAT ;COMPARE EXPECTED
55047 016550 023737 001124 001126      ;DATA WITH DATA READ FROM
55048 016550 023737 001124 001126      ;RHCS1

```

E09

```

5505 016556 001401 BEQ 87$ ;BRANCH IF GOOD
5506 016560 104026 ERROR 26
5507 ;AFTER SETTING "CLR" BIT #5
5508 ;IN RHCS2 TO INIT THE RH
5509 ;AND WRITING A PATTERN OF
5510 ;FLOATING ONES (WHICH IS
5511 ;1 2 4 10 20 40 100 200)
5512 ;INTO RHDB TO FILL IT
5513 ;THEN ON READING RHDB THE
5514 ;TOTAL 8 TIMES
5515 ;RHCS1 SHOULD HAVE DVA!PD/
5516 ;=4200
5517 ;BUT CONTAINED WHAT IS
5518 ;GIVEN IN BAD RHCS1
5519 016562 87$:
5520 ;CHECK THAT RHCS3 HAS 0
5521 016562 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
5522
5523 016570 017737 165336 001126 MOV @RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
5524 016576 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
5525 ;DATA WITH DATA READ FROM
5526 ;RHCS3
5527 016604 001401 BEQ 89$ ;BRANCH IF GOOD
5528 016606 104027 ERROR 27
5529 ;AFTER SETTING "CLR" BIT #5
5530 ;IN RHCS2 TO INIT THE RH
5531 ;AND WRITING A PATTERN OF
5532 ;FLOATING ONES (WHICH IS
5533 ;1 2 4 10 20 40 100 200)
5534 ;INTO RHDB TO FILL IT
5535 ;THEN ON READING RHDB THE
5536 ;TOTAL 8 TIMES
5537 ;RHCS3 SHOULD HAVE 0
5538 ;BUT CONTAINED WHAT IS
5539 ;GIVEN IN BAD RHCS3
5540 016610 89$:
5541 ;CHECK THAT RHBA HAS 0
5542 016610 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
5543
5544 016616 017737 165242 001126 MOV @RHBA,$BDDAT ;READ RHBA FOR COMPARISON
5545 016624 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
5546 ;DATA WITH DATA READ FROM
5547 ;RHBA
5548 016632 001401 BEQ 91$ ;BRANCH IF GOOD
5549 016634 104030 ERROR 30
5550 ;AFTER SETTING "CLR" BIT #5
5551 ;IN RHCS2 TO INIT THE RH
5552 ;AND WRITING A PATTERN OF
5553 ;FLOATING ONES (WHICH IS
5554 ;1 2 4 10 20 40 100 200)
5555 ;INTO RHDB TO FILL IT
5556 ;THEN ON READING RHDB THE
5557 ;TOTAL 8 TIMES
5558 ;RHBA SHOULD HAVE 0
5559 ;BUT CONTAINED WHAT IS
5560 ;GIVEN IN BAD RHBA

```

F09

```

5561 016636          91$:
5562
5563 016636 012737 000000 001124      :CHECK THAT RHBAE HAS 0
5564
5565 016644 017737 165260 001126      MOV      #0,$GDDAT      :GET GOOD = 0
5566 016652 023737 001124 001126      MOV      RHBAE,$BDDAT   :READ RHBAE FOR COMPARISON
5567
5568
5569 016660 001401      BEQ      93$           :COMPARE EXPECTED
5570 016662 104031      ERROR    31           :DATA WITH DATA READ FROM
5571
5572
5573
5574
5575
5576
5577
5578
5579
5580
5581
5582
5583
5584 016664          93$:
5585
5586 016664 012737 000000 001124      :CHECK THAT RHWC HAS 0
5587
5588 016672 017737 165164 001126      MOV      #0,$GDDAT      :GET GOOD = 0
5589 016700 023737 001124 001126      MOV      RHWC,$BDDAT   :READ RHWC FOR COMPARISON
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603 016712          95$:
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616

```

```

:RMBAE
:BRANCH IF GOOD
:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:AND WRITING A PATTERN OF
:FLOATING ONES (WHICH IS
:1 2 4 10 20 40 100 200)
:INTO RHDB TO FILL IT
:THEN ON READING RHDB THE
:TOTAL 8 TIMES
:RHBAE SHOULD HAVE 0
:BUT CONTAINED WHAT IS
:GIVEN IN BAD RHBAE

```

```

:RHC
:BRANCH IF GOOD
:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:AND WRITING A PATTERN OF
:FLOATING ONES (WHICH IS
:1 2 4 10 20 40 100 200)
:INTO RHDB TO FILL IT
:THEN ON READING RHDB THE
:TOTAL 8 TIMES
:RHWC SHOULD HAVE 0
:BUT CONTAINED WHAT IS
:GIVEN IN BAD RHWC

```

```

*****
*TEST 16          SILO TEST 6 (FLOATING ONES IN UPPER BYTE)

```

```

*
* AFTER A RH CLEAR IS GIVEN (SET BIT #5 IN RHCS2)
* RHCS2 IS CHECKED TO HAVE "IR" (BIT #6) HIGH,
* TOGETHER WITH UNIT NUMBER
* EIGHT WORDS, A PATTERN OF FLOATING ONES IN UPPER BYTE
* 400, 1000, 2000, 4000, 10000, 20000, 40000, 100000
* IS WRITTEN INTO RHDB
* "OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP

```

5617
5618
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5670
5671
5672

```

: * INSTRUCTION CALLED "WAT"
: * THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
: * "OR" HIGH TOGETHER WITH THE UNIT NUMBER
: * THEN RHDB IS READ AND COMPARED TO HAVE THE
: * RIGHT VALUE AFTER EACH OF THE EIGHT REACS.
: * THEN RHCS2 IS CHECKED TO HAVE "IR"
: * AND UNIT NUMBER
: * THEN RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
: * CHECKED TO HAVE THE APPROPRIATE VALUE
: *****
15:16: SCOPE
MOV #STACK.SP ;RESET STACK
MOV #16,2#1STNM ;SAVE TEST NUMBER
JSR PC,2#CLDISK ;GIVE RH INITIALIZE
;SETUP UNIT NUMBER
;CLEAR RHWC AND FUNCTION BITS IN RHCS1

;CHECK THAT RHCS2 HAS IR
MOV #IR,$GDDAT ;GET GOOD = 100
BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER

MOV 2#RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHCS2
;BRANCH IF GOOD
BEQ 65$
ERROR 6
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;AND HAVING DONE NOTHING
;ELSE
;RHCS2 SHOULD HAVE IR
;=100
;TOGETHER WITH UNIT NUMBER
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS2

016766 65$:
;WRITE EIGHT WORDS INTO RHDB TO FILL IT
;DATA IS 400,1000,2000,4000,10000,20000,40000
;100000
MOV #400,R1 ;GETTING READY TO FLOAT ONES
;IN UPPER BYTE
MOV #8,R2 ;COUNTER -8 DATA WORDS
1$: MOV R1,2#RHDB ;WRITE INTO RHDB
ASL R1 ;SHIFT 1 ONE POSITION LEFT
DEC R2 ;COUNT TO 8
BNE 1$ ;BRANCH IF 8 NOT DONE
;WAIT FOR OR BIT IN RHCS2 REGISTER TO SET
WAT ;TRAP TO WAIT.T SUBROUTINE
RHCS2 ;AND WAIT FOR OR BIT IN
OR ;RHCS2 REGISTER
;IF ERROR OCCURS HERE
;IT MEANS "OR" DID NOT

```

016712	000004		
016714	012706	001000	
016720	012737	000016	004656
016726	004737	040306	
016732	012737	000100	001124
016740	053737	005010	001124
016746	017737	165116	001126
016754	023737	001124	001126
016762	001401		
016764	104006		
016766			
016772	012702	000010	
016776	010177	165100	
017002	006301		
017004	005302		
017006	001373		
017010	104411		
017012	004070		
017014	000200		

H09

```

5673                                     :SET FOR THE FULL COUNT
5674                                     :DOWN OF THE WAIT.Y SUBROUTINE
5675                                     :THIS TIME IS APPROXIMATELY
5676                                     :LARGER THAN 500 MILLISECONCS
5677
5678                                     :CHECK THAT RHCS2 HAS OR
5679
5680 017016 012737 000200 001124      MOV      #CR,$GDDAT      :GET GOOD = 200
5681 017024 053737 005010 001124      BIS      UNIT,$GDDAT    :INCLUDE UNIT NUMBER
5682
5683 017032 017737 165032 001126      MOV      @RHCS2,$BDDAT  :READ RHCS2 FOR COMPARISON
5684 017040 023737 001124 001126      CMP      $GDDAT,$BDDAT  :COMPARE EXPECTED
5685                                     :DATA WITH DATA READ FROM
5686                                     :RHCS2
5687 017046 001401                      BEQ      67$            :BRANCH IF GOOD
5688 017050 104023                      ERROR    23
5689
5690                                     :AFTER SETTING "CLR" BIT #5
5691                                     :IN RHCS2 TO INIT THE RH
5692                                     :AND HAVING WRITTEN A
5693                                     :FLOATING ONES PATTERN INTO UPPER BYTE
5694                                     :400,1000,2000,4000,10000,20000
5695                                     :40000,100000
5696                                     :INTO RHDB TO FILL IT
5697                                     :RHCS2 SHOULD HAVE OR
5698                                     :=200
5699                                     :TOGETHER WITH UNIT NUMBER
5700                                     :BUT CONTAINED WHAT IS
5701 017052                               67$:
5702
5703
5704
5705 017052 012737 000001 004664      MOV      #1,SILONM      :FIRST WORD TO BE READ
5706                                     :CHECK THAT RHDB HAS 400
5707 017060 012737 000400 001124      MOV      #400,$GDDAT   :GET GOOD = 0
5708
5709 017066 017737 165010 001126      MOV      @RHDB,$BDDAT  :READ RHDB FOR COMPARISON
5710 017074 023737 001124 001126      CMP      $GDDAT,$BDDAT  :COMPARE EXPECTED
5711                                     :DATA WITH DATA READ FROM
5712                                     :RHDB
5713 017102 001401                      BEQ      69$            :BRANCH IF GOOD
5714 017104 104024                      ERROR    24
5715
5716                                     :AFTER SETTING "CLR" BIT #5
5717                                     :IN RHCS2 TO INIT THE RH
5718                                     :AND WRITING A PATTERN OF FLOATING
5719                                     :ONES (WHICH IS 400,1000,2000,4000,
5720                                     :10000,20000,40000,100000)
5721                                     :INTO RHDB TO FILL IT
5722                                     :THEN ON READING RHDB THE
5723                                     :FIRST TIME
5724                                     :RHDB SHOULD HAVE 400
5725                                     :BUT CONTAINED WHAT IS
5726 017106                               69$:
5727
5728

```

```

5729 017106 012737 000002 004664      MOV      #2,SILONM      ;SECOND WORD TO BE READ
5730      ;CHECK THAT RHDB HAS 1000
5731 017114 012737 001000 001124      MOV      #1000,$GDDAT  ;GET GOOD = 0
5732
5733 017122 017737 164754 001126      MOV      @RHDB,$BDDAT  ;READ RHDB FOR COMPARISON
5734 017130 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
5735      ;DATA WITH DATA READ FROM
5736      ;RHDB
5737 017136 001401      BEQ      71$           ;BRANCH IF GOOD
5738 017140 104024      ERROR    24
5739
5740      ;AFTER SETTING "CLR" BIT #5
5741      ;IN RHCS2 TO INIT THE RH
5742      ;AND WRITING A PATTERN OF FLOATING
5743      ;ONES (WHICH IS 400,1000,2000,4000,
5744      ;10000,20000,40000,100000)
5745      ;INTO RHDB TO FILL IT
5746      ;THEN ON READING RHDB THE
5747      ;SECOND TIME
5748      ;RHDB SHOULD HAVE 1000
5749      ;BUT CONTAINED WHAT IS
5750      ;GIVEN IN BAD RHDB
5751 017142      71$:
5752
5753 017142 012737 000003 004664      MOV      #3,SILONM      ;THIRD WORD TO BE READ
5754      ;CHECK THAT RHDB HAS 2000
5755 017150 012737 002000 001124      MOV      #2000,$GDDAT  ;GET GOOD = 0
5756
5757 017156 017737 164720 001126      MOV      @RHDB,$BDDAT  ;READ RHDB FOR COMPARISON
5758 017164 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
5759      ;DATA WITH DATA READ FROM
5760      ;RHDB
5761 017172 001401      BEQ      73$           ;BRANCH IF GOOD
5762 017174 104024      ERROR    24
5763
5764      ;AFTER SETTING "CLR" BIT #5
5765      ;IN RHCS2 TO INIT THE RH
5766      ;AND WRITING A PATTERN OF FLOATING
5767      ;ONES (WHICH IS 400,1000,2000,4000,
5768      ;10000,20000,40000,100000)
5769      ;INTO RHDB TO FILL IT
5770      ;THEN ON READING RHDB THE
5771      ;THIRD TIME
5772      ;RHDB SHOULD HAVE 2000
5773      ;BUT CONTAINED WHAT IS
5774      ;GIVEN IN BAD RHDB
5775 017176      73$:
5776
5777 017176 012737 000004 004664      MOV      #4,SILONM      ;FOURTH WORD TO BE READ
5778      ;CHECK THAT RHDB HAS 4000
5779 017204 012737 004000 001124      MOV      #4000,$GDDAT  ;GET GOOD = 0
5780
5781 017212 017737 164664 001126      MOV      @RHDB,$BDDAT  ;READ RHDB FOR COMPARISON
5782 017220 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
5783      ;DATA WITH DATA READ FROM
5784      ;RHDB

```



```

5841 ; THEN ON READING RHDB THE
5842 ; SIXTH TIME
5843 ; RHDB SHOULD HAVE 20000
5844 ; BUT CONTAINED WHAT IS
5845 ; GIVEN IN BAD RHDB
5846 017322          79$:
5847
5848
5849 017322 012737 000007 004664      MOV      #7,SILONM      ;SEVENTH WORD TO BE READ
5850 ;CHECK THAT RHDB HAS 40000
5851 017330 012737 040000 001124      MOV      #40000,$GDDAT ;GET GOOD = 0
5852
5853 017336 017737 164540 001126      MOV      @RHDB,$BDDAT  ;READ RHDB FOR COMPARISON
5854 017344 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
5855 ;DATA WITH DATA READ FROM
5856 ;RHDB
5857 017352 001401          BEQ      B1$          ;BRANCH IF GOOD
5858 017354 104024          ERROR     24
5859
5860 ; AFTER SETTING "CLR" BIT #5
5861 ; IN RHCS2 TO INIT THE RH
5862 ; AND WRITING A PATTERN OF FLOATING
5863 ; ONES (WHICH IS 400,1000,2000,4000,
5864 ; 10000,20000,40000,100000)
5865 ; INTO RHDB TO FILL IT
5866 ; THEN ON READING RHDB THE
5867 ; SEVENTH TIME
5868 ; RHDB SHOULD HAVE 40000
5869 ; BUT CONTAINED WHAT IS
5870 ; GIVEN IN BAD RHDB
5871 017356          81$:
5872
5873 017356 012737 000010 004664      MOV      #8,SILONM      ;EIGHTH WORD TO BE READ
5874 ;CHECK THAT RHDB HAS 100000
5875 017364 012737 100000 001124      MOV      #100000,$GDDAT ;GET GOOD = 0
5876
5877 017372 017737 164504 001126      MOV      @RHDB,$BDDAT  ;READ RHDB FOR COMPARISON
5878 017400 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
5879 ;DATA WITH DATA READ FROM
5880 ;RHDB
5881 017406 001401          BEQ      B3$          ;BRANCH IF GOOD
5882 017410 104024          ERROR     24
5883
5884 ; AFTER SETTING "CLR" BIT #5
5885 ; IN RHCS2 TO INIT THE RH
5886 ; AND WRITING A PATTERN OF FLOATING
5887 ; ONES (WHICH IS 400,1000,2000,4000,
5888 ; 10000,20000,40000,100000)
5889 ; INTO RHDB TO FILL IT
5890 ; THEN ON READING RHDB THE
5891 ; EIGHTH TIME
5892 ; RHDB SHOULD HAVE 100000
5893 ; BUT CONTAINED WHAT IS
5894 ; GIVEN IN BAD RHDB
5895 017412          83$:
5896

```

```

5897 ;CHECK THAT RHCS2 HAS IR
5898 017412 012737 000100 001124 MOV #IR,$GDDAT ;GET GOOD = 100
5899 017420 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
5900
5901 017426 017737 164436 001126 MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
5902 017434 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
5903 ;DATA WITH DATA READ FROM
5904 ;RHCS2
5905 017442 001401 BEQ 85$ ;BRANCH IF GOOD
5906 017444 104025 ERROR 25
5907
5908 ;AFTER SETTING "CLR" BIT #5
5909 ;IN RHCS2 TO INIT THE RH
5910 ;AND WRITING A PATTERN OF FLOATING
5911 ;ONES (WHICH IS 400,1000,2000,4000,
5912 ;10000,20000,40000,100000)
5913 ;INTO RHDB TO FILL IT
5914 ;THEN ON READING RHDB THE
5915 ;TOTAL 8 TIMES
5916 ;RHCS2 SHOULD HAVE IR
5917 ;=100
5918 ;TOGETHER WITH UNIT NUMBER
5919 ;BUT CONTAINED WHAT IS
5920 ;GIVEN IN BAD RHCS2
5921 017446 85$:
5922 017446 012737 004200 001124 MOV #DVA!RDY,$GDDAT ;CHECK THAT RHCS1 HAS DVA!RDY
5923 ;GET GOOD = 4200
5924 017454 017737 164400 001126 MOV @RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
5925 017462 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
5926 ;DATA WITH DATA READ FROM
5927 ;RHCS1
5928 017470 001401 BEQ 87$ ;BRANCH IF GOOD
5929 017472 104026 ERROR 26
5930
5931 ;AFTER SETTING "CLR" BIT #5
5932 ;IN RHCS2 TO INIT THE RH
5933 ;AND WRITING A PATTERN OF FLOATING
5934 ;ONES (WHICH IS 400,1000,2000,4000,
5935 ;10000,20000,40000,100000)
5936 ;INTO RHDB TO FILL IT
5937 ;THEN ON READING RHDB HE
5938 ;TOTAL 8 TIMES
5939 ;RHCS1 SHOULD HAVE DVA!RDY
5940 ;=4200
5941 ;BUT CONTAINED WHAT IS
5942 ;GIVEN IN BAD RHCS1
5943 017474 87$:
5944 017474 012737 000000 001124 MOV #0,$GDDAT ;CHECK THAT RHCS3 HAS 0
5945 ;GET GOOD = 0
5946 017502 017737 164424 001126 MOV @RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
5947 017510 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
5948 ;DATA WITH DATA READ FROM
5949 ;RHCS3
5950 017516 001401 BEQ 89$ ;BRANCH IF GOOD
5951 017520 104027 ERROR 27
5952 ;AFTER SETTING "CLR" BIT #5
    
```

```

5953                                     ; IN RHCS2 TO INIT THE RH
5954                                     ; AND WRITING A PATTERN OF FLOATING
5955                                     ; ONES (WHICH IS 400,1000,2000,4000,
5956                                     ; 10000,20000,40000,100000)
5957                                     ; INTO RHDB TO FILL IT
5958                                     ; THEN ON READING RHDB THE
5959                                     ; TOTAL 8 TIMES
5960                                     ; RHCS3 SHOULD HAVE 0
5961                                     ; BUT CONTAINED WHAT IS
5962                                     ; GIVEN IN BAD RHCS3
5963 017522                                89$:
5964                                     ; CHECK THAT RHBA HAS 0
5965 017522 012737 000000 001124          MOV     #0,$GDDAT ; GET GOOD = 0
5966                                     ; READ RHBA FOR COMPARISON
5967 017530 017737 164330 001126          MOV     @RHBA,$BDDAT
5968 017536 023737 001124 001126          CMP     $GDDAT,$BDDAT ; COMPARE EXPECTED
5969                                     ; DATA WITH DATA READ FROM
5970                                     ; RHBA
5971 017544 001401                          BEQ     91$
5972 017546 104030                          ERROR   30 ; BRANCH IF GOOD
5973
5974                                     ; AFTER SETTING "CLR" BIT #5
5975                                     ; IN RHCS2 TO INIT THE RH
5976                                     ; AND WRITING A PATTERN OF FLOATING
5977                                     ; ONES (WHICH IS 400,1000,2000,4000,
5978                                     ; 10000,20000,40000,100000)
5979                                     ; INTO RHDB TO FILL IT
5980                                     ; THEN ON READING RHDB THE
5981                                     ; TOTAL 8 TIMES
5982                                     ; RHBA SHOULD HAVE 0
5983                                     ; BUT CONTAINED WHAT IS
5984                                     ; GIVEN IN BAD RHBA
5984 017550                                91$:
5985                                     ; CHECK THAT RHBAE HAS 0
5986 017550 012737 000000 001124          MOV     #0,$GDDAT ; GET GOOD = 0
5987                                     ; READ RHBAE FOR COMPARISON
5988 017556 017737 164346 001126          MOV     @RHBAE,$BDDAT
5989 017564 023737 001124 001126          CMP     $GDDAT,$BDDAT ; COMPARE EXPECTED
5990                                     ; DATA WITH DATA READ FROM
5991                                     ; RHBAE
5992 017572 001401                          BEQ     93$
5993 017574 104031                          ERROR   31 ; BRANCH IF GOOD
5994
5995                                     ; AFTER SETTING "CLR" BIT #5
5996                                     ; IN RHCS2 TO INIT THE RH
5997                                     ; AND WRITING A PATTERN OF FLOATING
5998                                     ; ONES (WHICH IS 400,1000,2000,4000,
5999                                     ; 10000,20000,40000,100000)
6000                                     ; INTO RHDB TO FILL IT
6001                                     ; THEN ON READING RHDB THE
6002                                     ; TOTAL 8 TIMES
6003                                     ; RHBAE SHOULD HAVE 0
6004                                     ; BUT CONTAINED WHAT IS
6005                                     ; GIVEN IN BAD RHBAE
6005 017576                                93$:
6006                                     ; CHECK THAT RHWC HAS 0
6007 017576 012737 000000 001124          MOV     #0,$GDDAT ; GET GOOD = 0
6008

```



```

6065      ;RHCS2
6066 017674 001401      BEQ      65$      ;BRANCH IF GOOD
6067 017676 104006      ERROR    6
6068      ;AFTER SETTING "CLR" BIT #5
6069      ;IN RHCS2 TO INIT THE PH
6070      ;AND HAVING DONE NOTHING
6071      ;ELSE
6072      ;RHCS2 SHOULD HAVE IR
6073      ;=100
6074      ;TOGETHER WITH UNIT NUMBER
6075      ;BUT CONTAINED WHAT IS
6076      ;GIVEN IN BAD RHCS2
6077 017700      65$:
6078
6079      ;WRITE EIGHT WORDS INTO RHDB TO FILL IT
6080      ;DATA IS FLOATING ZEROS 177776, 177775, 177773, . 7767, 177757, 177737,
6081      ;177677, 177577
6082 017700 012701 177776      MOV      #177776,R1      ;GETTING READY TO FLOAT ZEROS
6083 017704 012702 000010      MOV      #8,R2          ;COUNTER -8 DATA WORDS
6084 017710 010177 164166      MOV      R1,RHDB        ;WRITE INTO RHDB
6085 017714 000261      SEC          ;SET CARRY
6086 017716 006101      ROL      R1          ;GET ZERO ONE BIT LEFT
6087 017720 005302      DEC      R2          ;COUNT TO 8
6088 017722 001372      BNE      1$          ;BRANCH IF 8 NOT DONE
6089      ;WAIT FOR OR BIT IN RHCS2 REGISTER TO SET
6090 017724 104411      WAT          ;TRAP TO WAIT.T SUBROUTINE
6091 017726 004070      RHCS2       ;AND WAIT FOR OR BIT IN
6092 017730 000200      OR          ;RHCS2 REGISTER
6093      ;IF ERROR OCCURS HERE
6094      ;IT MEANS "OR" DID NOT
6095      ;SET FOR THE FULL COUNT
6096      ;DOWN OF THE WAIT.T SUBROUTINE
6097      ;THIS TIME IS APPROXIMATELY
6098      ;LARGER THAN 500 MILLISECONDS
6099
6100
6101      ;CHECK THAT RHCS2 HAS OR
6102 017732 012737 000200 001124      MOV      #OR,$GDDAT     ;GET GOOD = 200
6103 017740 053737 005010 001124      BIS      UNIT,$GDDAT    ;INCLUDE UNIT NUMBER
6104
6105 017746 017737 164116 001126      MOV      RHCS2,$BDDAT   ;READ RHCS2 FOR COMPARISON
6106 017754 023737 001124 001126      CMP      $GDDAT,$BDDAT  ;COMPARE EXPECTED
6107      ;DATA WITH DATA READ FROM
6108      ;RHCS2
6109 017762 001401      BEQ      67$          ;BRANCH IF GOOD
6110 017764 104023      ERROR    23
6111      ;AFTER SETTING "CLR" BIT #5
6112      ;IN RHCS2 TO INIT THE RH
6113      ;AND HAVING WRITTEN A PATTERN
6114      ;OF FLOATING ZEROS - 177776, 177775,
6115      ;177773, 177767, 177757, 177737, 177677, 177577
6116      ;INTO RHDB TO FILL IT
6117      ;RHCS2 SHOULD HAVE OR
6118      ;=200
6119      ;TOGETHER WITH UNIT NUMBER
6120      ;BUT CONTAINED WHAT IS

```


C10

```

6121
6122 017766          67$:
6123
6124
6125
6126 017766 012737 000001 004664      MOV    #1,SILONM      ;FIRST WORD TO BE READ
        ;CHECK THAT RHDB HAS 177776
6127
6128 017774 012737 177776 001124      MOV    #177776,$GDDAT ;GET GOOD = 0
6129
6130 020002 017737 164074 001126      MOV    2RHDB,$BDDAT   ;READ RHDB FOR COMPARISON
6131 020010 023737 001124 001126      CMP    $GDDAT,$BDDAT  ;COMPARE EXPECTED
6132
6133
6134 020016 001401
6135 020020 104024      BEQ    69$           ;BRANCH IF GOOD
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145
6146
6147 020022          69$:
6148
6149
6150 020022 012737 000002 004664      MOV    #2,SILONM      ;SECOND WORD TO BE READ
        ;CHECK THAT RHDB HAS 177775
6151
6152 020030 012737 177775 001124      MOV    #177775,$GDDAT ;GET GOOD = 0
6153
6154 020036 017737 164040 001126      MOV    2RHDB,$BDDAT   ;READ RHDB FOR COMPARISON
6155 020044 023737 001124 001126      CMP    $GDDAT,$BDDAT  ;COMPARE EXPECTED
6156
6157
6158 020052 001401
6159 020054 104024      BEQ    71$           ;BRANCH IF GOOD
6160
6161
6162
6163
6164
6165
6166
6167
6168
6169
6170
6171 020056          71$:
6172
6173
6174 020056 012737 000003 004664      MOV    #3,SILONM      ;THIRD WORD TO BE READ
        ;CHECK THAT RHDB HAS 177773
6175
6176 020064 012737 177773 001124      MOV    #177773,$GDDAT ;GET GOOD = 0
    
```

```

;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;AND WRITING A PATTERN OF FLOATING
;ZEROS - 177776, 177775, 177773, 177767,
;177757, 177737, 177677, 177577
;INTO RHDB TO FILL IT
;THEN ON READING RHDB THE
;FIRST TIME
;RHDB SHOULD HAVE 177776
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHDB

;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;AND WRITING A PATTERN OF FLOATING
;ZEROS - 177776, 177775, 177773, 177767,
;177757, 177737, 177677, 177577
;INTO RHDB TO FILL IT
;THEN ON READING RHDB THE
;SECOND TIME
;RHDB SHOULD HAVE 177775
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHDB
    
```


6233
6234
6235
6236
6237
6238
6239
6240
6241
6242
6243
6244
6245
6246
6247
6248
6249
6250
6251
6252
6253
6254
6255
6256
6257
6258
6259
6260
6261
6262
6263
6264
6265
6266
6267
6268
6269
6270
6271
6272
6273
6274
6275
6276
6277
6278
6279
6280
6281
6282
6283
6284
6285
6286
6287
6288

020202

795:

020202 012737 000006 004664
020210 012737 177737 001124
020216 017737 163660 001126
020224 023737 001124 001126

MOV #6,SIL0NM
;CHECK THAT RHDB HAS 177737
MOV #177737,\$GDDAT
MOV @RHDB,\$BDDAT
CMP \$GDDAT,\$BDDAT

; IN RHCS2 TO INIT THE PH
; AND WRITING A PATTERN OF FLOATING
; ZEROS - 177776, 177775, 177773, 177767,
; 177757, 177737, 177677, 177577
; INTO RHDB TO FILL IT
; THEN ON READING RHDB THE
; FIFTH TIME
; RHDB SHOULD HAVE 177757
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHDB

; SIXTH WORD TO BE READ
; GET GOOD = 0

; READ RHDB FOR COMPARISON
; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHDB

BEG 795
ERROR 24

; BRANCH IF GOOD

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; AND WRITING A PATTERN OF FLOATING
; ZEROS - 177776, 177775, 177773, 177767,
; 177757, 177737, 177677, 177577
; INTO RHDB TO FILL IT
; THEN ON READING RHDB THE
; SIXTH TIME
; RHDB SHOULD HAVE 177737
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHDB

020236

795:

020236 012737 000007 004664
020244 012737 177677 001124
020252 017737 163624 001126
020260 023737 001124 001126

MOV #7,SIL0NM
;CHECK THAT RHDB HAS 177677
MOV #177677,\$GDDAT
MOV @RHDB,\$BDDAT
CMP \$GDDAT,\$BDDAT

; SEVENTH WORD TO BE READ
; GET GOOD = 0

; READ RHDB FOR COMPARISON
; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHDB

BEG 815
ERROR 24

; BRANCH IF GOOD

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; AND WRITING A PATTERN OF FLOATING
; ZEROS - 177776, 177775, 177773, 177767,
; 177757, 177737, 177677, 177577
; INTO RHDB TO FILL IT
; THEN ON READING RHDB THE
; SEVENTH TIME
; RHDB SHOULD HAVE 177677

F10

CERNADO MACY11 3011046 21-DEC-77 13:06 PAGE 123
 CERNADO.F11 21-DEC-77 12:48 17

SILO TEST 7 (FLOATING 0 IN LOWER BYTE)

SEG 0122

6289
6290
6291
6292
6293
6294
6295
6296
6297
6298
6299
6300
6301
6302
6303
6304
6305
6306
6307
6308
6309
6310
6311
6312
6313
6314
6315
6316
6317
6318
6319
6320
6321
6322
6323
6324
6325
6326
6327
6328
6329
6330
6331
6332
6333
6334
6335
6336
6337
6338
6339
6340
6341
6342
6343
6344

020272

81\$:

012737 000010 004664
 020300 012737 177577 001124
 020306 017737 163570 001126
 020314 023737 001124 001126

MOV #8,SILONM
 ;CHECK THAT RHDB HAS 177577
 MOV #177577,\$GDDAT
 MOV JRHDB,\$BDDAT
 CMP \$GDDAT,\$BDDAT

;BUT CONTAINED WHAT IS
 ;GIVEN IN BAD RHDB
 ;EIGHTH WORD TO BE READ
 ;GET GOOD = 0
 ;READ RHDB FOR COMPARISON
 ;COMPARE EXPECTED
 ;DATA WITH DATA READ FROM
 ;RHDB
 ;BRANCH IF GOOD

020322 001401
 020324 104024

BEG 83\$
 ERROR 24

;AFTER SETTING "CLR" BIT #5
 ;IN RHCS2 TO INIT THE RH
 ;AND WRITING A PATTERN OF FLOATING
 ;ZEROS - 177776, 177775, 177773, 177767,
 ;177757, 177737, 177677, 177577
 ;INTO RHDB TO FILL IT
 ;THEN ON READING RHDB THE
 ;EIGHTH TIME
 ;RHDB SHOULD HAVE 177577
 ;BUT CONTAINED WHAT IS
 ;GIVEN IN BAD RHDB

020326

83\$:

012737 000100 001124
 020334 053737 005010 001124
 020342 017737 163522 001126
 020350 023737 001124 001126

;CHECK THAT RHCS2 HAS IR
 MOV #IR,\$GDDAT
 BIS UNIT,\$GDDAT
 MOV JRHCS2,\$BDDAT
 CMP \$GDDAT,\$BDDAT

;GET GOOD = 100
 ;INCLUDE UNIT NUMBER
 ;READ RHCS2 FOR COMPARISON
 ;COMPARE EXPECTED
 ;DATA WITH DATA READ FROM
 ;RHCS2
 ;BRANCH IF GOOD

020356 001401
 020360 104025

BEG 85\$
 ERROR 25

;AFTER SETTING "CLR" BIT #5
 ;IN RHCS2 TO INIT THE RH
 ;AND WRITING A PATTERN OF FLOATING
 ;ZEROS - 177776, 177775, 177773, 177767,
 ;177757, 177737, 177677, 177577
 ;INTO RHDB TO FILL IT
 ;THEN ON READING RHDB THE
 ;TOTAL 8 TIMES
 ;RHCS2 SHOULD HAVE IR
 ;=100
 ;TOGETHER WITH UNIT NUMBER
 ;BUT CONTAINED WHAT IS
 ;GIVEN IN BAD RHCS2

020362

85\$:

012737 004200 001124

;CHECK THAT RHCS1 HAS DVA!RDY
 MOV #DVA!RDY,\$GDDAT ;GET GOOD = 4200

SILO TEST 8 (FLOATING ZEROS IN UPPER BYTE)

```

6457 177377, 177677, 175777, 173777, 167777, 157777, 137777, 77777
6458 IS WRITTEN INTO RHDB
6459 "OR" (BIT #7 IN RHCS2) IS WAITED FOR BY A TRAP
6460 INSTRUCTION CALLED "WAT"
6461 THEN RHCS2 IS CHECKED TO HAVE "IR" LOW AND
6462 "OR" HIGH TOGETHER WITH THE UNIT NUMBER
6463 THEN RHDB IS READ AND COMPARED TO HAVE THE
6464 RIGHT VALUE AFTER EACH OF THE EIGHT READS.
6465 THEN RHCS2 IS CHECKED TO HAVE "IR"
6466 AND UNIT NUMBER
6467 THEN RHCS1, RHCS3, RHBA, RHBAE, RHWC ARE
6468 CHECKED TO HAVE THE APPROPRIATE VALUE
6469 *****
6470 020540 000004 001000 004656 ST20: SCOPE
6471 020542 012706 001000 MOV #STACK, SP ; RESET STACK
6472 020546 012737 000020 004656 MOV #20, #STNM ; SAVE TEST NUMBER
6473
6474 020554 004737 040306 JSR PC, #CLDISK ; GIVE RH INITIALIZE
6475 ; SETUP UNIT NUMBER
6476 ; CLEAR RHWC AND FUNCTION BITS IN RHCS1
6477
6478
6479 ; CHECK THAT RHCS2 HAS IR
6480 020560 012737 000100 001124 MOV #IR, $GDDAT ; GET GOOD = 100
6481 020566 053737 005010 001124 BIS UNIT, $GDDAT ; INCLUDE UNIT NUMBER
6482
6483 020574 017737 163270 001126 MOV #RHCS2, $BDDAT ; READ RHCS2 FOR COMPARISON
6484 020602 023737 001124 001126 CMP $GDDAT, $BDDAT ; COMPARE EXPECTED
6485 ; DATA WITH DATA READ FROM
6486 ; RHCS2
6487 020610 001401 BEQ 655 ; BRANCH IF GOOD
6488 020612 104006 ERROR 6
6489 ; AFTER SETTING "CLR" BIT #
6490 ; IN RHCS2 TO INIT THE RH
6491 ; AND HAVING DONE NOTHING
6492 ; ELSE
6493 ; RHCS2 SHOULD HAVE IR
6494 ; =100
6495 ; TOGETHER WITH UNIT NUMBER
6496 ; BUT CONTAINED WHAT IS
6497 ; GIVEN IN BAD RHCS2
6498 020614 655:
6499
6500 ; WRITE EIGHT WORDS INTO RHDB TO FILL IT
6501 ; A PATTERN OF FLOATING ZEROS IN UPPER BYTE
6502 ; DATA IS - 177377, 176777, 175777, 173777, 167777,
6503 ; 157777, 137777, 77777
6504 020614 012701 177377 MOV #177377, R1 ; GETTING READY TO FLOAT ZEROS
6505 ; IN UPPER BYTE
6506 020620 012702 000010 MOV #8, R2 ; COUNTER -8 DATA WORDS
6507 020624 010177 163252 15: MOV R1, #RHDB ; WRITE INTO RHDB
6508 020630 000261 SEC ; SET CARRY
6509 020632 006101 ROL R1 ; GET ZERO ONE BIT LEFT
6510 020634 005302 DEC R2 ; COUNT TO 8
6511 020636 001372 BNE 15 ; BRANCH IF 8 NOT DONE
6512 ; WAIT FOR OR BIT IN RHCS2 REGISTER TO SET

```

```

6513 020640 104411          WAT          :TRAP TO WAIT.T SUBROUTINE
6514 020642 004070          RHCS2       :AND WAIT FOR OR BIT IN
6515 020644 000200          OR          :RHCS2 REGISTER
6516                                     :IF ERROR OCCURS HERE
6517                                     :IT MEANS "OR" DID NOT
6518                                     :SET FOR THE FULL COUNT
6519                                     :DOWN OF THE WAIT.T SUBROUTINE
6520                                     :THIS TIME IS APPROXIMATELY
6521                                     :LARGER THAN 500 MILLISECONDS
6522
6523                                     :CHECK THAT RHCS2 HAS OR
6524                                     :OR, $GDDAT :GET GOOD = 200
6525 020646 012737 000200 001124  MOV          :INCLUDE UNIT NUMBER
6526 020654 053737 005010 001124  BIS          :READ RHCS2 FOR COMPARISON
6527                                     :COMPARE EXPECTED
6528 020662 017737 163202 001126  MOV          :DATA WITH DATA READ FROM
6529 020670 023737 001124 001126  CMP          :RHCS2
6530                                     :BRANCH IF GOOD
6531 020676 001401          BEQ          67$
6532 020700 104023          ERROR        23
6533                                     :AFTER SETTING "CLR" BIT #5
6534                                     :IN RHCS2 TO INIT THE RH
6535                                     :AND HAVING WRITTEN A
6536                                     :PATTERN OF FLOATING ZEROS IN UPPER BYTE
6537                                     :177377, 176777, 175777, 173777,
6538                                     :167777, 157777, 137777, 77777
6539                                     :INTO RHDB TO FILL IT
6540                                     :RHCS2 SHOULD HAVE OR
6541                                     :=200
6542                                     :TOGETHER WITH UNIT NUMBER
6543                                     :BUT CONTAINED WHAT IS
6544                                     :GIVEN IN BAD RHCS2
6545
6546 020702          67$:
6547
6548
6549
6550 020702 012737 000001 004664  MOV          #1,SILONM :FIRST WORD TO BE READ
6551 :CHECK THAT RHDB HAS 177377
6552 020710 012737 177377 001124  MOV          #177377,$GDDAT :GET GOOD = 0
6553
6554 020716 017737 163160 001126  MOV          #RHDB,$BDDAT :READ RHDB FOR COMPARISON
6555 020724 023737 001124 001126  CMP          $GDDAT,$BDDAT :COMPARE EXPECTED
6556                                     :DATA WITH DATA READ FROM
6557                                     :RHDB
6558 020732 001401          BEQ          69$
6559 020734 104024          ERROR        24
6560                                     :AFTER SETTING "CLR" BIT #5
6561                                     :IN RHCS2 TO INIT THE RH
6562                                     :AND WRITING A PATTERN OF FLOATING
6563                                     :ZEROS IN UPPER BYTE
6564                                     :177377, 176777, 175777, 173777,
6565                                     :167777, 157777, 137777, 77777
6566                                     :INTO RHDB TO FILL IT
6567                                     :THEN ON READING RHDB THE
6568                                     :FIRST TIME

```


6625	021026	012737	000004	004664	MOV	#4,SILONM	:FOURTH WORD TO BE READ
6626					:CHECK	THAT RHDB HAS 173777	
6627	021034	012737	173777	001124	MOV	#173777,\$GDDAT	:GET GOOD = 0
6628							
6629	021042	017737	163034	001126	MOV	2RHDB,\$BDDAT	:READ RHDB FOR COMPARISON
6630	021050	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED
6631							:DATA WITH DATA READ FROM
6632							:RHDB
6633	021056	001401			BEQ	75\$:BRANCH IF GOOD
6634	021060	104024			ERROR	24	
6635							:AFTER SETTING "CLR" BIT #5
6636							:IN RHCS2 TO INIT THE RH
6637							:AND WRITING A PATTERN OF FLOATING
6638							:ZEROS IN UPPER BYTE
6639							:177377, 176777, 175777, 173777.
6640							:167777, 157777, 137777, 7777
6641							:INTO RHDB TO FILL IT
6642							:THEN ON READING RHDB THE
6643							:FOURTH TIME
6644							:RHDB SHOULD HAVE 173777
6645							:BUT CONTAINED WHAT IS
6646							:GIVEN IN BAD RHDB
6647	021062					75\$:	
6648							
6649							
6650	021062	012737	000005	004664	MOV	#5,SILONM	:FIFTH WORD TO BE READ
6651					:CHECK	THAT RHDB HAS 167777	
6652	021070	012737	167777	001124	MOV	#167777,\$GDDAT	:GET GOOD = 0
6653							
6654	021076	017737	163000	001126	MOV	2RHDB,\$BDDAT	:READ RHDB FOR COMPARISON
6655	021104	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED
6656							:DATA WITH DATA READ FROM
6657							:RHDB
6658	021112	001401			BEQ	77\$:BRANCH IF GOOD
6659	021114	104024			ERROR	24	
6660							:AFTER SETTING "CLR" BIT #5
6661							:IN RHCS2 TO INIT THE RH
6662							:AND WRITING A PATTERN OF FLOATING
6663							:ZEROS IN UPPER BYTE
6664							:177377, 176777, 175777, 173777.
6665							:167777, 157777, 137777, 7777
6666							:INTO RHDB TO FILL IT
6667							:THEN ON READING RHDB THE
6668							:FIFTH TIME
6669							:RHDB SHOULD HAVE 167777
6670							:BUT CONTAINED WHAT IS
6671							:GIVEN IN BAD RHDB
6672	021116					77\$:	
6673							
6674							
6675	021116	012737	000006	004664	MOV	#6,SILONM	:SIXTH WORD TO BE READ
6676					:CHECK	THAT RHDB HAS 157777	
6677	021124	012737	157777	001124	MOV	#157777,\$GDDAT	:GET GOOD = 0
6678							
6679	021132	017737	162744	001126	MOV	2RHDB,\$BDDAT	:READ RHDB FOR COMPARISON
6680	021140	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED

```

6681 ;DATA WITH DATA READ FROM
6682 ;RHDB
6683 021146 001401 BEQ 79$ ;BRANCH IF GOOD
6684 021150 104024 ERROR 24
6685 ;AFTER SETTING "CLR" BIT #5
6686 ;IN RHCS2 TO INIT THE RH
6687 ;AND WRITING A PATTERN OF FLOATING
6688 ;ZEROS IN UPPER BYTE
6689 ;177377, 176777, 175777, 173777,
6690 ;167777, 157777, 137777, 77777
6691 ;INTO RHDB TO FILL IT
6692 ;THEN ON READING RHDB THE
6693 ;SIXTH TIME
6694 ;RHDB SHOULD HAVE 157777
6695 ;BUT CONTAINED WHAT IS
6696 ;GIVEN IN BAD RHDB
6697 021152 79$:
6698
6699
6700 021152 012737 000007 004664 MOV #7,SILONM ;SEVENTH WORD TO BE READ
6701 ;CHECK THAT RHDB HAS 137777
6702 021160 012737 137777 001124 MOV #137777,$GDDAT ;GET GOOD = 0
6703
6704 021166 017737 162710 001126 MOV @RHDB,$BDDAT ;READ RHDB FOR COMPARISON
6705 021174 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
6706 ;DATA WITH DATA READ FROM
6707 ;RHDB
6708 021202 001401 BEQ 81$ ;BRANCH IF GOOD
6709 021204 104024 ERROR 24
6710 ;AFTER SETTING "CLR" BIT #5
6711 ;IN RHCS2 TO INIT THE RH
6712 ;AND WRITING A PATTERN OF FLOATING
6713 ;ZEROS IN UPPER BYTE
6714 ;177377, 176777, 175777, 173777,
6715 ;167777, 157777, 137777, 77777
6716 ;INTO RHDB TO FILL IT
6717 ;THEN ON READING RHDB THE
6718 ;SEVENTH TIME
6719 ;RHDB SHOULD HAVE 137777
6720 ;BUT CONTAINED WHAT IS
6721 ;GIVEN IN BAD RHDB
6722 021206 81$:
6723
6724
6725 021206 012737 000010 004664 MOV #8,SILONM ;EIGHTH WORD TO BE READ
6726 ;CHECK THAT RHDB HAS 77777
6727 021214 012737 077777 001124 MOV #77777,$GDDAT ;GET GOOD = 0
6728
6729 021222 017737 162654 001126 MOV @RHDB,$BDDAT ;READ RHDB FOR COMPARISON
6730 021230 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
6731 ;DATA WITH DATA READ FROM
6732 ;RHDB
6733 021236 001401 BEQ 83$ ;BRANCH IF GOOD
6734 021240 104024 ERROR 24
6735 ;AFTER SETTING "CLR" BIT #5
6736 ;IN RHCS2 TO INIT THE RH

```

```

6737 ;AND WRITING A PATTERN OF FLOATING
6738 ;ZEROS IN UPPER BYTE
6739 ;177377, 176777, 175777, 173777.
6740 ;167777, 157777, 137777, 77777
6741 ;INTO RHDB TO FILL IT
6742 ;THEN ON READING RHDB THE
6743 ;EIGHTH TIME
6744 ;RHDB SHOULD HAVE 77777
6745 ;BUT CONTAINED WHAT IS
6746 ;GIVEN IN BAD RHDB

```

6747 021242 83\$:

```

6748 ;CHECK THAT RHCS2 HAS IR
6749 MOV #IR,$GDDAT ;GET GOOD = 100
6750 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
6751 021242 012737 000100 001124
6752 021250 053737 005010 001124
6753
6754 021256 017737 162606 001126
6755 021264 023737 001124 001126
6756 MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
6757 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
6758 ;DATA WITH DATA READ FROM
6759 ;RHCS2
6760 BEQ 85$ ;BRANCH IF GOOD
6761 ERROR 25

```

```

6762 ;AFTER SETTING "CLR" BIT #5
6763 ;IN RHCS2 TO INIT THE RH
6764 ;AND WRITING A PATTERN OF FLOATING
6765 ;ZEROS IN UPPER BYTE
6766 ;177377, 176777, 175777, 173777.
6767 ;167777, 157777, 137777, 77777
6768 ;INTO RHDB TO FILL IT
6769 ;THEN ON READING RHDB THE
6770 ;TOTAL 8 TIMES
6771 ;RHCS2 SHOULD HAVE IR
6772 ;=100
6773 ;TOGETHER WITH UNIT NUMBER
6774 ;BUT CONTAINED WHAT IS
6775 ;GIVEN IN BAD RHCS2

```

6776 021276 85\$:

```

6777 ;CHECK THAT RHCS1 HAS DVA!RDY
6778 MOV #DVA!RDY,$GDDAT ;GET GOOD = 4200
6779
6780 021304 017737 162550 001126
6781 021312 023737 001124 001126
6782 MOV @RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
6783 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
6784 ;DATA WITH DATA READ FROM
6785 ;RHCS1
6786 BEQ 87$ ;BRANCH IF GOOD
6787 ERROR 26

```

```

6788 ;AFTER SETTING "CLR" BIT #5
6789 ;IN RHCS2 TO INIT THE RH
6790 ;AND WRITING A PATTERN OF FLOATING
6791 ;ZEROS IN UPPER BYTE
6792 ;177377, 176777, 175777, 173777.
6793 ;167777, 157777, 137777, 77777
6794 ;INTO RHDB TO FILL IT
6795 ;THEN ON READING RHDB THE
6796 ;TOTAL 8 TIMES

```



```

6849 021422 001401 BEQ 93$ ;BRANCH IF GOOD
6850 021424 104031 ERROR 31
6851 ;AFTER SETTING "CLR" BIT #5
6852 ;IN RHCS2 TO INIT THE RH
6853 ;AND WRITING A PATTERN OF FLOATING
6854 ;ZEROS IN UPPER BYTE
6855 ;177377, 176777, 175777, 173777.
6856 ;167777, 157777, 137777, 77777.
6857 ;INTO RHDB TO FILL IT
6858 ;THEN ON READING RHDB THE
6859 ;TOTAL 8 TIMES
6860 ;RHBAE SHOULD HAVE 0
6861 ;BUT CONTAINED WHAT IS
6862 ;GIVEN IN BAD RHBAE
6863 021426 93$:
6864 ;CHECK THAT RHWC HAS 0
6865 021426 012737 000000 001124 MOV #0, $GDDAT ;GET GOOD = 0
6866
6867 021434 017737 162422 001126 MOV @RHWC, $BDDAT ;READ RHWC FOR COMPARISON
6868 021442 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
6869 ;DATA WITH DATA READ FROM
6870 ;RHWC
6871 021450 001401 BEQ 95$ ;BRANCH IF GOOD
6872 021452 104032 ERROR 32
6873 ;AFTER SETTING "CLR" BIT #5
6874 ;IN RHCS2 TO INIT THE RH
6875 ;AND WRITING A PATTERN OF FLOATING
6876 ;ZEROS IN UPPER BYTE
6877 ;177377, 176777, 175777, 173777.
6878 ;167777, 157777, 137777, 77777.
6879 ;INTO RHDB TO FILL IT
6880 ;THEN ON READING RHDB THE
6881 ;TOTAL 8 TIMES
6882 ;RHWC SHOULD HAVE 0
6883 ;BUT CONTAINED WHAT IS
6884 ;GIVEN IN BAD RHWC
6885 021454 95$:
6886
6887
6888 ;*****
6889 ;*TEST 21 RHCS1 - MCPE BIT #13 (PARITY LINE = 0)
6890
6891 ;* AN RH CLEAR (SET BIT #5 IN RHCS2) IS GIVEN TO
6892 ;* CLEAR ALL DEVICE REGISTERS
6893 ;* THEN RHER1 (ERROR REGISTER 1) IS CHECKED TO HAVE
6894 ;* ZEROS
6895 ;* SET "PAT" (BIT #4 IN RHCS2) TO INVERT PARITY CHECKING
6896 ;* READ ANY DEVICE REGISTER - HERE RHER1
6897 ;* READ AND CHECK RHCS1 TO CONTAIN SC (BIT #15)
6898 ;* AND MCPE (BIT #13)
6899 ;* WRITE "1" INTO TRE (BIT #14 IN RHCS1)
6900 ;* READ RHCS1 SC, MCPE, AND RDY SHOULD BE SET
6901 ;* GIVE AN RH CLEAR (CLR - BIT #5 IN RHCS2)
6902 ;* CHECK RHCS1 TO HAVE RDY
6903 ;* CHECK RHCS2 TO HAVE ONLY IR AND UNIT NUMBER
6904 ;* CHECK RHCS3, RHBA, RHBAE, RHWC TO HAVE APPROPRIATE

```

```

6905          : *      VALUES.
6906          : *****
6907 021454 000004          TST21: SCOPE
6908 021456 012706 001000  MOV      #STACK,SP      ;RESET STACK
6909 021462 012737 000021 004656  MOV      #21,#STNM      ;SAVE TEST NUMBER
6910
6911 021470 004737 040306      JSR      PC,#CLDISK     ;GIVE RH INITIALIZE
6912                          ;SETUP UNIT NUMBER
6913                          ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
6914
6915
6916          ;CHECK THAT RHER1 HAS 0
6917 021474 012737 000000 001124  MOV      #0,$GDDAT      ;GET GOOD = 0
6918
6919 021502 017737 162366 001126  MOV      @RHER1,$BDDAT  ;READ RHER1 FOR COMPARISON
6920 021510 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE EXPECTED
6921                          ;DATA WITH DATA READ FROM
6922                          ;RHER1
6923 021516 001401          BEQ      65$           ;BRANCH IF GOOD
6924 021520 104033          ERROR    33
6925
6926          ;AFTER SETTING "CLR" BIT #15
6927          ;IN RHCS2 TO INIT THE RH
6928          ;AND HAVING DONE NOTHING ELSE
6929          ;RHER1 SHOULD HAVE 0
6930          ;BUT CONTAINED WHAT IS
6931          ;GIVEN IN BAD RHER1
6931 021522          65$:
6932
6933          ;TO INVERT THE PARITY CHECKING ON THE CONTROL BUS
6934          ;"PAT" BIT #4 IN RHCS2 IS SET
6935 021522 052777 000020 162340  BIS      #PAT,@RHCS2    ;SET PAT = 20 IN RHCS2
6936
6937
6938          ;CHECK THAT RHER1 HAS 0
6939 021530 012737 000000 001124  MOV      #0,$GDDAT      ;GET GOOD = 0
6940
6941 021536 017737 162332 001126  MOV      @RHER1,$BDDAT  ;READ RHER1 FOR COMPARISON
6942 021544 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE EXPECTED
6943                          ;DATA WITH DATA READ FROM
6944                          ;RHER1
6945 021552 001401          BEQ      67$           ;BRANCH IF GOOD
6946 021554 104034          ERROR    34
6947
6948          ;AFTER CLEARING THE RH AND
6949          ;DEVICE BY AN RH CLEAR
6950          ;AND READING RHER1 WITH
6951          ;WRONG PARITY CHECKING
6952          ;RHER1 SHOULD HAVE 0
6953          ;BUT CONTAINED WHAT IS
6954          ;GIVEN IN BAD RHER1
6954 021556          67$:
6955
6956          ;HAVING READ RHER1 WITH WRONG PARITY BEING
6957          ;CHECKED MOPE - BIT #13 AND SC BIT #15 IN RHCS1 SHOULD BE SET
6958
6959          ;CHECK THAT RHCS1 HAS 4200
6960 021556 012737 004200 001124  MOV      #4200,$GDDAT   ;GET GOOD = 124200

```


G11

```

7073
7074 021760 001401          BEQ      77$
7075 021762 104041          ERROR   41
7076
7077
7078
7079
7080
7081
7082
7083
7084
7085
7086
7087
7088
7089
7090 021764          77$:
7091          ;CHECK THAT RHBA HAS 0
7092 021764 012737 000000 001124  MOV     #0,$GDDAT ;GET GOOD = 0
7093
7094 021772 017737 162066 001126  MOV     @RHBA,$BDDAT ;READ RHBA FOR COMPARISON
7095 022000 023737 001124 001126  CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED
7096          ;DATA WITH DATA READ FROM
7097          ;RHBA
7098 022006 001401          BEQ      79$
7099 022010 104042          ERROR   42
7100
7101
7102
7103
7104
7105
7106
7107
7108
7109
7110
7111
7112
7113
7114 022012          79$:
7115          ;CHECK THAT RHBAE HAS 0
7116 022012 012737 000000 001124  MOV     #0,$GDDAT ;GET GOOD = 0
7117
7118 022020 017737 162104 001126  MOV     @RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
7119 022026 023737 001124 001126  CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED
7120          ;DATA WITH DATA READ FROM
7121          ;RHBAE
7122 022034 001401          BEQ      81$
7123 022036 104043          ERROR   43
7124
7125
7126
7127
7128
    
```

```

;RHCS3
;BRANCH IF GOOD
;AFTER CLEARING THE RH AND
;DEVICE REGISTERS BY AN
;RH CLEAR
;RHERI WAS CHECKED TO HAVE
;ZEROS
;WRONG PARITY CHECKING WAS
;SET BY "PAT" BIT IN RHCS2
;RHERI WAS READ TO SET
;MCPE AND SC IN RHCS1
;ON WRITING "1" INTO TRE
;THEN SETTING "CLR" IN RHCS1
;RHCS3 SHOULD HAVE 0
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS3
;CHECK THAT RHBA HAS 0
;GET GOOD = 0
;READ RHBA FOR COMPARISON
;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHBA
;BRANCH IF GOOD
;AFTER CLEARING THE RH AND
;DEVICE REGISTERS BY AN
;RH CLEAR
;RHERI WAS CHECKED TO HAVE
;ZEROS
;WRONG PARITY CHECKING WAS
;SET BY "PAT" BIT IN RHCS2
;RHERI WAS READ TO SET
;MCPE AND SC IN RHCS1
;ON WRITING "1" INTO TRE
;THEN SETTING "CLR" IN RHCS1
;RHBA SHOULD HAVE 0
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHBA
;CHECK THAT RHBAE HAS 0
;GET GOOD = 0
;READ RHBAE FOR COMPARISON
;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHBAE
;BRANCH IF GOOD
;AFTER CLEARING THE RH AND
;DEVICE REGISTERS BY AN
;RH CLEAR
;RHERI WAS CHECKED TO HAVE
;ZEROS
    
```

H11

```

7139 :WRONG PARITY CHECKING WAS
7140 :SET BY "PAT" BIT IN RHCS2
7141 :RHER1 WAS READ TO SET
7142 :MCPE AND SC IN RHCS1
7143 :ON WRITING "1" INTO TRE
7144 :THEN SETTING "CLR" IN RHCS1
7145 :RHBAE SHOULD HAVE 0
7146 :BUT CONTAINED WHAT IS
7147 :GIVEN IN BAD RHBAE

```

```

7148 022040 012737 000000 001124 81$:
7149 :CHECK THAT RHWC HAS 0
7150 MOV #0,$GDDAT ;GET GOOD = 0

```

```

7151 022046 017737 162010 001126 MOV RHWC,$BDDAT ;READ RHWC FOR COMPARISON
7152 022054 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED

```

```

7153 :DATA WITH DATA READ FROM
7154 :RHWC
7155 022062 001401 BEQ 83$ ;BRANCH IF GOOD
7156 022064 104044 EPRCR 44

```

```

7157 :AFTER CLEARING THE RH AND
7158 :DEVICE REGISTERS BY AN
7159 :RH CLEAR
7160 :RHER1 WAS CHECKED TO HAVE
7161 :ZEROS
7162 :WRONG PARITY CHECKING WAS
7163 :SET BY "PAT" BIT IN RHCS2
7164 :RHER1 WAS READ TO SET
7165 :MCPE AND SC IN RHCS1
7166 :ON WRITING "1" INTO TRE
7167 :THEN SETTING "CLR" IN RHCS1
7168 :RHWC SHOULD HAVE 0
7169 :BUT CONTAINED WHAT IS
7170 :GIVEN IN BAD RHWC

```

```

7171 022066 83$:

```

```

7172 :*****
7173 :TEST 22 RHCS1 - MCPE BIT #13 (PARITY LINE = 1)
7174 :
7175 : AN RH CLEAR (SET BIT #5 IN RHCS2) IS GIVEN TO CLEAR
7176 : ALL DEVICE REGISTERS
7177 : WRITE A ONE INTO DISK ADDRESS REGISTER (RHDA)
7178 : (IN TAPE DRIVES CALLED REGISTER)
7179 : SET "PAT" (RHCS2 BIT #4) THIS WILL INVERT THE PARITY CHECKING
7180 : READ RHDA AND COMPARE IT HAS ONE IN IT
7181 : THIS READING SHOULD SET SC, MCPE IN RHCS1
7182 : CHECK RHCS1 TO HAVE ONLY ROY SET
7183 : CHECK RHCS2, RHCS3, RHBA, RHBAE, RHWC TO HAVE APPROPRIATE
7184 : VALUES
7185 :*****

```

```

7186 022066 000004 ST22: SCOPE
7187 022070 012706 001000 MOV #STACK.SP ;RESET STACK
7188 022074 012737 000022 004656 MOV #22,$#TSTNM ;SAVE TEST NUMBER
7189 022102 004737 040306 JSR PC,$#CLDISK ;GIVE RH INITIALIZE
7190 :SETUP UNIT NUBER

```

```

7185                                     ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
7186
7187                                     ;WRITE A ONE INTO RHDA - DISK ADDRESS REGISTER
7188                                     ;THIS REGISTER IN TAPE DRIVES IS CALLED "RHFC FRAME COUNT"
7189                                     ;AFTER WRITING THIS "1" THEN ON READING THIS REGISTER
7190                                     ;THE CONTROL PARITY LINE WILL HAVE ZERO
7191 022106 012777 000001 161752      MOV     #BI 0, @RHDA      ;WRITE "1" INTO RHDA
7192                                     ;IN TAPE DRIVES CALLED RHFC
7193
7194                                     ;INVERT THE PARITY TO BE CHECKED BY SETTING "PAT" BIT #4 IN RHCS2
7195 022114 052777 000020 161746      BIS     #PAT, @RHCS2     ;SET PAT IN RHCS2
7196
7197
7198                                     ;CHECK THAT RHDA HAS BIT0
7199 022122 012737 000001 001124      MOV     #BIT0, $GDDAT    ;GET GOOD = 1
7200
7201 022130 017737 161732 001126      MOV     @RHDA, $BDDAT    ;READ RHDA FOR COMPARISON
7202 022136 023737 001124 001126      CMP     $GDDAT, $BDDAT   ;COMPARE EXPECTED
7203                                     ;DATA WITH DATA READ FROM
7204                                     ;RHDA
7205 022144 001401                       BEQ     65$              ;BRANCH IF GOOD
7206 022146 104045                       ERROR   45
7207                                     ;AFTER AN RH CLEAR "1"
7208                                     ;WAS WRITTEN INTO ADDRESS REGISTER
7209                                     ; (FRAME COUNT IN TAPE)
7210                                     ;PAT IN RHCS2 WAS SET
7211                                     ;THEN THE ADDRESS REGISTER
7212                                     ;WAS READ BACK
7213                                     ;RHDA SHOULD HAVE BIT0
7214                                     ;=1
7215                                     ;BUT CONTAINED WHAT IS
7216                                     ;GIVEN IN BAD RHDA
7217 022150                               65$:
7218
7219                                     ;CHECK THAT RHCS1 HAS 4200
7220 022150 012737 004200 001124      MOV     #4200, $GDDAT    ;GET GOOD = 124000
7221
7222 022156 017737 161676 001126      MOV     @RHCS1, $BDDAT   ;READ RHCS1 FOR COMPARISON
7223 022164 023737 001124 001126      CMP     $GDDAT, $BDDAT   ;COMPARE EXPECTED
7224                                     ;DATA WITH DATA READ FROM
7225                                     ;RHCS1
7226 022172 001401                       BEQ     67$              ;BRANCH IF GOOD
7227 022174 104046                       ERROR   46
7228                                     ;AFTER AN RH CLEAR "1"
7229                                     ;WAS WRITTEN INTO ADDRESS REGISTER
7230                                     ; (FRAME COUNT IN TAPE)
7231                                     ;PAT IN RHCS2 WAS SET
7232                                     ;THEN THE ADDRESS REGISTER
7233                                     ;WAS READ BACK
7234                                     ;RHCS1 SHOULD HAVE 4200
7235                                     ;=124000
7236                                     ;BUT CONTAINED WHAT IS
7237                                     ;GIVEN IN BAD RHCS1
7238 022176                               67$:
7239
7240 022176 012737 000120 001124      MOV     #IR!PAT, $GDDAT  ;GET GOOD = 120

```

```

7241 022204 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
7242
7243 022212 017737 161652 001126 MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
7244 022220 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7245 ;DATA WITH DATA READ FROM
7246 ;RHCS2
7247 022226 001401 BEQ 69$ ;BRANCH IF GOOD
7248 022230 104047 ERROR 47
7249
7250 ;AFTER AN RH CLEAR "1"
7251 ;WAS WRITTEN INTO ADDRESS REGISTER
7252 ;(FRAME COUNT IN TAPE)
7253 ;PAT IN RHCS2 WAS SET
7254 ;THEN THE ADDRESS REGISTER
7255 ;WAS READ BACK
7256 ;RHCS2 SHOULD HAVE IR!PAT
7257 ;=120
7258 ;TOGETHER WITH UNIT NUMBER
7259 ;BUT CONTAINED WHAT IS
7260 ;GIVEN IN BAD RHCS2
7261 022232 69$:
7262 022232 012737 000000 001124 MOV ;CHECK THAT RHCS3 HAS 0
7263 ;@0,$GDDAT ;GET GOOD = 0
7264 022240 017737 161666 001126 MOV @RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
7265 022246 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7266 ;DATA WITH DATA READ FROM
7267 ;RHCS3
7268 022254 001401 BEQ 71$ ;BRANCH IF GOOD
7269 022256 104050 ERROR 50
7270
7271 ;AFTER AN RH CLEAR "1"
7272 ;WAS WRITTEN INTO ADDRESS REGISTER
7273 ;(FRAME COUNT IN TAPE)
7274 ;PAT IN RHCS2 WAS SET
7275 ;THEN THE ADDRESS REGISTER
7276 ;WAS READ BACK
7277 ;RHCS3 SHOULD HAVE 0
7278 ;BUT CONTAINED WHAT IS
7279 ;GIVEN IN BAD RHCS3
7280 022260 71$:
7281 022260 012737 000000 001124 MOV ;CHECK THAT RHBA HAS 0
7282 ;@0,$GDDAT ;GET GOOD = 0
7283 022266 017737 161572 001126 MOV @RHBA,$BDDAT ;READ RHBA FOR COMPARISON
7284 022274 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7285 ;DATA WITH DATA READ FROM
7286 ;RHBA
7287 022302 001401 BEQ 73$ ;BRANCH IF GOOD
7288 022304 104051 ERROR 51
7289
7290 ;AFTER AN RH CLEAR "1"
7291 ;WAS WRITTEN INTO ADDRESS REGISTER
7292 ;(FRAME COUNT IN TAPE)
7293 ;PAT IN RHCS2 WAS SET
7294 ;THEN THE ADDRESS REGISTER
7295 ;WAS READ BACK
7296 ;RHBA SHOULD HAVE 0
7297 ;BUT CONTAINED WHAT IS

```

```

7297                                     ;GIVEN IN BAD RHBA
7298 022306 73$:
7299                                     ;CHECK THAT RHBAE HAS 0
7300 022306 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
7301
7302 022314 017737 161610 001126 MOV @RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
7303 022322 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7304                                     ;DATA WITH DATA READ FROM
7305                                     ;RHBAE
7306 022330 001401 BEQ 75$ ;BRANCH IF GOOD
7307 022332 104052 ERROR 52
7308                                     ;AFTER AN RH CLEAR "1"
7309                                     ;WAS WRITTEN INTO ADDRESS REGISTER
7310                                     ;(FRAME COUNT IN TAPE)
7311                                     ;PAT IN RHCS2 WAS SET
7312                                     ;THEN THE ADDRESS REGISTER
7313                                     ;WAS READ BACK
7314                                     ;RHBAE SHOULD HAVE 0
7315                                     ;BUT CONTAINED WHAT IS
7316                                     ;GIVEN IN BAD RHBAE
7317 022334 75$:
7318                                     ;CHECK THAT RHWC HAS 0
7319 022334 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
7320
7321 022342 017737 161514 001126 MOV @RHWC,$BDDAT ;READ RHWC FOR COMPARISON
7322 022350 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7323                                     ;DATA WITH DATA READ FROM
7324                                     ;RHWC
7325 022356 001401 BEQ 77$ ;BRANCH IF GOOD
7326 022360 104053 ERROR 53
7327                                     ;AFTER AN RH CLEAR "1"
7328                                     ;WAS WRITTEN INTO ADDRESS REGISTER
7329                                     ;(FRAME COUNT IN TAPE)
7330                                     ;PAT IN RHCS2 WAS SET
7331                                     ;THEN THE ADDRESS REGISTER
7332                                     ;WAS READ BACK
7333                                     ;RHWC SHOULD HAVE 0
7334                                     ;BUT CONTAINED WHAT IS
7335                                     ;GIVEN IN BAD RHWC
7336 022362 77$:
7337
7338
7339 ;*****
7340 ;*TEST 23 TEST DUPLICATED A16 (RHCS1 BIT #8)
7341
7342 ;* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
7343 ;* MOVE 400 (ONE INTO A16) IN RHCS1
7344 ;* READ RHCS1 TO CONTAIN 600 (BIT #8 AND RDY)
7345 ;* READ RHBAE TO CONTAIN "1" (BIT #0 HIGH)
7346 ;*
7347 ;*
7348 ;* MOVE 0 INTO RHBAE (WRITING 0 INTO RHBAE BIT #0)
7349 ;* READ RHBAE TO CONTAIN 0
7350 ;* READ RHCS1 TO CONTAIN ONLY RDY (BIT #8 IN RHCS1 IS ZERO)
7351 ;*
7352 ;*

```

```

7353
7354 022362 000004
7355 022364 012706 001000
7356 022370 012737 000023 004656
7357
7358 022376 004737 040306
7359
7360
7361
7362
7363 022402 012777 000400 161450
7364
7365
7366
7367
7368 022410 012737 004600 001124
7369
7370 022416 017737 161436 001126
7371 022424 023737 001124 001126
7372
7373
7374 022432 001401
7375 022434 104054
7376
7377
7378
7379
7380
7381
7382
7383
7384 022436
7385
7386
7387
7388 022436 012737 000001 001124
7389
7390 022444 017737 161460 001126
7391 022452 023737 001124 001126
7392
7393
7394 022460 001401
7395 022462 104055
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405 022464
7406
7407
7408 022464 005077 161440
    
```

```

*****
TST23: SCOPE
MOV #STACK,SP ;RESET STACK
MOV #23,STNM ;SAVE TEST NUMBER
JSR PC,CLDISK ;GIVE RH INITIALIZE
;SETUP UNIT NUMBER
;CLEAR RHWC AND FUNCTION BITS IN RHCS1
;WRITE "1" INTO A16 IN RHCS1
MOV #A16,RHCS1 ;MOVE 400 INTO RHCS1
;CHECK THAT RHCS1 HAS A16!DVA!RDY
MOV #A16!DVA!RDY,$GDDAT ;GET GOOD = 4600
MOV #RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHCS1
;BRANCH IF GOOD
BEQ 65$
ERROR 54
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;A16 WAS WRITTEN INTO RHCS1
;RHCS1 WAS READ
;RHCS1 SHOULD HAVE A16!DVA!RDY
;=4600
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS1
65$:
;CHECK THAT RHBAE HAS BIT0
MOV #BIT0,$GDDAT ;GET GOOD = 1
MOV #RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHBAE
;BRANCH IF GOOD
BEQ 67$
ERROR 55
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;A16 WAS WRITTEN INTO RHCS1
;RHCS1 WAS READ
;THEN RHBAE WAS READ
;RHBAE SHOULD HAVE BIT0
;=1
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHBAE
67$:
;NOW MOVE 0 INTO RHBAE (WRITING 0 INTO RHBAE BIT #0)
CLR #RHBAE ;WRITE ZERO INTO RHBAE
    
```

M11

```

7409
7410
7411
7412 022470 012737 000000 001124 ;CHECK THAT RHBAE HAS 0
MOV #0,$GDDAT ;GET GOOD = 0
7413
7414 022476 017737 161426 001126 MOV @RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
7415 022504 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7416 ;DATA WITH DATA READ FROM
7417 ;RHBAE
7418 022512 001401 BEQ 69$ ;BRANCH IF GOOD
7419 022514 104056 ERROR 56
7420 ;
7421 ;AFTER SETTING "CLR" BIT #5
7422 ;IN RHCS2 TO INIT THE RH
7423 ;A16 WAS WRITTEN INTO RHCS1
7424 ;RHCS1 WAS READ
7425 ;RHBAE WAS READ
7426 ;THEN ZERO WAS WRITTEN
7427 ;INTO RHBAE
7428 ;ON READING RHBAE
7429 ;RHBAE SHOULD HAVE 0
7430 ;BUT CONTAINED WHAT IS
7431 ;GIVEN IN BAD RHBAE
7432
7433 022516 69$:
7434
7435 022516 012737 004200 001124 ;CHECK THAT RHCS1 HAS DVA!RDY
MOV #DVA!RDY,$GDDAT ;GET GOOD = 4200
7436
7437 022524 017737 161330 001126 MOV @RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
7438 022532 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7439 ;DATA WITH DATA READ FROM
7440 ;RHCS1
7441 022540 001401 BEQ 71$ ;BRANCH IF GOOD
7442 022542 104057 ERROR 57
7443 ;
7444 ;AFTER SETTING "CLR" BIT #5
7445 ;IN RHCS2 TO INIT THE RH
7446 ;A16 WAS WRITTEN INTO RHCS1
7447 ;RHCS1 WAS READ
7448 ;RHBAE WAS READ
7449 ;THEN ZERO WAS WRITTEN
7450 ;INTO RHBAE
7451 ;RHBAE WAS READ
7452 ;ON READING RHCS1
7453 ;RHCS1 SHOULD HAVE DVA!RDY
7454 ;=4200
7455 ;BUT CONTAINED WHAT IS
7456 ;GIVEN IN BAD RHCS1
7457
7458 022544 71$:
7459
7460 ;*****
7461 ;*TEST 24 TEST DUPLICATED A17 (RHCS1 BIT #9)
7462
7463 ;* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
7464 ;* MOVE 400 (ONE INTO A17) IN RHCS1
7465 ;* READ RHCS1 TO CONTAIN 1200 (BIT #9 AND RDY)

```


READ RHBAE TO CONTAIN "2" (BIT #1 HIGH)

MCVE 0 INTO RHBAE (WRITING 0 INTO RHBAE BIT #1)
READ RHBAE TO CONTAIN 2
READ RHCSI TO CONTAIN ONLY RDY (BIT #9 IN RHCSI IS ZERO)

T24: SCOPE

MOV #STACK, SP ; RESET STACK
MOV #24, @#TSTNM ; SAVE TEST NUMBER
JSR PC, @#CLDISK ; GIVE RH INITIALIZE
; SETUP UNIT NUMBER
; CLEAR RHC AND FUNCTION BITS IN RHCSI

; WRITE "1" INTO A17 IN RHCSI
MOV #A17, @RHCSI ; MOVE 1200 INTO RHCSI

; CHECK THAT RHCSI HAS A17!DVA!RDY
MOV #A17!DVA!RDY, \$GDDAT ; GET GOOD = 4600

MCV @RHCSI, \$BDDAT ; READ RHCSI FOR COMPARISON
CMP \$GDDAT, \$BDDAT ; COMPARE EXPECTED

BEQ 65\$; BRANCH IF GOOD
ERROR 60

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; A17 WAS WRITTEN INTO RHCSI
; RHCSI WAS READ
; RHCSI SHOULD HAVE A17!DVA!RDY
; =4600
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHCSI

65\$:

; CHECK THAT RHBAE HAS BIT1
MOV #BIT1, \$GDDAT ; GET GOOD = 2

MOV @RHBAE, \$BDDAT ; READ RHBAE FOR COMPARISON
CMP \$GDDAT, \$BDDAT ; COMPARE EXPECTED

BEQ 67\$; BRANCH IF GOOD
ERROR 61

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; A17 WAS WRITTEN INTO RHCSI
; RHCSI WAS READ
; THEN RHBAE WAS READ

7465
7466
7467
7468
7469
7470
7471
7472
7473
7474 022544 000004
7475 022546 012706 001000
7476 022552 012737 000024 004656
7477
7478 022560 004737 040306
7479
7480
7481
7482
7483 022564 012777 001000 161266
7484
7485
7486
7487
7488 022572 012737 005200 001124
7489
7490 022600 017737 161254 001126
7491 022606 023737 001124 001126
7492
7493
7494 022614 001401
7495 022616 104060
7496
7497
7498
7499
7500
7501
7502
7503
7504 022620
7505
7506
7507
7508 022620 012737 000002 001124
7509
7510 022626 017737 161276 001126
7511 022634 023737 001124 001126
7512
7513
7514 022642 001401
7515 022644 104061
7516
7517
7518
7519
7520

```

7521                                     ;RHBAE SHOULD HAVE BIT1
7522                                     ;=2
7523                                     ;BUT CONTAINED WHAT IS
7524                                     ;GIVEN IN BAD RHBAE
7525 022646                               675:
7526
7527                                     ;NOW MOVE 0 INTO RHBAE (WRITING 0 INTO RHBAE BIT #0)
7528 022646 005077 161256                 CLR      @RHBAE      ;WRITE ZERO INTO RHBAE
7529
7530                                     ;CHECK THAT RHBAE HAS 0
7531 022652 012737 000000 001124           MOV      #0,$GDDAT   ;GET GOOD = 0
7532
7533                                     ;READ RHBAE FOR COMPARISON
7534 022660 017737 161244 001126           MOV      @RHBAE,$BDDAT
7535 022666 023737 001124 001126           CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
7536                                     ;DATA WITH DATA READ FROM
7537                                     ;RHBAE
7538 022674 001401                           BEQ      695
7539 022676 104062                           ERROR    62
7540                                     ;AFTER SETTING "CLR" BIT #5
7541                                     ;IN RHCS2 TO INIT THE RH
7542                                     ;A17 WAS WRITTEN INTO RHCSI
7543                                     ;RHCSI WAS READ
7544                                     ;RHBAE WAS READ
7545                                     ;THEN ZERO WAS WRITTEN
7546                                     ;INTO RHBAE
7547                                     ;ON READING RHBAE
7548                                     ;RHBAE SHOULD HAVE 0
7549                                     ;BUT CONTAINED WHAT IS
7550                                     ;GIVEN IN BAD RHBAE
7551 022700                               695:
7552
7553                                     ;CHECK THAT RHCSI HAS DVA!RDY
7554 022700 012737 004200 001124           MOV      #DVA!RDY,$GDDAT ;GET GOOD = 4200
7555
7556                                     ;READ RHCSI FOR COMPARISON
7557 022706 017737 161146 001126           MOV      @RHCSI,$BDDAT
7558 022714 023737 001124 001126           CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
7559                                     ;DATA WITH DATA READ FROM
7560                                     ;RHCSI
7561 022722 001401                           BEQ      715
7562 022724 104063                           ERROR    63
7563                                     ;AFTER SETTING "CLR" BIT #5
7564                                     ;IN RHCS2 TO INIT THE RH
7565                                     ;A17 WAS WRITTEN INTO RHCSI
7566                                     ;RHCSI WAS READ
7567                                     ;RHBAE WAS READ
7568                                     ;THEN ZERO WAS WRITTEN
7569                                     ;INTO RHBAE
7570                                     ;RHBAE WAS READ
7571                                     ;ON READING RHCSI
7572                                     ;RHCSI SHOULD HAVE DVA!RDY
7573                                     ;=4200
7574                                     ;BUT CONTAINED WHAT IS
7575                                     ;GIVEN IN BAD RHCSI
7576 022726                               715:
    
```

7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632

022726 000004
022730 012706 001000
022734 012737 000025 004656
022742 004737 040306
022746 012777 000100 161104
022754 012737 004300 001124
022762 017737 161072 001126
022770 023737 001124 001126
022776 001401
023000 104064
023002
023002 012737 000100 001124
023010 017737 161116 001126
023016 023737 001124 001126

```
*****  
: *TEST 25 TEST DUPLICATED IE (RHCS1 BIT #6)  
: * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)  
: * MOVE 100 (ONE INTO IE) IN RHCS1  
: * READ RHCS1 TO CONTAIN 300 (BIT #6 AND RDY)  
: * READ RHCS3 TO CONTAIN "100" (BIT #6 HIGH)  
: *  
: * MOVE 0 INTO RHCS3 (WRITING 0 INTO RHCS3 BIT #6)  
: * READ RHCS3 TO CONTAIN 100  
: * READ RHCS1 TO CONTAIN ONLY RDY (BIT #6 IN RHCS1 IS ZERO)  
: *  
*****  
↑ST25: SCOPE  
MOV #STACK.SP ;RESET STACK  
MOV #25, @↑STNM ;SAVE TEST NUMBER  
JSR PC, @CLDISK ;GIVE RH INITIALIZE  
;SETUP UNIT NUMBER  
;CLEAR RHWC AND FUNCTION BITS IN RHCS1  
;WRITE "1" INTO IE IN RHCS1  
MOV #IE, @RHCS1 ;MOVE 100 INTO RHCS1  
;CHECK THAT RHCS1 HAS IE!DVA!RDY  
MOV #IE!DVA!RDY, $GDDAT ;GET GOOD = 4300  
MOV @RHCS1, $BDDAT ;READ RHCS1 FOR COMPARISON  
CMP $GDDAT, $BDDAT ;COMPARE EXPECTED  
;DATA WITH DATA READ FROM  
;RHCS1  
;BRANCH IF GOOD  
BEG 65$  
ERROR 64  
;AFTER SETTING "CLR" BIT #5  
;IN RHCS2 TO INIT THE RH  
;IE WAS WRITTEN INTO RHCS1  
;RHCS1 WAS READ  
;RHCS1 SHOULD HAVE IE!DVA!RDY  
;=4300  
;BUT CONTAINED WHAT IS  
;GIVEN IN BAD RHCS1  
65$:  
;CHECK THAT RHCS3 HAS IE  
MOV #IE, $GDDAT ;GET GOOD = 1  
MOV @RHCS3, $BDDAT ;READ RHCS3 FOR COMPARISON  
CMP $GDDAT, $BDDAT ;COMPARE EXPECTED  
;DATA WITH DATA READ FROM
```



```

7689                                     : INTO RHCS3
7690                                     : RHCS3 WAS READ
7691                                     : ON READING RHCS1
7692                                     : RHCS1 SHOULD HAVE DVA!PDY
7693                                     : =4200
7694                                     : BUT CONTAINED WHAT IS
7695                                     : GIVEN IN BAD RHCS1
7696 023110                               718:
7697
7698
7699
7700 :*****
7701 :*TEST 26          RHCSI PROGRAM ERROR (PGE BIT #10)
7702
7703 :*
7704 :*   CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5
7705 :*   SET UP FOR A 10 WORD WRITE
7706 :*   SET "GO" (RHCSI BIT #0) TWICE IN TWO SUCCESSIVE
7707 :*   INSTRUCTIONS
7708 :*   THIS SHOULD SET SC AND TRE IN RHCS1
7709 :*   AND PGE SHOULD BE SET IN RHCS2
7710 :*   RHCS3, ARE CHECKED
7711 :*   THE NUMBER OF WORDS ARE LARGE ENOUGH SO THAT AFTER
7712 :*   ONE "BIS" INSTRUCTION TO SET "GO" IN RHCS1
7713 :*   THERE IS SUFFICIENT TIME TO GIVE ANOTHER "BIS" INSTRUCTION
7714 :*   TO SET "GO" BEFORE THE FIRST "GO" HAS TIME TO COMPLETE
7715 :*****
7715 023110 000004
7716 023112 012706 CJ1000
7717 023116 012737 000026 004656
7718
7719 023124 004737 040306
7720
7721
7722
7723
7724 023130 012777 177766 160724
7725 023136 012777 004210 160720
7726
7727
7728 023144 012737 177777 004644
7729 023152 004077 161462
7730 023156 004154
7731 023160 052777 000001 160672
7732 023166 000240
7733 023170 000240
7734 023172 000240
7735 023174 052777 000001 160656
7736
7737
7738
7739 023202 012737 144061 001124
7740
7741 023210 017737 160644 001126
7742 023216 023737 001124 001126
7743
7744

```

```

:RESET STACK
:SAVE TEST NUMBER
:GIVE RH INITIALIZE
:SETUP UNIT NUBER
:CLEAR RHWC AND FUNCTION BITS IN RHCSI
:SET UP RH FOR A 10 WORD WRITE
:WORL COUNT REGISTER=10
:BUS ADDRESS REGISTER=WRITBUF
:WRITE 10 WORDS FROM "WRFROM" INTO
:WHAT EVER RH DEVICE IS AVAILABLE
:DO NOT GIVE GO IN COMMAND ROUTINE
:GO TO DO COMMAND
:WRITE DATA
:SET GO
:TWO
:MICROSECOND
:DELAY
:SET GO WITHOUT TIME TO COMPLETE LAST GO
:CHECK THAT RHCSI HAS SC!DVA!TRE!61
:GET GOOD = 144000
:READ RHCSI FOR COMPARISON
:COMPARE EXPECTED
:DATA WITH DATA READ FROM
:RHCSI

```

F12

CERHADO MACY11 3011046 21-DEC-77 13:06 PAGE 149
CERHAD.P11 21-DEC-77 12:48 126

RHCS1 PROGRAM ERROR PGE BIT #10

SEQ C:48

```

7745 023224 001401          BEQ      65$          ;BRANCH IF GOOD
7746 023226 104070          ERROR    0           ;
7747                                     ;AFTER SETTING "CLR" BIT #5
7748                                     ;IN RHCS2 TO INIT THE PH
7749                                     ;AND GIVING TWO SUCCESSIVE
7750                                     ;"GO" WITHOUT GIVING TIME FOR
7751                                     ;THE FIRST TO COMPLETE
7752                                     ;RHCS1 SHOULD HAVE SC!DVA!TRE!S:
7753                                     ;=144000
7754                                     ;BUT CONTAINED WHAT IS
7755                                     ;GIVEN IN BAC RHCS1
7756 023230          65$:
7757                                     ;CHECK THAT RHCS2 HAS PGE!OR
7758 023230 012737 002200 001124  MOV     #PGE!OR,$GDDAT ;GET GOOD = 2000
7759 023236 053737 005010 001124  BIS     UNIT,$GDDAT   ;INCLUDE UNIT NUMBER
7760
7761 023244 017737 160620 001126  MOV     @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARI
7762 023252 023737 001124 001126  CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED
7763                                     ;DATA WITH DATA READ FROM
7764                                     ;RHCS2
7765 023260 001401          BEQ      67$          ;BRANCH IF GOOD
7766 023262 104071          ERROR    71         ;
7767                                     ;AFTER SETTING "CLR" BIT #5
7768                                     ;IN RHCS2 TO INIT THE RH
7769                                     ;AND GIVING TWO SUCCESSIVE
7770                                     ;"GO" WITHOUT GIVING TIME FOR
7771                                     ;THE FIRST TO COMPLETE
7772                                     ;RHCS2 SHOULD HAVE PGE!OR
7773                                     ;=2000
7774                                     ;TOGETHER WITH UNIT NUMBER
7775                                     ;BUT CONTAINED WHAT IS
7776                                     ;GIVEN IN BAD RHCS2
7777 023264          67$:
7778                                     ;CHECK THAT RHCS3 HAS DBL
7779 023264 012737 002000 001124  MOV     #DBL,$GDDAT  ;GET GOOD = 0
7780
7781 023272 017737 160634 001126  MOV     @RHCS3,$BDDAT ;READ RHCS3 FOR COMPAPIS
7782 023300 023737 001124 001126  CMP     $GDDAT,$BDDAT ;COMPARE EXPECTED
7783                                     ;DATA WITH DATA READ FROM
7784                                     ;RHCS3
7785 023306 001401          BEQ      69$          ;BRANCH IF GOOD
7786 02331C 104072          ERROR    72         ;
7787                                     ;AFTER SETTING "CLR" BIT #5
7788                                     ;IN RHCS2 TO INIT THE RH
7789                                     ;AND GIVING TWO SUCCESSIVE
7790                                     ;"GO" WITHOUT GIVING TIME FOR
7791                                     ;THE FIRST TO COMPLETE
7792                                     ;RHCS3 SHOULD HAVE DBL
7793                                     ;BUT CONTAINED WHAT IS
7794                                     ;GIVEN IN BAD RHCS3
7795 023312          69$:
7796
7797
7798
7799
7800
::*****
: *TEST 27          RHCS2 - BUS ADDRESS INHIBIT BIT #3

```

RHCS2 - BUS ADDRESS INHIBIT BIT #3

```

7801      * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR RHCS2 BIT #6
7802      * SET UP FOR A 2 WORD WRITE FROM A BUFFER TAGGED "WRFROM"
7803      * SET BUS ADDRESS INHIBIT RHCS2 BIT #3 "BAI"
7804      * AT THE END OF WRITE CHECK
7805      * RHCS1 TO HAVE ONLY RDY AND COMMAND
7806      * RHCS2 TO HAVE BAI
7807      * RHCS3 TO HAVE 0
7808      * RHBA TO HAVE ADDRESS OF "WRFROM"
7809      * RHBAE AND RHWC TO HAVE ZEROS
7810
7811      * NOW SET UP READ FOR THE SAME DATA WITH
7812      * BAI SET TO READ INTO A BUFFER TAGGED "REINTO"
7813      * AFTER READ CHECK
7814      * RHCS1 TO HAVE ONLY RDY AND COMMAND
7815      * RHCS2 TO HAVE BAI
7816      * RHCS3 TO HAVE 0
7817      * RHBA TO HAVE ADDRESS OF "REINTO"
7818      * RHBAE AND RHWC TO HAVE ZEROS
7819      * DATA IN REINTO BUFFER IS CHECKED WITH DATA IN
7820      * WRFROM BUFFER
7821      * *****
7822      * ST27: SCOPE
7823      * MOV      #STACK SP      ;RESET STACK
7824      * MOV      #27,#STNM     ;SAVE TEST NUMBER
7825
7826      * JSR      PC,#CLDISK    ;GIVE RH INITIALIZE
7827      *          ;SETUP UNIT NUBER
7828      *          ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
7829
7830      * ;SET UP 10 WORD WRITE FROM BUFFER TAGGED "WRFROM"
7831      * ;WITH BAI IN RHCS2
7832      * MOV      #-10,#RHWC    ;WORD COUNT REGISTER=10
7833      * BIS      #BAI,#RHCS2  ;BUS ADDRESS INHIBIT IN RHCS2
7834      * MOV      #WRFROM,#RHBA ;BUS ADDRESS REGISTER=WRFROM
7835      * JSR      RD,#COMND     ;GO TO DO COMMAND
7836      * WRIDAT      ;WRITE DATA
7837
7838
7839      * ;CHECK THAT RHCS1 HAS DVA!RDY!60
7840      * MOV      #DVA!RDY!60,#GDDAT ;GET GOOD = 4252
7841
7842      * MOV      #RHCS1,#BDDAT  ;READ RHCS1 FOR COMPARISON
7843      * CMP      #GDDAT,#BDDAT ;COMPARE EXPECTED
7844      *          ;DATA WITH DATA READ FROM
7845      *          ;RHCS1
7846      * BEQ      65$           ;BRANCH IF GOOD
7847      * ERROR    76
7848
7849      * ;AFTER SETTING "CLR" BIT #5
7850      * ;IN RHCS2 TO INIT THE RH
7851      * ;A 2 WORD WRITE WAS DONE
7852      * ;WITH BAI BIT IN RHCS2 SET
7853      * ;AT END OF WRITE
7854      * ;RHCS1 SHOULD HAVE DVA!RDY!60
7855      * ;=4252
7856      * ;BUT CONTAINED WHAT IS
7857      * ;GIVEN IN BAD RHCS1

```



```

7913                                     ;GIVEN IN BAD RHBA
7914 023520 715:
7915                                     ;CHECK THAT RHBAE HAS 0
7916 023520 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
7917
7918 023526 017737 160376 001126 MOV DRHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
7919 023534 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7920                                     ;DATA WITH DATA READ FROM
7921                                     ;RHBAE
7922 023542 001401 BEQ 735 ;BRANCH IF GOOD
7923 023544 104102 ERROR 102
7924
7925                                     ;AFTER SETTING "CLR" BIT #5
7926                                     ;IN RHCS2 TO INIT THE RH
7927                                     ;A 2 WORD WRITE WAS DONE
7928                                     ;WITH BAI BIT IN RHCS2 SET
7929                                     ;AT END OF WRITE
7930                                     ;RHBAE SHOULD HAVE 0
7931                                     ;BUT CONTAINED WHAT IS
7932                                     ;GIVEN IN BAD RHBAE
7933 023546 735:
7934 023546 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
7935
7936 023554 017737 160302 001126 MOV DRHWC,$BDDAT ;READ RHWC FOR COMPARISON
7937 023562 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
7938                                     ;DATA WITH DATA READ FROM
7939                                     ;RHWC
7940 023570 001401 BEQ 755 ;BRANCH IF GOOD
7941 023572 104103 ERROR 103
7942
7943                                     ;AFTER SETTING "CLR" BIT #5
7944                                     ;IN RHCS2 TO INIT THE RH
7945                                     ;A 2 WORD WRITE WAS DONE
7946                                     ;WITH BAI BIT IN RHCS2 SET
7947                                     ;AT END OF WRITE
7948                                     ;RHWC SHOULD HAVE 0
7949                                     ;BUT CONTAINED WHAT IS
7950 023574 755:
7951
7952
7953
7954
7955 *****
7956 *TEST 30 RHCS2 MOPE BIT #8 AND RHCS3 IPCKO BIT #0
7957 * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5
7958 * SET IPCKO (RHCS3 BIT #0)
7959 * MOVE ALL ZEROS INTO RHDB ONCE
7960 * THIS SHOULD SET TRE SC IN RHCS1
7961 * READ RHDB ONCE THIS SHOULD SET MOPE (RHCS2 BIT #8)
7962 * CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC BIT #15
7963 * "CLR" (RHCS2 BIT #5) IS GIVEN
7964 * ALL ERRORS ARE CLEARED
7965 * CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC
7966 *****
7967 023574 000004 TEST30: SCOPE
7968 023576 012706 001000 MOV #STACK,SP ;RESET STACK

```

```

7969 023602 012737 000030 004656      MOV      #30, @TSTNM      ;SAVE TEST NUMBER
7970
7971 023610 004737 040306              JSP      PC, @CLDISK      ;GIVE RH INITIALIZE
7972                                ;SETUP UNIT NUBER
7973                                ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
7974 023614 012737 000000 004662      MOV      #0, IP          ;IPCK BIT NO. FOR PRINTOUT
7975
7976 023622 052777 000001 160302      BIS      #IPCKO, @RHCS3 ;SET IPCKO IN RHCS3-BIT #0
7977 023630 005037 004210              CLR      WRFROM          ;WRITE ZERO INTO LOCATION
7978 023634 013777 004210 160240      MOV      WRFROM, @RHDB   ;WRITE ZEROS INTO RHDB ONCE
7979
7980                                ;CHECK THAT RHCS1 HAS SC:TR:DV:RDY
7981 023642 012737 144200 001124      MOV      #SC:TR:DV:RDY, $GDDAT ;GET GOOD = 144200
7982
7983 023650 017737 160204 001126      MOV      @RHCS1, $BDDAT  ;READ RHCS1 FOR COMPARISON
7984 023656 023737 001124 001126      CMP      $GDDAT, $BDDAT  ;COMPARE EXPECTED
7985                                ;DATA WITH DATA READ FROM
7986                                ;RHCS1
7987 023664 001401              BEQ      65$             ;BRANCH IF GOOD
7988 023666 104140      ERROR   140
7989
7990                                ;AFTER SETTING "CLR" BIT #5
7991                                ;IN RHCS2 TO INIT THE RH
7992                                ;IPCKO BIT #0 RHCS3 WAS SET
7993                                ;ZEROS WERE MOVED INTO RHDB
7994                                ;RHCS1 SHOULD HAVE SC:TR:DV:RDY
7995                                ;=144200
7996                                ;BUT CONTAINED WHAT IS
7997                                ;GIVEN IN BAD RHCS1
7997 023670              65$:
7998 023670 017737 160206 001200      MOV      @RHDB, $TMP1   ;READ RHDB ONCE
7999
8000
8001                                ;CHECK THAT RHCS3 HAS 1
8002 023676 012737 000001 001124      MOV      #1, $GDDAT     ;GET GOOD = 1
8003
8004 023704 017737 160222 001126      MOV      @RHCS3, $BDDAT ;READ RHCS3 FOR COMPARISON
8005 023712 023737 001124 001126      CMP      $GDDAT, $BDDAT ;COMPARE EXPECTED
8006                                ;DATA WITH DATA READ FROM
8007                                ;RHCS3
8008 023720 001401              BEQ      67$             ;BRANCH IF GOOD
8009 023722 104104      ERROR   104
8010
8011                                ;AFTER SETTING "CLR" BIT #5
8012                                ;IN RHCS2 TO INIT THE RH
8013                                ;IPCKO BIT #0 RHCS3 WAS SET
8014                                ;ZEROS WERE MOVED INTO
8015                                ;RHDB
8016                                ;RHDB WAS READ THEN
8017                                ;RHCS3 SHOULD HAVE 1
8018                                ;=1
8019                                ;BUT CONTAINED WHAT IS
8020                                ;GIVEN IN BAD RHCS3
8020 023724              67$:
8021
8022                                ;CHECK THAT RHCS2 HAS MDPE:IR
8023 023724 012737 000500 001124      MOV      #MDPE:IR, $GDDAT ;GET GOOD = 500
8024 023732 053737 005010 001124      BIS      UNIT, $GDDAT   ;INCLUDE UNIT NUMBER
    
```



```

0137                                     : WAS GIVEN THIS SHOULD
0138                                     : CLEAR ALL ERRORS
0139                                     : RHBA SHOULD HAVE 0
0140                                     : BUT CONTAINED WHAT IS
0141                                     : GIVEN IN BAD RHBA
0142 024122                               77$:
0143                                     : CHECK THAT RHBAE HAS 0
0144 024122 012737 000000 001124         MOV      #0,$GDDAT      ; GET GOOD = 0
0145
0146 024130 017737 157774 001126         MOV      @RHBAE,$BDDAT ; READ RHBAE FOR COMPARISON
0147 024136 023737 001124 001126         CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
0148                                     : DATA WITH DATA READ FROM
0149                                     : RHBAE
0150 024144 001401                         BEQ      79$           ; BRANCH IF GOOD
0151 024146 104112                         ERROR    112
0152
0153                                     : AFTER SETTING "CLR" BIT #5
0154                                     : IN RHCS2 TO INIT THE RH
0155                                     : IPCK0 BIT #0 RHCS3 WAS SET
0156                                     : ZEROS WERE MOVED INTO
0157                                     : RHDB
0158                                     : RHDB WAS READ THEN
0159                                     : AN RH CLEAR (RHCS2 BIT #5)
0160                                     : WAS GIVEN THIS SHOULD
0161                                     : CLEAR ALL ERRORS
0162                                     : RHBAE SHOULD HAVE 0
0163                                     : BUT CONTAINED WHAT IS
0164                                     : GIVEN IN BAD RHBAE
0164 024150                               79$:
0165
0166 024150 012737 000000 001124         MOV      #0,$GDDAT      ; GET GOOD = 0
0167
0168 024156 017737 157700 001126         MOV      @RHWC,$BDDAT  ; READ RHWC FOR COMPARISON
0169 024164 023737 001124 001126         CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
0170                                     : DATA WITH DATA READ FROM
0171                                     : RHWC
0172 024172 001401                         BEQ      81$           ; BRANCH IF GOOD
0173 024174 104113                         ERROR    113
0174
0175                                     : AFTER SETTING "CLR" BIT #5
0176                                     : IN RHCS2 TO INIT THE RH
0177                                     : IPCK0 BIT #0 RHCS3 WAS SET
0178                                     : ZEROS WERE MOVED INTO
0179                                     : RHDB
0180                                     : RHDB WAS READ THEN
0181                                     : AN RH CLEAR (RHCS2 BIT #5)
0182                                     : WAS GIVEN THIS SHOULD
0183                                     : CLEAR ALL ERRORS
0184                                     : RHWC SHOULD HAVE 0
0185                                     : BUT CONTAINED WHAT IS
0186                                     : GIVEN IN BAD RHWC
0186 024176                               81$:
0187
0188
0189
0190
0191                                     : *****
0192                                     : *TEST 31      RHSC2 MDPE BIT #8 AND RHCS3 IPCK1 BIT #1
                                     : *          CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)

```

```

8193      ;*      SET IPCK1 (RHCS3 BIT #1)
8194      ;*      MOVE ALL ZEROS INTO RHDB ONCE
8195      ;*      THIS SHOULD SET TRE SC IN RHCS1
8196      ;*      READ RHDB ONCE THIS SHOULD SET MOPE (RHCS2 BIT #8)
8197      ;*      CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC (BIT #15)
8198      ;*      "CLR" (RHCS2 BIT #5) IS GIVEN
8199      ;*      ALL ERRORS WRE CLEARED
8200      ;*      CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC
8201
8202      ;*****
8203      024176 000004      †ST31: SCOPE
8204      024200 012706 001000      MOV      #STACK,SP      ;RESET STACK
8205      024204 012737 000031 004656      MOV      #31,#†STNM      ;SAVE TEST NUMBER
8206
8207      024212 004737 040306      JSR      PC,#CLDISK      ;GIVE RH INITIALIZE
8208      ;*      SETUP UNIT NUBER
8209      ;*      CLEAR RHWC AND FUNCTION BITS IN RHCS1
8210      024216 012737 000001 004662      MOV      #1,IP      ;IPCK BIT NO. FOR PRINTOUT
8211
8212      024224 052777 000002 157700      BIS      #IPCK1,#RHCS3 ;SET IPCK1 IN RHCS3-BIT #1
8213      024232 005037 004210      CLR      WRFROM      ;WRITE ZERO INTO LOCATION
8214      024236 013777 004210 157636      MOV      WRFROM,#RHDB ;WRITE ZEROS INTO RHDB ONCE
8215
8216      ;CHECK THAT RHCS1 HAS SC:TRE!DVA!RDY
8217      024244 012737 144200 001124      MOV      #SC:TRE!DVA!RDY,$GDDAT ;GET GOOD = 144200
8218
8219      024252 017737 157602 001126      MOV      #RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
8220      024260 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
8221      ;*      DATA WITH DATA READ FROM
8222      ;*      RHCS1
8223      024266 001401      BEQ      65$          ;BRANCH IF GOOD
8224      024270 104140      ERROR    140
8225
8226      ;AFTER SETTING "CLR" BIT #5
8227      ;*      IN RHCS2 TO INIT THE RH
8228      ;*      IPCK1 BIT #1 RHCS3 WAS SET
8229      ;*      ZEROS WERE MOVED INTO RHDB
8230      ;*      RHCS1 SHOULD HAVE SC:TRE!DVA!RDY
8231      ;*      =144200
8232      ;*      BUT CONTAINED WHAT IS
8233      ;*      GIVEN IN BAD RHCS1
8234      024272 017737 157604 001200      65$: MOV      #RHDB,$TMP1 ;READ RHDB ONCE
8235
8236
8237      ;CHECK THAT RHCS3 HAS 2
8238      024300 012737 000002 001124      MOV      #2,$GDDAT ;GET GOOD = 2
8239
8240      024306 017737 157620 001126      MOV      #RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
8241      024314 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
8242      ;*      DATA WITH DATA READ FROM
8243      ;*      RHCS3
8244      024322 001401      BEQ      67$          ;BRANCH IF GOOD
8245      024324 104104      ERROR    104
8246
8247      ;AFTER SETTING "CLR" BIT #5
8248      ;*      IN RHCS2 TO INIT THE RH
8249      ;*      IPCK1 BIT #1 RHCS3 WAS SET

```



```

0305 ;RHCS1 SHOULD HAVE RDY!DVA
0306 ;=4200
0307 ;BUT CONTAINED WHAT IS
0308 ;GIVEN IN BAD RHCS1
0309 024414 ;18: ;CHECK THAT RHCS2 HAS IR
0310 MOV #IR,$GDDAT ;GET GOOD = 100
0311 024414 012737 000100 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
0312 024422 053737 005010 001124
0313
0314 024430 017737 157434 001126 MOV #RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
0315 024436 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
0316 ;DATA WITH DATA READ FROM
0317 ;RHCS2
0318 024444 001401 BEQ 73$ ;BRANCH IF GOOD
0319 024446 104107
0320 ;AFTER SETTING "CLR" BIT #5
0321 ;IN RHCS2 TO INIT THE RH
0322 ;IPCK1 BIT #1 RHCS3 WAS SET
0323 ;ZEROS WERE MOVED INTO
0324 ;RHDB
0325 ;RHDB WAS READ THEN
0326 ;AN RH CLEAR (RHCS2 BIT #5)
0327 ;WAS GIVEN THIS SHOULD
0328 ;CLEAR ALL ERRORS
0329 ;RHCS2 SHOULD HAVE IR
0330 ;=100
0331 ;TOGETHER WITH UNIT NUMBER
0332 ;BUT CONTAINED WHAT IS
0333 ;GIVEN IN BAD RHCS2
0334 024450 ;73$: ;CHECK THAT RHCS3 HAS 0
0335 MOV #0,$GDDAT ;GET GOOD = 0
0336 024450 012737 000000 001124
0337
0338 024456 017737 157450 001126 MOV #RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
0339 024464 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
0340 ;DATA WITH DATA READ FROM
0341 ;RHCS3
0342 024472 001401 BEQ 75$ ;BRANCH IF GOOD
0343 024474 104110
0344 ;AFTER SETTING "CLR" BIT #5
0345 ;IN RHCS2 TO INIT THE RH
0346 ;IPCK1 BIT #1 RHCS3 WAS SET
0347 ;ZEROS WERE MOVED INTO
0348 ;RHDB
0349 ;RHDB WAS READ THEN
0350 ;AN RH CLEAR (RHCS2 BIT #5)
0351 ;WAS GIVEN THIS SHOULD
0352 ;CLEAR ALL ERRORS
0353 ;RHCS3 SHOULD HAVE 0
0354 ;BUT CONTAINED WHAT IS
0355 ;GIVEN IN BAD RHCS3
0356 024476 ;75$: ;CHECK THAT RHBA HAS 0
0357 MOV #0,$GDDAT ;GET GOOD = 0
0358 024476 012737 000000 001124
0359
0360 024504 017737 157354 001126 MOV #RHBA,$BDDAT ;READ RHBA FOR COMPARISON
    
```


0411
0412
0413
0414
0415
0416
0417
0418
0419
0420
0421
0422
0423
0424
0425
0426
0427
0428
0429
0430
0431
0432
0433
0434
0435
0436
0437
0438
0439
0440
0441
0442
0443
0444
0445
0446
0447
0448
0449
0450
0451
0452
0453
0454
0455
0456
0457
0458
0459
0460
0461
0462
0463
0464
0465
0466
0467
0468
0469
0470
0471
0472

024600

81\$:

; WAS GIVEN THIS SHOULD
; CLEAR ALL ERRORS
; RHWC SHOULD HAVE 0
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHWC

*TEST 32 RHSC2 MOPE BIT #8 AND RHCS3 IPCK2 BIT #2
* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5
* SET IPCK2 (RHCS3 BIT #2)
* MOVE ALL ZEROS INTO RHDB TWICE
* THIS SHOULD NOT SET TRE SC IN RHCS1
* READ RHDB ONCE THIS SHOULD SET MOPE (RHCS2 BIT #8,
* CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC (BIT #15)
* READ RHDB A SECOND TIME. CHECK RHCS1, RHCS2, RHCS3
* "CLR" (RHCS2 BIT #5) IS GIVEN
* ALL ERRORS ARE CLEARED
* CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

024600 000004
024602 012706 001000
024606 012737 000032 004656
024614 004737 040306
024620 012737 000002 004662
024626 052777 000004 157276
024634 005037 004210
024640 013777 004210 157234
024646 013777 004210 157226
024654 012737 004200 001124
024662 017737 157172 001126
024670 023737 001124 001126
024676 001401
024700 104140

*S*32: SCOPE
MOV #STACK, SP ; RESET STACK
MOV #32, @#TSTNM ; SAVE TEST NUMBER
JSR PC, @#CLDISK ; GIVE RH INITIALIZE
; SETUP UNIT NUMBER
; CLEAR RHWC AND FUNCTION BITS IN RHCS1
MOV #2, IP ; IPCK BIT NO. FOR PRINTOUT
BIS #IPCK2, @RHCS3 ; SET IPCK2 IN RHCS3-BIT #2
CLR WRFROM ; WRITE ZERO INTO LOCATION
MOV WRFROM, @RHDB ; WRITE ZEROS INTO RHDB ONCE
MOV WRFROM, @RHDB ; WRITE ZERO SECOND TIME
; CHECK THAT RHCS1 HAS DVA!RDY
MOV #DVA!RDY, \$GDDAT ; GET GOOD = 4200
MOV @RHCS1, \$BDDAT ; READ RHCS1 FOR COMPARISON
CMP \$GDDAT, \$BDDAT ; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHCS1
BEQ 65\$; BRANCH IF GOOD
ERROR 140

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; IPCK2 BIT #2 RHCS3 WAS SET
; ZEROS WERE MOVED INTO RHDB
; TWICE
; RHCS1 SHOULD HAVE DVA!RDY
; =4200
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHCS1

024702

65\$:

```

0473 024702 017737 157174 001200      MOV      @RHDB,$TMP1      ;READ RHDB ONCE
0474
0475
0476
0477 024710 012737 000004 001124      ;CHECK THAT RHCS3 HAS 4
MOV      #4,$GDDAT      ;GET GOOD = 4
0478
0479 024716 017737 157210 001126      MOV      @RHCS3,$BDDAT   ;READ RHCS3 FOR COMPARISON
0480 024724 023737 001124 001126      CMP      $GDDAT,$BDDAT   ;COMPARE EXPECTED
                                ;DATA WITH DATA READ FROM
                                ;RHCS3
0481
0482
0483 024732 001401      BEQ      67$             ;BRANCH IF GOOD
0484 024734 104104      ERROR   104
0485
0486
0487
0488
0489
0490
0491
0492
0493
0494
0495 024736      67$:
0496
0497
0498 024736 012737 000700 001124      ;CHECK THAT RHCS2 HAS MDPE!OR!IR
MOV      @MDPE!OR!IR,$GDDAT ;GET GOOD = 700
0499 024744 053737 005010 001124      BIS      UNIT,$GDDAT     ;INCLUDE UNIT NUMBER
0500
0501 024752 017737 157112 001126      MOV      @RHCS2,$BDDAT   ;READ RHCS2 FOR COMPARISON
0502 024760 023737 001124 001126      CMP      $GDDAT,$BDDAT   ;COMPARE EXPECTED
                                ;DATA WITH DATA READ FROM
                                ;RHCS2
0503
0504
0505 024766 001401      BEQ      69$             ;BRANCH IF GOOD
0506 024770 104105      ERROR   105
0507
0508
0509
0510
0511
0512
0513
0514
0515
0516
0517
0518 024772      69$:
0519
0520 024772 017737 157104 001202      MOV      @RHDB,$TMP2     ;READ RHDB SECOND TIME
0521 ;CHECK THAT RHCS1 HAS SC!TRE!DVA!RDY
0522 025000 012737 144200 001124      MOV      #SC!TRE!DVA!RDY,$GDDAT ;GET GOOD = 144200
0523
0524 025006 017737 157046 001126      MOV      @RHCS1,$BDDAT   ;READ RHCS1 FOR COMPARISON
0525 025014 023737 001124 001126      CMP      $GDDAT,$BDDAT   ;COMPARE EXPECTED
                                ;DATA WITH DATA READ FROM
                                ;RHCS1
0526
0527
0528 025022 001401      BEQ      71$             ;BRANCH IF GOOD
    
```

```

8529 025024 104140          ERROR 140
8530
8531
8532
8533
8534
8535
8536
8537
8538
8539
8540 025026          715:
8541
8542 025026 012737 000500 001124      MOV      #MDPE!IR,$GDDAT ;CHECK THAT RHCS2 HAS MDPE!IR
8543 025034 053737 005010 001124      BIS      UNIT,$GDDAT ;GET GOOD = 500
8544
8545 025042 017737 157022 001126      MOV      @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
8546 025050 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
8547
8548
8549 025056 001401          BEQ      735 ;DATA WITH DATA READ FROM
8550 025060 104105          ERROR 105 ;RHCS2
8551
8552
8553
8554
8555
8556
8557
8558
8559
8560
8561
8562 025062          735:
8563
8564 025062 012737 000004 001124      MOV      #4,$GDDAT ;CHECK THAT RHCS3 HAS 4
8565
8566 025070 017737 157036 001126      MOV      @RHCS3,$BDDAT ;GET GOOD = 4
8567 025076 023737 001124 001126      CMP      $GDDAT,$BDDAT ;READ RHCS3 FOR COMPARISON
8568
8569
8570 025104 001401          BEQ      755 ;COMPARE EXPECTED
8571 025106 104104          ERROR 104 ;DATA WITH DATA READ FROM
8572
8573
8574
8575
8576
8577
8578
8579
8580
8581
8582 025110          755:
8583
8584 025110 004737 040306          JSR      PC,#CLDISK ;RHCS3
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;IPCK2 BIT #2 RHCS3 WAS SET
;ZEROS WERE MOVED INTO
;RHOB
;RHOB WAS READ THEN
;RHCS1 SHOULD HAVE SCITRE!DVA!PC
;=144200
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS1
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;IPCK2 BIT #2 RHCS3 WAS SET
;ZEROS WERE MOVED INTO
;RHOB
;RHOB WAS READ THEN
;RHCS2 SHOULD HAVE MDPE!IR
;=500
;TOGETHER WITH UNIT NUMBER
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS2
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;IPCK2 BIT #2 RHCS3 WAS SET
;ZEROS WERE MOVED INTO
;RHOB
;RHOB WAS READ THEN
;RHCS3 SHOULD HAVE 4
;=4
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS3
;GIVE RH INITIALIZE
    
```

H13

CEPHAD0 MACY11 30 1046 21-DEC-77 13:06 PAGE 164
CEPHAD0.F11 21-DEC-77 12:48 732

RHSC2 MDPE BIT #8 AND RHCS3 IPCK2 BIT #2

SEQ 0163

```
0585 :SETUP UNIT NUMBER
0586 :CLEAR RHWC AND FUNCTION BITS IN RHCS1
0587
0588
0589 ;CHECK THAT RHCS1 HAS RDY!DVA
0590 025114 012737 004200 001124 MOV #RDY!DVA,$GDDAT ;GET GOOD = 4200
0591
0592 025122 017737 156732 001126 MOV @RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
0593 025130 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
0594 ;DATA WITH DATA READ FROM
0595 ;RHCS1
0596 025136 001401 BEQ 77$ ;BRANCH IF GOOD
0597 025140 104106 ERROR 106
0598
0599 ;AFTER SETTING "CLR" BIT #5
0600 ;IN RHCS2 TO INIT THE RH
0601 ;IPCK2 BIT #2 RHCS3 WAS SET
0602 ;ZEROS WERE MOVED INTO
0603 ;RHOB
0604 ;RHOB WAS READ THEN
0605 ;AN RH CLEAR (RHCS2 BIT #5
0606 ;WAS GIVEN THIS SHOULD
0607 ;CLEAR ALL ERRORS
0608 ;RHCS1 SHOULD HAVE RDY!DVA
0609 ;=4200
0610 ;BUT CONTAINED WHAT IS
0611 ;GIVEN IN BAD RHCS1
0612
0613 025142 77$:
0614 025142 012737 000100 001124 MOV #IR,$GDDAT ;CHECK THAT RHCS2 HAS IR
0615 025150 053737 005010 001124 BIS UNIT,$GDDAT ;GET GOOD = 100
0616 ;INCLUDE UNIT NUMBER
0617
0618 025156 017737 156706 001126 MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
0619 025164 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
0620 ;DATA WITH DATA READ FROM
0621 ;RHCS2
0622 025172 001401 BEQ 79$ ;BRANCH IF GOOD
0623 025174 104107 ERROR 107
0624
0625 ;AFTER SETTING "CLR" BIT #5
0626 ;IN RHCS2 TO INIT THE RH
0627 ;IPCK2 BIT #2 RHCS3 WAS SET
0628 ;ZEROS WERE MOVED INTO
0629 ;RHOB
0630 ;RHOB WAS READ THEN
0631 ;AN RH CLEAR (RHCS2 BIT #5
0632 ;WAS GIVEN THIS SHOULD
0633 ;CLEAR ALL ERRORS
0634 ;RHCS2 SHOULD HAVE IR
0635 ;=100
0636 ;TOGETHER WITH UNIT NUMBER
0637 ;BUT CONTAINED WHAT IS
0638 ;GIVEN IN BAD RHCS2
0639
0640 025176 79$:
0641 025176 012737 000000 001124 MOV #0,$GDDAT ;CHECK THAT RHCS3 HAS 0
0642 ;GET GOOD = 0
0643
0644 025204 017737 156722 001126 MOV @RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
```



```

8697                                     ; WAS GIVEN THIS SHOULD
8698                                     ; CLEAR ALL ERRORS
8699                                     ; RHBAE SHOULD HAVE 0
8700                                     ; BUT CONTAINED WHAT IS
8701                                     ; GIVEN IN BAD RHBAE
8702 025300                                BSS:
8703                                     ; CHECK THAT RHWC HAS 0
8704 025300 012737 000000 001124          MOV     #0,SGDDAT      ; GET GOOD = 0
8705
8706 025306 017737 156550 001126          MOV     @RHWC,SBDDAT   ; READ RHWC FOR COMPARISON
8707 025314 023737 001124 001126          CMP     SGDDAT,SBDDAT ; COMPARE EXPECTED
8708                                     ; DATA WITH DATA READ FROM
8709                                     ; RHWC
8710 025322 001401                          BEQ     B7$            ; BRANCH IF GOOD
8711 025324 104113                          ERROR   113
8712
8713                                     ; AFTER SETTING "CLR" BIT #5
8714                                     ; IN RHCS2 TO INIT THE PH
8715                                     ; IPCK2 BIT #2 RHCS3 WAS SET
8716                                     ; ZEROS WERE MOVED INTO
8717                                     ; RHDB
8718                                     ; RHDB WAS READ THEN
8719                                     ; AN RH CLEAR (RHCS2 BIT #5
8720                                     ; WAS GIVEN THIS SHOULD
8721                                     ; CLEAR ALL ERRORS
8722                                     ; RHWC SHOULD HAVE 0
8723                                     ; BUT CONTAINED WHAT IS
8724                                     ; GIVEN IN BAD RHWC
8725 025326                                BSS:
8726
8727
8728
8729
8730
8731
8732
8733
8734
8735
8736
8737
8738
8739
8740
8741
8742 025326 000004                          *TEST 33
8743 025330 012706 001000                  ; *****
8744 025334 012737 000033 004656          ; RHSC2 MOPE BIT #8 AND RHCS3 IPCK3 BIT #3
8745                                     ; CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5
8746 025342 004737 040306                  ; SET IPCK3 (RHCS3 BIT #3)
8747                                     ; MOVE ALL ZEROS INTO RHDB TWICE
8748                                     ; THIS SHOULD NOT SET TRE SC IN RHCS1
8749 025346 012737 000003 004662          ; READ RHDB ONCE THIS SHOULD SET MOPE (RHCS2 BIT #8)
8750                                     ; CHECK RHCS1 FOR RDY (BIT #7), TRE (BIT #14), SC (BIT #15)
8751 025354 052777 000010 156550          ; READ RHDB A SECOND TIME. CHECK RHCS1, RHCS2, RHCS3
8752 025362 005037 004210                  ; "CLR" (RHCS2 BIT #5) IS GIVEN
8753                                     ; ALL ERRORS ARE CLEARED
8754                                     ; CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC
8755
8756
8757
8758
8759
8760
8761
8762
8763
8764
8765
8766
8767
8768
8769
8770
8771
8772
8773
8774
8775
8776
8777
8778
8779
8780
8781
8782
8783
8784
8785
8786
8787
8788
8789
8790
8791
8792
8793
8794
8795
8796
8797
8798
8799
8800

```

RHSC2 MOPE BIT #8 AND RHCS3 IPCK3 BIT #3

```

8753 025366 013777 004210 156506 MOV WRFROM,QRHDB ;WRITE ZEROS INTO RHDB ONCE
8754 025374 013777 004210 156500 MOV WRFROM,QRHDB ;WRITE ZERO SECOND TIME
8755
8756 ;CHECK THAT RHCS1 HAS DVA!RDY
8757 025402 012737 004200 001124 MOV #DVA!RDY,$GDDAT ;GET GOOD = 4200
8758
8759 025410 017737 156444 001126 MOV QRHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
8760 025416 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
8761 ;DATA WITH DATA READ FROM
8762 ;RHCS1
8763 025424 001401 BEQ 65$ ;BRANCH IF GOOD
8764 025426 104140 ERROR 140
8765
8766 ;AFTER SETTING "CLR" BIT #5
8767 ;IN RHCS2 TO INIT THE RH
8768 ;IPCK3 BIT #3 RHCS3 WAS SET
8769 ;ZEROS WERE MOVED INTO R1JB
8770 ;TWICE
8771 ;RHCS1 SHOULD HAVE DVA!RDY
8772 ;=4200
8773 ;BUT CONTAINED WHAT IS
8774 ;GIVEN IN BAD RHCS1
8774 025430 65$:
8775 025430 017737 156446 001200 MOV QRHDB,$TMP1 ;READ RHDB ONCE
8776
8777
8778 ;CHECK THAT RHCS3 HAS 10
8779 025436 012737 000010 001124 MOV #10,$GDDAT ;GET GOOD = 10
8780
8781 025444 017737 156462 001126 MOV QRHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
8782 025452 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
8783 ;DATA WITH DATA READ FROM
8784 ;RHCS3
8785 025460 001401 BEQ 67$ ;BRANCH IF GOOD
8786 025462 104104 ERROR 104
8787
8788 ;AFTER SETTING "CLR" BIT #5
8789 ;IN RHCS2 TO INIT THE RH
8790 ;IPCK3 BIT #3 RHCS3 WAS SET
8791 ;ZEROS WERE MOVED INTO
8792 ;RHDB
8793 ;RHDB WAS READ THEN
8794 ;RHCS3 SHOULD HAVE 10
8795 ;=10
8796 ;BUT CONTAINED WHAT IS
8797 ;GIVEN IN BAD RHCS3
8797 025464 67$:
8798
8799
8800 ;CHECK THAT RHCS2 HAS MOPE!OR!IR
8800 025464 012737 000700 001124 MOV #MOPE!OR!IR,$GDDAT ;GET GOOD = 700
8801 025472 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
8802
8803 025500 017737 156364 001126 MOV QRHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
8804 025506 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
8805 ;DATA WITH DATA READ FROM
8806 ;RHCS2
8807 025514 001401 BEQ 69$ ;BRANCH IF GOOD
8808 025516 104105 ERROR 105
    
```



```

8977                                     : WAS GIVEN THIS SHOULD
8978                                     : CLEAR ALL ERRORS
8979                                     : RHBA SHOULD HAVE 0
8980                                     : BUT CONTAINED WHAT IS
8981                                     : GIVEN IN BAD RHBA
8982 026000                               83$:
8983                                     : CHECK THAT RHBAE HAS 0
8984 026000 012737 000000 001124         MOV    #0,$GDDAT      ; GET GOOD = 0
8985
8986 026006 017737 156116 001126         MOV    @RHBAE,$BDDAT ; READ RHBAE FOR COMPARISON
8987 026014 023737 001124 001126         CMP    $GDDAT,$BDDAT ; COMPARE EXPECTED
8988                                     : DATA WITH DATA READ FROM
8989                                     : RHBAE
8990 026022 001401                         BEQ    85$            ; BRANCH IF GOOD
8991 026024 104112                         ERROR  112
8992
8993                                     : AFTER SETTING "CLR" BIT #5
8994                                     : IN RHCS2 TO INIT THE RH
8995                                     : IPCK3 BIT #3 RHCS3 WAS SET
8996                                     : ZEROS WERE MOVED INTO
8997                                     : RHDB
8998                                     : RHDB WAS READ THEN
8999                                     : AN RH CLEAR (RHCS2 BIT #5
9000                                     : WAS GIVEN THIS SHOULD
9001                                     : CLEAR ALL ERRORS
9002                                     : RHBAE SHOULD HAVE 0
9003                                     : BUT CONTAINED WHAT IS
9004                                     : GIVEN IN BAD RHBAE
9005
9006 026026                               85$:
9007                                     : CHECK THAT RHWC HAS 0
9008 026026 012737 000000 001124         MOV    #0,$GDDAT      ; GET GOOD = 0
9009
9010 026034 017737 156022 001126         MOV    @RHWC,$BDDAT  ; READ PHWC FOR COMPARISON
9011 026042 023737 001124 001126         CMP    $GDDAT,$BDDAT ; COMPARE EXPECTED
9012                                     : DATA WITH DATA READ FROM
9013                                     : RHWC
9014 026050 001401                         BEQ    87$            ; BRANCH IF GOOD
9015 026052 104113                         ERROR  113
9016
9017                                     : AFTER SETTING "CLR" BIT #5
9018                                     : IN RHCS2 TO INIT THE RH
9019                                     : IPCK3 BIT #3 RHCS3 WAS SET
9020                                     : ZEROS WERE MOVED INTO
9021                                     : RHDB
9022                                     : RHDB WAS READ THEN
9023                                     : AN RH CLEAR (RHCS2 BIT #5)
9024                                     : WAS GIVEN THIS SHOULD
9025                                     : CLEAR ALL ERRORS
9026                                     : RHWC SHOULD HAVE 0
9027                                     : BUT CONTAINED WHAT IS
9028                                     : GIVEN IN BAD RHWC
9029
9030 026054                               87$:
9031
9032

```

```

:*****
:*TEST 34          RHCS2-WCE BIT #14 AND RHCS3-WCE OW BIT #12

```

```

9033
9034          : * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
9035          : * WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
9036          : * SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
9037          : * ODD WORD BOUNDARY
9038          : * DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
9039          : * ODD WORD BOUNDARY
9040          : * THIS SHOULD SET WCE OW (RHCS3 BIT #12)
9041          : * AND WCE (RHCS2 BIT #14)
9042          : * "CLR" (RHCS2 BIT #5) IS GIVEN
9043          : * ALL ERRORS ARE CLEARED
9044          : * CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC
9045
9046          : *****
9047 026054 000004          †ST34: SCOPE
9048 026056 012706 001000      MOV      #STACK_SP      ;RESET STACK
9049 026062 012737 000034 004656  MOV      #34,#STSTNM    ;SAVE TEST NUMBER
9050
9051 026070 004737 040306      JSR      PC,#CLDISK     ;GIVE RH INITIALIZE
9052                                ;SETUP UNIT NUBER
9053                                ;CLEAR RHWC AND FUNCTION BITS IN RHCS:
9054
9055          ;WRITE ONE WORD OF ALL ONES ON THE DEVICE
9056 026074 012737 177777 004210  MOV      #-1,WRFROM     ;ALL ONES INTO WRITE FROM
9057 026102 012777 177767 155752  MOV      #-9,#RHWC      ;NINE WORD FOR WORD COUNT REGISTER
9058 026110 012777 004210 155746  MOV      #WRFACM,#RHBA  ;WRITE FROM BUFFER INTO BUS ADDRESS
9059
9060 026116 004077 156516      JSR      RD,#COMND      ;GO TO DO COMMAND
9061 026122 004154                                WRIDAT    ;WRITE DATA
9062
9063          ;SET UP FOR WRITE CHECK
9064 026124 005037 003002      CLR      BUFOW          ;WRITE DATA FOR WRITE CHECK ERROR
9065 026130 012777 177767 155724  MOV      #-9,#RHWC      ;NINE WORD FOR WORD COUNT REGISTER
9066 026136 012777 003002 155720  MOV      #BUFOW,#RHBA   ;WRITE FROM BUFFER INTO BUS ADDRESS
9067                                ;FROM ODD WORD BOUNDARY
9068          ;WRITE CHECK ON ODD WORD BOUNDARY
9069 026144 004077 156470      JSR      RD,#COMND      ;GO TO DO COMMAND
9070 026150 004150                                WRCKEK    ;WRITE CHECK DATA
9071
9072
9073
9074
9075          ;CHECK THAT RHCS2 HAS WCE!OR!IR
9076 026152 012737 040300 001124  MOV      #WCE!OR!IR,$GDDAT ;GET GOOD = 40300
9077 026160 053737 005010 001124  BIS      UNIT,$GDDAT     ;INCLUDE UNIT NUMBER
9078
9079 026166 017737 155676 001126  MOV      #RHCS2,$BDDAT   ;READ RHCS2 FOR COMPARISON
9080 026174 023737 001124 001126  CMP      $GDDAT,$BDDAT  ;COMPARE EXPECTED
9081                                ;DATA WITH DATA READ FROM
9082                                ;RHCS2
9083 026202 001401          BEQ      65$            ;BRANCH IF GOOD
9084 026204 104114          ERRCR    114
9085
9086          ;AFTER SETTING "CLR" BIT #5
9087          ;IN RHCS2 TO INIT THE RH
9088          ;ONE WORD OF ALL ONES IS
          ;WRITTEN ON THE DEVICE
    
```

```

9089                                     ; A WRITE CHECK WAS DONE
9090                                     ; ON AN ODD WORD BOUNDARY THEN
9091                                     ; RHCS2 SHOULD HAVE WCE!OR!IP
9092                                     ; =40300
9093                                     ; TOGETHER WITH UNIT NUMBER
9094                                     ; BUT CONTAINED WHAT IS
9095                                     ; GIVEN IN BAD RHCS2
9096 026206                               65$:
9097
9098                                     ; CHECK THAT RHCS3 HAS DBL!WCEOW
9099 026206 012737 012000 001124        MOV    #DBL!WCEOW,$GDDAT ; GET GOOD = 12000
9100
9101 026214 017737 155712 001126        MOV    @RHCS3,$BDDAT ; READ RHCS3 FOR COMPARISON
9102 026222 023737 001124 001126        CMP    $GDDAT,$BDDAT ; COMPARE EXPECTED
9103                                     ; DATA WITH DATA READ FROM
9104                                     ; RHCS3
9105 026230 001401                          BEQ    67$ ; BRANCH IF GOOD
9106 026232 104115                          ERROR  115
9107
9108                                     ; AFTER SETTING "CLR" BIT #5
9109                                     ; IN RHCS2 TO INIT THE RH
9110                                     ; ONE WORD OF ALL ONES IS
9111                                     ; WRITTEN ON THE DEVICE
9112                                     ; A WRITE CHECK WAS DONE
9113                                     ; ON AN ODD WORD BOUNDARY THEN
9114                                     ; RHCS3 SHOULD HAVE DBL!WCEOW
9115                                     ; =12000
9116                                     ; BUT CONTAINED WHAT IS
9117 026234                               67$:
9118
9119
9120 026234 004737 040306                    JSR    PC,@#CLDISK ; GIVE RH INITIALIZE
9121                                     ; SETUP UNIT NUMBER
9122                                     ; CLEAR RHWC AND FUNCTION BITS IN RHCS1
9123
9124
9125                                     ; CHECK THAT RHCS1 HAS RDY!DVA
9126 026240 012737 004200 001124        MOV    #RDY!DVA,$GDDAT ; GET GOOD = 4200
9127
9128 026246 017737 155606 001126        MOV    @RHCS1,$BDDAT ; READ RHCS1 FOR COMPARISON
9129 026254 023737 001124 001126        CMP    $GDDAT,$BDDAT ; COMPARE EXPECTED
9130                                     ; DATA WITH DATA READ FROM
9131                                     ; RHCS1
9132 026262 001401                          BEQ    69$ ; BRANCH IF GOOD
9133 026264 104116                          ERROR  116
9134
9135                                     ; AFTER SETTING "CLR" BIT #5
9136                                     ; IN RHCS2 TO INIT THE RH
9137                                     ; ONE WORD OF ALL ONES IS
9138                                     ; WRITTEN ON THE DEVICE
9139                                     ; A WRITE CHECK WAS DONE
9140                                     ; ON AN ODD WORD BOUNDARY THEN
9141                                     ; AN RH CLEAR (RHCS2 BIT #5)
9142                                     ; WAS GIVEN THIS SHOULD
9143                                     ; CLEAR ALL ERRORS
9144                                     ; RHCS1 SHOULD HAVE RDY!DVA
9145                                     ; =4200
    
```



```

9201 ;RHBA
9202 026372 001401 BEQ 75$ ;BRANCH IF GOOD
9203 026374 104121 EROR 121
9204 ;AFTER SETTING "CLR" BIT #5
9205 ;IN RHCS2 TO INIT THE RH
9206 ;ONE WORD OF ALL ONES IS
9207 ;WRITTEN ON THE DEVICE
9208 ;A WRITE CHECK WAS DONE
9209 ;ON AN ODD WORD BOUNDARY THEN
9210 ;AN RH CLEAR (RHCS2 BIT #5)
9211 ;WAS GIVEN THIS SHOULD
9212 ;CLEAR ALL ERRORS
9213 ;RHBA SHOULD HAVE 0
9214 ;BUT CONTAINED WHAT IS
9215 ;GIVEN IN BAD RHBA
9216 026376 75$:
9217 ;CHECK THAT RHBAE HAS 0
9218 026376 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
9219
9220 026404 017737 155520 001126 MOV $RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
9221 026412 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
9222 ;DATA WITH DATA READ FROM
9223 ;RHBAE
9224 026420 001401 BEQ 77$ ;BRANCH IF GOOD
9225 026422 104122 EROR 122
9226 ;AFTER SETTING "CLR" BIT #5
9227 ;IN RHCS2 TO INIT THE RH
9228 ;ONE WORD OF ALL ONES IS
9229 ;WRITTEN ON THE DEVICE
9230 ;A WRITE CHECK WAS DONE
9231 ;ON AN ODD WORD BOUNDARY THEN
9232 ;AN RH CLEAR (RHCS2 BIT #5)
9233 ;WAS GIVEN THIS SHOULD
9234 ;CLEAR ALL ERRORS
9235 ;RHBAE SHOULD HAVE 0
9236 ;BUT CONTAINED WHAT IS
9237 ;GIVEN IN BAD RHBAE
9238 026424 77$:
9239 ;CHECK THAT RHBAE HAS 0
9240 026424 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
9241
9242 026432 017737 155424 001126 MOV $RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
9243 026440 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
9244 ;DATA WITH DATA READ FROM
9245 ;RHBAE
9246 026446 001401 BEQ 77$ ;BRANCH IF GOOD
9247 026450 104123 EROR 122
9248 ;AFTER SETTING "CLR" BIT #5
9249 ;IN RHCS2 TO INIT THE RH
9250 ;ONE WORD OF ALL ONES IS
9251 ;WRITTEN ON THE DEVICE
9252 ;A WRITE CHECK WAS DONE
9253 ;ON AN ODD WORD BOUNDARY THEN
9254 ;AN RH CLEAR (RHCS2 BIT #5)
9255 ;WAS GIVEN THIS SHOULD
9256 ;CLEAR ALL ERRORS

```


G14

```

9257
9258
9259
9260 026452
9261
9262
9263
9264
9265
9266
9267
9268
9269
9270
9271
9272
9273
9274
9275
9276
9277
9278
9279
9280 026452 000004
9281 026454 012706 001000
9282 026460 012737 000035 004656
9283
9284 026466 004737 040306
9285
9286
9287
9288
9289 026472 012737 177777 004210
9290 026500 012777 177767 155354
9291 026506 012777 004210 155350
9292
9293 026514 004077 156120
9294 026520 004154
9295
9296
9297 026522 005037 003002
9298 026526 012777 177767 155326
9299 026534 012777 003002 155322
9300
9301
9302 026542 004077 156072
9303 026546 004150
9304
9305
9306
9307
9308
9309 026550 012737 040300 001124
9310 026556 053737 005010 001124
9311
9312 026564 017737 155300 001126
    
```

:RHWC SHOULD HAVE 0
 :BUT CONTAINED WHAT IS
 :GIVEN IN BAD RHWC

-95:

```

*****
*TEST 35      RHCS2-WCE BIT #14 AND RHCS3-WCE OW BIT #12
*****
*
* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
* WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
* SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
* ODD WORD BOUNDARY
* DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
* ODD WORD BOUNDARY
* THIS SHOULD SET WCE OW (RHCS3 BIT #12)
* AND WCE (RHCS2 BIT #14)
* "CLR" (RHCS2 BIT #5) IS GIVEN
* ALL ERRORS ARE CLEARED
* CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC
*****
*TEST35:  SCOPE
          MOV      #STACK,SP      ;RESET STACK
          MOV      #35, @#1STNM   ;SAVE TEST NUMBER
          JSR      PC, @#CLDISK   ;GIVE RH INITIALIZE
                                   ;SETUP UNIT NUBER
                                   ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
          ;WRITE ONE WORD OF ALL ONES ON THE DEVICE
          MOV      #-1, WRFROM     ;ALL ONES INTO WRITE FROM
          MOV      #-9, @RHWC     ;NINE WORD FOR WORD COUNT REGISTER
          MOV      #WRFROM, @RHBA ;WRITE FROM BUFFER INTO BUS ADDRESS
          JSR      RD, @COMND      ;GO TO DO COMMAND
          WRIDAT                    ;WRITE DATA
          ;SET UP FOR WRITE CHECK
          CLR      BUFOW          ;WRITE DATA FOR WRITE CHECK ERROR
          MOV      #-9, @RHWC     ;NINE WORD FOR WORD COUNT REGISTER
          MOV      #BUFOW, @RHBA  ;WRITE FROM BUFFER INTO BUS ADDRESS
                                   ;FROM ODD WORD BOUNDARY
          ;WRITE CHECK ON ODD WORD BOUNDARY
          JSR      RD, @COMND      ;GO TO DO COMMAND
          WRCHK                    ;WRITE CHECK DATA
          ;CHECK THAT RHCS2 HAS WCE!OR!IR
          MOV      #WCE!OR!IR, $GDDAT ;GET GOOD = 40300
          BIS      UNIT, $GDDAT    ;INCLUDE UNIT NUMBER
          MOV      @RHCS2, $BDDAT  ;READ RHCS2 FOR COMPARISON
    
```



```

933  : ONE WORD OF ALL ONES IS
934  : WRITTEN ON THE DEVICE
935  : A WRITE CHECK WAS DONE
936  : ON AN ODD WORD BOUNDARY THEN
937  : AN RH CLEAR (RHCS2 BIT #5)
938  : WAS GIVEN THIS SHOULD
939  : CLEAR ALL ERRORS
940  : RHCS1 SHOULD HAVE RDY!DVA
941  : =4200
942  : BUT CONTAINED WHAT IS
943  : GIVEN IN BAD RHCS1
9380 026664 695:
9381 : CHECK THAT RHCS2 HAS IR
9382 026664 012737 000100 001124 MOV #IR,$GDDAT ; GET GOOD = 100
9383 026672 053737 005010 001124 BIS UNIT,$GDDAT ; INCLUDE UNIT NUMBER
9384
9385 026700 017737 155164 001126 MOV @RHCS2,$BDDAT ; READ RHCS2 FOR COMPARISON
9386 026706 023737 001124 001126 CMP $GDDAT,$BDDAT ; COMPARE EXPECTED
9387 : DATA WITH DATA READ FROM
9388 : RHCS2
9389 : BRANCH IF GOOD
9390 026714 001401 BEQ 715
9391 026716 104117 ERROR 117
9392
9393 : AFTER SETTING "CLR" BIT #5
9394 : IN RHCS2 TO INIT THE RH
9395 : ONE WORD OF ALL ONES IS
9396 : WRITTEN ON THE DEVICE
9397 : A WRITE CHECK WAS DONE
9398 : ON AN ODD WORD BOUNDARY THEN
9399 : AN RH CLEAR (RHCS2 BIT #5)
9400 : WAS GIVEN THIS SHOULD
9401 : CLEAR ALL ERRORS
9402 : RHCS2 SHOULD HAVE IR
9403 : =100
9404 : TOGETHER WITH UNIT NUMBER
9405 : BUT CONTAINED WHAT IS
9406 : GIVEN IN BAD RHCS2
9407 026720 715:
9408 : CHECK THAT RHCS3 HAS 0
9409 026720 012737 000000 001124 MOV #0,$GDDAT ; GET GOOD = 0
9410 026726 017737 155200 001126 MOV @RHCS3,$BDDAT ; READ RHCS3 FOR COMPARISON
9411 026734 023737 001124 001126 CMP $GDDAT,$BDDAT ; COMPARE EXPECTED
9412 : DATA WITH DATA READ FROM
9413 : RHCS3
9414 : BRANCH IF GOOD
9415 026742 001401 BEQ 735
9416 026744 104120 ERROR 120
9417
9418 : AFTER SETTING "CLR" BIT #5
9419 : IN RHCS2 TO INIT THE RH
9420 : ONE WORD OF ALL ONES IS
9421 : WRITTEN ON THE DEVICE
9422 : A WRITE CHECK WAS DONE
9423 : ON AN ODD WORD BOUNDARY THEN
9424 : AN RH CLEAR (RHCS2 BIT #5)
9425 : WAS GIVEN THIS SHOULD
9426 : CLEAR ALL ERRORS
9427 : RHCS3 SHOULD HAVE 0

```

```

9425                                     ;BUT CONTAINED WHAT IS
9426                                     ;GIVEN IN BAD RHCS3
9427 026746                               73$:
9428                                     ;CHECK THAT RHBA HAS 0
9429 026746 012737 000000 001124      MOV      #0,$GDDAT      ;GET GOOD = 0
9430
9431 026754 017737 155104 001.26      MOV      @RHBA,$BDDAT  ;READ RHBA FOR COMPARISON
9432 026762 023737 001124 001.26      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
9433                                     ;DATA WITH DATA READ FROM
9434                                     ;RHBA
9435 026770 001401                          BEQ      75$           ;BRANCH IF GOOD
9436 026772 104121                          ERROR    121
9437
9438                                     ;AFTER SETTING "CLR" BIT #5
9439                                     ;IN RHCS2 TO INIT THE RH
9440                                     ;ONE WORD OF ALL ONES IS
9441                                     ;WRITTEN ON THE DEVICE
9442                                     ;A WRITE CHECK WAS DONE
9443                                     ;ON AN ODD WORD BOUNDARY THEN
9444                                     ;AN RH CLEAR (RHCS2 BIT #5)
9445                                     ;WAS GIVEN THIS SHOULD
9446                                     ;CLEAR ALL ERRORS
9447                                     ;RHBA SHOULD HAVE 0
9448                                     ;BUT CONTAINED WHAT IS
9449                                     ;GIVEN IN BAD RHBA
9450 026774                               75$:
9451 026774 012737 000000 001124      MOV      #0,$GDDAT      ;GET GOOD = 0
9452
9453 027002 017737 155122 001126      MOV      @RHBAE,$BDDAT ;READ RHBAE FOR COMPARISON
9454 027010 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
9455                                     ;DATA WITH DATA READ FROM
9456                                     ;RHBAE
9457 027016 001401                          BEQ      77$           ;BRANCH IF GOOD
9458 027020 104122                          ERROR    122
9459
9460                                     ;AFTER SETTING "CLR" BIT #5
9461                                     ;IN RHCS2 TO INIT THE RH
9462                                     ;ONE WORD OF ALL ONES IS
9463                                     ;WRITTEN ON THE DEVICE
9464                                     ;A WRITE CHECK WAS DONE
9465                                     ;ON AN ODD WORD BOUNDARY THEN
9466                                     ;AN RH CLEAR (RHCS2 BIT #5)
9467                                     ;WAS GIVEN THIS SHOULD
9468                                     ;CLEAR ALL ERRORS
9469                                     ;RHBAE SHOULD HAVE 0
9470                                     ;BUT CONTAINED WHAT IS
9471                                     ;GIVEN IN BAD RHBAE
9472 027022                               77$:
9473 027022 012737 000000 001124      MOV      #0,$GDDAT      ;GET GOOD = 0
9474
9475 027030 017737 155026 001126      MOV      @RHWC,$BDDAT  ;READ RHWC FOR COMPARISON
9476 027036 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
9477                                     ;DATA WITH DATA READ FROM
9478                                     ;RHWC
9479 027044 001401                          BEQ      79$           ;BRANCH IF GOOD
9480 027046 104123                          ERROR    123

```

9481
9482
9483
9484
9485
9486
9487
9488
9489
9490
9491
9492
9493 027050
9494
9495
9496
9497
9498
9499
9500
9501
9502
9503
9504
9505
9506
9507
9508
9509
9510
9511
9512
9513 027050 000004
9514 027052 012706 001000
9515 027056 012737 000036 004656
9516
9517 027064 004737 040306
9518
9519
9520
9521
9522 027070 012737 177777 004210
9523 027076 012777 177767 154756
9524 027104 012777 004210 154752
9525
9526 027112 004077 155522
9527 027116 004154
9528
9529
9530 027120 005037 003000
9531 027124 012777 177767 154730
9532 027132 012777 003000 154724
9533
9534
9535 027140 004077 155474
9536 027144 004150

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; ONE WORD OF ALL ONES IS
; WRITTEN ON THE DEVICE
; A WRITE CHECK WAS DONE
; ON AN ODD WORD BOUNDARY THEN
; AN RH CLEAR (RHCS2 BIT #5
; WAS GIVEN THIS SHOULD
; CLEAR ALL ERRORS
; RHWC SHOULD HAVE 0
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHWC

95:

*TEST 36 RHCS2-WCE BIT #14 AND RHCS3-WCE EW BIT #11

* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
* WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
* SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
* EVEN WORD BOUNDARY
* DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
* EVEN WORD BOUNDARY
* THIS SHOULD SET WCE EW (RHCS3 BIT #11)
* AND WCE (RHCS2 BIT #14)
* "CLR" (RHCS2 BIT #5) IS GIVEN
* ALL ERRORS ARE CLEARED
* CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

†ST36: SCOPE
MOV #STACK, SP ;RESET STACK
MOV #36, @†STNM ;SAVE TEST NUMBER
JSR PC, @CLDISK ;GIVE RH INITIALIZE
;SETUP UNIT NUBER
;CLEAR RHWC AND FUNCTION BITS IN RHCS1
;WRITE ONE WORD OF ALL ONES ON THE DEVICE
MOV #-1, WRFROM ;ALL ONES INTO WRITE FROM
MOV #-9, @RHWC ;NINE WORD FOR WORD COUNT REGISTER
MOV #WRFROM, @RHBA ;WRITE FROM BUFFER INTO BUS ADDRESS
JSR RD, @COMND ;GO TO DO COMMAND
WRIDAT ;WRITE DATA
;SET UP FOR WRITE CHECK
CLR BUFEW ;WRITE DATA FOR WRITE CHECK ERROR
MOV #-9, @RHWC ;NINE WORD FOR WORD COUNT REGISTER
MOV #BUFEW, @RHBA ;WRITE FROM BUFFER INTO BUS ADDRESS
;FROM EVEN WORD BOUNDARY
;WRITE CHECK ON EVEN WORD BOUNDARY
JSR RD, @COMND ;GO TO DO COMMAND
WRCHK ;WRITE CHECK DATA

```

9537
9538
9539
9540
9541 ;CHECK THAT RHCS2 HAS WCE!OR!IR
9542 027146 012737 040300 001124 MOV #WCE!OR!IR,$GDDAT ;GET GOOD = 40300
9543 027154 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
9544
9545 027162 017737 154702 001126 MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
9546 027170 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
9547 ;DATA WITH DATA READ FROM
9548 ;RHCS2
9549 027176 001401 BEQ 65$ ;BRANCH IF GOOD
9550 027200 104114 ERROR 114
9551 ;AFTER SETTING "CLR" BIT #5
9552 ;IN RHCS2 TO INIT THE RH
9553 ;ONE WORD OF ALL ONES IS
9554 ;WRITTEN ON THE DEVICE
9555 ;A WRITE CHECK WAS DONE
9556 ;ON AN EVEN WORD BOUNDARY THEN
9557 ;RHCS2 SHOULD HAVE WCE!OR!IR
9558 ;=40300
9559 ;TOGETHER WITH UNIT NUMBER
9560 ;BUT CONTAINED WHAT IS
9561 ;GIVEN IN BAD RHCS2
9562 027202 65$:
9563
9564 ;CHECK THAT RHCS3 HAS DBL!WCEEW
9565 027202 012737 006000 001124 MOV #DBL!WCEEW,$GDDAT ;GET GOOD = 6000
9566
9567 027210 017737 154716 001126 MOV @RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
9568 027216 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
9569 ;DATA WITH DATA READ FROM
9570 ;RHCS3
9571 027224 001401 BEQ 67$ ;BRANCH IF GOOD
9572 027226 104115 ERROR 115
9573 ;AFTER SETTING "CLR" BIT #5
9574 ;IN RHCS2 TO INIT THE RH
9575 ;ONE WORD OF ALL ONES IS
9576 ;WRITTEN ON THE DEVICE
9577 ;A WRITE CHECK WAS DONE
9578 ;ON AN EVEN WORD BOUNDARY THEN
9579 ;RHCS3 SHOULD HAVE DBL!WCEEW
9580 ;=6000
9581 ;BUT CONTAINED WHAT IS
9582 ;GIVEN IN BAD RHCS3
9583 027230 67$:
9584
9585
9586 027230 004737 040306 JSR PC,@CLDISK ;GIVE RH INITIALIZE
9587 ;SETUP UNIT NUMBER
9588 ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
9589
9590
9591 ;CHECK THAT RHCS1 HAS RDY!DVA
9592 027234 012737 004200 001124 MOV #RDY!DVA,$GDDAT ;GET GOOD = 4200
    
```



```

9649                                     : IN RHCS2 TO INIT THE RH
9650                                     : ONE WORD OF ALL ONES IS
9651                                     : WRITTEN ON THE DEVICE
9652                                     : A WRITE CHECK WAS DONE
9653                                     : ON AN EVEN WORD BOUNDARY THEN
9654                                     : AN RH CLEAR (RHCS2 BIT #5)
9655                                     : WAS GIVEN THIS SHOULD
9656                                     : CLEAR ALL ERRORS
9657                                     : RHCS3 SHOULD HAVE 0
9658                                     : BUT CONTAINED WHAT IS
9659                                     : GIVEN IN BAD RHCS3
9660 027344                               735:
9661                                     : CHECK THAT RHBA HAS 0
9662 027344 012737 000000 001124          MOV      #0,$GDDAT      ; GET GOOD = 0
9663
9664 027352 017737 154506 001126          MOV      @RHBA,$BDDAT   ; READ RHBA FOR COMPARISON
9665 027360 023737 001124 001126          CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
9666                                     : DATA WITH DATA READ FROM
9667                                     : RHBA
9668 027366 001401                          BEQ      755            ; BRANCH IF GOOD
9669 027370 104121                          ERROR    121
9670
9671                                     : AFTER SETTING "CLR" BIT #5
9672                                     : IN RHCS2 TO INIT THE RH
9673                                     : ONE WORD OF ALL ONES IS
9674                                     : WRITTEN ON THE DEVICE
9675                                     : A WRITE CHECK WAS DONE
9676                                     : ON AN EVEN WORD BOUNDARY THEN
9677                                     : AN RH CLEAR (RHCS2 BIT #5)
9678                                     : WAS GIVEN THIS SHOULD
9679                                     : CLEAR ALL ERRORS
9680                                     : RHBA SHOULD HAVE 0
9681                                     : BUT CONTAINED WHAT IS
9682                                     : GIVEN IN BAD RHBA
9683 027372                               755:
9684 027372 012737 000000 001124          MOV      #0,$GDDAT      ; GET GOOD = 0
9685
9686 027400 017737 154524 001126          MOV      @RHBAE,$BDDAT ; READ RHBAE FOR COMPARISON
9687 027406 023737 001124 001126          CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
9688                                     : DATA WITH DATA READ FROM
9689                                     : RHBAE
9690 027414 001401                          BEQ      775            ; BRANCH IF GOOD
9691 027416 104122                          ERROR    122
9692
9693                                     : AFTER SETTING "CLR" BIT #5
9694                                     : IN RHCS2 TO INIT THE RH
9695                                     : ONE WORD OF ALL ONES IS
9696                                     : WRITTEN ON THE DEVICE
9697                                     : A WRITE CHECK WAS DONE
9698                                     : ON AN EVEN WORD BOUNDARY THEN
9699                                     : AN RH CLEAR (RHCS2 BIT #5)
9700                                     : WAS GIVEN THIS SHOULD
9701                                     : CLEAR ALL ERRORS
9702                                     : RHBAE SHOULD HAVE 0
9703                                     : BUT CONTAINED WHAT IS
9704 027420                               775:
    
```



```

9705
9706 027420 012737 000000 001124
9707
9708 027426 017737 154430 001126
9709 027434 023737 001124 001126
9710
9711
9712 027442 001401
9713 027444 104123
9714
9715
9716
9717
9718
9719
9720
9721
9722
9723
9724
9725
9726 027446
9727
9728
9729
9730
9731
9732
9733
9734
9735
9736
9737
9738
9739
9740
9741
9742
9743
9744
9745
9746 027446 000004
9747 027450 012706 001000
9748 027454 012737 000037 004656
9749
9750 027462 004737 040306
9751
9752
9753
9754
9755 027466 012737 177777 004210
9756 027474 012777 177767 154360
9757 027502 012777 004210 154354
9758
9759 027510 004077 155124
9760 027514 004154

```

```

;CHECK THAT RHWC HAS 0
MOV #0,$GDDAT ;GET GOOD = 0
;READ RHWC FOR COMPARISON
MOV @RHWC,$BDDAT ;COMPARE EXPECTED
CMP $GDDAT,$BDDAT ;DATA WITH DATA READ FROM
;RHWC
;BRANCH IF GOOD
BEG 795
ERROR 123
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;ONE WORD OF ALL ONES IS
;WRITTEN ON THE DEVICE
;A WRITE CHECK WAS DONE
;ON AN EVEN WORD BOUNDARY. THE
;AN RH CLEAR (RHCS2 BIT #5)
;WAS GIVEN THIS SHOULD
;CLEAR ALL ERRORS
;RHWC SHOULD HAVE 0
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHWC

```

```

*****
;TEST 37 RHCS2-WCE BIT #14 AND RHCS3-WCE EW BIT #11
;
; CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
; WRITE NINE WORD OF ALL NINES ON TO THE DEVICE
; SET UP TO DO WRITE CHECK ON THAT WORD FROM AN
; EVEN WORD BOUNDARY
; DO A ONE WORD WRITE CHECK WITH RHBA FROM AN
; EVEN WORD BOUNDARY
; THIS SHOULD SET WCE EW (RHCS3 BIT #11)
; AND WCE (RHCS2 BIT #14)
; "CLR" (RHCS2 BIT #5) IS GIVEN
; ALL ERRORS ARE CLEARED
; CHECK RHCS1, RHCS2, RHCS3, RHBA, RHBAE, RHWC

```

```

*****
;TEST37: SCOPE
MOV #STACK,SP ;RESET STACK
MOV #37,@#1STNM ;SAVE TEST NUMBER
JSR PC,@#CLDISK ;GIVE RH INITIALIZE
;SETUP UNIT NUBER
;CLEAR RHWC AND FUNCTION BITS IN RHCS1
;WRITE ONE WORD OF ALL ONES ON THE DEVICE
MOV #-1,WRFROM ;ALL ONES INTO WRITE FROM
MOV #-9,@RHWC ;NINE WORD FOR WORD COUNT REGISTER
MOV #WRFROM,@RHBA ;WRITE FROM BUFFER INTO BUS ADDRESS
JSR PC,@COMND ;GO TO DO COMMAND
WRIDAT ;WRITE DATA

```

```

9761
9762
9763 027516 005037 003000
9764 027522 012777 177767 154332
9765 027530 012777 003000 154326
9766
9767
9768 027536 004077 155076
9769 027542 004150
9770
9771
9772
9773
9774
9775 027544 012737 040300 001124
9776 027552 053737 005010 001124
9777
9778 027560 017737 154304 001126
9779 027566 023737 001124 001126
9780
9781
9782 027574 001401
9783 027576 104114
9784
9785
9786
9787
9788
9789
9790
9791
9792
9793
9794
9795 027600
9796
9797
9798 027600 012737 006000 001124
9799
9800 027606 017737 154320 001126
9801 027614 023737 001124 001126
9802
9803
9804 027622 001401
9805 027624 104115
9806
9807
9808
9809
9810
9811
9812
9813
9814
9815
9816 027626
    
```

```

;SET UP FOR WRITE CHECK
CLR      BUFEW
MOV      #9, 2RHWC
MOV      #BUFEW, 2RHBA
;WRITE DATA FOR WRITE CHECK ERPOP
;NINE WORD FOR WORD COUNT REGISTER
;WRITE FROM BUFFER INTO BUS ADDRESS
;FROM EVEN WORD BOUNDARY
;WRITE CHECK ON EVEN WORD BOUNDARY
JSR      RD, 2COMND
WRCHEK
;GO TO DO COMMAND
;WRITE CHECK DATA

;CHECK THAT RHCS2 HAS WCE!OR!IR
MOV      #WCE!OR!IR, $GDDAT
BIS      UNIT, $GDDAT
;GET GOOD = 40300
;INCLUDE UNIT NUMBER

MOV      2RHCS2, $BDDAT
CMP      $GDDAT, $BDDAT
;READ RHCS2 FOR COMPARISON
;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHCS2
BEQ      65$
ERROR    114
;BRANCH IF GOOD

;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;ONE WORD OF ALL ONES IS
;WRITTEN ON THE DEVICE
;A WRITE CHECK WAS DONE
;ON AN EVEN WORD BOUNDARY THEN
;RHCS2 SHOULD HAVE WCE!OR!IR
;=40300
;TOGETHER WITH UNIT NUMBER
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS2

;CHECK THAT RHCS3 HAS DBL!WCEEW
MOV      #DBL!WCEEW, $GDDAT
;GET GOOD = 6000

MOV      2RHCS3, $BDDAT
CMP      $GDDAT, $BDDAT
;READ RHCS3 FOR COMPARISON
;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHCS3
BEQ      67$
ERROR    115
;BRANCH IF GOOD

;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;ONE WORD OF ALL ONES IS
;WRITTEN ON THE DEVICE
;A WRITE CHECK WAS DONE
;ON AN EVEN WORD BOUNDARY THEN
;RHCS3 SHOULD HAVE DBL!WCEEW
;=6000
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS3
    
```

```

9817
9818
9819 027626 004737 040306 JSR PC, @MCLEISK ;GIVE RH INITIALIZE
9820 ;SETUP UNIT NUBER
9821 ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
9822
9823
9824 ;CHECK THAT RHCS1 HAS RDY!DVA
9825 027632 012737 004200 001124 MOV #RDY!DVA, $GDDAT ;GET GOOD = 4200
9826
9827 027640 017737 154214 001126 MOV @RHCS1, $BDDAT ;READ RHCS1 FOR COMPARISON
9828 027646 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
9829 ;DATA WITH DATA READ FROM
9830 ;RHCS1
9831 027654 001401 BEQ 69$ ;BRANCH IF GOOD
9832 027656 104116 ERROR 116
9833
9834 ;AFTER SETTING "CLR" BIT #5
9835 ;IN RHCS2 TO INIT THE RH
9836 ;ONE WORD OF ALL ONES IS
9837 ;WRITTEN ON THE DEVICE
9838 ;A WRITE CHECK WAS DONE
9839 ;ON AN EVEN WORD BOUNDARY THEN
9840 ;AN RH CLEAR (RHCS2 BIT #5)
9841 ;WAS GIVEN THIS SHOULD
9842 ;CLEAR ALL ERRORS
9843 ;RHCS1 SHOULD HAVE RDY!DVA
9844 ;=4200
9845 ;BUT CONTAINED WHAT IS
9846 ;GIVEN IN BAD RHCS1
9847 027660 69$:
9848 027660 012737 000100 001124 ;CHECK THAT RHCS2 HAS IR
9849 027666 053737 005010 001124 MOV #IR, $GDDAT ;GET GOOD = 100
9850 BIS UNIT, $GDDAT ;INCLUDE UNIT NUMBER
9851 027674 017737 154170 001126 MOV @RHCS2, $BDDAT ;READ RHCS2 FOR COMPARISON
9852 027702 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
9853 ;DATA WITH DATA READ FROM
9854 ;RHCS2
9855 027710 001401 BEQ 71$ ;BRANCH IF GOOD
9856 027712 104117 ERROR 117
9857
9858 ;AFTER SETTING "CLR" BIT #5
9859 ;IN RHCS2 TO INIT THE RH
9860 ;ONE WORD OF ALL ONES IS
9861 ;WRITTEN ON THE DEVICE
9862 ;A WRITE CHECK WAS DONE
9863 ;ON AN EVEN WORD BOUNDARY THEN
9864 ;AN RH CLEAR (RHCS2 BIT #5)
9865 ;WAS GIVEN THIS SHOULD
9866 ;CLEAR ALL ERRORS
9867 ;RHCS2 SHOULD HAVE IR
9868 ;=100
9869 ;TOGETHER WITH UNIT NUMBER
9870 ;BUT CONTAINED WHAT IS
9871 027714 71$:
9872 ;CHECK THAT RHCS3 HAS C
    
```


G15

```

9985 030070 012702 003000      MOV      #BFEVEN,R2      ; START THREE WORD BUFFER
9986                                ; FROM AN EVEN WORD BOUNDARY
9987 030074 012722 052525      1$:  MOV      #52525,(R2)+   ; MOVE THREE 52525 INTO BUFFER
9988 030100 005301              DEC      R1              ; COUNT
9989 030102 001374              BNE     1$              ; BRANCH IF THREE NOT DONE
9990 030104 012777 003000 153752  MOV      #BFEVEN,DRHBA   ; SET BUS ADDRESS TO START
9991                                ; FROM AN EVEN WORD BOUNDARY
9992 030112 012777 177767 153742  MOV      #-9,DRHWC ; WORD COUNT NINE
9993 030120 004077 154514      JSR     R0,DCOMND       ; GO TO DO COMMAND
9994 030124 004154              WRIDAT                    ; WRITE DATA
9995
9996
9997                                ; CHECK THAT RHCS3 HAS 0
9998 030126 012737 000000 001124      MOV      #0,$GDDAT      ; GET GOOD = 0
9999
10000 030134 017737 153772 001126      MOV      DRHCS3,$BDDAT  ; READ RHCS3 FOR COMPARISON
10001 030142 023737 001124 001126      CMP      $GDDAT,$BDDAT  ; COMPARE EXPECTED
10002                                ; DATA WITH DATA READ FROM
10003                                ; RHCS3
10004 030150 001401              BEQ     65$              ; BRANCH IF GOOD
10005 030152 104124              ERROR   124
10006                                ; AFTER SETTING "CLR" BIT #5
10007                                ; IN RHCS2 TO INIT THE RH
10008                                ; A NINE WORD WRITE FROM AN
10009                                ; EVEN WORD BOUNDARY WAS DONE
10010                                ; THEN
10011                                ; RHCS3 SHOULD HAVE 0
10012                                ; BUT CONTAINED WHAT IS
10013                                ; GIVEN IN BAD RHCS3
10014 030154 042777 000076 153676      65$: BIC      #76,DRHCS1     ; CLEAR FUNCTION BITS
10015 030154 042777 000076 153676      ; CHECK THAT RHCS1 HAS RDY!DVA
10016                                ; RDY!DVA
10017 030162 012737 004200 001124      MOV      #RDY!DVA,$GDDAT ; GET GOOD = 4200
10018
10019 030170 017737 153664 001126      MOV      DRHCS1,$BDDAT  ; READ RHCS1 FOR COMPARISON
10020 030176 023737 001124 001126      CMP      $GDDAT,$BDDAT  ; COMPARE EXPECTED
10021                                ; DATA WITH DATA READ FROM
10022                                ; RHCS1
10023 030204 001401              BEQ     67$              ; BRANCH IF GOOD
10024 030206 104125              ERROR   125
10025                                ; AFTER SETTING "CLR" BIT #5
10026                                ; IN RHCS2 TO INIT THE RH
10027                                ; A NINE WORD WRITE FROM AN
10028                                ; EVEN WORD BOUNDARY WAS DONE
10029                                ; THEN
10030                                ; RHCS1 SHOULD HAVE RDY!DVA
10031                                ; =4200
10032                                ; BUT CONTAINED WHAT IS
10033                                ; GIVEN IN BAD RHCS1
10034 030210 67$:
10035                                ; CHECK THAT RHCS2 HAS IR
10036 030210 012737 000100 001124      MOV      #IR,$GDDAT     ; GET GOOD = 100
10037 030216 053737 005010 001124      BIS      UNIT,$GDDAT    ; INCLUDE UNIT NUMBER
10038
10039 030224 017737 153640 001126      MOV      DRHCS2,$BDDAT  ; READ RHCS2 FOR COMPARISON
10040 030232 023737 001124 001126      CMP      $GDDAT,$BDDAT  ; COMPARE EXPECTED
    
```

H15

```

10041                                     : DATA WITH DATA READ FROM
10042                                     : RHCS2
10043 C30240 001401                         BEQ      69$
10044 030242 104126                         ERRORP   126
10045                                     : AFTER SETTING "CLR" BIT #5
10046                                     : IN RHCS2 TO INIT THE RH
10047                                     : A NINE WORD WRITE FROM AN
10048                                     : EVEN WORD BOUNDARY WAS DONE
10049                                     : THEN
10050                                     : RHCS2 SHOULD HAVE IR
10051                                     : =100
10052                                     : TOGETHER WITH UNIT NUMBER
10053                                     : BUT CONTAINED WHAT IS
10054                                     : GIVEN IN BAD RHCS2
10055 C30244                                 69$:
10056                                     : CHECK THAT RHWC HAS 0
10057 030244 012737 000000 001124           MOV      #0,$GDDAT ; GET GOOD = 0
10058                                     :
10059 030252 017737 153604 001126           MOV      @RHWC,$BDDAT ; READ RHWC FOR COMPARISON
10060 030260 023737 001124 001126           CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
10061                                     : DATA WITH DATA READ FROM
10062                                     : RHWC
10063 030266 001401                         BEQ      71$
10064 030270 104131                         ERRORP   101
10065                                     : AFTER SETTING "CLR" BIT #5
10066                                     : IN RHCS2 TO INIT THE RH
10067                                     : A NINE WORD WRITE FROM AN
10068                                     : EVEN WORD BOUNDARY WAS DONE
10069                                     : THEN
10070                                     : RHWC SHOULD HAVE 0
10071                                     : BUT CONTAINED WHAT IS
10072                                     : GIVEN IN BAD RHWC
10073 C30271                                 71$:
10074
10075
10076
10077
10078
10079
10080
10081
10082
10083
10084
10085
10086
10087
10088 030272 000004                         *ST41: SCOPE
10089 030274 012706 001000
10090 030270 012737 000041 004656           MOV      #STACK,SP ; RESET STACK
10091                                     :
10092 030306 004737 040306                   JSR      PC,@CLDISK ; SAVE TEST NUMBER
10093                                     :
10094                                     : GIVE RH INITIALIZE
10095                                     : SETUP UNIT NUMBER
10096                                     : CLEAR RHWC AND FUNCTION BITS IN RHCS1
10097                                     :
10098                                     : SET UP FOR A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY

```

10097	030312	012701	000003		MOV	#3,R1	:COUNT OF THREE
10098	030316	012702	003000		MOV	#BFEVEN,R2	:START THREE WORD BUFFER
10099							:FROM AN EVEN WORD BOUNDARY
10100	030322	012722	052525	15:	MOV	#52525,R21+	:MOVE THREE 52525 INTO BUFFER
10101	030326	005301			DEC	R1	:COUNT
10102	030330	001374			BNE	15	:BRANCH IF THREE NOT DONE
10103	030332	012777	003000	153524	MOV	#BFEVEN,DRHBA	:SET BUS ADDRESS TO START
10104							:FROM AN EVEN WORD BOUNDARY
10105	030340	012777	177766	153514	MOV	#-10,DRHWC	:WORD COUNT TEN
10106	030346	004077	154266		JSR	RD,DRCOMND	:GO TO DO COMMAND
10107	030352	004154			WRDAT		:WRITE DATA
10108							
10109							
10110							:CHECK THAT RHCS3 HAS DBL
10111	030354	012737	002000	001124	MOV	#DBL,\$GDDAT	:GET GOOD = 2000
10112							
10113	030362	017737	153544	001126	MOV	DRHCS3,\$BDDAT	:READ RHCS3 FOR COMPARISON
10114	030370	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED
10115							:DATA WITH DATA READ FROM
10116							:RHCS3
10117	030376	001401			BEG	655	:BRANCH IF GOOD
10118	030400	104124			ERROR	124	
10119							:AFTER SETTING "CLR" BIT #5
10120							:IN RHCS2 TO INIT THE RH
10121							:A TEN WORD WRITE FROM AN
10122							:EVEN WORD BOUNDARY WAS DONE
10123							:THEN
10124							:RHCS3 SHOULD HAVE DBL
10125							: =2000
10126							:BUT CONTAINED WHAT IS
10127							:GIVEN IN BAD RHCS3
10128	030402						
10129	030402	042777	000076	153450	655:	BIC	#76,DRHCS1
10130							:CLEAR FUNCTION BITS
10131	030410	012737	004200	001124	:CHECK THAT RHCS1 HAS RDY!DVA	MOV	#RDY!DVA,\$GDDAT
10132							:GET GOOD = 4200
10133	030416	017737	153436	001126	MOV	DRHCS1,\$BDDAT	:READ RHCS1 FOR COMPARISON
10134	030424	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED
10135							:DATA WITH DATA READ FROM
10136							:RHCS1
10137	030432	001401			BEG	675	:BRANCH IF GOOD
10138	030434	104125			ERROR	125	
10139							:AFTER SETTING "CLR" BIT #5
10140							:IN RHCS2 TO INIT THE RH
10141							:A TEN WORD WRITE FROM AN
10142							:EVEN WORD BOUNDARY WAS DONE
10143							:THEN
10144							:RHCS1 SHOULD HAVE RDY!DVA
10145							: =4200
10146							:BUT CONTAINED WHAT IS
10147							:GIVEN IN BAD RHCS1
10148	030436				675:		
10149							:CHECK THAT RHCS2 HAS IR
10150	030436	012737	000100	001124	MOV	#IR,\$GDDAT	:GET GOOD = 100
10151	030444	053737	005010	001124	BIS	UNIT,\$GDDAT	:INCLUDE UNIT NUMBER
10152							


```

10153 030452 017737 153412 001126      MOV      2RHCS2,$BDDAT  ;READ RHCS2 FOR COMPARISON
10154 030460 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
10155                                     ;DATA WITH DATA READ FROM
10156                                     ;RHCS2
10157 030466 001401      BEQ      69$           ;BRANCH IF GOOD
10158 030473 104126      ERROR   126
10159                                     ;AFTER SETTING "CLR" BIT #5
10160                                     ;IN RHCS2 TO INIT THE RH
10161                                     ;A TEN WORD WRITE FROM AN
10162                                     ;EVEN WORD BOUNDARY WAS DONE
10163                                     ;THEN
10164                                     ;RHCS2 SHOULD HAVE IR
10165                                     ;=100
10166                                     ;TOGETHER WITH UNIT NUMBR
10167                                     ;BUT CONTAINED WHAT IS
10168                                     ;GIVEN IN BAD RHCS2
10169 030472      69$:
10170                                     ;CHECK THAT RHCW HAS 0
10171 030472 012737 000000 001124      MOV      #0,$GDDAT    ;GET GOOD = 0
10172
10173 030500 017737 153356 001126      MOV      2RHWC,$BDDAT ;READ RHCW FOR COMPARISON
10174 030506 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
10175                                     ;DATA WITH DATA READ FROM
10176                                     ;RHCW
10177 030514 001401      BEQ      71$           ;BRANCH IF GOOD
10178 030516 104131      ERROR   131
10179                                     ;AFTER SETTING "CLR" BIT #5
10180                                     ;IN RHCS2 TO INIT THE RH
10181                                     ;A TEN WORD WRITE FROM AN
10182                                     ;EVEN WORD BOUNDARY WAS DONE
10183                                     ;THEN
10184                                     ;RHCW SHOULD HAVE 0
10185                                     ;BUT CONTAINED WHAT IS
10186                                     ;GIVEN IN BAD RHCW
10187 030520      71$:
10188
10189
10190
10191
10192
10193
10194
10195
10196
10197
10198
10199
10200
10201
10202 030520 000004      *ST42: SCOPE
10203 030522 012706 001000      MOV      #STACK,SP   ;RESET STACK
10204 030526 012737 000042 004656      MOV      #42,2*STNM  ;SAVE TEST NUMBER
10205
10206 030534 004737 040306      JSR      PC,2*CLDISK ;GIVE RH INITIALIZE
10207                                     ;SETUP UNIT NUBER
10208                                     ;CLEAR RHCW AND FUNCTION BITS IN RHCS1

```

K15

```

10209
10210 ;SET UP FOR A TEN WORD WRITE FROM AN ODD WORD BOUNDARY
10211 030540 012701 000003 MOV #3,R1 ;COUNT OF THREE
10212 030544 012702 003002 MOV #BF0DD,R2 ;START THREE WORD BUFFER
10213 ;FROM AN ODD WORD BOUNDARY
10214 030550 012722 052525 15: MOV #52525,(R2)+ ;MOVE THREE 52525 INTO BUFFER
10215 030554 005301 DEC R1 ;COUNT
10216 030556 001374 BNE 15 ;BRANCH IF THREE NOT DONE
10217 030560 012777 003002 153276 MOV #BF0DD,ARHBA ;SET BUS ADDRESS TO START
10218 ;FROM AN ODD WORD BOUNDARY
10219 030566 012777 177766 153266 MOV #-10,ARHWC ;WORD COUNT TEN
10220 030574 004077 154040 JSR R0,ACOMND ;GO TO DO COMMAND
10221 030600 004154 WRIDAT ;WRITE DATA
10222
10223
10224 ;CHECK THAT RHCS3 HAS 0
10225 030602 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
10226
10227 030610 017737 153316 001126 MOV ARHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
10228 030616 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10229 ;DATA WITH DATA READ FROM
10230 ;RHCS3
10231 030624 001401 BEQ 65$ ;BRANCH IF GOOD
10232 030626 104124 ERROR 124
10233 ;AFTER SETTING "CLR" BIT #5
10234 ;IN RHCS2 TO INIT THE RH
10235 ;A TEN WORD WRITE FROM AN
10236 ;ODD WORD BOUNDARY WAS DONE
10237 ;THEN
10238 ;RHCS3 SHOULD HAVE 0
10239 ;BUT CONTAINED WHAT IS
10240 ;GIVEN IN BAD RHCS3
10241 030630 042777 000076 153222 65$: BIC #76,ARHCS1 ;CLEAR FUNCTION BITS
10242 030630 012737 004200 001124 ;CHECK THAT RHCS1 HAS RDY!DVA
10243 MOV #RDY!DVA,$GDDAT ;GET GOOD = 4200
10244 030636 012737 004200 001124
10245
10246 030644 017737 153210 001126 MOV ARHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
10247 030652 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10248 ;DATA WITH DATA READ FROM
10249 ;RHCS1
10250 030660 001401 BEQ 67$ ;BRANCH IF GOOD
10251 030662 104125 ERROR 125
10252 ;AFTER SETTING "CLR" BIT #5
10253 ;IN RHCS2 TO INIT THE RH
10254 ;A TEN WORD WRITE FROM AN
10255 ;ODD WORD BOUNDARY WAS DONE
10256 ;THEN
10257 ;RHCS1 SHOULD HAVE RDY!DVA
10258 ;=4200
10259 ;BUT CONTAINED WHAT IS
10260 ;GIVEN IN BAD RHCS1
10261 030664 053737 000100 001124 67$: ;CHECK THAT RHCS2 HAS IR
10262 MOV #IR,$GDDAT ;GET GOOD = 100
10263 030664 012737 000100 001124
10264 030672 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
  
```

```

10265
10266 030700 017737 153164 001126      MOV      @RHCS2,%BDDAT      ;READ RHCS2 FOR COMPARISON
10267 030706 023737 001124 001126      CMP      %GDDAT,%BDDAT     ;COMPARE EXPECTED
10268                                     ;DATA WITH DATA READ FROM
10269                                     ;RHCS2
10270 030714 001401      BEQ      69$               ;BRANCH IF GOOD
10271 030716 104126      ERROR   126
10272                                     ;AFTER SETTING "CLR" BIT #5
10273                                     ;IN RHCS2 TO INIT THE RH
10274                                     ;A TEN WORD WRITE FROM AN
10275                                     ;ODD WORD BOUNDARY WAS DONE
10276                                     ;THEN
10277                                     ;RHCS2 SHOULD HAVE IR
10278                                     ;=100
10279                                     ;TOGETHER WITH UNIT NUMBER
10280                                     ;BUT CONTAINED WHAT IS
10281                                     ;GIVEN IN BAD RHCS2
10282 030720      69$:
10283                                     ;CHECK THAT RHW C HAS 0
10284 030720 012737 000000 001124      MOV      #0,%GDDAT        ;GET GOOD = 0
10285
10286 030726 017737 153130 001126      MOV      @RHW,%BDDAT      ;READ RHW C FOR COMPARISON
10287 030734 023737 001124 001126      CMP      %GDDAT,%BDDAT     ;COMPARE EXPECTED
10288                                     ;DATA WITH DATA READ FROM
10289                                     ;RHW C
10290 030742 001401      BEQ      71$               ;BRANCH IF GOOD
10291 030744 104131      ERROR   131
10292                                     ;AFTER SETTING "CLR" BIT #5
10293                                     ;IN RHCS2 TO INIT THE RH
10294                                     ;A TEN WORD WRITE FROM AN
10295                                     ;ODD WORD BOUNDARY WAS DONE
10296                                     ;THEN
10297                                     ;RHW C SHOULD HAVE 0
10298                                     ;BUT CONTAINED WHAT IS
10299                                     ;GIVEN IN BAD RHW C
10300 030746      71$:
10301
10302
10303
10304
10305
10306
10307
10308
10309
10310
10311
10312
10313
10314
10315 030746 000004      *ST43: SCOPE
10316 030750 012706 001000      MOV      @STACK,%SP        ;RESET STACK
10317 030754 012737 000043 004656      MOV      #43,@*STNM        ;SAVE TEST NUMBER
10318
10319 030762 004737 040306      JSR      PC,@*CLDISK       ;GIVE RH INITIALIZE
10320                                     ;SETUP UNIT NUMBER

```

```

*****
*TEST 43      TEST DBL (RHCS3 BIT #10) TEST D
*****
*
* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
* SET UP FOR A ELEVEN WORD WRITE FROM AN EVEN WORD BOUNDARY
* DO A ELEVEN WORD WRITE FROM AN EVEN WORD BOUNDARY
* THIS SHOULD NOT SET RHCS3 BIT #10 DBL
* CHECK RHCS3
* CHECK RHCS1, RHCS2, RHBA, RHBAE, RHW C
*****

```

M15

CERHAD MACY11 3001046 21-DEC-77 13:06 PAGE 135
 CERHAD.P11 21-DEC-77 12:48 743

TEST DBL RHCS3 BIT #10) TEST D

SEQ 0194

```

10321                                     ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
10322
10323                                     ;SET UP FOR A ELEVEN WORD WRITE FROM AN EVEN WORD BOUNDARY
10324 030766 012701 000003 MOV #3,R1 ;COUNT OF THREE
10325 030772 012702 003000 MOV #BFEEVEN,R2 ;START THREE WORD BUFFER
10326                                     ;FROM AN EVEN WORD BOUNDARY
10327 030776 012722 052525 15: MOV #52525,(R2)+ ;MOVE THREE 52525 INTO BUFFER
10328 031002 005301 DEC R1 ;COUNT
10329 031004 001374 BNE 15 ;BRANCH IF THREE NOT DONE
10330 031006 012777 003000 153050 MOV #BFEEVEN,ARHBA ;SET BUS ADDRESS TO START
10331                                     ;FROM AN EVEN WORD BOUNDARY
10332 031014 012777 177765 153040 MOV #-11,ARHWC ;WORD COUNT ELEVEN
10333 031022 004077 153612 JSR RD,ACOMND ;GO TO DO COMMAND
10334 031026 004154 WRIDAT ;WRITE DATA
10335
10336
10337                                     ;CHECK THAT RHCS3 HAS 0
10338 031030 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
10339
10340 031036 017737 153070 001126 MOV ARHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
10341 031044 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10342                                     ;DATA WITH DATA READ FROM
10343                                     ;RHCS3
10344 031052 001401 BEQ 65$ ;BRANCH IF GOOD
10345 031054 104124 ERROR 124
10346                                     ;AFTER SETTING "CLR" BIT #5
10347                                     ;IN RHCS2 TO INIT THE RH
10348                                     ;A ELEVEN WORD WRITE FROM AN
10349                                     ;EVEN WORD BOUNDARY WAS DONE
10350                                     ;THEN
10351                                     ;RHCS3 SHOULD HAVE 0
10352                                     ;BUT CONTAINED WHAT IS
10353                                     ;GIVEN IN BAD RHCS3
10354 031056 042777 000076 152774 65$: BIC #76,ARHCS1 ;CLEAR FUNCTION BITS
10355 031056 042777 000076 152774 ;CHECK THAT RHCS1 HAS RDY!DVA
10356 031064 012737 004200 001124 MOV #RDY!DVA,$GDDAT ;GET GOOD = 4200
10357 031072 017737 152762 001126 MOV ARHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
10358 031100 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10359                                     ;DATA WITH DATA READ FROM
10360                                     ;RHCS1
10361                                     ;BRANCH IF GOOD
10362 031106 001401 BEQ 67$
10363 031110 104125 ERROR 125
10364                                     ;AFTER SETTING "CLR" BIT #5
10365                                     ;IN RHCS2 TO INIT THE RH
10366                                     ;A ELEVEN WORD WRITE FROM AN
10367                                     ;EVEN WORD BOUNDARY WAS DONE
10368                                     ;THEN
10369                                     ;RHCS1 SHOULD HAVE RDY!DVA
10370                                     ;=4200
10371                                     ;BUT CONTAINED WHAT IS
10372                                     ;GIVEN IN BAD RHCS1
10373 031112 67$:
10374 031112 012737 000100 001124 ;CHECK THAT RHCS2 HAS 1R
10375 031112 012737 000100 001124 MOV #1R,$GDDAT ;GET GOOD = 100
10376

```

```

10377 031120 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBEF
10378
10379 031126 017737 152736 001126 MOV DRHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
10380 031134 023737 001124 001125 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10381 ;DATA WITH DATA READ FROM
10382 ;RHCS2
10383 031142 001401 BEQ 69$ ;BRANCH IF GOOD
10384 031144 104126 ERROR 126
10385 ;AFTER SETTING "CLR" BIT #5
10386 ;IN RHCS2 TO INIT THE RH
10387 ;A ELEVEN WORD WRITE FROM AN
10388 ;EVEN WORD BOUNDARY WAS DONE
10389 ;THEN
10390 ;RHCS2 SHOULD HAVE IR
10391 ;=100
10392 ;TOGETHER WITH UNIT NUMBE
10393 ;BUT CONTAINED WHAT IS
10394 ;GIVEN IN BAD RHCS2
10395 031146 69$:
10396 ;CHECK THAT RHWC HAS 0
10397 031146 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
10398
10399 031154 017737 152702 001126 MOV DRHWC,$GDDAT ;READ RHWC FOR COMPARISON
10400 031162 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10401 ;DATA WITH DATA READ FROM
10402 ;RHWC
10403 031170 001401 BEQ 71$ ;BRANCH IF GOOD
10404 031172 104131 ERROR 131
10405 ;AFTER SETTING "CLR" BIT #5
10406 ;IN RHCS2 TO INIT THE RH
10407 ;A ELEVEN WORD WRITE FROM
10408 ;EVEN WORD BOUNDARY WAS DONE
10409 ;THEN
10410 ;RHWC SHOULD HAVE 0
10411 ;BUT CONTAINED WHAT IS
10412 ;GIVEN IN BAD RHWC
10413 031174 71$:
10414
10415
10416
10417 ;*****
10418 ;*TEST 44 TEST DBL (RHCS3 BIT #10) TEST E
10419
10420 ;* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
10421 ;* SET UP FOR A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY
10422 ;* WITH BAI IN RHCS2 BIT #3 SET
10423 ;* DO A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY
10424 ;* THIS SHOULD NOT SET RHCS3 BIT #10 DBL
10425 ;* CHECK RHCS3
10426 ;* CHECK RHCS1,RHCS2,RHBA,RHBAE,RHWC
10427
10428 ;*****
10429 031174 000004 ST44: SCOPE
10430 031176 012706 MOV #STACK,SP ;RESET STACK
10431 031202 012737 000044 004656 MOV #44,#STNM ;SAVE TEST NUMBER
10432

```

```

10433 031210 004737 040306          :GIVE RH INITIALIZE
10434          :SETUP UNIT NUMBER
10435          :CLEAR RHWC AND FUNCTION BITS IN RHWC
10436
10437          :SET UP FOR A TEN WORD WRITE FROM AN EVEN WORD BOUNDARY
10438 031214 012701 000003          :COUNT OF THREE
10439 031220 012702 003000          :START THREE WORD BUFFER
10440          :FROM AN EVEN WORD BOUNDARY
10441 031224 012722 052525          15: MOV    #52525, R2+    :MOVE THREE 52525 INTO BUFFER
10442 031230 005301          :COUNT
10443 031232 001374          :BRANCH IF THREE NOT DONE
10444 031234 012777 003000 152622  :SET BUS ADDRESS TO START
10445          :FROM AN EVEN WORD BOUNDARY
10446 031242 012777 177766 152612  :MOV    #-10, #RHWC :WORD COUNT TEN
10447 031250 052777 000010 152612  BIS    #BAI, #RHCS2 :SET BAI IN RHCS2
10448 031256 004077 153356          JSP    RD, #COMMC   :GO TO DO COMMAND
10449 031262 004154          WRIDAT :WRITE DATA
10450
10451          :CHECK THAT RHCS3 HAS 0
10452 031264 012737 000000 001124          MOV    #0, $GDDAT  :GET GOOD = 0
10453
10454 031272 017737 152634 001126          MOV    #RHCS3, $BDDAT :READ RHCS3 FOR COMPARISON
10455 031300 023737 001124 001126          CMP    $GDDAT, $BDDAT :COMPARE EXPECTED
10456          :DATA WITH DATA READ FROM
10457          :RHCS3
10458          :BRANCH IF GOOD
10459 031306 001401          BEQ    65$
10460 031310 104124          ERROR  124
10461          :AFTER SETTING "CLR" BIT #5
10462          :IN RHCS2 TO INIT THE RH
10463          :A TEN WORD WRITE FROM AN
10464          :EVEN WORD BOUNDARY WAS DONE
10465          :WITH BAI IN RHCS2 SET
10466          :THEN
10467          :RHCS3 SHOULD HAVE 0
10468          :BUT CONTAINED WHAT IS
10469          :GIVEN IN BAD RHCS3
10470
10471 031312          65$:
10472 031312 042777 000076 152540          BIC    #76, #RHCS1  :CLEAR FUNCTION BITS
10473          :CHECK THAT RHCS1 HAS RDY!DVA
10474 031320 012737 004200 001124          MOV    #RDY!DVA, $GDDAT :GET GOOD = 4200
10475
10476 031326 017737 152526 001126          MOV    #RHCS1, $BDDAT :READ RHCS1 FOR COMPARISON
10477 031334 023737 001124 001126          CMP    $GDDAT, $BDDAT :COMPARE EXPECTED
10478          :DATA WITH DATA READ FROM
10479          :RHCS1
10480 031342 001401          BEQ    67$
10481 031344 104125          ERROR  125
10482          :AFTER SETTING "CLR" BIT #5
10483          :IN RHCS2 TO INIT THE RH
10484          :A TEN WORD WRITE FROM AN
10485          :EVEN WORD BOUNDARY WAS DONE
10486          :WITH BAI IN RHCS2 SET
10487          :THEN
10488          :RHCS1 SHOULD HAVE RDY!DVA
          :=4200

```

```

10489 ;BUT CONTAINED WHAT IS
10490 ;GIVEN IN BAD RHCS1
10491 031346 675:
10492 ;CHECK THAT RHCS2 HAS IR!BAI
10493 031346 012737 000110 001124 MOV #IR!BAI,$GDDAT ;GET GOOD = 110
10494 031354 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
10495
10496 031362 017737 152502 001126 MOV #RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
10497 031370 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10498 ;DATA WITH DATA READ FROM
10499 ;RHCS2
10500 031376 001401 BEQ 695 ;BRANCH IF GOOD
10501 031400 104126 ERROR 126

```

```

10502 ;AFTER SETTING "CLR" BIT #5
10503 ;IN RHCS2 TO INIT THE RH
10504 ;A TEN WORD WRITE FROM AN
10505 ;EVEN WORD BOUNDARY WAS DONE
10506 ;WITH BAI IN RHCS2 SET
10507 ;THEN
10508 ;RHCS2 SHOULD HAVE IR!BAI
10509 ;=110
10510 ;TOGETHER WITH UNIT NUMBER
10511 ;BUT CONTAINED WHAT IS
10512 ;GIVEN IN BAD RHCS2

```

```

10513 031402 695:
10514 ;CHECK THAT RHWC HAS 0
10515 031402 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
10516
10517 031410 017737 152446 001126 MOV #RHWC,$BDDAT ;READ RHWC FOR COMPARISON
10518 031416 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10519 ;DATA WITH DATA READ FROM
10520 ;RHWC
10521 031424 001401 BEQ 715 ;BRANCH IF GOOD
10522 031426 104131 ERROR 131

```

```

10523 ;AFTER SETTING "CLR" BIT #5
10524 ;IN RHCS2 TO INIT THE RH
10525 ;A TEN WORD WRITE FROM AN
10526 ;EVEN WORD BOUNDARY WAS DONE
10527 ;WITH BAI IN RHCS2 SET
10528 ;THEN
10529 ;RHWC SHOULD HAVE 0
10530 ;BUT CONTAINED WHAT IS
10531 ;GIVEN IN BAD RHWC

```

```

10532 031430 715:
10533
10534
10535 ;*****
10536 ;*TEST 45 TEST DBL (RHCS3 BIT #10) TEST F
10537
10538
10539 ;* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
10540 ;* SET UP FOR A ELEVEN WORD WRITE FROM AN ODD WORD BOUNDARY
10541 ;* DO A ELEVEN WORD WRITE FROM AN ODD WORD BOUNDARY
10542 ;* THIS SHOULD SET RHCS3 BIT #10 DBL
10543 ;* CHECK RHCS3
10544 ;* CHECK RHCS1,RHCS2,RHBA,RHBAE,RHWC

```

```

10545
10546
10547 031430 000004
10548 031432 012706 001000
10549 031436 012737 000045 0C4656
10550
10551 031444 004737 040306
10552
10553
10554
10555
10556 031450 012701 000003
10557 031454 012702 003002
10558
10559 031460 012722 052525
10560 031464 005301
10561 031466 001374
10562 031470 012777 003002 152366
10563
10564 031476 012777 177765 152356
10565 031504 004077 153130
10566 031510 004154
10567
10568
10569
10570 031512 012737 002000 001124
10571
10572 031520 017737 152406 001126
10573 031526 023737 001124 001126
10574
10575
10576 031534 001401
10577 031536 104124
10578
10579
10580
10581
10582
10583
10584
10585
10586
10587 031540
10588 031540 042777 000076 152312
10589
10590 031546 012737 004200 001124
10591
10592 031554 017737 152300 001126
10593 031562 023737 001124 001126
10594
10595
10596 031570 001401
10597 031572 104125
10598
10599
10600

```

```

*****
1545: SCOPE
MOV #STACK,SP ;RESET STACK
MOV #45,#STNM ;SAVE TEST NUMBER
JSR PC,#CLDISK ;GIVE RH INITIALIZE
;SETUP UNIT NUMBER
;CLEAR RHWC AND FUNCTION BITS IN PHCS1
;SET UP FOR A ELEVEN WORD WRITE FROM AN ODD WORD BOUNDARY
MOV #3,R1 ;COUNT OF THREE
MOV #BFODD,R2 ;START THREE WORD BUFFER
;FROM AN ODD WORD BOUNDARY
15: MOV #52525,(R2)+ ;MOVE THREE 52525 INTO BUFFER
DEC R1 ;COUNT
BNE 15 ;BRANCH IF THREE NOT DONE
MOV #BFODD,#RHBA ;SET BUS ADDRESS TO START
;FROM AN ODD WORD BOUNDARY
MOV #-11,#RHWC ;WORD COUNT ELEVEN
JSR R0,#COMND ;GO TO DO COMMAND
WRIDAT ;WRITE DATA
;CHECK THAT RHCS3 HAS DBL
MOV #DBL,#GDDAT ;GET GOOD = 2000
MOV #RHCS3,#BDDAT ;READ RHCS3 FOR COMPARISON
CMP #GDDAT,#BDDAT ;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHCS3
;BRANCH IF GOOD
BEQ 655
ERROR 124
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;A ELEVEN WORD WRITE FROM AN
;ODD WORD BOUNDARY WAS DONE
;THEN
;RHCS3 SHOULD HAVE DBL
;=2000
;BUT CONTAINED WHAT IS
;GIVEN IN BAD RHCS3
655: BIC #76,#RHCS1 ;CLEAR FUNCTION BITS
;CHECK THAT RHCS1 HAS RDY!DVA
MOV #RDY!DVA,#GDDAT ;GET GOOD = 4200
MOV #RHCS1,#BDDAT ;READ RHCS1 FOR COMPARISON
CMP #GDDAT,#BDDAT ;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHCS1
;BRANCH IF GOOD
BEQ 675
ERROR 125
;AFTER SETTING "CLR" BIT #5
;IN RHCS2 TO INIT THE RH
;A ELEVEN WORD WRITE FROM AN

```


E16

```

10601                                     : ODD WORD BOUNDARY WAS DONE
10602                                     : THEN
10603                                     : RHCS1 SHOULD HAVE PDY!DVA
10604                                     : =4200
10605                                     : BUT CONTAINED WHAT IS
10606                                     : GIVEN IN BAD RHCS1
10607 031574                               67$:
10608                                     : CHECK THAT RHCS2 HAS IR
10609 031574 012737 000100 001124          MOV   #IR,$GDDAT      : GET GOOD = 100
10610 031602 053737 005010 001124          BIS   UNIT,$GDDAT    : INCLUDE UNIT NUMBER
10611
10612 031610 017737 152254 001126          MOV   @RHCS2,$BDDAT  : READ RHCS2 FOR COMPARISON
10613 031616 023737 001124 001126          CMP   $GDDAT,$BDDAT : COMPARE EXPECTED
10614                                     : DATA WITH DATA READ FROM
10615                                     : RHCS2
10616 031624 001401                          BEQ   69$             : BRANCH IF GOOD
10617 031626 104126
10618                                     : AFTER SETTING "CLR" BIT #5
10619                                     : IN RHCS2 TO INIT THE RH
10620                                     : A ELEVEN WORD WRITE FROM AN
10621                                     : ODD WORD BOUNDARY WAS DONE
10622                                     : THEN
10623                                     : RHCS2 SHOULD HAVE IR
10624                                     : =100
10625                                     : TOGETHER WITH UNIT NUMBER
10626                                     : BUT CONTAINED WHAT IS
10627                                     : GIVEN IN BAD RHCS2
10628 031630                               69$:
10629                                     : CHECK THAT RHWC HAS 0
10630 031630 012737 000000 001124          MOV   #0,$GDDAT     : GET GOOD = 0
10631
10632 031636 017737 152220 001126          MOV   @RHWC,$BDDAT  : READ RHWC FOR COMPARISON
10633 031644 023737 001124 001126          CMP   $GDDAT,$BDDAT : COMPARE EXPECTED
10634                                     : DATA WITH DATA READ FROM
10635                                     : RHWC
10636 031652 001401                          BEQ   71$             : BRANCH IF GOOD
10637 031654 104131
10638                                     : AFTER SETTING "CLR" BIT #5
10639                                     : IN RHCS2 TO INIT THE RH
10640                                     : A ELEVEN WORD WRITE FROM AN
10641                                     : ODD WORD BOUNDARY WAS DONE
10642                                     : THEN
10643                                     : RHWC SHOULD HAVE 0
10644                                     : BUT CONTAINED WHAT IS
10645                                     : GIVEN IN BAD RHWC
10646 031656                               71$:
10647
10648
10649
10650
10651 :*****
10652 :*TEST 46          TEST DBL (RHCS3 BIT #10) TEST G
10653
10654 :*          CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5.
10655 :*          SET UP FOR A NINE WORD READ FROM AN EVEN WORD BOUNDARY
10656 :*          DO A NINE WORD READ FROM AN EVEN WORD BOUNDARY

```

F16

CERHADD MACY11 30 1046 21-DEC-77 13:06 PAGE 201
 CERHAD.F11 21-DEC-77 13:48 146 TEST DBL (RHCS3 BIT #10) TEST

000 0000

```

10657          : * THIS SHOULD NOT SET RHCS3 BIT #10 DBL
10658          : * CHECK RHCS3
10659          : * CHECK RHCS1,RHCS2,RHBA,RHBAE,RHWC
10660
10661          :*****
10662 031656 000004          †ST46: SCOPE
10663 031660 012706 001000  MOV      #STACK,SP          :RESET STACK
10664 031664 012737 000046 004656  MOV      #46,2†STNM         :SAVE TEST NUMBER
10665
10666 031672 004737 040306          JSR      PC,2#CLDISK        :GIVE RH INITIALIZE
10667                                          :SETUP UNIT NUBER
10668                                          :CLEAR RHWC AND FUNCTION BITS IN RHCS.
10669
10670          :SET UP FOR A NINE WORD READ FROM AN EVEN WORD BOUNDARY
10671 031676 012701 000003  MOV      #3,R1              :COUNT OF THREE
10672 031702 012702 003000  MOV      #BF EVEN,R2        :START THREE WORD BUFFER
10673                                          :FROM AN EVEN WORD BOUNDARY
10674 031706 012722 052525          IS:  MOV      #52525,(R2)+       :MOVE THREE 52525 INTO BUFFER
10675 031712 005301          DEC      R1                  :COUNT
10676 031714 001374          BNE     1$                  :BRANCH IF THREE NOT DONE
10677 031716 012777 003000 152140  MOV      #BF EVEN,2RHBA     :SET BUS ADDRESS TO START
10678                                          :FROM AN EVEN WORD BOUNDARY
10679 031724 012777 177767 152130  MOV      #-9,2RHWC          :WORD COUNT NINE
10680 031732 004077 152702          JSR      RD,2COMND          :GO TO DO COMMAND
10681 031736 004160          READAT                      :READ DATA
10682
10683
10684          :CHECK THAT RHCS3 HAS 0
10685 031740 012737 000000 001124  MOV      #0,$GDDAT          :GET GOOD = 0
10686
10687 031746 017737 152160 001126  MOV      2RHCS3,$BDDAT      :READ RHCS3 FOR COMPARISON
10688 031754 023737 001124 001126  CMP      $GDDAT,$BDDAT      :COMPARE EXPECTED
10689                                          :DATA WITH DATA READ FROM
10690                                          :RHCS3
10691 031762 001401          BEQ     65$                  :BRANCH IF GOOD
10692 031764 104124          ERROR  124
10693
10694          :AFTER SETTING "CLR" BIT #5
10695          :IN RHCS2 TO INIT THE RH
10696          :A NINE WORD READ FROM AN
10697          :EVEN WORD BOUNDARY WAS DONE
10698          :THEN
10699          :RHCS3 SHOULD HAVE 0
10700          :BUT CONTAINED WHAT IS
10701          :GIVEN IN BAD RHCS3
10701 031766          65$: BIC      #76,2RHCS1          :CLEAR FUNCTION BITS
10702 031766 042777 000076 152064  :CHECK THAT RHCS1 HAS RDY!DVA
10703          MOV      #RDY!DVA,$GDDAT :GET GOOD = 4200
10704 031774 012737 004200 001124  MOV      2RHCS1,$BDDAT      :READ RHCS1 FOR COMPARISON
10705          CMP      $GDDAT,$BDDAT  :COMPARE EXPECTED
10706          :DATA WITH DATA READ FROM
10707          :RHCS1
10708          :BRANCH IF GOOD
10709 032016 001401          BEQ     67$                  :BRANCH IF GOOD
10710 032020 104125          ERROR  125
10711
10712          :AFTER SETTING "CLR" BIT #5
  
```

G16

```

10713 : IN RHCS2 TO INIT THE RH
10714 : A NINE WORD READ FROM AN
10715 : EVEN WORD BOUNDARY WAS DONE
10716 : THEN
10717 : RHCS1 SHOULD HAVE RDY!DVA
10718 : =4200
10719 : BUT CONTAINED WHAT IS
10720 : GIVEN IN BAD RHCS1

```

```

10721 032022 67$: :CHECK THAT RHCS2 HAS IR
10722 : MOV #IR,$GDDAT ;GET GOOD = 100
10723 032022 012737 000100 001124 : BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
10724 032030 053737 005010 001124
10725
10726 032036 017737 152026 001126 : MOV #RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
10727 032044 023737 001124 001126 : CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10728 : DATA WITH DATA READ FROM
10729 : RHCS2
10730 032052 001401 : BEQ 69$ ;BRANCH IF GOOD
10731 032054 104126 : ERROR 126

```

```

10732 : AFTER SETTING "CLR" BIT #5
10733 : IN RHCS2 TO INIT THE RH
10734 : A NINE WORD READ FROM AN
10735 : EVEN WORD BOUNDARY WAS DONE
10736 : THEN
10737 : RHCS2 SHOULD HAVE IR
10738 : =100
10739 : TOGETHER WITH UNIT NUMBER
10740 : BUT CONTAINED WHAT IS
10741 : GIVEN IN BAD RHCS2

```

```

10742 032056 69$: :CHECK THAT RHWC HAS 0
10743 : MOV #0,$GDDAT ;GET GOOD = 0
10744 032056 012737 000000 001124
10745
10746 032064 017737 151772 001126 : MOV #RHWC,$BDDAT ;READ RHWC FOR COMPARISON
10747 032072 023737 001124 001126 : CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10748 : DATA WITH DATA READ FROM
10749 : RHWC
10750 032100 001401 : BEQ 71$ ;BRANCH IF GOOD
10751 032102 104131 : ERROR 131

```

```

10752 : AFTER SETTING "CLR" BIT #5
10753 : IN RHCS2 TO INIT THE RH
10754 : A NINE WORD READ FROM AN
10755 : EVEN WORD BOUNDARY WAS DONE
10756 : THEN
10757 : RHWC SHOULD HAVE 0
10758 : BUT CONTAINED WHAT IS
10759 : GIVEN IN BAD RHWC

```

```

10760 032104 71$:
10761
10762
10763
10764 :*****
10765 :*TEST 47 TEST DBL (RHCS3 BIT #10) TEST H
10766
10767 :* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5
10768 :* SET UP FOR A TEN WORD READ FROM AN EVEN WORD BOUNDARY

```

H16

CERHADO MAC111 30 1046 21-DEC-77 13:06 PAGE 203
CERHAD.F11 21-DEC-77 12:48 T47

TEST DBL (RHCS3 BIT #10) TEST H

SEG 0202

```
10769 : DO A TEN WORD READ FROM AN EVEN WORD BOUNDARY
10770 : THIS SHOULD SET RHCS3 BIT #10 DBL
10771 :* CHECK RHCS3
10772 :* CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
10773 :*****
10775 032104 000004 :ST47: SCOPE
10776 032106 012706 001000 MOV #STACK, SP ;RESET STACK
10777 032112 012737 000047 004656 MOV #47, #STNM ;SAVE TEST NUMBER
10778 :
10779 032120 004737 040306 JSR PC, #CLDISK ;GIVE RH INITIALIZE
10780 : ;SETUP UNIT NUMBER
10781 : ;CLEAR RHWC AND FUNCTION BITS IN RHCS1
10782 :
10783 :SET UP FOR A TEN WORD READ FROM AN EVEN WORD BOUNDARY
10784 032124 012701 000003 MOV #3, R1 ;COUNT OF THREE
10785 032130 012702 003000 MOV #BF-EVEN, R2 ;START THREE WORD BUFFER
10786 : ;FROM AN EVEN WORD BOUNDARY
10787 032134 012722 052525 13: MOV #52525, (R2)+ ;MOVE THREE 52525 INTO BUFFER
10788 032140 005301 DEC R1 ;COUNT
10789 032142 001374 BNE 1$ ;BRANCH IF THREE NOT DONE
10790 032144 012777 003000 151712 MOV #BF-EVEN, #RHBA ;SET BUS ADDRESS TO START
10791 : ;FROM AN EVEN WORD BOUNDARY
10792 032152 012777 177766 151702 MOV #-10, #RHWC ;WORD COUNT TEN
10793 032160 004077 152454 JSR R0, #COMND ;GO TO DO COMMAND
10794 032164 004160 READAT ;READ DATA
10795 :
10796 :CHECK THAT RHCS3 HAS DBL
10797 :
10798 032166 012737 002000 001124 MOV #DBL, $GDDAT ;GET GOOD = 2000
10799 :
10800 032174 017737 151732 001126 MOV #RHCS3, $BDDAT ;READ RHCS3 FOR COMPARISON
10801 032202 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
10802 : ;DATA WITH DATA READ FROM
10803 : ;RHCS3
10804 032210 001401 BEQ 65$ ;BRANCH IF GOOD
10805 032212 104124 ERROR 124
10806 : ;AFTER SETTING "CLR" BIT #5
10807 : ;IN RHCS2 TO INIT THE RH
10808 : ;A TEN WORD READ FROM AN
10809 : ;EVEN WORD BOUNDARY WAS DONE
10810 : ;THEN
10811 : ;RHCS3 SHOULD HAVE DBL
10812 : ;=2000
10813 : ;BUT CONTAINED WHAT IS
10814 : ;GIVEN IN BAD RHCS3
10815 032214 042777 000076 151636 65$: BIC #76, #RHCS1 ;CLEAR FUNCTION BITS
10816 032214 :CHECK THAT RHCS1 HAS RDY: DVA
10817 :
10818 032222 012737 004200 001124 MOV #RDY: DVA, $GDDAT ;GET GOOD = 4200
10819 :
10820 032230 017737 151624 001126 MOV #RHCS1, $BDDAT ;READ RHCS1 FOR COMPARISON
10821 032236 023737 001124 001126 CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
10822 : ;DATA WITH DATA READ FROM
10823 : ;RHCS1
10824 032244 001401 BEQ 67$ ;BRANCH IF GOOD
```

```

10825 032246 104125 ERROR 125
10826
10827
10828
10829
10830
10831
10832
10833
10834
10835 032250 675:
10836
10837 032250 012737 000100 001124
10838 032256 053737 005010 001124
10839
10840 032264 017737 151600 001126
10841 032272 023737 001124 001126
10842
10843
10844 032300 001401
10845 032302 104126
10846
10847
10848
10849
10850
10851
10852
10853
10854
10855
10856 032304 695:
10857
10858 032304 012737 000000 001124
10859
10860 032312 017737 151544 001126
10861 032320 023737 001124 001126
10862
10863
10864 032326 001401
10865 032330 104131
10866
10867
10868
10869
10870
10871
10872
10873
10874 032332 715:
10875
10876
10877
10878
10879
10880

```

```

:CHECK THAT RHCS2 HAS IR
MOV #IR,$GDDAT ;GET GOOD = 100
BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHCS2
BEQ 695 ;BRANCH IF GOOD
ERROR 125

```

```

:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:A TEN WORD READ FROM AN
:EVEN WORD BOUNDARY WAS DONE
:THEN
:RHCS1 SHOULD HAVE RDY!DVA
:=4200
: BUT CONTAINED WHAT IS
: GIVEN IN BAD RHCS1

```

```

:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:A TEN WORD READ FROM AN
:EVEN WORD BOUNDARY WAS DONE
:THEN
:RHCS2 SHOULD HAVE IR
:=100
:TOGETHER WITH UNIT NUMBER
: BUT CONTAINED WHAT IS
: GIVEN IN BAD RHCS2

```

```

:CHECK THAT RHWC HAS 0
MOV #0,$GDDAT ;GET GOOD = 0
MOV @RHWC,$BDDAT ;READ RHWC FOR COMPARISON
CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
;DATA WITH DATA READ FROM
;RHWC
BEQ 715 ;BRANCH IF GOOD
ERROR 131

```

```

:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:A TEN WORD READ FROM AN
:EVEN WORD BOUNDARY WAS DONE
:THEN
:RHWC SHOULD HAVE 0
: BUT CONTAINED WHAT IS
: GIVEN IN BAD RHWC

```

```

:*****
:*TEST 50 TEST DBL (RHCS3 BIT #10) TEST I

```

J16

```

10881      : * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR RHCS2 BIT #5
10882      : * SET UP FOR A TEN WORD READ FROM AN ODD WORD BOUNDAR:
10883      : * DO A TEN WORD READ FROM AN ODD WORD BOUNDAR
10884      : * THIS SHOULD NOT SET RHCS3 BIT #10 DBL
10885      : * CHECK RHCS3
10886      : * CHECK RHCS1,RHCS2,RHBA,RHBAE,RHWC
10887
10888      :*****
10889 032332 000004      †ST50: SCOPE
10890 032334 012706 001000      MOV #STACK,SP ;RESET STACK
10891 032340 012737 000050 004657      MOV #50,‡STNM ;SAVE TEST NUMBER
10892
10893 032346 004737 040306      JSR PC,‡CLDISK ;GIVE RH INITIALIZE
10894 ;SETUP UNIT NUBER
10895 ;CLEAR RHWC AND FUNCTION BITS IN RHCS:
10896
10897 ;SET UP FOR A TEN WORD READ FROM AN ODD WORD BOUNDARY
10898 032352 012701 000003      MOV #3,R1 ;COUNT OF THREE
10899 032356 012702 003002      MOV #BF0DD,R2 ;START THREE WORD BUFFER
10900 ;FROM AN ODD WORD BOUNDARY
10901 032362 012722 052525      1$: MOV #52525,(R2)+ ;MOVE THREE 52525 INTO BUFFER
10902 032366 005301 ;COUNT
10903 032370 001374 ;BRANCH IF THREE NOT DONE
10904 032372 012777 003002 151464      MOV #BF0DD,‡RHBA ;SET BUS ADDRESS TO START
10905 ;FROM AN ODD WORD BOUNDAR:
10906 032400 012777 177766 151454      MOV #-10,‡RHWC ;WORD COUNT TEN
10907 032406 004077 152226      JSR R0,‡COMND ;GO TO DO COMMAND
10908 032412 004160      READAT ;READ DATA
10909
10910
10911 ;CHECK THAT RHCS3 HAS 0
10912 C32414 012737 000000 001124      MOV #0,$GDDAT ;GET GOOD = 0
10913
10914 032422 017737 151504 001126      MOV ‡RHCS3,$BDDAT ;READ RHCS3 FOR COMPARISON
10915 032430 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10916 ;DATA WITH DATA READ FROM
10917 ;RHCS3
10918 032436 001401 ;BRANCH IF GOOD
10919 032440 104124      BEQ ERROR 65$
10920 ;AFTER SETTING "CLR" BIT #5
10921 ;IN RHCS2 TO INIT THE RH
10922 ;A TEN WORD READ FROM AN
10923 ;ODD WORD BOUNDARY WAS DONE
10924 ;THEN
10925 ;RHCS3 SHOULD HAVE 0
10926 ;BUT CONTAINED WHAT IS
10927 ;GIVEN IN BAD RHCS3
10928 032442 ;CLEAR FUNCTION BITS
10929 032442 042777 000076 151410      65$: BIC #76,‡RHCS1
10930 ;CHECK THAT RHCS1 HAS RDY!DVA
10931 032450 012737 004200 001124      MOV #RDY!DVA,$GDDAT ;GET GOOD = 4200
10932
10933 032456 017737 151376 001126      MOV ‡RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
10934 032464 023737 001124 001126      CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
10935 ;DATA WITH DATA READ FROM
10936 ;RHCS:

```

K16

```

10937 032472 001401          BEQ      67$      : BRANCH IF GOOD
10938 032474 104125          ERROR    125     :
10939                                     : AFTER SETTING "CLR" BIT #5
10940                                     : IN RHCS2 TO INIT THE RH
10941                                     : A TEN WORD READ FROM AN
10942                                     : ODD WORD BOUNDARY WAS DONE
10943                                     : THEN
10944                                     : RHCS1 SHOULD HAVE RDY/DVA
10945                                     : =4200
10946                                     : BUT CONTAINED WHAT IS
10947                                     : GIVEN IN BAD RHCS1
10948 032476          67$:
10949                                     : CHECK THAT RHCS2 HAS IR
10950 032476 012737 000100 001124  MOV      #IR,$GDDAT : GET GOOD = 100
10951 032504 053737 005010 001124  BIS      UNIT,$GDDAT : INCLUDE UNIT NUMBER
10952                                     :
10953 032512 017737 151352 001126  MOV      @RHCS2,$BDDAT : READ RHCS2 FOR COMPARISON
10954 032520 023737 001124 001126  CMP      $GDDAT,$BDDAT : COMPARE EXPECTED
10955                                     : DATA WITH DATA READ FROM
10956                                     : RHCS2
10957 032526 001401          BEQ      69$      : BRANCH IF GOOD
10958 032530 104126          ERROR    126     :
10959                                     : AFTER SETTING "CLR" BIT #5
10960                                     : IN RHCS2 TO INIT THE RH
10961                                     : A TEN WORD READ FROM AN
10962                                     : ODD WORD BOUNDARY WAS DONE
10963                                     : THEN
10964                                     : RHCS2 SHOULD HAVE IR
10965                                     : =100
10966                                     : TOGETHER WITH UNIT NUMBER
10967                                     : BUT CONTAINED WHAT IS
10968                                     : GIVEN IN BAD RHCS2
10969 032532          69$:
10970                                     : CHECK THAT RHWC HAS 0
10971 032532 012737 000000 001124  MOV      #0,$GDDAT   : GET GOOD = 0
10972                                     :
10973 032540 017737 151316 001126  MOV      @RHWC,$BDDAT : READ RHWC FOR COMPARISON
10974 032546 023737 001124 001126  CMP      $GDDAT,$BDDAT : COMPARE EXPECTED
10975                                     : DATA WITH DATA READ FROM
10976                                     : RHWC
10977 032554 001401          BEQ      71$      : BRANCH IF GOOD
10978 032556 104131          ERROR    131     :
10979                                     : AFTER SETTING "CLR" BIT #5
10980                                     : IN RHCS2 TO INIT THE RH
10981                                     : A TEN WORD READ FROM AN
10982                                     : ODD WORD BOUNDARY WAS DONE
10983                                     : THEN
10984                                     : RHWC SHOULD HAVE 0
10985                                     : BUT CONTAINED WHAT IS
10986                                     : GIVEN IN BAD RHWC
10987 032560          71$:
10988
10989
10990
10991
10992

```

*TEST 51 TEST DBL (RHCS3 BIT #10 TEST I

L16

```

10993
10994
10995
10996
10997
10998
10999
11000
11001
11002 032560 000004
11003 032562 012706 001000
11004 032566 012737 000051 004656
11005
11006 032574 004737 040306
11007
11008
11009
11010
11011 032600 012701 000003
11012 032604 012702 003000
11013
11014 032610 012722 052525
11015 032614 005301
11016 032616 001374
11017 032620 012777 003000 151236
11018
11019 032626 012777 177765 151226
11020 032634 004077 152000
11021 032640 004160
11022
11023
11024
11025 032642 012737 000000 001124
11026
11027 032650 017737 151256 001126
11028 032656 023737 001124 001126
11029
11030
11031 032664 001401
11032 032666 104124
11033
11034
11035
11036
11037
11038
11039
11040
11041 032670
11042 032670 042777 000076 151162
11043
11044 032676 012737 004200 001124
11045
11046 032704 017737 151150 001126
11047 032712 023737 001124 001126
11048

```

```

: * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5
: * SET UP FOR A ELEVEN WORD READ FROM AN EVEN WORD BOUNDARY
: * DO A ELEVEN WORD READ FROM AN EVEN WORD BOUNDARY
: * THIS SHOULD NOT SET RHCS3 BIT #10 DBL
: * CHECK RHCS3
: * CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
: *****
151: SCOPE
MOV #STACK, SP ; RESET STACK
MOV #51, @STNM ; SAVE TEST NUMBER
JSR PC, @CLDISK ; GIVE RH INITIALIZE
; SETUP UNIT NUMBER
; CLEAR RHWC AND FUNCTION BITS IN RHCS:
; SET UP FOR A ELEVEN WORD READ FROM AN EVEN WORD BOUNDARY
MOV #3, R1 ; COUNT OF THREE
MOV #BF-EVEN, R2 ; START THREE WORD BUFFER
; FROM AN EVEN WORD BOUNDARY
15: MOV #52525, (R2)+ ; MOVE THREE 52525 INTO BUFFER
DEC R1 ; COUNT
BNE 15 ; BRANCH IF THREE NOT DONE
MOV #BF-EVEN, @RHBA ; SET BUS ADDRESS TO START
; FROM AN EVEN WORD BOUNDARY
MOV #-11, @RHWC ; WORD COUNT ELEVEN
JSR RD, @COMND ; GO TO DO COMMAND
READAT ; READ DATA
; CHECK THAT RHCS3 HAS 0
MOV #0, $GDDAT ; GET GOOD = 0
MOV @RHCS3, $BDDAT ; READ RHCS3 FOR COMPARISON
CMP $GDDAT, $BDDAT ; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHCS3
; BRANCH IF GOOD
BEG 65$
ERROR 124
; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; A ELEVEN WORD READ FROM AN
; EVEN WORD BOUNDARY WAS DONE
; THEN
; RHCS3 SHOULD HAVE 0
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHCS3
65$: BIC #76, @RHCS1 ; CLEAR FUNCTION BITS
; CHECK THAT RHCS1 HAS RDY!DVA
MOV #RDY!DVA, $GDDAT ; GET GOOD = 4200
MOV @RHCS1, $BDDAT ; READ RHCS1 FOR COMPARISON
CMP $GDDAT, $BDDAT ; COMPARE EXPECTED
; DATA WITH DATA READ FROM

```


M16

```

11049
11050 032720 001401          BEQ      67$          :RHCS1
11051 032722 104125          ERRJR   125          :BRANCH IF GOOD
11052
11053
11054
11055
11056
11057
11058
11059
11060
11061 032724          67$:
11062
11063 032724 012737 000100 001124      :CHECK THAT RHCS2 HAS IR
11064 032732 053737 005010 001124      MOV      #IR,$GDDAT  :GET GOOD = 100
11065
11066 032740 017737 151124 001126      BIS      UNI+,$GDDAT :INCLUDE UNIT NUMBER
11067 032746 023737 001124 001126      MOV      @RHCS2,$BDDAT :READ RHCS2 FOR COMPARISON
11068
11069
11070 032754 001401          CMP      $GDDAT,$BDDAT :COMPARE EXPECTED
11071 032756 104126          :DATA WITH DATA READ FROM
11072
11073
11074
11075
11076
11077
11078
11079
11080
11081
11082 032760          69$:
11083
11084 032760 012737 000000 001124      :CHECK THAT RHWC HAS 0
11085
11086 032766 017737 151070 001126      MOV      #0,$GDDAT   :GET GOOD = 0
11087 032774 023737 001124 001126      MOV      @RHWC,$BDDAT :READ RHWC FOR COMPARISON
11088
11089
11090 033002 001401          CMP      $GDDAT,$BDDAT :COMPARE EXPECTED
11091 033004 104131          :DATA WITH DATA READ FROM
11092
11093
11094
11095
11096
11097
11098
11099
11100 033006          71$:
11101
11102
11103
11104

```

```

:RHCS1
:BRANCH IF GOOD

:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:A ELEVEN WORD READ FROM AN
:EVEN WORD BOUNDARY WAS DONE
:THEN
:RHCS1 SHOULD HAVE RDY!DVA
:=4200
:BUT CONTAINED WHAT IS
:GIVEN IN BAD RHCS1

:CHECK THAT RHCS2 HAS IR
:GET GOOD = 100
:INCLUDE UNIT NUMBER

:READ RHCS2 FOR COMPARISON
:COMPARE EXPECTED
:DATA WITH DATA READ FROM
:RHCS2
:BRANCH IF GOOD

:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:A ELEVEN WORD READ FROM AN
:EVEN WORD BOUNDARY WAS DONE
:THEN
:RHCS2 SHOULD HAVE IR
:=100
:TOGETHER WITH UNIT NUMBER
:BUT CONTAINED WHAT IS
:GIVEN IN BAD RHCS2

:CHECK THAT RHWC HAS 0
:GET GOOD = 0

:READ RHWC FOR COMPARISON
:COMPARE EXPECTED
:DATA WITH DATA READ FROM
:RHWC
:BRANCH IF GOOD

:AFTER SETTING "CLR" BIT #5
:IN RHCS2 TO INIT THE RH
:A ELEVEN WORD READ FROM AN
:EVEN WORD BOUNDARY WAS DONE
:THEN
:RHWC SHOULD HAVE 0
:BUT CONTAINED WHAT IS
:GIVEN IN BAD RHWC

```

::*****

B01

```

11105          ;*TEST 52          TEST DBL (RHCS3 BIT #10) TEST K
11106
11107          ;*          CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5
11108          ;*          SET UP FOR A ELEVEN WORD READ FROM AN ODD WORD BOUNDAR
11109          ;*          DO A ELEVEN WORD READ FROM AN ODD WORD BOUNDAR
11110          ;*          THIS SHOULD SET RHCS3 BIT #10 DBL
11111          ;*          CHECK RHCS3
11112          ;*          CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
11113
11114          ;*****
11115 033006 000004          †T52: SCOPE
11116 033010 012706 001000          MOV      #STACK, SP          ;RESET STACK
11117 033014 012737 000052 004656          MOV      #52, @†STNM          ;SAVE TEST NUMBER
11118
11119 033022 004737 040306          JSR      PC, @#CLDISK          ;GIVE RH INITIALIZE
11120
11121
11122
11123          ;SET UP FOR A ELEVEN WORD READ FROM AN ODD WORD BOUNDAR
11124 033026 012701 000003          MOV      #3, R1          ;COUNT OF THREE
11125 033032 012702 003002          MOV      #BFODD, R2          ;START THREE WORD BUFFER
11126
11127 033036 012722 052525          15:    MOV      #52525, (R2)+          ;MOVE THREE 52525 INTO BUFFER
11128 033042 005301          DEC      R1          ;COUNT
11129 033044 001374          BNE     15          ;BRANCH IF THREE NOT DONE
11130 033046 012777 003002 151010          MOV      #BFODD, @RHBA          ;SET BUS ADDRESS TO START
11131
11132 033054 012777 177765 151000          MOV      #-11, @RHWC          ;WORD COUNT ELEVEN
11133 033062 004077 151552          JSR      RD, @COMND          ;GO TO DO COMMAND
11134 033066 004160          READAT          ;READ DATA
11135
11136
11137          ;CHECK THAT RHCS3 HAS DBL
11138 033070 012737 002000 001124          MOV      #DBL, $GDDAT          ;GET GOOD = 2000
11139
11140 033076 017737 151030 001126          MOV      @RHCS3, $BDDAT          ;READ RHCS3 FOR COMPARISON
11141 033104 023737 001124 001126          CMP     $GDDAT, $BDDAT          ;COMPARE EXPECTED
11142
11143
11144 033112 001401          BEQ     65$          ;BRANCH IF GOOD
11145 033114 104124          ERROR   124
11146
11147          ;AFTER SETTING "CLR" BIT #5
11148          ;IN RHCS2 TO INIT THE RH
11149          ;A ELEVEN WORD READ FROM AN
11150          ;ODD WORD BOUNDARY WAS DONE
11151          ;THEN
11152          ;RHCS3 SHOULD HAVE DBL
11153          ;=2000
11154          ;BUT CONTAINED WHAT IS
11155          ;GIVEN IN BAD RHCS3
11155 033116          65$:  BIC     #76, @RHCS1          ;CLEAR FUNCTION BITS
11156 033116 042777 000076 150734          ;CHECK THAT RHCS1 HAS RDY!DVA
11157          MOV      #RDY!DVA, $GDDAT          ;GET GOOD = 4200
11158 033124 012737 004200 001124          MOV     @RHCS1, $BDDAT          ;READ RHCS1 FOR COMPARISON
11159
11160 033132 017737 150722 001126

```

C01

```

11161 033140 023737 001124 001126      CMP      $GDDAT,$BDDAT      ;COMPARE EXPECTED
11162                                     ;DATA WITH DATA READ FROM
11163                                     ;RHCS1
11164 033146 001401      BEQ      67$                ;BRANCH IF GOOD
11165 033150 104125      ERROR    12$
11166                                     ;AFTER SETTING "CLR" BIT #5
11167                                     ;IN RHCS2 TO INIT THE RH
11168                                     ;A ELEVEN WORD READ FROM AN
11169                                     ;ODD WORD BOUNDARY WAS DONE
11170                                     ;THEN
11171                                     ;RHCS1 SHOULD HAVE RDI!DVA
11172                                     ;=4200
11173                                     ;BUT CONTAINED WHAT IS
11174                                     ;GIVEN IN BAD RHCS1
11175 033152                67$:
11176                                     ;CHECK THAT RHCS2 HAS IR
11177 033152 012737 000100 001124      MOV      #IR,$GDDAT        ;GET GOOD = 100
11178 033160 053737 005010 001124      BIS      UNIT,$GDDAT      ;INCLUDE UNIT NUMBER
11179
11180 033166 017737 150676 001126      MOV      @RHCS2,$BDDAT    ;READ RHCS2 FOR COMPARISON
11181 033174 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;COMPARE EXPECTED
11182                                     ;DATA WITH DATA READ FROM
11183                                     ;RHCS2
11184 033202 001401      BEQ      69$                ;BRANCH IF GOOD
11185 033204 104126      ERROR    12$
11186                                     ;AFTER SETTING "CLR" BIT #5
11187                                     ;IN RHCS2 TO INIT THE RH
11188                                     ;A ELEVEN WORD READ FROM AN
11189                                     ;ODD WORD BOUNDARY WAS DONE
11190                                     ;THEN
11191                                     ;RHCS2 SHOULD HAVE IR
11192                                     ;=100
11193                                     ;TOGETHER WITH UNIT NUMBER
11194                                     ;BUT CONTAINED WHAT IS
11195                                     ;GIVEN IN BAD RHCS2
11196 033206                69$:
11197                                     ;CHECK THAT RHWC HAS 0
11198 033206 012737 000000 001124      MOV      #0,$GDDAT        ;GET GOOD = 0
11199
11200 033214 017737 150642 001126      MOV      @RHWC,$BDDAT    ;READ RHWC FOR COMPARISON
11201 033222 023737 001124 001126      CMP      $GDDAT,$BDDAT    ;COMPARE EXPECTED
11202                                     ;DATA WITH DATA READ FROM
11203                                     ;RHWC
11204 033230 001401      BEQ      71$                ;BRANCH IF GOOD
11205 033232 104131      ERROR    13$
11206                                     ;AFTER SETTING "CLR" BIT #5
11207                                     ;IN RHCS2 TO INIT THE RH
11208                                     ;A ELEVEN WORD READ FROM AN
11209                                     ;ODD WORD BOUNDARY WAS DONE
11210                                     ;THEN
11211                                     ;RHWC SHOULD HAVE 0
11212                                     ;BUT CONTAINED WHAT IS
11213                                     ;GIVEN IN BAD RHWC
11214 033234                71$:
11215
11216

```

001

```

11217
11218
11219
11220
11221
11222
11223
11224
11225
11226
11227
11228
11229
11230 033234 000004
11231 033236 012706 001000
11232 033242 012737 000053 004656
11233
11234 033250 004737 040306
11235
11236
11237
11238
11239 033254 012701 000003
11240 033260 012702 003000
11241
11242 033264 012722 052525 15:
11243 033270 005301
11244 033272 001374
11245 033274 012777 003000 150562
11246
11247 033302 012777 177766 150552
11248 033310 052777 000010 150552
11249 033316 004077 151316
11250 033322 004160
11251
11252
11253
11254 033324 012737 000000 001124
11255
11256 033332 017737 150574 001126
11257 033340 023737 001124 001126
11258
11259
11260 033346 001401
11261 033350 104124
11262
11263
11264
11265
11266
11267
11268
11269
11270
11271 033352
11272 033352 042777 000076 150500 65%:

```

```

*****
TEST 53 TEST DBL (RHCS3 BIT #10) TEST L

* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
* SET UP FOR A TEN WORD READ FROM AN EVEN WORD BOUNDARY
* WITH BAI IN RHCS2 BIT #3 SET
* DO A TEN WORD READ FROM AN EVEN WORD BOUNDARY
* THIS SHOULD NOT SET RHCS3 BIT #10 DBL
* CHECK RHCS3
* CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC

*****
ST53: SCOPE
MOV #STACK, SP ; RESET STACK
MOV #53, #STSTNM ; SAVE TEST NUMBER
JSR PC, #CLDISK ; GIVE RH INITIALIZE
; SETUP UNIT NUMBER
; CLEAR RHWC AND FUNCTION BITS IN RHCS1
; SET UP FOR A TEN WORD READ FROM AN EVEN WORD BOUNDARY
MOV #3, R1 ; COUNT OF THREE
MOV #BFEBVEN, R2 ; START THREE WORD BUFFER
; FROM AN EVEN WORD BOUNDARY
15: MOV #52525, (R2)+ ; MOVE THREE 52525 INTO BUFFER
DEC R1 ; COUNT
BNE 15 ; BRANCH IF THREE NOT DONE
MOV #BFEBVEN, #RHBA ; SET BUS ADDRESS TO START
; FROM AN EVEN WORD BOUNDARY
BIS #BAI, #RHCS2 ; SET BAI IN RHCS2
JSR #0, #COMND ; GO TO DO COMMAND
READAT ; READ DATA

; CHECK THAT RHCS3 HAS 0
MOV #0, #GDDAT ; GET GOOD = 0
MOV #RHCS3, #BDDAT ; READ RHCS3 FOR COMPARISON
CMP #GDDAT, #BDDAT ; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHCS3
; BRANCH IF GOOD
65%:
; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; A TEN WORD READ FROM AN
; EVEN WORD BOUNDARY WAS DONE
; WITH BAI IN RHCS2 SET
; THEN
; RHCS3 SHOULD HAVE 0
; BUT CONTAINED WHAT IS
; GIVEN IN BAD RHCS3
; CLEAR FUNCTION BITS

```

E01

CERHADD MACY11 3011046 21-DEC-77 13:06 PAGE 212
 CERHAD.P11 21-DEC-77 12:48 753

TEST DBL (RHCS3 BIT #10) TEST L

SEG 0211

```

11273 ;CHECK THAT RHCS1 HAS RDY!DVA
11274 033360 012737 004200 001124 MOV #RDY!DVA,$GDDAT ;GET GOOD = 4200
11275
11276 033366 017737 150466 001126 MOV @RHCS1,$BDDAT ;READ RHCS1 FOR COMPARISON
11277 033374 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
11278 ;DATA WITH DATA READ FROM
11279 ;RHCS1
11280 033402 001401 BEQ 67$ ;BRANCH IF GOOD
11281 033404 104125 ERROR 125
11282 ;AFTER SETTING "CLR" BIT #5
11283 ;IN RHCS2 TO INIT THE RH
11284 ;A TEN WORD READ FROM AN
11285 ;EVEN WORD BOUNDARY WAS DONE
11286 ;WITH BAI IN RHCS2 SET
11287 ;THEN
11288 ;RHCS1 SHOULD HAVE RDY!DVA
11289 ;=4200
11290 ;BUT CONTAINED WHAT IS
11291 ;GIVEN IN BAD RHCS1
11292 033406 67$:
11293 ;CHECK THAT RHCS2 HAS IR!BAI
11294 033406 012737 000110 001124 MOV #IR!BAI,$GDDAT ;GET GOOD = 110
11295 033414 053737 005010 001124 BIS UNIT,$GDDAT ;INCLUDE UNIT NUMBER
11296
11297 033422 017737 150442 001126 MOV @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
11298 033430 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
11299 ;DATA WITH DATA READ FROM
11300 ;RHCS2
11301 033436 001401 BEQ 69$ ;BRANCH IF GOOD
11302 033440 104126 ERROR 126
11303 ;AFTER SETTING "CLR" BIT #5
11304 ;IN RHCS2 TO INIT THE RH
11305 ;A TEN WORD READ FROM AN
11306 ;EVEN WORD BOUNDARY WAS DONE
11307 ;WITH BAI IN RHCS2 SET
11308 ;THEN
11309 ;RHCS2 SHOULD HAVE IR!BAI
11310 ;=110
11311 ;TOGETHER WITH UNIT NUMBER
11312 ;BUT CONTAINED WHAT IS
11313 ;GIVEN IN BAD RHCS2
11314 033442 69$:
11315 ;CHECK THAT RHWC HAS 0
11316 033442 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
11317
11318 033450 017737 150406 001126 MOV @RHWC,$BDDAT ;READ RHWC FOR COMPARISON
11319 033456 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
11320 ;DATA WITH DATA READ FROM
11321 ;RHWC
11322 033464 001401 BEQ 71$ ;BRANCH IF GOOD
11323 033466 104131 ERROR 131
11324 ;AFTER SETTING "CLR" BIT #5
11325 ;IN RHCS2 TO INIT THE RH
11326 ;A TEN WORD READ FROM AN
11327 ;EVEN WORD BOUNDARY WAS DONE
11328 ;WITH BAI IN RHCS2 SET
  
```

F01

```

11329                                     ; THEN
11330                                     ; RHWC SHOULD HAVE 0
11331                                     ; BUT CONTAINED WHAT IS
11332                                     ; GIVEN IN BAD RHWC
11333 033470                               13:
11334
11335
11336
11337
11338
11339                                     ;*****
11340                                     ;>TEST 54      TEST DBL (RHCS3 BIT #10) TEST M
11341
11342                                     ;*      CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
11343                                     ;*      SET UP FOR A ONE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
11344                                     ;*      DO A ONE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
11345                                     ;*      THIS SHOULD NOT SET RHCS3 BIT #10 DBL
11346                                     ;*      CHECK RHCS3
11347                                     ;*      CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
11348
11349 033470 000004                               ;*****
11350 033472 012706 001000                       †ST54: SCOPE
11351 033476 012737 000054 004656             MOV     #STACK, SP      ; RESET STACK
11352                                     MOV     #54, #STNM      ; SAVE TEST NUMBER
11353 033504 004737 040306                       JSR     FC, #CLDISK     ; GIVE RH INITIALIZE
11354                                     ; SETUP UNIT NUMBER
11355                                     ; CLEAR RHWC AND FUNCTION BITS IN RHCS1
11356
11357                                     ; SET UP FOR A ONE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
11358 033510 012701 000003                       MOV     #3, R1          ; COUNT OF THREE
11359 033514 012702 003000                       MOV     #BFEBVEN, R2    ; START THREE WORD BUFFER
11360                                     ; FROM AN EVEN WORD BOUNDARY
11361 033520 012722 052525                       1$:  MOV     #52525, (R2)+ ; MOVE THREE 52525 INTO BUFFER
11362 033524 005301                               DEC     R1              ; COUNT
11363 033526 001374                               BNE     1$,             ; BRANCH IF THREE NOT DONE
11364 033530 012777 003000 150326             MOV     #BFEBVEN, #RHBA ; SET BUS ADDRESS TO START
11365                                     ; FROM AN EVEN WORD BOUNDARY
11366 033536 012777 177777 150316             MOV     #-1, #RHWC      ; WORD COUNT ONE
11367 033544 004077 151070                       JSR     RD, #COMND      ; GO TO DO COMMAND
11368 033550 004204                               REVWRT ; REVERSEWRITE DATA
11369
11370
11371                                     ; CHECK THAT RHCS3 HAS 0
11372 033552 012737 000000 001124             MOV     #0, #GDDAT     ; GET GOOD = 0
11373
11374 033560 017737 150346 001126             MOV     #RHCS3, #BDDAT ; READ RHCS3 FOR COMPARISON
11375 033566 023737 001124 001126             CMP     #GDDAT, #BDDAT ; COMPARE EXPECTED
11376                                     ; DATA WITH DATA READ FROM
11377                                     ; RHCS3
11378 033574 001401                               BEQ     65$,            ; BRANCH IF GOOD
11379 033576 104124                               ERROR  124
11380
11381                                     ; AFTER SETTING "CLR" BIT #5
11382                                     ; IN RHCS2 TO INIT THE RH
11383                                     ; A ONE WORD WRITE REVERSE FROM AN
11384                                     ; EVEN WORD BOUNDARY WAS DONE
11385                                     ; THEN
  
```


J01

```

11553                                     ; IN RHCS2 TO INIT THE RH
11554                                     ; A TWO WORD WRITE REVERSE FROM AN
11555                                     ; EVEN WORD BOUNDARY WAS DONE
11556                                     ; THEN
11557                                     ; RHWC SHOULD HAVE 0
11558                                     ; BUT CONTAINED WHAT IS
11559                                     ; GIVEN IN BAD RHWC
11560 034144 718:
11561
11562
11563
11564 ::*****
11565 *TEST 56 TEST DBL (RHCS3 BIT #10) TEST 0
11566
11567 * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
11568 * SET UP FOR A TWO WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY
11569 * DO A TWO WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY
11570 * THIS SHOULD SET RHCS3 BIT #10 DBL
11571 * CHECK RHCS3
11572 * CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
11573
11574 ::*****
11575 034144 000004 †ST56: SCOPE
11576 034146 012706 001000 MOV #STACK, SP ; RESET STACK
11577 034152 012737 000056 004656 MOV #56, @†STNM ; SAVE TEST NUMBER
11578
11579 034160 004737 040306 JSR PC, @CLDISK ; GIVE RH INITIALIZE
11580 ; SETUP UNIT NUMBER
11581 ; CLEAR RHWC AND FUNCTION BITS IN RHCS1
11582
11583 ; SET UP FOR A TWO WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY
11584 034164 012701 000003 MOV #3, R1 ; COUNT OF THREE
11585 034170 012702 003002 MOV #BF0DD, R2 ; START THREE WORD BUFFER
11586 ; FROM AN ODD WORD BOUNDARY
11587 034174 012722 052525 1$: MOV #52525, (R2)+ ; MOVE THREE 52525 INTO BUFFER
11588 034200 005301 DEC R1 ; COUNT
11589 034202 001374 BNE 1$ ; BRANCH IF THREE NOT DONE
11590 034204 012777 003002 147652 MOV #BF0DD, @RHBA ; SET BUS ADDRESS TO START
11591 ; FROM AN ODD WORD BOUNDARY
11592 034212 012777 177776 147642 MOV #-2, @RHWC ; WORD COUNT TWO
11593 034220 004077 150414 JSR RD, @COMND ; GO TO DO COMMAND
11594 034224 004204 REVWRT ; REVERSEWRITE DATA
11595
11596
11597 ; CHECK THAT RHCS3 HAS DBL
11598 034226 012737 002000 001124 MOV #DBL, $GDDAT ; GET GOOD = 2000
11599
11600 034234 017737 147672 001126 MOV @RHCS3, $BDDAT ; READ RHCS3 FOR COMPARISON
11601 034242 023737 001124 001126 CMP $GDDAT, $BDDAT ; COMPARE EXPECTED
11602 ; DATA WITH DATA READ FROM
11603 ; RHCS3
11604 034250 001401 BEQ 65$ ; BRANCH IF GOOD
11605 034252 104124 ERROR 124
11606
11607 ; AFTER SETTING "CLR" BIT #5
11608 ; IN RHCS2 TO INIT THE RH
; A TWO WORD WRITE REVERSE FROM AN
    
```

K01

```

11609                                     : ODD WORD BOUNDARY WAS DONE
11610                                     : THEN
11611                                     : RHCS3 SHOULD HAVE DBL
11612                                     : =2000
11613                                     : BUT CONTAINED WHAT IS
11614                                     : GIVEN IN BAD RHCS3
11615 034254                                65$: BIC      #76, @RHCS1      : CLEAR FUNCTION BITS
11616 034254 042777 000076 147576          : CHECK THAT RHCS1 HAS SC!RDY!DVA
11617                                     : MOV      #SC!RDY!DVA, $GDDAT ; GET GOOD = 104200
11618 034262 012737 104200 001124
11619
11620 034270 017737 147564 001126          MOV      @RHCS1, $BDDAT ; READ RHCS1 FOR COMPARISON
11621 034276 023737 001124 001126          CMP      $GDDAT, $BDDAT ; COMPARE EXPECTED
11622                                     : DATA WITH DATA READ FROM
11623                                     : RHCS1
11624 034304 001401                          BEQ      67$           ; BRANCH IF GOOD
11625 034306 104125                          ERROR    125
11626
11627                                     : AFTER SETTING "CLR" BIT #5
11628                                     : IN RHCS2 TO INIT THE RH
11629                                     : A TWO WORD WRITE REVERSE FROM AN
11630                                     : ODD WORD BOUNDARY WAS DONE
11631                                     : THEN
11632                                     : RHCS1 SHOULD HAVE SC!RDY!DVA
11633                                     : =104200
11634                                     : BUT CONTAINED WHAT IS
11635                                     : GIVEN IN BAD RHCS1
11635 034310                                67$:
11636                                     : CHECK THAT RHCS2 HAS IR!OR
11637 034310 012737 000300 001124          MOV      #IR!OR, $GDDAT ; GET GOOD = 300
11638 034316 053737 005010 001124          BIS      UNIT, $GDDAT ; INCLUDE UNIT NUMBER
11639
11640 034324 017737 147540 001126          MOV      @RHCS2, $BDDAT ; READ RHCS2 FOR COMPARISON
11641 034332 023737 001124 001126          CMP      $GDDAT, $BDDAT ; COMPARE EXPECTED
11642                                     : DATA WITH DATA READ FROM
11643                                     : RHCS2
11644 034340 001401                          BEQ      69$           ; BRANCH IF GOOD
11645 034342 104126                          ERROR    126
11646
11647                                     : AFTER SETTING "CLR" BIT #5
11648                                     : IN RHCS2 TO INIT THE RH
11649                                     : A TWO WORD WRITE REVERSE FROM AN
11650                                     : ODD WORD BOUNDARY WAS DONE
11651                                     : THEN
11652                                     : RHCS2 SHOULD HAVE IR!OR
11653                                     : =300
11654                                     : TOGETHER WITH UNIT NUMBER
11655                                     : BUT CONTAINED WHAT IS
11656                                     : GIVEN IN BAD RHCS2
11656 034344                                69$:
11657                                     : CHECK THAT RHWC HAS 0
11658 034344 012737 000000 001124          MOV      #0, $GDDAT ; GET GOOD = 0
11659
11660 034352 017737 147504 001126          MOV      @RHWC, $BDDAT ; READ RHWC FOR COMPARISON
11661 034360 023737 001124 001126          CMP      $GDDAT, $BDDAT ; COMPARE EXPECTED
11662                                     : DATA WITH DATA READ FROM
11663                                     : RHWC
11664 034366 001401                          BEJ      71$           ; BRANCH IF GOOD

```

L01

```

11665 034370 104131          ERROR 131
11666
11667
11668
11669
11670
11671
11672
11673
11674 034372          13:
11675
11676
11677
11678
11679
11680
11681
11682
11683
11684
11685
11686
11687
11688
11689 034372 000004
11690 034374 012706 001000
11691 034400 012737 000057 004656
11692
11693 034406 004737 040306
11694
11695
11696
11697
11698 034412 012701 000003
11699 034416 012702 003000
11700
11701 034422 012722 052525          13:
11702 034426 005301
11703 034430 001374
11704 034432 012777 003000 147424
11705
11706 034440 012777 177775 147414
11707 034446 004077 150166
11708 034452 004204
11709
11710
11711
11712 034454 012737 002000 001124
11713
11714 034462 017737 147444 001126
11715 034470 023737 001124 001126
11716
11717
11718 034476 001401
11719 034500 104124
11720

: AFTER SETTING "CLR" BIT #5
: IN RHCS2 TO INIT THE RH
: A TWO WORD WRITE REVERSE FROM #1
: ODD WORD BOUNDARY WAS DONE
: THEN
: RHWC SHOULD HAVE 0
: BUT CONTAINED WHAT IS
: GIVEN IN BAD RHWC

:*****
: *TEST 57          TEST DBL (RHCS3 BIT #10) TEST P
:*****
: *
: * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
: * SET UP FOR A THREE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
: * DO A THREE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
: * THIS SHOULD SET RHCS3 BIT #10 DBL
: * CHECK RHCS3
: * CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
:*****
:*****
: *ST5: SCOPE
: MOV      #STACK, SP          ; RESET STACK
: MOV      #57, @#1STNM       ; SAVE TEST NUMBER
: JSR      PC, @#CLDISK       ; GIVE RH INITIALIZE
:                               ; SETUP UNIT NUMBER
:                               ; CLEAR RHWC AND FUNCTION BITS IN RHCS1
:
: * SET UP FOR A THREE WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
: MOV      #3, R1              ; COUNT OF THREE
: MOV      #BFEEVEN, R2        ; START THREE WORD BUFFER
:                               ; FROM AN EVEN WORD BOUNDARY
: *      MOV      #52525, (R2)+ ; MOVE THREE 52525 INTO BUFFER
: *      DEC      R1              ; COUNT
: *      BNE     1$              ; BRANCH IF THREE NOT DONE
: *      MOV      #BFEEV, @RHBA  ; SET BUS ADDRESS TO START
: *                               ; FROM AN EVEN WORD BOUNDARY
: *      MOV      #-3, @RHWC     ; WORD COUNT THREE
: *      JSR      RD, @COMND     ; GO TO DO COMMAND
: *      REVWRT                    ; REVERSEWRITE DATA
:
: * CHECK THAT RHCS3 HAS DBL
: MOV      #DBL, $GDDAT        ; GET GOOD = 2000
:
: MOV      @RHCS3, $BDDAT      ; READ RHCS3 FOR COMPARISON
: CMP      $GDDAT, $BDDAT      ; COMPARE EXPECTED
:                               ; DATA WITH DATA READ FROM
:                               ; RHCS3
: BEQ      65$                 ; BRANCH IF GOOD
: ERROR    124
:
: AFTER SETTING "CLR" BIT #5
    
```


NO1

```

11777                                     :RHWC
11778 034614 001401                       :BRANCH IF GOOD
11779 034616 104131                       :
11780                                     :AFTER SETTING "CLR" BIT #5
11781                                     :IN RHCS2 TO INIT THE RH
11782                                     :A THREE WORD WRITE REVERSE FROM AN
11783                                     :EVEN WORD BOUNDARY WAS DONE
11784                                     :THEN
11785                                     :RHWC SHOULD HAVE 0
11786                                     :BUT CONTAINED WHAT IS
11787                                     :GIVEN IN BAD RHWC
11788 034620                               :
11789                                     :
11790                                     :
11791                                     :
11792                                     :*****
11793                                     :*TEST 60      TEST DBL (RHCS3 BIT #10) TEST 0
11794                                     :
11795                                     :*
11796                                     :*   CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
11797                                     :*   SET UP FOR A THREE WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY
11798                                     :*   DO A THREE WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY
11799                                     :*   THIS SHOULD NOT SET RHCS3 BIT #10 DBL
11800                                     :*   CHECK RHCS3
11801                                     :*   CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
11802                                     :*****
11803 034620 000004                         †ST60: SCOPE
11804 034622 012706 J01000                 MOV     #STACK_SP      ;RESET STACK
11805 034626 012737 000060 004656         MOV     #60,#STNM      ;SAVE TEST NUMBER
11806                                     :
11807 034634 004737 040306                 JSR     PC,#CLDISK     ;GIVE RH INITIALIZE
11808                                     :SETUP UNIT NUBER
11809                                     :CLEAR RHWC AND FUNCTION BITS IN RHCS1
11810                                     :
11811                                     :SET UP FOR A THREE WORD WRITE REVERSE FROM AN ODD WORD BOUNDARY
11812 034640 012701 000003                 MOV     #3,R1          ;COUNT OF THREE
11813 034644 012702 003002                 MOV     #BF0DD,R2      ;START THREE WORD BUFFER
11814                                     :FROM AN ODD WORD BOUNDARY
11815 034650 012722 052525                 1$:   MOV     #52525,(R2)+ ;MOVE THREE 52525 INTO BUFFER
11816 034654 005301                         DEC     R1              ;COUNT
11817 034656 001374                         BNE    1$              ;BRANCH IF THREE NOT DONE
11818 034660 012777 003002 147176         MOV     #BF0DD,#RHBA   ;SET BUS ADDRESS TO START
11819                                     :FROM AN ODD WORD BOUNDARY
11820 034666 012777 177775 147166         MOV     #-3,#RHWC      ;WORD COUNT THREE
11821 034674 004077 147740                 JSR     R0,#COMND      ;GO TO DO COMMAND
11822 034700 004204                         REVWRT                ;REVERSEWRITE DATA
11823                                     :
11824                                     :
11825                                     :CHECK THAT RHCS3 HAS 0
11826 034702 012737 000000 001124         MOV     #0,$GDDAT      ;GET GOOD = 0
11827                                     :
11828 034710 017737 147216 001126         MOV     #RHCS3,$BDDAT  ;READ RHCS3 FOR COMPARISON
11829 034716 023737 001124 001126         CMP     $GDDAT,$BDDAT  ;COMPARE EXPECTED
11830                                     :DATA WITH DATA READ FROM
11831                                     :RHCS3
11832 034724 001401                         BEQ     65$            ;BRANCH IF GOOD

```

```

11833 034726 104124          ERROR 124
11834
11835
11836
11837
11838
11839
11840
11841
11842 034730          65$:
11843 034730 042777 000076 147122  BIC #76, @RHCS1 ;CLEAR FUNCTION BITS
11844 ;CHECK THAT RHCS1 HAS SC!RDY!DVA ;RDY!DVA
11845 034736 012737 104200 001124  MOV #SC!RDY!DVA, $GDDAT ;GET GOOD = 104200
11846
11847 034744 017737 147110 001126  MOV @PHCS1, $BDDAT ;READ RHCS1 FOR COMPARISON
11848 034752 023737 001124 001126  CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
11849 ;DATA WITH DATA READ FROM
11850 ;RHCS1
11851 034760 001401          BEQ 67$ ;BRANCH IF GOOD
11852 034762 104125          ERROR 125
11853
11854
11855
11856
11857
11858
11859
11860
11861
11862 034764          67$:
11863 ;CHECK THAT RHCS2 HAS IR!OR
11864 034764 012737 000300 001124  MOV #IR!OR, $GDDAT ;GET GOOD = 300
11865 034772 053737 005010 001124  BIS UNIT, $GDDAT ;INCLUDE UNIT NUMBER
11866
11867 035000 017737 147064 001126  MOV @PHCS2, $BDDAT ;READ RHCS2 FOR COMPARISON
11868 035006 023737 001124 001126  CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
11869 ;DATA WITH DATA READ FROM
11870 ;RHCS2
11871 035014 001401          BEQ 69$ ;BRANCH IF GOOD
11872 035016 104126          ERROR 126
11873
11874
11875
11876
11877
11878
11879
11880
11881
11882
11883 035020          69$:
11884 ;CHECK THAT RHWC HAS 0
11885 035020 012737 000000 001124  MOV #0, $GDDAT ;GET GOOD = 0
11886
11887 035026 017737 147030 001126  MOV @RHWC, $BDDAT ;READ RHWC FOR COMPARISON
11888 035034 023737 001124 001126  CMP $GDDAT, $BDDAT ;COMPARE EXPECTED
    
```

; AFTER SETTING "CLR" BIT #5
 ; IN RHCS2 TO INIT THE RH
 ; A THREE WORD WRITE REVERSE FROM AN
 ; ODD WORD BOUNDARY WAS DONE
 ; THEN
 ; RHCS3 SHOULD HAVE 0
 ; BUT CONTAINED WHAT IS
 ; GIVEN IN BAD RHCS3

; AFTER SETTING "CLR" BIT #5
 ; IN RHCS2 TO INIT THE RH
 ; A THREE WORD WRITE REVERSE FROM AN
 ; ODD WORD BOUNDARY WAS DONE
 ; THEN
 ; RHCS1 SHOULD HAVE SC!RDY!DVA
 ; =104200
 ; BUT CONTAINED WHAT IS
 ; GIVEN IN BAD RHCS1

; AFTER SETTING "CLR" BIT #5
 ; IN RHCS2 TO INIT THE RH
 ; A THREE WORD WRITE REVERSE FROM AN
 ; ODD WORD BOUNDARY WAS DONE
 ; THEN
 ; RHCS2 SHOULD HAVE IR!OR
 ; =300
 ; TOGETHER WITH UNIT NUMBER
 ; BUT CONTAINED WHAT IS
 ; GIVEN IN BAD RHCS2

```

11889                                     ;DATA WITH DATA READ FROM
11890                                     ;RHC
11891 035042 001401                         BEQ      715
11892 035044 104131                         ERROR   131
11893                                     ;AFTER SETTING "CLR" BIT #5
11894                                     ;IN RHCS2 TO INIT THE RH
11895                                     ;A THREE WORD WRITE REVERSE FROM AN
11896                                     ;ODD WORD BOUNDARY WAS DONE
11897                                     ;THEN
11898                                     ;RHC SHOULD HAVE 0
11899                                     ;BUT CONTAINED WHAT IS
11900                                     ;GIVEN IN BAD RHC
11901 035046                                715:
11902
11903
11904
11905
11906
11907
11908
11909
11910
11911
11912
11913
11914
11915
11916
11917 035046 000004
11918 035050 012706 001000
11919 035054 012737 000061 004656
11920
11921 035062 004737 040306
11922
11923
11924
11925
11926 035066 012701 000003
11927 035072 012702 003000
11928
11929 035076 012722 052525
11930 035102 005301
11931 035104 001374
11932 035106 012777 003000 146750
11933
11934 035114 012777 177776 146740
11935 035122 052777 000010 146740
11936 035130 004077 147504
11937 035134 004204
11938
11939
11940
11941 035136 012737 000000 001124
11942
11943 035144 017737 146762 001126
11944 035152 023737 001124 001126

```

```

:*****
:TEST 61      TEST DBL (RHCS3 BIT #10) TEST R
:
:  CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
:  SET UP FOR A TWO WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
:  WITH BAI IN RHCS2 BIT #3 SET
:  DO A TWO WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
:  THIS SHOULD NOT SET RHCS3 BIT #10 DBL
:  CHECK RHCS3
:  CHECK RHCS1, RHCS2, RHBA, RHBAE, RHC
:*****
†ST61:  SCOPE
        MOV      #STACK, SP      ;RESET STACK
        MOV      #61, @†STNM     ;SAVE TEST NUMBER
        JSR      PC, @CLDISK     ;GIVE RH INITIALIZE
                                     ;SETUP UNIT NUMBER
                                     ;CLEAR RHC AND FUNCTION BITS IN RHCS1
        ;SET JP FOR A TWO WORD WRITE REVERSE FROM AN EVEN WORD BOUNDARY
        MOV      #3, R1          ;COUNT OF THREE
        MOV      #8FEVEN, R2     ;START THREE WORD BUFFER
                                     ;FROM AN EVEN WORD BOUNDARY
        IS:    MOV      #52525, (R2)+ ;MOVE THREE 52525 INTO BUFFER
        DEC      R1              ;COUNT
        BNE     1$              ;BRANCH IF THREE NOT DONE
        MOV      #8FEVEN, @RHBA  ;SET BUS ADDRESS TO START
                                     ;FROM AN EVEN WORD BOUNDARY
        BIS      #BAI, @RHCS2    ;SET BAI IN RHCS2
        JSR      RO, @COMND      ;GO TO DO COMMAND
        REVWRT                                     ;REVERSEWRITE DATA
        ;CHECK THAT RHCS3 HAS 0
        MOV      #0, $GDDAT      ;GET GOOD = 0
        MOV      @RHCS3, $BDDAT  ;READ RHCS3 FOR COMPARISON
        CMP      $GDDAT, $BDDAT  ;COMPARE EXPECTED

```


E02

```

12001 035254          69$:
12002
12003 035254 012737 000000 001124      :CHECK THAT RHWC HAS 0
12004                                     MOV      #0,$GDDAT      ;GET GOOD = 0
12005 035262 017737 146574 001126      MOV      @RHWC,$BDDAT  ;READ RHWC FOR COMPARISON
12006 035270 023737 001124 001126      CMP      $GDDAT,$BDDAT ;COMPARE EXPECTED
12007                                     ;DATA WITH DATA READ FROM
12008                                     ;RHWC
12009 035276 001401      BEQ      71$          ;BRANCH IF GOOD
12010 035300 104131      ERROR    131
12011
12012                                     ;AFTER SETTING "CLR" BIT #5
12013                                     ;IN RHCS2 TO INIT THE RH
12014                                     ;A TWO WORD WRITE REVERSE FROM AN
12015                                     ;EVEN WORD BOUNDARY WAS DONE
12016                                     ;WITH BAI IN RHCS2 SET
12017                                     ;THEN
12018                                     ;RHWC SHOULD HAVE 0
12019                                     ;BUT CONTAINED WHAT IS
12020                                     ;GIVEN IN BAD RHWC
12020 035302          71$:
12021
12022
12023
12024
12025
12026
12027
12028
12029
12030
12031
12032
12033
12034
12035
12036
12037 035302 000004      ;*****
12038 035304 012706 001000      ;TEST 62      TEST DBL (RHCS3 BIT #10) TEST S
12039 035310 012737 000062 004656      ;
12040
12041 035316 004737 040306      ;*
12042                                     ;* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
12043                                     ;* SET UP FOR A NINE WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
12044                                     ;* DO A NINE WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
12045                                     ;* THIS SHOULD SET RHCS3 BIT #10 DBL
12046                                     ;* CHECK RHCS3
12047                                     ;* CHECK RHCS1,RHCS2,RHBA,RHBAE,RHWC
12048                                     ;* IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE
12049
12050
12051
12052
12053
12054
12055
12056
12057
12058
12059
12060
12061
12062
12063
12064
12065
12066
12067
12068
12069
12070
12071
12072
12073
12074
12075
12076
12077
12078
12079
12080
12081
12082
12083
12084
12085
12086
12087
12088
12089
12090
12091
12092
12093
12094
12095
12096
12097
12098
12099
12100
12101
12102
12103
12104
12105
12106
12107
12108
12109
12110
12111
12112
12113
12114
12115
12116
12117
12118
12119
12120
12121
12122
12123
12124
12125
12126
12127
12128
12129
12130
12131
12132
12133
12134
12135
12136
12137
12138
12139
12140
12141
12142
12143
12144
12145
12146
12147
12148
12149
12150
12151
12152
12153
12154
12155
12156
12157
12158
12159
12160
12161
12162
12163
12164
12165
12166
12167
12168
12169
12170
12171
12172
12173
12174
12175
12176
12177
12178
12179
12180
12181
12182
12183
12184
12185
12186
12187
12188
12189
12190
12191
12192
12193
12194
12195
12196
12197
12198
12199
12200
12201
12202
12203
12204
12205
12206
12207
12208
12209
12210
12211
12212
12213
12214
12215
12216
12217
12218
12219
12220
12221
12222
12223
12224
12225
12226
12227
12228
12229
12230
12231
12232
12233
12234
12235
12236
12237
12238
12239
12240
12241
12242
12243
12244
12245
12246
12247
12248
12249
12250
12251
12252
12253
12254
12255
12256
12257
12258
12259
12260
12261
12262
12263
12264
12265
12266
12267
12268
12269
12270
12271
12272
12273
12274
12275
12276
12277
12278
12279
12280
12281
12282
12283
12284
12285
12286
12287
12288
12289
12290
12291
12292
12293
12294
12295
12296
12297
12298
12299
12300
12301
12302
12303
12304
12305
12306
12307
12308
12309
12310
12311
12312
12313
12314
12315
12316
12317
12318
12319
12320
12321
12322
12323
12324
12325
12326
12327
12328
12329
12330
12331
12332
12333
12334
12335
12336
12337
12338
12339
12340
12341
12342
12343
12344
12345
12346
12347
12348
12349
12350
12351
12352
12353
12354
12355
12356
12357
12358
12359
12360
12361
12362
12363
12364
12365
12366
12367
12368
12369
12370
12371
12372
12373
12374
12375
12376
12377
12378
12379
12380
12381
12382
12383
12384
12385
12386
12387
12388
12389
12390
12391
12392
12393
12394
12395
12396
12397
12398
12399
12400
12401
12402
12403
12404
12405
12406
12407
12408
12409
12410
12411
12412
12413
12414
12415
12416
12417
12418
12419
12420
12421
12422
12423
12424
12425
12426
12427
12428
12429
12430
12431
12432
12433
12434
12435
12436
12437
12438
12439
12440
12441
12442
12443
12444
12445
12446
12447
12448
12449
12450
12451
12452
12453
12454
12455
12456
12457
12458
12459
12460
12461
12462
12463
12464
12465
12466
12467
12468
12469
12470
12471
12472
12473
12474
12475
12476
12477
12478
12479
12480
12481
12482
12483
12484
12485
12486
12487
12488
12489
12490
12491
12492
12493
12494
12495
12496
12497
12498
12499
12500
12501
12502
12503
12504
12505
12506
12507
12508
12509
12510
12511
12512
12513
12514
12515
12516
12517
12518
12519
12520
12521
12522
12523
12524
12525
12526
12527
12528
12529
12530
12531
12532
12533
12534
12535
12536
12537
12538
12539
12540
12541
12542
12543
12544
12545
12546
12547
12548
12549
12550
12551
12552
12553
12554
12555
12556
12557
12558
12559
12560
12561
12562
12563
12564
12565
12566
12567
12568
12569
12570
12571
12572
12573
12574
12575
12576
12577
12578
12579
12580
12581
12582
12583
12584
12585
12586
12587
12588
12589
12590
12591
12592
12593
12594
12595
12596
12597
12598
12599
12600
12601
12602
12603
12604
12605
12606
12607
12608
12609
12610
12611
12612
12613
12614
12615
12616
12617
12618
12619
12620
12621
12622
12623
12624
12625
12626
12627
12628
12629
12630
12631
12632
12633
12634
12635
12636
12637
12638
12639
12640
12641
12642
12643
12644
12645
12646
12647
12648
12649
12650
12651
12652
12653
12654
12655
12656
12657
12658
12659
12660
12661
12662
12663
12664
12665
12666
12667
12668
12669
12670
12671
12672
12673
12674
12675
12676
12677
12678
12679
12680
12681
12682
12683
12684
12685
12686
12687
12688
12689
12690
12691
12692
12693
12694
12695
12696
12697
12698
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
12727
12728
12729
12730
12731
12732
12733
12734
12735
12736
12737
12738
12739
12740
12741
12742
12743
12744
12745
12746
12747
12748
12749
12750
12751
12752
12753
12754
12755
12756
12757
12758
12759
12760
12761
12762
12763
12764
12765
12766
12767
12768
12769
12770
12771
12772
12773
12774
12775
12776
12777
12778
12779
12780
12781
12782
12783
12784
12785
12786
12787
12788
12789
12790
12791
12792
12793
12794
12795
12796
12797
12798
12799
12800
12801
12802
12803
12804
12805
12806
12807
12808
12809
12810
12811
12812
12813
12814
12815
12816
12817
12818
12819
12820
12821
12822
12823
12824
12825
12826
12827
12828
12829
12830
12831
12832
12833
12834
12835
12836
12837
12838
12839
12840
12841
12842
12843
12844
12845
12846
12847
12848
12849
12850
12851
12852
12853
12854
12855
12856
12857
12858
12859
12860
12861
12862
12863
12864
12865
12866
12867
12868
12869
12870
12871
12872
12873
12874
12875
12876
12877
12878
12879
12880
12881
12882
12883
12884
12885
12886
12887
12888
12889
12890
12891
12892
12893
12894
12895
12896
12897
12898
12899
12900
12901
12902
12903
12904
12905
12906
12907
12908
12909
12910
12911
12912
12913
12914
12915
12916
12917
12918
12919
12920
12921
12922
12923
12924
12925
12926
12927
12928
12929
12930
12931
12932
12933
12934
12935
12936
12937
12938
12939
12940
12941
12942
12943
12944
12945
12946
12947
12948
12949
12950
12951
12952
12953
12954
12955
12956
12957
12958
12959
12960
12961
12962
12963
12964
12965
12966
12967
12968
12969
12970
12971
12972
12973
12974
12975
12976
12977
12978
12979
12980
12981
12982
12983
12984
12985
12986
12987
12988
12989
12990
12991
12992
12993
12994
12995
12996
12997
12998
12999
13000

```



```

12113 ; IN RHCS2 TO INIT THE RH
12114 ; A NINE WORD READ REVERSE FROM AN
12115 ; EVEN WORD BOUNDARY WAS DONE
12116 ; THEN
12117 ; RHCS2 SHOULD HAVE IR
12118 ; =100
12119 ; TOGETHER WITH UNIT NUMBER
12120 ; BUT CONTAINED WHAT IS
12121 ; GIVEN IN BAD RHCS2
12122 035512          69$:
12123 ; CHECK THAT RHWC HAS 0
12124 035512 012737 000000 001124  MOV      #0,$GDDAT ; GET GOOD = 0
12125
12126 035520 017737 146336 001126  MOV      @RHWC,$BDDAT ; READ RHWC FOR COMPARISON
12127 035526 023737 001124 001126  CMP      $GDDAT,$BDDAT ; COMPARE EXPECTED
12128 ; DATA WITH DATA READ FROM
12129 ; RHWC
12130 035534 001401          BEQ      71$ ; BRANCH IF GOOD
12131 035536 104131          ERROR   131
12132
12133 ; AFTER SETTING "CLR" BIT #5
12134 ; IN RHCS2 TO INIT THE RH
12135 ; A NINE WORD READ REVERSE FROM AN
12136 ; EVEN WORD BOUNDARY WAS DONE
12137 ; THEN
12138 ; RHWC SHOULD HAVE 0
12139 ; BUT CONTAINED WHAT IS
12140 ; GIVEN IN BAD RHWC
12140 035540          71$:
12141
12142
12143
12144 ;*****
12145 ;*TEST 63          TEST DBL (RHCS3 BIT #10) TEST T
12146
12147 ;*
12148 ;* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
12149 ;* SET UP FOR A TEN WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
12150 ;* DO A TEN WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
12151 ;* THIS SHOULD NOT SET RHCS3 BIT #10 DBL
12152 ;* CHECK RHCS3
12153 ;* CHECK RHCS1,RHCS2,RHBA,RHBAE,RHWC
12154 ;* IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE
12155 ;*****
12156 035540 000004          TST63: SCOPE
12157 035542 012706 001000  MOV      #STACK,SP ; RESET STACK
12158 035546 012737 000063 004656  MOV      #63,@TSTNM ; SAVE TEST NUMBER
12159
12160 035554 004737 000306  JSR      PC,@CLDISK ; GIVE RH INITIALIZE
12161 ; SETUP UNIT NUMBER
12162 ; CLEAR RHWC AND FUNCTION BITS IN RHCS:
12163 035560 022737 041674 004640  CMP      #CMNDTM,COMND ; IS TU THERE
12164
12165 035566 001103          BNE     TST64 ; BRANCH IF TU NOT THERE
12166
12167
12168 ; SET UP FOR A TEN WORD READ REVERSE FROM AN EVEN WORD BOUNDARY

```

H02

CERHACO MACY11 30 1046 21-DEC-77 13:06 PAGE 228
 CERHAC.F11 21-DEC-77 12:48 163

TEST DBL RHCS3 BIT #10; TEST T

SEG 0227

12169	035570	012701	000003		MOV	#3,R1	:COUNT OF THREE	
12170	035574	012702	003000		MOV	#BF,EVEN,R2	:START THREE WORD BUFFER	
12171							:FROM AN EVEN WORD BOUNDARY	
12172	035600	012722	052525	13:	MOV	#52525,R2)+	:MOVE THREE 52525 INTO BUFFER	
12173	035604	005301			DEC	R1	:COUNT	
12174	035606	001374			BNE	15	:BRANCH IF THREE NOT DONE	
12175	035610	012777	003000	146246	MOV	#BF,EVEN,ARHBA	:SET BUS ADDRESS TO START	
12176							:FROM AN EVEN WORD BOUNDARY	
12177	035616	012777	177766	146236	MOV	#-10.,ARHWC	:WORD COUNT TEN	
12178								
12179	035624	004077	147010		JSR	RD,ACOMND	:GO TO DO COMMAND	
12180	035630	004200			REVRED		:REVERSE READ	
12181								
12182							:CHECK THAT RHCS3 HAS 0	
12183	035632	012737	000000	001124	MOV	#0,\$GDDAT	:GET GOOD = 0	
12184								
12185	035640	017737	146266	001126	MOV	ARHCS3,\$BDDAT	:READ RHCS3 FOR COMPARISON	
12186	035646	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED	
12187							:DATA WITH DATA READ FROM	
12188							:RHCS3	
12189	035654	001401			BEQ	65\$:BRANCH IF GOOD	
12190	035656	104124			ERROR	124		
12191							:AFTER SETTING "CLR" BIT #5	
12192							:IN RHCS2 TO INIT THE RH	
12193							:A TEN WORD READ REVERSE FROM AN	
12194							:EVEN WORD BOUNDARY WAS DONE	
12195							:THEN	
12196							:RHCS3 SHOULD HAVE 0	
12197							:BUT CONTAINED WHAT IS	
12198							:GIVEN IN BAD RHCS3	
12199	035660							
12200	035660	042777	000076	146172	65\$:	BIC	#76,ARHCS1	:CLEAR FUNCTION BITS
12201							:CHECK THAT RHCS1 HAS RDY!DVA	
12202	035666	012737	004200	001124	MOV	#RDY!DVA,\$GDDAT	:GET GOOD = 4200	
12203								
12204	035674	017737	146160	001126	MOV	ARHCS1,\$BDDAT	:READ RHCS1 FOR COMPARISON	
12205	035702	023737	001124	001126	CMP	\$GDDAT,\$BDDAT	:COMPARE EXPECTED	
12206							:DATA WITH DATA READ FROM	
12207							:RHCS1	
12208	035710	001401			BEQ	67\$:BRANCH IF GOOD	
12209	035712	104125			ERROR	125		
12210							:AFTER SETTING "CLR" BIT #5	
12211							:IN RHCS2 TO INIT THE RH	
12212							:A TEN WORD READ REVERSE FROM AN	
12213							:EVEN WORD BOUNDARY WAS DONE	
12214							:THEN	
12215							:RHCS1 SHOULD HAVE RDY!DVA	
12216							:=4200	
12217							:BUT CONTAINED WHAT IS	
12218							:GIVEN IN BAD RHCS1	
12219	035714				67\$:			
12220							:CHECK THAT RHCS2 HAS IR	
12221	035714	012737	000100	001124	MOV	#IR,\$GDDAT	:GET GOOD = 100	
12222	035722	053737	005010	001124	BIS	UNIT,\$GDDAT	:INCLUDE UNIT NUMBER	
12223								
12224	035730	017737	146134	001126	MOV	ARHCS2,\$BDDAT	:READ RHCS2 FOR COMPARISON	

```

12225 035736 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
12226 ;DATA WITH DATA READ FROM
12227 ;RHCS2
12228 035744 001401 BEQ 695 ;BRANCH IF GOOD
12229 035746 104126 ERROR 126
12230 ;AFTER SETTING "CLR" BIT #5
12231 ;IN RHCS2 TO INIT THE RH
12232 ;A TEN WORD READ REVERSE FROM AN
12233 ;EVEN WORD BOUNDARY WAS DONE
12234 ;THEN
12235 ;RHCS2 SHOULD HAVE IR
12236 ;=100
12237 ;TOGETHER WITH UNIT NUMBER
12238 ;BUT CONTAINED WHAT IS
12239 ;GIVEN IN BAD RHCS2
12240 035750 695:
12241 ;CHECK THAT RHCW HAS 0
12242 035750 012737 000000 001124 MOV #0,$GDDAT ;GET GOOD = 0
12243
12244 035756 017737 146100 001126 MOV @RHCW,$BDDAT ;READ RHCW FOR COMPARISON
12245 035764 023737 001124 001126 CMP $GDDAT,$BDDAT ;COMPARE EXPECTED
12246 ;DATA WITH DATA READ FROM
12247 ;RHCW
12248 035772 001401 BEQ 715 ;BRANCH IF GOOD
12249 035774 104131 ERROR 131
12250 ;AFTER SETTING "CLR" BIT #5
12251 ;IN RHCS2 TO INIT THE RH
12252 ;A TEN WORD READ REVERSE FROM AN
12253 ;EVEN WORD BOUNDARY WAS DONE
12254 ;THEN
12255 ;RHCW SHOULD HAVE 0
12256 ;BUT CONTAINED WHAT IS
12257 ;GIVEN IN BAD RHCW
12258 035776 715:
12259
12260
12261
12262
12263 :*****
12264 :*TEST 64 TEST DBL (RHCS3 BIT #10) TEST U
12265
12266 :* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5,
12267 :* SET UP FOR A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY)
12268 :* DO A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
12269 :* THIS SHOULD SET RHCS3 BIT #10 DBL
12270 :* CHECK RHCS3
12271 :* CHECK RHCS1,RHCS2,RHBA,RHBAE,RHCW
12272 :* IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE
12273 :*****
12274 035776 000004 001000 004656 TEST64: SCOPE
12275 036000 012706 000064 MOV #STACK,SP ;RESET STACK
12276 036004 012737 MOV #64,@TESTNM ;SAVE TEST NUMBER
12277
12278 036012 004737 040306 JSR PC,@CLDISK ;GIVE RH INITIALIZE
12279 ;SETUP UNIT NUMBER
12280 ;CLEAR RHCW AND FUNCTION BITS IN RHCS1

```

J02

```

12281 036016 022737 041674 004640      CMP      #CMNDTM,COMND      ;IS TJ THERE
12282
12283 036024 001103                        BNE      TST65      ;BRANCH IF TU NOT THERE
12284
12285
12286
12287 036026 012701 000003                    ;SET UP FOR A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
12288 036032 012702 003002      MOV      #3,R1      ;COUNT OF THREE
12289
12290 036036 012722 052525      1$:      MOV      #52525.(R2)+    ;START THREE WORD BUFFER
12291 036042 005301                        ;FROM AN ODD WORD BOUNDARY
12292 036044 001374                        ;MOVE THREE 52525 INTO BUFFER
12293 036046 012777 003002 146010      DEC      R1      ;COUNT
12294
12295 036054 012777 177766 146000      BNE      1$      ;BRANCH IF THREE NOT DONE
12296
12297 036062 004077 146552      MOV      #BF0DD,ARHBA    ;SET BUS ADDRESS TO START
12298 036066 004200                        ;FROM AN ODD WORD BOUNDARY
12299
12300
12301 036070 012737 002000 001124      MOV      #-10.,ARHWC    ;WORD COUNT TEN
12302
12303 036076 017737 146030 001126      JSR      RO,ACOMND      ;GO TO DO COMMAND
12304 036104 023737 001124 001126      REVRED                      ;REVERSE READ
12305
12306
12307 036112 001401                        ;CHECK THAT RHCS3 HAS DBL
12308 036114 104124                        MOV      #DBL,$GDDAT    ;GET GOOD = 2000
12309
12310
12311
12312
12313
12314
12315
12316
12317
12318 036116 001401                        MOV      ARHCS3,$BDDAT  ;READ RHCS3 FOR COMPARISON
12319 036116 042777 000076 145734      65$:      CMP      $GDDAT,$BDDAT  ;COMPARE EXPECTED
12320
12321 036124 012737 004200 001124      ;DATA WITH DATA READ FROM
12322
12323 036132 017737 145722 001126      ;RHCS3
12324 036140 023737 001124 001126      BEQ      65$          ;BRANCH IF GOOD
12325
12326
12327 036146 001401                        ;AFTER SETTING "CLR" BIT #5
12328 036150 104125                        ;IN RHCS2 TO INIT THE RH
12329
12330
12331
12332
12333
12334
12335
12336
    
```

K02

```

12337                                     ;GIVEN IN BAD RHCS1
12338 036152                               675:
12339                                     ;CHECK THAT RHCS2 HAS IR
12340 036152 012737 000100 001124      MOV   #IR,$GDDAT ;GET GOOD = 100
12341 036160 053737 005010 001124      BIS   JUNIT,$GDDAT ;INCLUDE UNIT NUMBER
12342
12343 036166 017737 145676 001126      MOV   @RHCS2,$BDDAT ;READ RHCS2 FOR COMPARISON
12344 036174 023737 001124 001126      CMP   $GDDAT,$BDDAT ;COMPARE EXPECTED
12345                                     ;DATA WITH DATA READ FROM
12346                                     ;RHCS2
12347 036202 001401                       BEQ   695          ;BRANCH IF GOOD
12348 036204 104126                       ERROR 126
12349
12350                                     ;AFTER SETTING "CLR" BIT #5
12351                                     ;IN RHCS2 TO INIT THE RH
12352                                     ;A TEN WORD READ REVERSE FROM AN
12353                                     ;ODD WORD BOUNDARY WAS DONE
12354                                     ;THEN
12355                                     ;RHCS2 SHOULD HAVE IR
12356                                     ;=100
12357                                     ;TOGETHER WITH UNIT NUMBER
12358                                     ;BUT CONTAINED WHAT IS
12359                                     ;GIVEN IN BAD RHCS2
12359 036206                               695:
12360
12361 036206 012737 000000 001124      MOV   #0,$GDDAT ;GET GOOD = 0
12362
12363 036214 017737 145642 001126      MOV   @RHWC,$BDDAT ;READ RHWC FOR COMPARISON
12364 036222 023737 001124 001126      CMP   $GDDAT,$BDDAT ;COMPARE EXPECTED
12365                                     ;DATA WITH DATA READ FROM
12366                                     ;RHWC
12367 036230 001401                       BEQ   715          ;BRANCH IF GOOD
12368 036232 104131                       ERROR 131
12369
12370                                     ;AFTER SETTING "CLR" BIT #5
12371                                     ;IN RHCS2 TO INIT THE RH
12372                                     ;A TEN WORD READ REVERSE FROM AN
12373                                     ;ODD WORD BOUNDARY WAS DONE
12374                                     ;THEN
12375                                     ;RHWC SHOULD HAVE 0
12376                                     ;BUT CONTAINED WHAT IS
12377 036234                               715:
12378                                     ;GIVEN IN BAD RHWC

```

```

12381 ::*****
12382 :*TEST 65 TEST DBL (RHCS3 BIT #10) TEST V
12383
12384 :* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
12385 :* SET UP FOR A ELEVEN WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
12386 :* DO A ELEVEN WORD READ REVERSE FROM AN EVEN WORD BOUNDARY
12387 :* THIS SHOULD SET RHCS3 BIT #10 DBL
12388 :* CHECK RHCS3
12389 :* CHECK RHCS1,RHCS2,RHBA,RHBAE,RHWC
12390 :* IF MAG. TAPE NOT USED THIS TEST IS NOT DONE
12391
12392 ::*****

```


M02

CERHADO MACY11 30(1046) 21-DEC-77 13:06 PAGE 233
CERHAD.P11 21-DEC-77 12:48 T65

TEST DBL (RHCS3 BIT #10) TEST V

SEQ 0232

```

12449                                     ; IN RHCS2 TO INIT THE RH
12450                                     ; A ELEVEN WORD READ REVERSE FROM AN
12451                                     ; EVEN WORD BOUNDARY WAS DONE
12452                                     ; THEN
12453                                     ; RHCS1 SHOULD HAVE RDY!DVA
12454                                     ; =4200
12455                                     ; BUT CONTAINED WHAT IS
12456                                     ; GIVEN IN BAD RHCS1
12457 036410                               675:
12458                                     ; CHECK THAT RHCS2 HAS IR
12459 036410 012737 000100 001124          MOV   #IR,$GDDAT      ; GET GOOD = 100
12460 036416 053737 005010 001124          BIS   UNIT,$GDDAT    ; INCLUDE UNIT NUMBER
12461
12462 036424 017737 145440 001126          MOV   @RHCS2,$BDDAT  ; READ RHCS2 FOR COMPARISON
12463 036432 023737 001124 001126          CMP   $GDDAT,$BDDAT ; COMPARE EXPECTED
12464                                     ; DATA WITH DATA READ FROM
12465                                     ; RHCS2
12466 036440 001401                          BEQ   695             ; BRANCH IF GOOD
12467 036442 104126                          ERROR 126
12468                                     ; AFTER SETTING "CLR" BIT #5
12469                                     ; IN RHCS2 TO INIT THE RH
12470                                     ; A ELEVEN WORD READ REVERSE FROM AN
12471                                     ; EVEN WORD BOUNDARY WAS DONE
12472                                     ; THEN
12473                                     ; RHCS2 SHOULD HAVE IR
12474                                     ; =100
12475                                     ; TOGETHER WITH UNIT NUMBER
12476                                     ; BUT CONTAINED WHAT IS
12477                                     ; GIVEN IN BAD RHCS2
12478 036444                               695:
12479                                     ; CHECK THAT RHWC HAS 0
12480 036444 012737 000000 001124          MOV   #0,$GDDAT     ; GET GOOD = 0
12481
12482 036452 017737 145404 001126          MOV   @RHWC,$BDDAT  ; READ RHWC FOR COMPARISON
12483 036460 023737 001124 001126          CMP   $GDDAT,$BDDAT ; COMPARE EXPECTED
12484                                     ; DATA WITH DATA READ FROM
12485                                     ; RHWC
12486 036466 001401                          BEQ   715             ; BRANCH IF GOOD
12487 036470 104131                          ERROR 131
12488                                     ; AFTER SETTING "CLR" BIT #5
12489                                     ; IN RHCS2 TO INIT THE RH
12490                                     ; A ELEVEN WORD READ REVERSE FROM AN
12491                                     ; EVEN WORD BOUNDARY WAS DONE
12492                                     ; THEN
12493                                     ; RHWC SHOULD HAVE 0
12494                                     ; BUT CONTAINED WHAT IS
12495                                     ; GIVEN IN BAD RHWC
12496 036472                               715:
12497
12498
12499
12500                                     ; *****
12501                                     ; *TEST 66          TEST DBL (RHCS3 BIT #10) TEST W
12502
12503                                     ; * CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
12504                                     ; * SET UP FOR A ELEVEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY

```

```

12505      : * DO A ELEVEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
12506      : * THIS SHOULD NOT SET RHCS3 BIT #10 DBL
12507      : * CHECK RHCS3
12508      : * CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
12509      : * IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE
12510
12511      : *****
12512 036472 000004      †T66: SCOPE
12513 036474 012706 001000      MOV #3, R1 ; RESET STACK
12514 036500 012737 000066 004656      MOV #66, #TSTNM ; SAVE TEST NUMBER
12515
12516 036506 004737 040306      JSR PC, #CLDISK ; GIVE RH INITIALIZE
12517 ; SETUP UNIT NUMBER
12518 ; CLEAR RHWC AND FUNCTION BITS IN RHCS1
12519 036512 022737 041674 004640      CMP #CMNDTM, COMND ; IS TU THERE
12520
12521 036520 001103      BNE TST67 ; BRANCH IF TU NOT THERE
12522
12523 ; SET UP FOR A ELEVEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
12524
12525 036522 012701 000003      MOV #3, R1 ; COUNT OF THREE
12526 036526 012702 003002      MOV #BF00D, R2 ; START THREE WORD BUFFER
12527 ; FROM AN ODD WORD BOUNDARY
12528 036532 012722 052525      1$: MOV #52525, (R2)+ ; MOVE THREE 52525 INTO BUFFER
12529 036536 005301      DEC R1 ; COUNT
12530 036540 001374      BNE 1$ ; BRANCH IF THREE NOT DONE
12531 036542 012777 003002 145314      MOV #BF00D, #RHBA ; SET BUS ADDRESS TO START
12532 ; FROM AN ODD WORD BOUNDARY
12533 036550 012777 177765 145304      MOV #-11., #RHWC ; WORD COUNT ELEVEN
12534
12535 036556 004077 146056      JSR RO, #COMND ; GO TO DO COMMAND
12536 036562 004200      REVRED ; REVERSE READ
12537
12538 ; CHECK THAT RHCS3 HAS 0
12539 036564 012737 000000 001124      MOV #0, $GDDAT ; GET GOOD = 0
12540
12541 036572 017737 145334 001126      MOV #RHCS3, $BDDAT ; READ RHCS3 FOR COMPARISON
12542 036600 023737 001124 001126      CMP $GDDAT, $BDDAT ; COMPARE EXPECTED
12543 ; DATA WITH DATA READ FROM
12544 ; RHCS3
12545 036606 001401      BEQ 65$ ; BRANCH IF GOOD
12546 036610 104124      ERROR 124
12547
12548 ; AFTER SETTING "CLR" BIT #5
12549 ; IN RHCS2 TO INIT THE RH
12550 ; A ELEVEN WORD READ REVERSE FROM AN
12551 ; ODD WORD BOUNDARY WAS DONE
12552 ; THEN
12553 ; RHCS3 SHOULD HAVE 0
12554 ; BUT CONTAINED WHAT IS
12555 ; GIVEN IN BAD RHCS3
12555 036612      65$: BIC #76, #RHCS1 ; CLEAR FUNCTION BITS
12556 036612 042777 000076 145240      ; CHECK THAT RHCS1 HAS RDY!DVA
12557 036620 012737 004200 001124      MOV #RDY!DVA, $GDDAT ; GET GOOD = 4200
12558
12559
12560 036626 017737 145226 001126      MOV #RHCS1, $BDDAT ; READ RHCS1 FOR COMPARISON

```



```

12617
12618
12619
12620
12621
12622
12623
12624
12625
12626
12627
12628
12629
12630
12631 036730 000004
12632 036732 012706 001000
12633 036736 012737 000067 004656
12634
12635 036744 004737 040306
12636
12637
12638 036750 022737 041674 004640
12639
12640 036756 001106
12641
12642
12643
12644 036760 012701 000003
12645 036764 012702 003002
12646
12647 036770 012722 052525
12648 036774 005301
12649 036776 001374
12650 037000 012777 003002 145056
12651
12652 037006 012777 177766 145046
12653 037014 052777 000010 145046
12654
12655 037022 004077 145612
12656 037026 004200
12657
12658
12659 037030 012737 000000 001124
12660
12661 037036 017737 145070 001126
12662 037044 023737 001124 001126
12663
12664
12665 037052 001401
12666 037054 104124
12667
12668
12669
12670
12671
12672

```

```

*****
*TEST 67 TEST DBL (RHCS3 BIT #10) TEST W
*
* CLEAR THE SUBSYSTEM BY A CONTROLLER CLEAR (RHCS2 BIT #5)
* SET UP FOR A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
* WITH BAI IN RHCS2 BIT #3 SET
* DO A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
* THIS SHOULD NOT SET RHCS3 BIT #10 DBL
* CHECK RHCS3
* CHECK RHCS1, RHCS2, RHBA, RHBAE, RHWC
* IF MAG. TAPE NOT USED. THIS TEST IS NOT DONE
*****
TST67: SCOPE
MOV #STACK, SP ; RESET STACK
MOV #67, #TSTNM ; SAVE TEST NUMBER
JSR PC, #CLDISK ; GIVE RH INITIALIZE
; SETUP UNIT NUMBER
; CLEAR RHWC AND FUNCTION BITS IN RHCS1
CMP #CMNDTM, COMND ; IS TU THERE
BNE TST70 ; BRANCH IF TU NOT THERE

; SET UP FOR A TEN WORD READ REVERSE FROM AN ODD WORD BOUNDARY
MOV #3, R1 ; COUNT OF THREE
MOV #BFODD, R2 ; START THREE WORD BUFFER
; FROM AN ODD WORD BOUNDARY
1$: MOV #52525, (R2)+ ; MOVE THREE 52525 INTO BUFFER
DEC R1 ; COUNT
BNE 1$ ; BRANCH IF THREE NOT DONE
MOV #BFODD, #RHBA ; SET BUS ADDRESS TO START
; FROM AN ODD WORD BOUNDARY
MOV #-10, #RHWC ; WORD COUNT TEN
BIS #BAI, #RHCS2 ; SET BAI IN RHCS2

JSR #0, #COMND ; GO TO DO COMMAND
REVRED ; REVERSE READ

; CHECK THAT RHCS3 HAS 0
MOV #0, #GDDAT ; GET GOOD = 0

MOV #RHCS3, #BDDAT ; READ RHCS3 FOR COMPARISON
CMP #GDDAT, #BDDAT ; COMPARE EXPECTED
; DATA WITH DATA READ FROM
; RHCS3
; BRANCH IF GOOD
BEQ 65$
ERROR 124

; AFTER SETTING "CLR" BIT #5
; IN RHCS2 TO INIT THE RH
; A TEN WORD READ REVERSE FROM AN
; ODD WORD BOUNDARY WAS DONE
; WITH BAI IN RHCS2 SET
; THEN

```

```

12673                                     :RHCS3 SHOULD HAVE 0
12674                                     :BUT CONTAINED WHAT IS
12675                                     :GIVEN IN BAD RHCS3
12676 037056                               65$:
12677 037056 042777 000076 144774      BIC      #76,DRHCS1      :CLEAR FUNCTION BITS
12678                                     :CHECK THAT RHCS1 HAS RDY!DVA
12679 037064 012737 004200 001124      MOV      #RD,!DVA,$GDDAT :GET GOOD = 4200
12680
12681 037072 017737 144762 001126      MOV      DRHCS1,$BDDAT   :READ RHCS1 FOR COMPARISON
12682 037100 023737 001124 001126      CMP      $GDDAT,$BDDAT  :COMPARE EXPECTED
12683                                     :DATA WITH DATA READ FROM
12684                                     :RHCS1
12685 037106 001401                          BEQ      67$
12686 037110 104125                          ERROR    125
12687                                     :AFTER SETTING "CLR" BIT #5
12688                                     :IN RHCS2 TO INIT THE RH
12689                                     :A TEN WORD READ REVERSE FROM AN
12690                                     :ODD WORD BOUNDARY WAS DONE
12691                                     :WITH BAI IN RHCS2 SET
12692                                     :THEN
12693                                     :RHCS1 SHOULD HAVE RDY!DVA
12694                                     :=4200
12695                                     :BUT CONTAINED WHAT IS
12696                                     :GIVEN IN BAD RHCS1
12697 037112                               67$:
12698                                     :CHECK THAT RHCS2 HAS IR!BAI
12699 037112 012737 000110 001124      MOV      #IR!BAI,$GDDAT :GET GOOD = !10
12700 037120 053737 005010 001124      BIS      UNIT,$GDDAT    :INCLUDE UNIT NUMBER
12701
12702 037126 017737 144736 001126      MOV      DRHCS2,$BDDAT   :READ RHCS2 FOR COMPARISON
12703 037134 023737 001124 001126      CMP      $GDDAT,$BDDAT  :COMPARE EXPECTED
12704                                     :DATA WITH DATA READ FROM
12705                                     :RHCS2
12706 037142 001401                          BEQ      69$
12707 037144 104126                          ERROR    126
12708                                     :AFTER SETTING "CLR" BIT #5
12709                                     :IN RHCS2 TO INIT THE RH
12710                                     :A TEN WORD READ REVERSE FROM AN
12711                                     :ODD WORD BOUNDARY WAS DONE
12712                                     :WITH BAI IN RHCS2 SET
12713                                     :THEN
12714                                     :RHCS2 SHOULD HAVE IR!BAI
12715                                     :=110
12716                                     :TOGETHER WITH UNIT NUMBER
12717                                     :BUT CONTAINED WHAT IS
12718                                     :GIVEN IN BAD RHCS2
12719 037146                               69$:
12720                                     :CHECK THAT RHWC HAS 0
12721 037146 012737 000000 001124      MOV      #0,$GDDAT      :GET GOOD = 0
12722
12723 037154 017737 144702 001126      MOV      DRHWC,$BDDAT   :READ RHWC FOR COMPARISON
12724 037162 023737 001124 001126      CMP      $GDDAT,$BDDAT  :COMPARE EXPECTED
12725                                     :DATA WITH DATA READ FROM
12726                                     :RHWC
12727 037170 001401                          BEQ      71$
12728 037172 104131                          ERROR    131

```

12739
12740
12741
12742
12743
12744
12745
12746
12747
12748
12749
12750
12751
12752
12753
12754
12755
12756
12757
12758
12759
12760
12761
12762
12763
12764
12765
12766
12767
12768
12769
12770
12771
12772
12773
12774
12775
12776
12777
12778
12779
12780
12781
12782
12783
12784

037174

713:

: AFTER SETTING "CLR" BIT #5
: IN RHCS2 TO INIT THE RH
: A TEN WORD READ REVERSE FROM A'
: ODD WORD BOUNDARY WAS DONE
: WITH BAI IN RHCS2 SET
: THEN
: RHWC SHOULD HAVE 0
: BUT CONTAINED WHAT IS
: GIVEN IN BAD RHWC

```
*****
: TEST 70      END OF ONE RH
: *          THIS IS THE END OF TEST FOR ONE RH
: *          IF THERE ARE MORE RH THEN THE PROGRAM
: *          JUMPS TO TEST 2 FOR NEXT RH TEST
: *          END PASS IS REACHED ONLY AFTER ALL RH ARE COMPLETE
*****
TST70: SCOPE
MOV      #1, $TIMES      ;; DO 1 ITERATION
CLR      PS              ;; REINSTATE PS TO 0
TYPE     65$             ;; TYPE ASCIZ STRING
BR       64$             ;; GET OVER THE ASCIZ
;;65$: .ASCIZ <15><12>/END OF TEST FOR RH NO/
64$:    MOV      WORUNT, -(SP) ;; GET WORKING RH NUMBER LEFT
        MOV      TSTUNT, R2   ;; GET TOTAL NO OF RH FOUND
        SUB      (SP)+, R2    ;; NO OF RH TESTED
        ASL      R2           ;; INDEX FOR RH TESTED
        MOV      FRHNM(R2), -(SP) ;; TYPE OF RH NO.
        TYPDS
        TYPE     67$         ;; TYPE ASCIZ STRING
        BR       66$         ;; GET OVER THE ASCIZ
;;67$: .ASCIZ /          BASE /
66$:    MOV      RHCS1, -(SP) ;; TYPE BASE ADDRESS
        TYPOC
        TYPE     69$         ;; TYPE ASCIZ STRING
        BR       68$         ;; GET OVER THE ASCIZ
;;69$: .ASCIZ <15><12>/TOTAL NO OF ERRORS ON THIS PASS/
68$:    MOV      $ERTTL, -(SP) ;; TYPE TOTAL NO OF ERRORS
        TYPDS
        TYPE     71$         ;; TYPE ASCIZ STRING
        BR       70$         ;; GET OVER THE ASCIZ
;;71$: .ASCIZ <15><12>
70$:    TYPE     73$         ;; TYPE ASCIZ STRING
        BR       72$         ;; GET OVER THE ASCIZ
;;73$: .ASCIZ 15 12
72$:
```

000004
012737 000001 001212
005037 177776
104401 037216
000414
037246
013746 005510
013702 005506
162602
006302
016246 005436
104405
104401 037276
000410
037316
013746 004060
104402
104401 037332
000421
037374
013746 001112
104405
104401 037410
000402
037414
104401 037422
000402
037426

F03

CERHAD0 MACY11 301046 21-DEC-77 13:06 PAGE 239
CERHAD.F11 21-DEC-77 12:48 770

SEQ 0239

12785	037426	005037	001112	CLR	SERTTL	:CLEAR TOTAL NO OF ERRORS
12786	037432	005337	005510	DEC	WORUNT	:DECREASE NO. OF RH TESTED
12787	037436	001402		BEG	IS	
12788	037440	000137	010610	JMP	TST2	
12789	037444	013737	005506	MOV	TSTUNT,WOPUNT	:MAKE WORKING RH SAME
12790						:AS TOTAL RH

END OF PASS ROUTINE

.SBTTL END OF PASS ROUTINE

12791
12792
12793
12794
12795
12796
12797
12798
12799 037452
12800 037452 000004
12801 037454 005037 001102
12802 037460 005037 001212
12803 037464 005237 001100
12804 037470 042737 100000 001100
12805 037476 005327
12806 037500 000001
12807 037502 003022
12808 037504 012737
12809 037506 000001
12810 037510 037500
12811 037512 104401 037557
12812 037516 013746 001100
12813 037522 104405
12814 037524 104401 037554
12815 037530 013700 000042
12816 037534 001405
12817 037536 000005
12818 037540 004710
12819 037542 000240
12820 037544 000240
12821 037546 000240
12822 037550
12823 037550 000137
12824 037552 010610
12825 037554 377 377 000
12826 037557 015 042412 042116
12827 037564 050040 051501 020123
12828 037572 000043
12829
12830
12831
12832
12833
12834
12835
12836
12837
12838
12839
12840
12841
12842
12843 037574
12844 037574 104401 037602
12845 037600 000433
12846

```

*****
*INCREMENT THE PASS NUMBER ($PASS)
*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO TST2

```

```

$EOP:
SCOPE
CLR $TSTNM ;; ZERO THE TEST NUMBER
CLR $TIMES ;; ZERO THE NUMBER OF ITERATIONS
INC $PASS ;; INCREMENT THE PASS NUMBER
BIC #10000,$PASS ;; DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;; LOOP?

$EOPCT: .WORD 1
BGT $DOAGN ;; YES
MOV (PC)+,(PC)+ ;; RESTORE COUNTER

$ENDCT: .WORD 1
$EOPCT
TYPE $ENDMG ;; TYPE "END PASS #"
MOV $PASS,-(SP) ;; SAVE $PASS FOR TYPEOUT
TYPDS ;; GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $ENULL ;; TYPE A NULL CHARACTER
$GET42: MOV #42,RO ;; GET MONITOR ADDRESS
BEQ $DOAGN ;; BRANCH IF NO MONITOR
RESET ;; CLEAR THE WORLD
$ENDAD: JSR PC,(RO) ;; GO TO MONITOR
NOP ;; SAVE ROOM
NOP ;; FOR
NOP ;; ACT11

$DOAGN:
JMP #4(PC)+ ;; RETURN
$RTNAD: .WORD TST2
$ENULL: .BYTE -1,-1,0 ;; NULL CHARACTER STRING
$ENDMG: .ASCIZ <15><12> END PASS #

```

.SBTTL SUBROUTINES

```

*****
* THIS ROUTINE HANDLES TIME OUT TRAPS THROUGH LOC. 4
*****

```

```

TIEOUT:
TYPE 65$ ;; TYPE ASCIZ STRING
BR 64$ ;; GET OVER THE ASCIZ
:65$: .ASCIZ <15><12> TIMEOUT OCCURED THROUGH LOCATION 4 FROM PROGRAM P.

```

```

12847 037670
12848 037670 162716 000002
12849 037674 104402
12850 037676 104401 037704
12851 037702 000410
12852
12853 037724
12854 037724 104402
12855 037726 000137 043104
12856
12857
12858
12859
12860
12861
12862
12863
12864
12865
12866
12867 037732
12868 037732 104401 037740
12869 037736 000417
12870
12871 037776
12872 037776 104401 040004
12873 040002 000413
12874
12875 040032
12876 040032 162716 000002
12877 040036 104402
12878 040040 104401 040046
12879 040044 000407
12880
12881 040064
12882 040064 104402
12883 040066 104401 040074
12884 040072 000421
12885
12886 040136
12887 040136 017746 145354
12888 040142 104402

```

```

64$: SUB #2,(SP) ;TYPE PC
      TYPOC
      TYPE 67$ ;:TYPE ASCIZ STRING
      BR 66$ ;:GET OVER THE ASCIZ
::67$: .ASCIZ '<15><12> PSW WAS /
66$: TYPOC
      JMP $$SCOPE ;RETURN TO TEST

::*****
::* THIS ROUTINE HANDLES PARITY ERRORS
::*****
PARITY:
      TYPE 65$ ;:TYPE ASCIZ STRING
      BR 64$ ;:GET OVER THE ASCIZ
::65$: .ASCIZ '<15><12> PARITY TRAP THRU VECTOR 114/
64$: TYPE 67$ ;:TYPE ASCIZ STRING
      BR 66$ ;:GET OVER THE ASCIZ
::67$: .ASCIZ '<15><12> FROM PROGRAM PC /
66$: SUB #2,(SP) ;TYPE PC
      TYPOC
      TYPE 69$ ;:TYPE ASCIZ STRING
      BR 68$ ;:GET OVER THE ASCIZ
::69$: .ASCIZ '<15><12> PSW WAS /
68$: TYPOC
      TYPE 71$ ;:TYPE ASCIZ STRING
      BR 70$ ;:GET OVER THE ASCIZ
::71$: .ASCIZ '<15><1> HIGH ERROR ADDRESS REGISTER
70$: MOV $HERADD,-(SP) ;TYPE HIGH ERROR
      TYPOC

```

12889	040144	104401	040152		TYPE	735	::TYPE ASCIZ STRING
12890	040150	000421			BR	725	::GET OVER THE ASCIZ
12891				735:	.ASCIZ	<15><12 /LOW ERROR ADDRESS REGISTER	
12892	040214			725:			
12893	040214	017746	145274		MOV	2LERADD,-(SP)	:TYPE LOW ERROR
12894	040220	104402			TYPOC		

J03

CERHADO MACY11 30110467 21-DEC-77 13:06 PAGE 243
 CERHAD.P11 21-DEC-77 12:48 SUBROUTINES

SEG 0242

12895	040222	104401	040230						
12896	040226	000422							
12897									
12898	040274								
12899	040274	017746	145220						
12900	040300	104402							
12901	040302	000137	043104						
12902									
12903									
12904									
12905									
12906									
12907									
12908									
12909	040306	012777	000040	143554	CLDISK:	MOV	#CLR, @RHCS2		;CLEAR ALL REG.
12910	040314	013777	005010	143546		MOV	@UNIT, @RHCS2		;REINSTATE UNIT NO.
12911	040322	042777	000077	143530		BIC	#77, @RHCS1		;CLEAR FUNCTION BITS
12912	040330	005077	143526			CLR	@RHW		;CLEAR RHW
12913	040334	000207				RTS	PC		

K03

```

12914
12915
12916
12917
12918
12919
12920 040336 000000
12921
12922
12923
12924
12925 040340 011637 040336
12926 040344 162737 000004 040336
12927 040352 011346
12928 040354 052716 000100
12929 040360 000406
12930 040362 011637 040336
12931 040366 162737 000004 040336
12932 040374 011346
12933 040376 011146
12934 040400 042716 173577
12935 040404 022726 004200
12936
12937 040410 001403
12938 040412 011137 001122
12939 040416 104062
12940
12941
12942 040420 042716 102000
12943 040424 022726 010700
12944 040430 001403
12945 040432 011337 001122
12946 040436 104061
12947
12948
12949 040440 000207
12950
    ; THIS CHECKS DEVICE AVAILABLE (DVA) AND READY (RDY) IN RHCSI
    ; AND CHECKS MEDIUM ON LINE (MOL), DEVICE PRESENT (DPR), DEVICE READY (DRY), IN PHL :
    PCJSR: 0 ,PC OF JSR

    CHECK: MOV (SP),J#PCJSR ;SAVE PC OF JSR+4
           SUB #4,J#PCJSR ;GET PC OF JSR
           MOV @R3,-(SP) ;GET RHDS1
           BIS #VV,(SP) ;DONT CHECK VV BIT
           BR CHECKC ;GOTO COMMON CHECK ROUT!!
    CHECKT: MOV (SP),J#PCJSR ;SAVE PC OF JSR+4
           SUB #4,J#PCJSR ;GET PC OF JSR
           MOV @R3,-(SP) ;GET RHDS1 & DO VV CHECK AT 3$
           MOV @R1,-(SP) ;GET CSI
           BIC #173577,(SP) ;CLEAR UNWANTED BITS
           CMP #DVA!RDY,(SP)+ ;RHCSI SHOULD HAVE DEVICE AVAILABLE
           BEQ 3$ ;AND BE READY
           MOV @R1,J#SBDADR ;BRANCH IF GOOD
           ERROR 62 ;BAD DATA REGISTER (RHDS1)
           ;RHCSI DID NOT HAVE DEVICE
           ;AVAILABLE RIGHT AT THE START
           ;ALL OTHER BITS SHOULD BE 0
    3$: BIC #ATA!LBT,(SP) ;CLEAR UNWANTED BITS
        CMP #MOL!DPR!DRY!VV,(SP)+ ;RHDS1 SHOULD HAVE THESE SET
        BEQ 7$ ;BRANCH IF GOOD
        MOV @R3,J#SBDADR ;BAD DATA IN REGISTER (RHDS1)
        ERROR 61 ;RHDS1 HAS SOME BITS OTHER
           ;THAN MOL, DRY, DPR, VV SET
           ;ALL OTHER BITS SHOULD BE 0
    7$: RTS PC ;RETURN TO TEST MC.
    
```

```

12951
12952
12953
12954
12955
12956
12957
12958
12959
12960
12961      ; THIS IS A WAIT LOOP WHEN NO P-CLOCK IS AVAILABLE
12962      ; NO TIMING IS DONE
12963      ; CALL IS
12964      ;
12965      ;       WAT
12966      ;       A       ; POINTER TO REGISTER ADDRESS
12967      ;       B       ; BIT WAITED FOR
12968      ; R3-IS A TEMPORARY COUNTER
12969      TIMCNT: 177777      ; COUNT FOR WAIT LOOP
12970
12971      WAIT.T:
12972      MOV      R0,-(SP)      ;: PUSH R0 ON STACK
12973      MOV      R3,-(SP)      ;: PUSH R3 ON STACK
12974      MOV      4(SP),R0      ;: R0 HAS ADDRESS OF NEXT LOCATION
12975      MOV      R0,@#WAITPC   ;: WAT PC +2 IS IN WAITPC
12976      SUB      #2,@#WAITPC   ;: WAT PC IS IN WAITPC
12977      MOV      @#WAITPC,R0   ;: WAIT ON REGISTER ADDRESS
12978      MOV      (R0)+,@#WAITRE ;: WAIT ON BIT
12979      MOV      R0,4(SP)      ;: RESTORE RETURN ON STACK
12980
12981      ; THIS HAS THE TWO COUNT DOWNS FROM 177777
12982      ; FOR WAITING FOR A BIT TO SET
12983      TST      WAT0          ;: WAITING FOR 1?
12984      BNE      4$           ;: BRANCH IF WAITING FOR 0
12985      MOV      @#TIMCNT,R3   ;: R3 HAS TEMPORARY COUNT
12986      BIT      @#WAITBT,@#WAITRE ;: IS REQUIRED BIT THERE
12987      BNE      3$           ;: BRANCH IF YES
12988      DEC      R3           ;: COUNT IF REQUIRED BIT NOT THERE
12989      BNE      1$           ;:
12990      MOV      @#TIMCNT,R3   ;: SECOND COUNT DOWN FROM 177777
12991      BIT      @#WAITBT,@#WAITRE ;: IS REQUIRED BIT THERE
12992      BNE      3$           ;: BRANCH IF YES
12993      DEC      R3           ;: COUNT IF REQUIRED BIT NOT THERE
12994      BNE      2$           ;:
12995      MOV      @#WAITRE,@#SDDAT ;: REGISTER CONTENTS FOR TYPEOUT
12996      ERROR   13$          ;: WAS WAITING FOR A BIT TO SET
12997      ;: BIT IN QUESTION IS IN "BIT WAITED"
12998      ;: REGISTER IN QUESTION IS IN "REG ADDR"
12999      BR      3$           ;: BRANCH OUT
13000
13001
13002
13003      ; THIS HAS THE TWO COUNT DOWNS FROM 177777
13004      ; FOR WAITING FOR A BIT TO RESET
13005      MOV      @#TIMCNT,R3   ;: R3 HAS TEMPORARY COUNT
13006      MOV      #20,R0       ;: R0 HAS TEMPORARY COUNT
13007      BIT      @#WAITBT,@#WAITRE ;: IS REQUIRED BIT RESET?

```

13007	040600	001413			BEG	3\$: BRANCH IF YES
13008	040602	005303			DEC	R3		: COUNT IF REQUIRED BIT HASN'T RESET YET
13009	040604	001372			BNE	5\$: TRY AGAIN
13010	040606	005300			DEC	R0		: DECREMENT R0 COUNT
13011	040610	001370			BNE	5\$: START OVER IF COUNT ISN'T ZERO YET
13012	040612	017737	144052	001126	MOV	@WAITRE,@#SBDAT		: REGISTER CONTENTS FOR TIMEOUT
13013	040620	104136			ERROR	136		: WAS WAITING FOR A BIT TO RESET
13014								: BIT IN QUESTION IS IN "BIT WAITED"
13015								: REGISTER IN QUESTION IS IN "REG ADDR"
13016	040622	017737	144042	001126	MOV	@WAITRE,@#SBDAT		: REGISTER CONTENTS FOR TIMEOUT
13017	040630				3\$:			
13018	040630	012603			MOV	(SP)+,R3		: POP STACK INTO R3
13019	040632	012600			MOV	(SP),R0		: POP STACK INTO R0
13020	040634	000002			RTI			: RETURN TO MAIN TEST

13021
 13022
 13023
 13024
 13025
 13026
 13027
 13028
 13029
 13030
 13031
 13032
 13033
 13034
 13035
 13036
 13037
 13038
 13039
 13040
 13041
 13042
 13043
 13044
 13045
 13046
 13047
 13048
 13049
 13050
 13051
 13052
 13053
 13054
 13055
 13056
 13057
 13058
 13059
 13060
 13061
 13062
 13063
 13064
 13065
 13066
 13067
 13068
 13069
 13070
 13071
 13072
 13073
 13074
 13075
 13076

040636 000000
 040640
 040640 005037 177776
 040644 012737 177777 045064
 040652 104401 040660
 040656 000421
 040722
 040722 013746 004656
 040726 104402
 040730 104401 040736
 040734 000414
 040766
 040766 013746 001110
 040772 104402
 040774 104401 001223
 041000 104401 041006
 041004 000430
 041066
 041066 104401 041074
 041072 000430
 041154
 041154 104401 041162
 041160 000422
 041226
 041226 104410
 041230 062716 000002
 041234 012637 001106
 041240 104401 041246
 041244 000417

: HERE IS A DETAILED EXPLANATION OF HOW THE LOOP ON ERROR WORKS.
 : ON HITTING AN ERROR IF THE LOOP ON ERROR SWITCH IS SET, THE
 : PROGRAM GOES BACK - USUALLY BACK TO THE BEGINNING OF THE TEST.
 : WHEN THIS OPERATOR SELECTABLE SCOPE LOOP IS USED THEN THE POINT
 : THE PROGRAM GOES BACK TO CAN BE CHANGED.
 : THE RESTRICTIONS TO THE POINT WHERE THE PROGRAM CAN GO ARE: -
 : 1. IT MUST BE WITHIN THE TEST UNDER CONSIDERATION
 : 2. LOOP ON ERROR SWITCH MUST BE SET
 : 3. THE ERROR MUST OCCUR WITHIN THE TEST UNDER CONSIDERATION
 : IF THE ERROR DOES NOT OCCUR WITHIN THE TEST UNDER CONSIDERATION
 : THE PROGRAM WILL REVERT TO NORMAL OPERATION. HOWEVER, IF LOOP ON
 : TEST SWITCH IS SET AND THIS OPERATOR SELECTABLE SCOPE LOOP IS USED
 : THEN THE PROGRAM WILL LOOP BACK TO THE SELECTED POINT WHEN IT
 : COMES TO THE END OF THE TEST UNDER CONSIDERATION.
 : AFTER LOOPING FOR SOME TIME IF THE LOOP SWITCH IS PUT DOWN THEN
 : NORMAL OPERATION WILL CONTINUE.

TESTAD: 0 ; FIRST ADDRESS OF TEST
 OPERSEL:
 CLR PS ; MAKE PROCESSOR STATUS ZERO
 MOV #-1, 2#PRITEM ; CLEAR PREVIOUS ITEM NUMBER
 TYPE 65\$; TYPE ASCIZ STRING
 BR 64\$; GET OVER THE ASCIZ
 65\$: .ASCIZ <15><12>/THE PROGRAM WAS IN TEST NUMBER /
 64\$: MOV 2#TSTNM, -(SP) ; GET READY TO TYPE TEST
 TYPOC ; NUMBER
 TYPE 67\$; TYPE ASCIZ STRING
 BR 66\$; GET OVER THE ASCIZ
 67\$: .ASCIZ <15><12>/THE LOOP BACK PC WAS /
 66\$: MOV 2#SLPERR, -(SP) ; GET READY TO TYPE LOOP BACK PC
 TYPOC
 TYPE , \$CRLF ; TYPE ASCIZ STRING
 TYPE 69\$; GET OVER THE ASCIZ
 BR 68\$; GET OVER THE ASCIZ
 69\$: .ASCIZ <15><12>/SET SWITCH FOR LOOP ON ERROR OR LOOP ON TEST
 68\$: TYPE 71\$; TYPE ASCIZ STRING
 BR 70\$; GET OVER THE ASCIZ
 71\$: .ASCIZ <15><12>/TYPE THE FIRST PC OF THE TEST TO BE LOOPED ON
 70\$: TYPE 73\$; TYPE ASCIZ STRING
 BR 72\$; GET OVER THE ASCIZ
 73\$: .ASCIZ <15><12>/ FOLLOWED BY A CARRIAGE RETURN <15><12
 72\$: RDOCT
 ADD #2, (SP) ; GET LPADR
 MOV (SP)+, 2#SLPADR
 TYPE 75\$; TYPE ASCIZ STRING
 BR 74\$; GET OVER THE ASCIZ


```

13077
13078 041304
13079 041304 104401 041312
13080 041310 000440
13081
13082 041412
13083 041412 104410
13084 041414 012637 001110
13085 041420 013746 001106
13086 041424 000002
13087
13088
13089
13090
13091
13092
13093
13094
13095
13096
13097
13098
13099
13100 041426
13101 041426 010046
13102 041430 010146
13103 041432 010246
13104 041434 012700 004062
13105 041440 012701 000002
13106 041444 012702 000010
13107 041450 013021
13108 041452 005302
13109 041454 001375
13110 041456 012602
13111 041460 012601
13112 041462 012600
13113 041464 000207

```

```

::75S: .ASCIZ <15><12> TYPE THE PC WHERE YOU WANT/
74S:
TYPE 77S ::TYPE ASCIZ STRING
BR 76S ::GET OVER THE ASCIZ
::77S: .ASCIZ <15><12> THE PROGRAM TO LOOP BACK TO FOLLOWED BY A CARRIAGE RETURN //15
76S:
RDOCT
MOV (SP)+,2*SLPERR ;GET LPERP
MOV 2*SLPADR,-(SP)
RTI

```

```

;THIS SAVES THE CONTENTS OF ALL HARDWARE REGISTERS
;IN MEMORY LOCATIONS TAGED FROM "WC" TO "EC2"
;THIS IS DONE SO THAT COMPARES ARE DONE WITH SAVED LOCATIONS
;AND NOT THE REGISTERS THEMSELVES. THIS WILL MAKE
;ERROR PRINTOUTS FOR GOOD AND BAD DATA ALWAYS DIFFRENT

```

```

PUTREG:
MOV R0,-(SP) ::PUSH R0 ON STACK
MOV R1,-(SP) ::PUSH R1 ON STACK
MOV R2,-(SP) ::PUSH R2 ON STACK
MOV #RHC,R0 ;STARTING ADDRESS OF REG
MOV #WC,R1 ;STARTING ADDRESS OF WERE SAVED
MOV #RHC-RHC+2/2,R2 ;NUMBER OF REG. INTO R2
10S: MOV 2(R0)+,(R1)+ ;SAVE HARDWARE REG.
DEC R2
BNE 10S
MOV (SP)+,R2 ::POP STACK INTO R2
MOV (SP)+,R1 ::POP STACK INTO R1
MOV (SP)+,R0 ::POP STACK INTO R0
RTS PC

```

```

13114
13115      .SBTTL  RS DATA TRANSFER SUBROUTINE
13116
13117      ;*
13118      ;* THIS SUBROUTINE WRITES OR READS OR DOES A
13119      ;* WRITE CHECK ON CYLINDER 0 TRACK 0 SECTOR 0
13120      ;* IT ASSUMES THE RH REGISTERS ARE APPROPRIATELY
13121      ;* FILLED
13122      ;* WHEN IT RETURNS FROM THIS SUBROUTINE TO THE
13123      ;* MAIN PROGRAM IT MEANS THE TRANSFER IS COMPLETE
13124      ;*
13125      ;* THE CALL IS BY A JSR RD, @COMND COMAND
13126 041466 012037 004642      CMNDRS: MOV      (RD)+, COMAND      ;GET COMMAND
13127
13128      ;CHECK THAT RHDS1 HAS DRY!DPR!MOL
13129 041472 012737 010600 001124      MOV      @DRY!DPR!MOL, $GDDAT      ;GET GOOD = 10300
13130
13131 041500 017737 142366 001126      MOV      @RHDS1, $BDDAT      ;READ RHDS1 FOR COMPARISON
13132 041506 023737 001124 001126      CMP      $GDDAT, $BDDAT      ;COMPARE EXPECTED
13133      ;DATA WITH DATA READ FROM
13134      ;RHDS1
13135 041514 001401      BEQ      65$      ;BRANCH IF GOOD
13136 041516 104133      ERROR   133
13137
13138      ;BEFORE ANY COMMAND IS GIVEN
13139      ;RHDS1 SHOULD HAVE DRY!DPR!MOL
13140      ;=10300
13141      ;BUT CONTAINED WHAT IS
13142      ;GIVEN IN BAD RHDS1
13143
13144 041520      65$:
13145 041520 005077 142342      CLR      @RHDA      ;TRANSFER ON CYLINDER 0
13146      ;TRANSFER ON SECTOR 0
13147      ;TRACK 0
13148 041524 017777 143112 142326      MOV      @COMAND, @RHCS1      ;SET UP COMMAND
13149 041532 005737 004644      TST     NOGO      ;IS GO TO BE GIVEN
13150 041536 001006      BNE     1$      ;BRANCH IF NO
13151 041540 052777 000001 142312      BIS     @GO, @RHCS1      ;SET GO
13152
13153      ;WAIT FOR RDY BIT IN RHCS1 REGISTER TO SET
13154 041546 104411      WAT     ;TRAP TO WAIT.T SUBROUTINE
13155 041550 004060      RHCS1   ;AND WAIT FOR RDY BIT IN
13156 041552 000200      RDY     ;RHCS1 REGISTER
13157      ;IF ERROR OCCURS HERE
13158      ;IT MEANS "RDY" DID NOT
13159      ;SET FOR THE FULL COUNT
13160      ;DOWN OF THE WAIT.T SUBROUTINE
13161      ;THIS TIME IS APPROXIMATELY
13162 041554 000200      1$:    RTS     RD      ;LARGER THAN 500 MILLISECONDS
13163
13164      .SBTTL  RPO4 DATA TRANSFER SUBROUTINE
13165
13166      ;*
13167      ;* THIS SUBROUTINE WRITE/READ/WRITE CHECK ON THE RPO4 CYLINDER 0,
13168      ;* TRACK 0, SECTOR 0.
13169      ;* IT ASSUMES THE RH REGISTERS ARE APPROPRIATELY FILLED.
13170      ;* WHEN IT RETURNS FROM THIS SUBROUTINE TO THE MAIN

```

```

13170          : * PROGRAM IT MEANS THE COMMAND IS COMPLETE
13171          : *
13172          : * THE CALL IS BY A JSR RD, @COMND COMMAND.
13173
13174 041556 012037 004642 CMNDRP: MOV      (RD)+, COMAND      ; GET COMMAND
13175
13176 041562 013746 004172      MOV      PKACK, -(SP)      ; GET READY TO SET VV
13177 041566 052716 000001      BIS      #GO, (SP)      ; GET PACK ACKNOWLEDGE COMMAND
13178 041572 012677 142262      MOV      (SP)+, @RHCS1    ; SET VV
13179
13180          ; CHECK THAT RHDS1 HAS VV! DRY! DPR! MOL
13181 041576 012737 010700 001124      MOV      #VV! DRY! DPR! MOL, $GDDAT ; GET GOOD = 10700
13182
13183 041604 017737 142262 001126      MOV      @RHDS1, $BDDAT    ; READ RHDS1 FOR COMPARISON
13184 041612 023737 001124 001126      CMP      $GDDAT, $BDDAT    ; COMPARE EXPECTED
13185          ; DATA WITH DATA READ FROM
13186          ; RHDS1
13187 041620 001401          BEQ      65$              ; BRANCH IF GOOD
13188 041622 104132          ERROR 132
13189          ; BEFORE ANY COMMAND IS GIVEN
13190          ; RHDS1 SHOULD HAVE VV! DRY! DPR! MOL
13191          ; =10700
13192          ; BUT CONTAINED WHAT IS
13193          ; GIVEN IN BAD RHDS1
13194 041624          65$:
13195
13196 041624 005077 142264      CLR      @RHCA            ; TRANSFER ON CYLINDER 0
13197 041630 005077 142232      CLR      @RHDS1         ; TRANSFER ON SECTOR 0
13198          ; TRACK 0
13199 041634 012777 014000 142250      MOV      #ECI! FMT22, @RHOF ; INHIBIT ECC
13200          ; FORMAT 16 BIT PER WORD
13201 041642 017777 142774 142210      MOV      @COMAND, @RHCS1  ; SET UP COMMAND
13202 041650 005737 004644          TST      NOGO            ; IS GO TO BE GIVEN
13203 041654 001006          BNE      1$              ; BRANCH IF NO
13204 041656 052777 000001 142174      BIS      #GO, @RHCS1     ; SET GO
13205
13206          ; WAIT FOR RDY BIT IN RHCS1 REGISTER TO SET
13207 041664 104411          WAT                      ; TRAP TO WAIT.T SUBROUTINE
13208 041666 004060          RHCS1                  ; AND WAIT FOR RDY BIT IN
13209 041670 000200          RDY                    ; RHCS1 REGISTER
13210          ; IF ERROR OCCURS HERE
13211          ; IT MEANS "RDY" DID NOT
13212          ; SET FOR THE FULL COUNT
13213          ; DOWN OF THE WAIT.T SUBROUTINE
13214          ; THIS TIME IS APPROXIMATELY
13215          ; LARGER THAN 500 MILLISECONDS
13216 041672 000200          1$: RTS      RD
13217
13218          .SBTTL TMO2 DATA TRANSFER SUBROUTINE
13219
13220          : * THIS SUBROUTINE WRITES/READS/WRITE CHECKS ON THE TMO2-TUIE
13221          : * ON THE FIRST BLOCK FROM BEGINNING OF TAPE (BOT)
13222          : * IT ASSUMES THE RH REGISTERS ARE APPROPRIATELY FILLED.
13223          : * WHEN IT RETURNS FROM THIS SUBROUTINE TO THE
13224          : * MAIN PROGRAM IT MEANS THE COMMAND IS COMPLETE.
13225          : *

```

E04

CERHADD MACY11 3011046, 21-DEC-77 13:06 PAGE 251
 CERHAD.F11 21-DEC-77 12:48

*M02 DATA TRANSFER SUBROUTINE

EE 0250

```

13226          : *      THE CALL IS
13227          : *      JSR      RO,COMND
13228
13229 041674 012037 004642          CMND* M: MOV      (RO)+,COMAND
13230 041700 013777 004660 142204  MOV      SLAVE,DRHTC          ;GET SLAVE NUMBER
13231 041706 012737 010600 001124  MOV      #MOL!DPR!DRY,$GDDAT ;GET GOOD DATA
13232 041714 017737 142152 001126  MOV      DRHLS1,$BDDAT       ;GET RHDS1
13233 041722 042737 000062 001126  BIC      #BOT!BIT05!BIT04,$BDDAT ;DISREGARD BOT
13234 041730 023737 001124 001126  CMP      $GDDAT,$BDDAT       ;COMPARE EXPECTED DATA WITH
13235                                     ;DATA READ FROM RHDS1
13236 041736 001401                                     ;BRANCH IF GOOD
13237 041740 104134          ERROR 134          ;BEFORE ANY COMMAND
13238                                     ;IS GIVEN DRIVE STATUS REGISTER
13239                                     ;DID NOT CONTAIN WHAT IS IN
13240                                     ;GOOD
13241 041742 032777 000002 142122 15:  BIT      #BOT,DRHDS1          ;IS IT ALREADY AT BOT
13242 041750 001035          BNE      2$                    ;BRANCH IF YES
13243                                     ;IF NOT AT BOT GIVE A REWIND COMMAND
13244 041752 013777 004176 142100  MOV      REWIND,DRHCS1        ;LOAD RHCS1 WITH REWIND COMMAND
13245 041760 052777 000001 142072  BIS      #GO,DRHCS1
13246                                     ;WAIT FOR PIP BIT IN RHDS1 REGISTER TO SET
13247 041766 104411          WAT                                     ;TRAP TO WAIT.T SUBROUTINE
13248 041770 004072          RHDS1          ;AND WAIT FOR PIP BIT IN
13249 041772 020000          PIP          ;RHDS1 REGISTER
13250                                     ;IF ERROR OCCURS HERE
13251                                     ;IT MEANS "PIP" DID NOT
13252                                     ;SET FOR THE FULL COUNT
13253                                     ;DOWN OF THE WAIT.T SUBROUTINE
13254                                     ;THIS TIME IS APPROXIMATELY
13255                                     ;LARGER THAN 500 MILLISECONDS
13256                                     ;WAIT FOR PIP BIT IN RHDS1 REGISTER TO SET
13257 041774 012737 177777 004646  MOV      #-1,WATO          ;WAIT FOR PIP TO GO FROM 1 TO 0
13258 042002 104411          WAT                                     ;TRAP TO WAIT.T SUBROUTINE
13259 042004 004072          RHDS1          ;AND WAIT FOR PIP BIT IN
13260 042006 020000          PIP          ;RHDS1 REGISTER
13261                                     ;IF ERROR OCCURS HERE
13262                                     ;IT MEANS "PIP" DID NOT
13263                                     ;SET FOR THE FULL COUNT
13264                                     ;DOWN OF THE WAIT.T SUBROUTINE
13265                                     ;THIS TIME IS APPROXIMATELY
13266                                     ;LARGER THAN 500 MILLISECONDS
13267 042010 005037 004646  CLR      WATO
13268                                     ;WAIT FOR BOT BIT IN RHDS1 REGISTER TO SET
13269 042014 104411          WAT                                     ;TRAP TO WAIT.T SUBROUTINE
13270 042016 004072          RHDS1          ;AND WAIT FOR BOT BIT IN
13271 042020 000002          BOT          ;RHDS1 REGISTER
13272                                     ;IF ERROR OCCURS HERE
13273                                     ;IT MEANS "BOT" DID NOT
13274                                     ;SET FOR THE FULL COUNT
13275                                     ;DOWN OF THE WAIT.T SUBROUTINE
13276                                     ;THIS TIME IS APPROXIMATELY
13277                                     ;LARGER THAN 500 MILLISECONDS
13278
13279                                     ;CLEAR ATTENTIONS
13280 042022 013746 004142  MOV      DCLEAR,-(SP)
13281 042026 052716 000001  BIS      #GO,(SP)
  
```

```

13282 042032 012677 142022      MOV      (SP)+,DRHCSI
13283
13284      ;WAIT FOR DRY BIT IN RHDS1 REGISTER TO SET
13285 042036 104411      WAT      ;TRAP TO WAIT.T SUBROUTINE
13286 042040 004072      RHDS1   ;AND WAIT FOR DRY BIT IN
13287 042042 000200      DRY     ;RHDS1 REGISTER
13288      ;IF ERROR OCCURS HERE
13289      ;IT MEANS "DRY" DID NOT
13290      ;SET FOR THE FULL COUNT
13291      ;DOWN OF THE WAIT.T SUBROUTINE
13292      ;THIS TIME IS APPROXIMATELY
13293      ;LARGER THAN 500 MILLISECONDS
13294 042044      2$:    ;CHECK IF COMMAND IS REVERSE READ
13295      ;IF IT IS REVERSE READ, SPACE FORWARD ONE BLOCK
13296 042044 023777 004200 142570    CMP      REVRED,DCOMAND ;IS IT REVERSE READ
13297 042052 001035      BNE     3$ ;BRANCH IF NOT
13298 042054 052777 001700 142030    BIS     #BPI8:NML,DRHTC ;800 BPI, NORMAL
13299 042062 042777 000010 142022    BIC     #EPAR,DRHTC ;ODD PARITY
13300
13301 042070 012777 177777 141770    MOV     #-1,DRHFC ;FRAME COUNT 1
13302      ;FOR SPACE FORWARD ONE BLOCK
13303 042076 013777 004202 141754    MOV     SPACFD,DRHCSI ;SPACE FORWARD
13304 042104 052777 000001 141746    BIS     #GO,DRHCSI ;GO
13305      ;WAIT FOR PIP BIT IN RHDS1 REGISTER TO SET
13306 042112 104411      WAT      ;TRAP TO WAIT.T SUBROUTINE
13307 042114 004072      RHDS1   ;AND WAIT FOR PIP BIT IN
13308 042116 020000      PIP     ;RHDS1 REGISTER
13309      ;IF ERROR OCCURS HERE
13310      ;IT MEANS "PIP" DID NOT
13311      ;SET FOR THE FULL COUNT
13312      ;DOWN OF THE WAIT.T SUBROUTINE
13313      ;THIS TIME IS APPROXIMATELY
13314      ;LARGER THAN 500 MILLISECONDS
13315      ;WAIT FOR PIP BIT IN RHDS1 REGISTER TO SET
13316 042120 012737 177777 004646    MOV     #1,WATO ;WAIT FOR PIP TO GO FROM 1 TO 0
13317 042126 104411      WAT      ;TRAP TO WAIT.T SUBROUTINE
13318 042130 004072      RHDS1   ;AND WAIT FOR PIP BIT IN
13319 042132 020000      PIP     ;RHDS1 REGISTER
13320      ;IF ERROR OCCURS HERE
13321      ;IT MEANS "PIP" DID NOT
13322      ;SET FOR THE FULL COUNT
13323      ;DOWN OF THE WAIT.T SUBROUTINE
13324      ;THIS TIME IS APPROXIMATELY
13325      ;LARGER THAN 500 MILLISECONDS
13326 042134 005037 004646      CLR     WATO
13327      ;WAIT FOR DRY BIT IN RHDS1 REGISTER TO SET
13328 042140 104411      WAT      ;TRAP TO WAIT.T SUBROUTINE
13329 042142 004072      RHDS1   ;AND WAIT FOR DRY BIT IN
13330 042144 000200      DRY     ;RHDS1 REGISTER
13331      ;IF ERROR OCCURS HERE
13332      ;IT MEANS "DRY" DID NOT
13333      ;SET FOR THE FULL COUNT
13334      ;DOWN OF THE WAIT.T SUBROUTINE
13335      ;THIS TIME IS APPROXIMATELY
13336      ;LARGER THAN 500 MILLISECONDS
13337 042146      3$:    ;NOW GIVE COMMAND
    
```

13338	042146	052777	001700	141736	BIS	#BPI8,NML,DRHTC	:800 BPI, NORMAL
13339	042154	042777	000010	141730	BIC	#EPAR,DRHTC	:ODD PARITY
13340	042162	017746	141674		MOV	DRHWC,-(SP)	:GET WORD COUNT
13341	042166	005416			NEG	(SP)	:GET POSITIVE NUMBER
13342	042170	061616			ADD	(SP),(SP)	:DOUBLE WORD COUNT
13343	042172	005416			NEG	(SP)	:GET NEGATIVE NUMBER FOR
13344							:FRAME COUNT REGISTER
13345	042174	012677	141666		MOV	(SP)+,DRHFC	:FILL FRAME COUNT
13346	042200	017777	142436	141652	MOV	DCOMAND,DRHCS1	:GET COMMAND
13347							
13348	042206	005737	004644		TST	NOGO	:IS GO TO BE GIVEN
13349	042212	001006			BNE	45	:BRANCH IF NO
13350	042214	052777	000001	141636	BIS	#GO,DRHCS1	:GO
13351							
13352							:WAIT FOR DRY BIT IN RHDS1 REGISTER TO SET
13353	042222	104411			WAT		:TRAP TO WAIT.T SUBROUTINE
13354	042224	004072			RHDS1		:AND WAIT FOR DRY BIT IN
13355	042226	000200			DRY		:RHDS1 REGISTER
13356							:IF ERROR OCCURS HERE
13357							:IT MEANS "DRY" DID NOT
13358							:SET FOR THE FULL COUNT
13359							:DOWN OF THE WAIT.T SUBROUTINE
13360							:THIS TIME IS APPROXIMATELY
13361							:LARGER THAN 500 MILLISECONDS
13362	042230						
13363					45:		
13364	042230	104411					:WAIT FOR RDY BIT IN RHCS1 REGISTER TO SET
13365	042232	004060			WAT		:TRAP TO WAIT.T SUBROUTINE
13366	042234	000200			RHCS1		:AND WAIT FOR RDY BIT IN
13367					RDY		:RHCS1 REGISTER
13368							:IF ERROR OCCURS HERE
13369							:IT MEANS "RDY" DID NOT
13370							:SET FOR THE FULL COUNT
13371							:DOWN OF THE WAIT.T SUBROUTINE
13372							:THIS TIME IS APPROXIMATELY
13373	042236	000200					:LARGER THAN 500 MILLISECONDS
					RTS	RC	

```

13374 042240 104401 042246
13375 042240 000424
13376 042240
13377 042240
13378 042240
13379 042240
13380 042240 104401 042246
13381 042240 000424
13382 042316
13383 042316 013746 004060
13384 042322 104402
13385 042324 104401 042332
13386 042330 000425
13387
13388 042404
13389 042404 104410
13390 042406 012700 004102
13391 042412 012701 000024
13392 042416 042710 177700
13393 042422 051620
13394 042424 005301
13395 042426 001373
13396 042430 104401 042436
13397 042434 000417
13398
13399 042474
13400 042474 013746 002716
13401 042500 104402
13402 042502 104401 042510
13403 042506 000437
13404
13405 042606
13406 042606 104410
13407 042610 012637 002716
13408 042614 104401 042622
13409 042620 000421
13410
13411 042664
13412 042664 104401 042672
13413 042670 000414
13414
13415 042722
13416 042722 013746 004060
13417 042726 104402
13418 042730 104401 042736
13419 042734 000415
13420
13421 042770
13422 042770 013746 002716
13423 042774 104402
13424 042776 104401 043004
13425 043002 000416
13426
13427 043040
13428 043040 000000
13429

```

```

* THIS ROUTINE WILL ALLOW THE CHANGE OF THE BASE
* ADDRESS FROM 17E700 TO ANY TYPED VALUE

BASECH:
TYPE 65$          ;; TYPE ASCIZ STRING
BR 64$           ;; GET OVER THE ASCIZ
65$: .ASCIZ <15><12>PRESENT BASE ADDRESS OF REGISTERS IS
64$: MOV 2#RHCS1,-(SP) ;GET READY TO TYPE OLD BASE
      TYPOC
      TYPE 67$          ;; TYPE ASCIZ STRING
      BR 66$           ;; GET OVER THE ASCIZ
67$: .ASCIZ <15><12>TYPE NEW BASE ADDRESS FOLLOWED BY 'CR'
66$: RDOCT
      MOV 2#RHD8,RO      ;GET STARTING ADDRESS OF REGISTERS
      MOV 20,R1          ;NUMBER OF REGISTERS
1$: BIC 2#C7,(RO)       ;CLEAR OLD BASE
     BIS (SP),(RO)+     ;SET NEW BASE
     DEC R1              ;COUNT
     BNE 1$             ;BRANCH IF 20 NOT DONE
     TYPE 69$          ;; TYPE ASCIZ STRING
     BR 68$           ;; GET OVER THE ASCIZ
69$: .ASCIZ <15><12>PRESENT VECTOR ADDRESS IS /
68$: MOV 2#RPVEC,-(SP) ;GET READY TO TYPE OLD VECTOR ADDRESS
      TYPOC
      TYPE 71$          ;; TYPE ASCIZ STRING
      BR 70$           ;; GET OVER THE ASCIZ
71$: .ASCIZ <15><12>TYPE NEW VECTOR ADDRESS OR RETYPE OLD ONE FOLLOWED BY "CR"
70$: RDOCT
      MOV (SP)+,2#RPVEC ;SETUP VECTOR ADDRESS
      TYPE 73$          ;; TYPE ASCIZ STRING
      BR 72$           ;; GET OVER THE ASCIZ
73$: .ASCIZ <15><12>RESTART PROGRAM FROM 200 OR 210
72$: TYPE 75$          ;; TYPE ASCIZ STRING
      BR 74$           ;; GET OVER THE ASCIZ
75$: .ASCIZ <15><12>NEW BASE WILL REMAIN/
74$: MOV 2#RHCS1,-(SP)
      TYPOC
      TYPE 77$          ;; TYPE ASCIZ STRING
      BR 76$           ;; GET OVER THE ASCIZ
77$: .ASCIZ <15><12>NEW VECTOR WILL REMAIN /
76$: MOV 2#RPVEC,-(SP)
      TYPOC
      TYPE 79$          ;; TYPE ASCIZ STRING
      BR 78$           ;; GET OVER THE ASCIZ
79$: .ASCIZ <15><12>UNTIL PROGRAM IS RELOADED
78$: HALT

```

```

:0430 043042
:0431 043042 104401 043050
:0432 043046 000411
:0433
:0434 043072
:0435 043072 104402
:0436 043074 012777 043042 137614
:0437 043102 000000
:0438
:0439
:0440
:0441
:0442

```

```

RPVECT:
TYPE 655 ::TYPE ASCIZ STRING
BR 648 ::GET OVER THE ASCIZ
::655: .ASCIZ /TRAPED FROM PC =
648:
TYP0C :TYPE FROM PC
MOV 0RPVECT,0RPVEC :RESTORE TRAP RPO4 VECTOR
HALT :CHANGE TO CONTINUE

```

13442
13443
13444
13445
13446
13447
13448
13449
13450
13451
13452
13453
13454
13455
13456
13457
13458
13459
13460
13461
13462
13463
13464
13465
13466
13467
13468
13469
13470
13471
13472
13473
13474
13475
13476
13477
13478
13479
13480
13481
13482
13483
13484
13485
13486
13487
13488
13489
13490
13491
13492
13493
13494
13495
13496
13497

043104
043104 005037 004646
043110 005037 004644
043114 032777 040000 136016
043122 001111
043124 000416
043126 013746 000004
043132 012737 043152 000004
043140 005737 177060
043144 012637 000004
043150 000463
043152 022626
043154 012637 000004
043160 000423
043162
043162 032777 000400 135750
043170 001404
043172 127737 135742 001102
043200 001462
043202 105737 001103
043206 001421
043210 123737 001115 001103
043216 101015
043220 032777 001000 135712
043226 001404
043230 013737 001110 001106
043236 000443
043240 105037 001100
043244 005037 001212
043250 000415
043252 032777 004000 135660
043260 001011
043262 005737 001100
043266 001406
043270 005237 001104
043274 023737 001212 001104
043302 002021
043304 012737 000001 001104

.SBTTL SCOPE HANDLER ROUTINE

: THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
: AND LOAD THE TEST NUMBER(\$TSTNM) INTO THE DISPLAY REG. (DISPLAY 7:0)
: AND LOAD THE ERROR FLAG (\$ERFLG) INTO DISPLAY 15:08
: THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
: SW14=1 LOOP ON TEST
: SW11=1 INHIBIT ITERATIONS
: SW09=1 LOOP ON ERROR
: SW08=1 LOOP ON TEST IN SWR<7:0>
: CALL
: * SCOPE ;:SCOPE=IOT

\$SCOPE: CLR WATO ;WAIT FOR BIT TO SET
;USED IN "WAT" WAIT ROUTINE
CLR NOGO ;DO GIVE GO IN COMMAND ROUTINE
1\$: BIT #BIT14,\$SWR ;LOOP ON PRESENT TEST?
BNE \$OVER ;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
\$XTSTR: BR 6\$;IF RUNNING ON THE "XOR" TESTER CHANGE
;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV #ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV #SS,\$ERRVEC ;SET FOR TIMEOUT
TST #177060 ;TIME OUT ON XOR?
MOV (SP)+,\$ERRVEC ;RESTORE THE ERROR VECTOR
BR \$SVLAD ;GO TO THE NEXT TEST
5\$: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,\$ERRVEC ;RESTORE THE ERROR VECTOR
BR 7\$;LOOP ON THE PRESENT TEST
6\$: ;*****END OF CODE FOR THE XOR TESTER*****
BIT #BIT08,\$SWR ;LOOP ON SPEC. TEST?
BEQ 2\$;BR IF NO
CMPB \$SWR,\$TSTNM ;ON THE RIGHT TEST? SWR 7:0
BEQ \$OVER ;BR IF YES
2\$: TSTB \$ERFLG ;HAS AN ERROR OCCURRED?
BEQ 3\$;BR IF NO
CMPB \$ERMAX,\$ERFLG ;MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 3\$;BR IF NO
BIT #BIT09,\$SWR ;LOOP ON ERROR?
BEQ 4\$;BR IF NO
7\$: MOV \$LPERR,\$LPADR ;SET LOOP ADDRESS TO LAST SCOPE
BR \$OVER
4\$: CLRB \$ERFLG ;ZERO THE ERROR FLAG
CLR \$TIMES ;CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 1\$;ESCAPE TO THE NEXT TEST
3\$: BIT #BIT11,\$SWR ;INHIBIT ITERATIONS?
BNE 1\$;BR IF YES
TST \$PASS ;IF FIRST PASS OF PROGRAM
BEQ 1\$;INHIBIT ITERATIONS
INC \$ICNT ;INCREMENT ITERATION COUNT
CMP \$TIMES,\$ICNT ;CHECK THE NUMBER OF ITERATIONS MADE
BGE \$OVER ;BR IF MORE ITERATION REQUIRED
1\$: MOV #1,\$ICNT ;REINITIALIZE THE ITERATION COUNTER

13498	043312	013737	043362	001212		MOV	\$MXCNT,\$TIMES	:: SET NUMBER OF ITERATIONS TO DO
13499	043320	105237	001102		\$SVLAD:	INCB	\$TSTNM	:: COUNT TEST NUMBERS
13500	043324	011637	001106			MOV	(SP), \$LPADR	:: SAVE SCOPE LOOP ADDRESS
13501	043330	011637	001110			MOV	(SP), \$LPERR	:: SAVE ERROR LOOP ADDRESS
13502	043334	005037	001214			CLR	\$ESCAPE	:: CLEAR THE ESCAPE FROM ERROR ADDRESS
13503	043340	112737	000001	001115		MOVB	#1, \$ERMAX	:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
13504	043346	013777	001102	135566	\$OVER:	MOV	\$TS NM, @DISPLAY	:: DISPLAY TEST NUMBER
13505	043354	013716	001106			MOV	\$LPADR, (SP)	:: FUDGE RETURN ADDRESS
13506	043360	000002				RTI		:: FIXES PS
13507	043362	000004			\$MXCNT:	4		:: MAX. NUMBER OF ITERATIONS

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

13508
13509
13510
13511
13512
13513
13514
13515
13516
13517
13518
13519
13520
13521
13522
13523
13524
13525
13526
13527
13528
13529
13530
13531
13532
13533
13534
13535
13536
13537
13538
13539
13540
13541
13542
13543
13544
13545
13546
13547
13548
13549
13550
13551
13552
13553
13554
13555
13556
13557
13558
13559
13560
13561
13562
13563

043364
043364 010046
043366 010146
043370 010246
043372 010346
043374 010546
043376 012746 020200
043402 016605 000020
043406 100004
043410 005405
043412 112766 000055 000001
043420 005000
043422 012703 043600
043426 112723 000040
043432 005002
043434 016001 043570
043440 160105
043442 002402
043444 005202
043446 000774
043450 060105
043452 005702
043454 001002
043456 10571E
043460 100407
043462 106316
043464 103003
043466 116663 000001 177777
043474 052702 000060
043500 052702 000040
043504 110223
043506 005720
043510 020027 000010
043514 002746
043516 003002
043520 010502
043522 000764
043524 105726
043526 100003
043530 116663 177777 177776
043536 105013
043540 012605
043542 012603
043544 012602

```
*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
*REPLACED WITH SPACES.
*CALL:
*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
*      TYPDS    ;;GO TO THE ROUTINE

$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB    #'-',1(SP)     ;;MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0            ;;ZERO THE CONSTANTS INDEX
MOV      #5DBLK,R3     ;;SETUP THE OUTPUT POINTER
MOVB    #'',(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2            ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
3$:      SUB      R1,R5        ;;FORM THIS BCD DIGIT
BLT     4$            ;;BR IF DONE
INC     R2            ;;INCREASE THE BCD DIGIT BY 1
BR      3$
4$:      ADD     R1,R5        ;;ADD BACK THE CONSTANT
TST     R2            ;;CHECK IF BCD DIGIT=0
BNE     5$            ;;FALL THROUGH IF 0
TSTB   (SP)           ;;STILL DOING LEADING 0'S
BMI     7$            ;;BR IF YES
5$:      ASLB    (SP)         ;;MSD?
BCC     6$            ;;BR IF NO
MOVB   1(SP),-1(R3)   ;;YES--SET THE SIGN
BIS    #'0,R2        ;;MAKE THE BCD DIGIT ASCII
BIS    #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST    (R0)+         ;;JUST INCREMENTING
CMP    R0,#10        ;;CHECK THE TABLE INDEX
BLT    2$            ;;GO DO THE NEXT DIGIT
BGT    8$            ;;GO TO EXIT
MOV    R5,R2         ;;GET THE LSD
BR     6$            ;;GO CHANGE TO ASCII
8$:      TSTB   (SP)+       ;;WAS THE LSD THE FIRST NON-ZERO?
BPL    9$            ;;BR IF NO
MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
9$:      CLRB   (R3)         ;;SET THE TERMINATOR
MOV    (SP)+,R5      ;;POP STACK INTO R5
MOV    (SP)+,R3      ;;POP STACK INTO R3
MOV    (SP)+,R2      ;;POP STACK INTO R2
```

M04

CERHADO MACY11 301046, 21-DEC-77 13:06 PAGE 259
 CERHAD.P11 21-DEC-77 12:48

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0258

```

13564 043546 012601          MOV      (SP)+,R1          ;; POP STACK INTO R1
13565 043550 012600          MOV      (SP)+,R0          ;; POP STACK INTO R0
13566 043552 104401 043600    TYPE      $DBLK           ;; NOW TYPE THE NUMBER
13567 043556 016666 000002 000004  MOV      2(SP),4(SP)      ;; ADJUST THE STACK
13568 043564 012616          MOV      (SP)+,(SP)
13569 043566 000002          RTI                          ;; RETURN TO USER
13570 043570 023420          $DTBL: 10000.
13571 043572 001750          1000.
13572 043574 000144          100.
13573 043576 000012          10.
13574 043600 000004          $DBLK: .BLKW 4
13575                                     .SBTTL TYPE ROUTINE
13576
13577                                     ;*****
13578                                     ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
13579                                     ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
13580                                     ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
13581                                     ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
13582                                     ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
13583                                     ;*
13584                                     ;*CALL:
13585                                     ;*1) USING A TRAP INSTRUCTION
13586                                     ;*   TYPE      ,MESADR          ;; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
13587                                     ;*OR
13588                                     ;*   TYPE
13589                                     ;*   MESADR
13590                                     ;*
13591
13592 043610 105737 001157    $TYPE: TSTB      $TFPLG          ;; IS THERE A TERMINAL?
13593 043614 100002          BPL      1$                ;; BR IF YES
13594 043616 000000          HALT                    ;; HALT HERE IF NO TERMINAL
13595 043620 000407          BR      3$                ;; LEAVE
13596 043622 010046          1$: MOV      RO,-(SP)        ;; SAVE RO
13597 043624 017600 000002    MOV      2(SP),R0         ;; GET ADDRESS OF ASCIZ STRING
13598 043630 112046          2$: MOV      (R0)+,-(SP)   ;; PUSH CHARACTER TO BE TYPED ONTO STACK
13599 043632 001005          BNE      4$                ;; BR IF IT ISN'T THE TERMINATOR
13600 043634 005726          TST      (SP)+           ;; IF TERMINATOR POP IT OFF THE STACK
13601 043636 012600          60$: MOV      (SP)+,R0     ;; RESTORE RO
13602 043640 062716 000002    3$: ADD      #2,(SP)        ;; ADJUST RETURN PC
13603 043644 000002          RTI                          ;; RETURN
13604 043646 122716 000011    4$: CMP      #HT,(SP)      ;; BRANCH IF <HT>
13605 043652 001430          BEQ      8$                ;; BRANCH IF NOT <CRLF>
13606 043654 122716 000200    CMP      #CRLF,(SP)
13607 043660 001006          BNE      5$                ;; POP <CR><LF> EQUIV
13608 043662 005726          TST      (SP)+           ;; TYPE A CR AND LF
13609 043664 104401          TYPE
13610 043666 001223          $CRLF
13611 043670 105037 044024    CLRB     $CHARCNT        ;; CLEAR CHARACTER COUNT
13612 043674 000755          BR      2$                ;; GET NEXT CHARACTER
13613 043676 004737 043760    5$: JSR      PC,$TYPE      ;; GO TYPE THIS CHARACTER
13614 043702 123726 001156    6$: CMP      $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
13615 043706 001350          BNE      2$                ;; IF NO GO GET NEXT CHAR.
13616 043710 013746 001154    MOV      $NULL,-(SP)     ;; GET # OF FILLER CHARS. NEEDED
13617                                     ;; AND THE NULL CHAR.
13618 043714 105366 000001    7$: DECB     1(SP)         ;; DOES A NULL NEED TO BE TYPED?
13619 043720 002770          BLT      6$                ;; BR IF NO--GO POP THE NULL OFF OF STACK
    
```

```

13620 043722 004737 043760 JSR PC,$TYPEC ;:GO TYPE A NULL
13621 043726 105337 044024 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
13622 043732 000770 BR 7$ ;:LOOP
13623
13624 ;HORIZONTAL TAB PROCESSOR
13625
13626 043734 112716 000040 8$: MOVB #' (SP) ;:REPLACE TAB WITH SPACE
13627 043740 004737 043760 9$: JSR PC,$TYPEC ;:TYPE A SPACE
13628 043744 132737 000007 044024 BITB #',$CHARCNT ;:BRANCH IF NOT AT
13629 043752 001372 BNE 9$ ;:TAB STOP
13630 043754 005726 TST (SP)+ ;:POP SPACE OFF STACK
13631 043756 000724 BR 2$ ;:GET NEXT CHARACTER
13632 043760 105777 135164 $TYPEC: TSTB $STPS ;:WAIT UNTIL PRINTER IS READY
13633 043764 100375 BPL $TYPEC
13634 043766 116677 000002 135156 MOVB 2(SP),$STPB ;:LOAD CHAR TO BE TYPED INTO DATA REG.
13635 043774 122766 000015 000002 CMPB #'CR,2(SP) ;:IS CHARACTER A CARRIAGE RETURN?
13636 044002 001003 BNE 1$ ;:BRANCH IF NO
13637 044004 105037 044024 CLRB $CHARCNT ;:YES--CLEAR CHARACTER COUNT
13638 044010 000406 BR $TYPEX ;:EXIT
13639 044012 122766 000012 000002 1$: CMPB #'LF,2(SP) ;:IS CHARACTER A LINE FEED?
13640 044020 001402 BEQ $TYPEX ;:BRANCH IF YES
13641 044022 105227 INCB (PC)+ ;:COUNT THE CHARACTER
13642 044024 000000 $CHARCNT: .WORD 0 ;:CHARACTER COUNT STORAGE
13643 044026 000207 $TYPEX: RTS PC
13644
13645 .SBTTL TTY INPUT ROUTINE
13646
13647 ;:*****
13648 ;:ENABL LSB
13649 044030 000000 $TKCNT: .WORD 0 ;:NUMBER OF ITEMS IN QUEUE
13650 044032 000000 $TKQIN: .WORD 0 ;:INPUT POINTER
13651 044034 000000 $TKQOUT: .WORD 0 ;:OUTPUT POINTER
13652 044036 000011 $TKQSRT: .BLKB 9. ;:TTY KEYBOARD QUEUE
13653 044047 $TKQEND=.
13654 044050 .EVEN
13655
13656 ;*TK INITIALIZE ROUTINE
13657 ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
13658 ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
13659
13660 ;:CALL:
13661 ;* JSR PC,$TKINT
13662 ;* RETURN
13663
13664 044050 005037 044030 $TKINT: CLR $TKCNT ;:CLEAR COUNT OF ITEMS IN QUEUE
13665 044054 012737 044036 044032 MOV $TKQSRT,$TKQIN ;:MOVE THE STARTING ADDRESS OF THE
13666 044062 013737 044032 044034 MOV $TKQIN,$TKQOUT ;:QUEUE INTO THE INPUT & OUTPUT POINTERS.
13667 044070 012737 044120 000060 MOV $TKSRV,$TKVEC ;:INITIALIZE THE KEYBOARD VECTOR
13668 044076 012737 000200 000062 MOV #200,$TKVEC+2 ;:"BR" LEVEL 4
13669 044104 005777 135036 TST $TKB ;:CLEAR DONE FLAG
13670 044110 012777 000100 135026 MOV #100,$TKS ;:ENABLE TTY KEYBOARD INTERRUPT
13671 044116 000207 RTS PC ;:RETURN TO CALLER
13672
13673 ;*TK SERVICE ROUTINE
13674 ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
13675 ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING

```

```

13676      ;*IT IN THE QUEUE.
13677      ;*IF THE CHARACTER IS A "CONTROL-C" (1C) $TKINT IS CALLED AND
13678      ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (OPERSEL)
13679
13680      044120 117746 135022      $TKSRV: MOVB    2$TKB, -(SP)      ;; PICKUP THE CHARACTER
13681      044124 042716 177600      BIC    #1C177, (SP)      ;; STRIP THE JUNK
13682      044130 021627 000003      CMP    (SP), #3          ;; IS IT A CONTROL C?
13683      044134 001007              BNE    1$                ;; BRANCH IF NO
13684      044136 104401 044527      TYPE   $CNTLC           ;; TYPE A CONTROL-C (1C)
13685      044142 004737 044050      JSR    PC, $TKINT       ;; INIT THE KEYBOARD
13686      044146 005726              TST    (SP)+            ;; CLEAN UP STACK
13687      044150 000137 040640      JMP    OPERSEL          ;; CONTROL C RESTART
13688
13689      044154
13690      044154 022737 000011 044030 1$:      CMP    #9, $TKCNT      ;; IS THE QUEUE FULL?
13691      044162 001004              BNE    3$                ;; BRANCH IF NO
13692      044164 104401 001216      TYPE   $BELL           ;; RING THE TTY BELL
13693      044170 005726              TST    (SP)+            ;; CLEAN CHARACTER OFF OF STACK
13694      044172 000451              BR     5$                ;; EXIT
13695      044174 021627 000023      3$:      CMP    (SP), #23       ;; IS IT A CONTROL-S?
13696      044200 001021              BNE    32$               ;; BRANCH IF NO
13697      044202 005077 134736      CLR    2$TKS           ;; DISABLE TTY KEYBOARD INTERRUPTS
13698      044206 005726              TST    (SP)+            ;; CLEAN CHAR OFF STACK
13699      044210 105777 134730      31$:     TSTB   2$TKS           ;; WAIT FOR A CHAR
13700      044214 100375              BPL    31$              ;; LOOP UNTIL ITS THERE
13701      044216 117746 134724      MOVB   2$TKB, -(SP)     ;; GET THE CHARACTER
13702      044222 042716 177600      BIC    #1C177, (SP)     ;; MAKE IT 7-BIT ASCII
13703      044226 022627 000021      CMP    (SP)+, #21       ;; IS IT A CONTROL-Q?
13704      044232 001366              BNE    31$              ;; BRANCH IF NO
13705      044234 012777 000100 134702      MOV    #100, 2$TKS      ;; REENABLE TTY KEYBOARD INTERRUPTS
13706      044242 000002              RTI                    ;; RETURN
13707      044244 005237 044030      32$:     INC    $TKCNT          ;; COUNT THIS CHARACTER
13708      044250 021627 000140      CMP    (SP), #140       ;; IS IT UPPER CASE?
13709      044254 002405              BLT    4$                ;; BRANCH IF YES
13710      044256 021627 000175      CMP    (SP), #175       ;; IS IT A SPECIAL CHAR?
13711      044262 003002              BGT    4$                ;; BRANCH IF YES
13712      044264 042716 000040      BIC    #40, (SP)        ;; MAKE IT UPPER CASE
13713      044270 112677 177536      4$:      MOVB   (SP)+, 2$TKQIN   ;; AND PUT IT IN QUEUE
13714      044274 005237 044032      INC    $TKQIN           ;; UPDATE THE POINTER
13715      044300 023727 044032 044047      CMP    $TKQIN, #2$TKQEND ;; GO OFF THE END?
13716      044306 001003              BNE    5$                ;; BRANCH IF NO
13717      044310 012737 044036 044032      MOV    #2$TKQSR, $TKQIN ;; RESET THE POINTER
13718      044316 000002      5$:      RTI                    ;; RETURN
13719
13720      .DSABL  LSB
13721
13722
13723      ;*****
13724      ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
13725      ;*CALL:
13726      ;*      RDCHR          ;; GET A CHARACTER FROM THE QUEUE
13727      ;*      RETURN HERE   ;; CHARACTER IS ON THE STACK
13728      ;*                  ;; WITH PARITY BIT STRIPPED OFF
13729
13730
13731      044320 011646      $RDCHR: MOV    (SP), -(SP)      ;; PUSH DOWN THE PC AND
    
```

```

13732 044322 016666 000004 000002      MOV     4(SP),2 SP      ;; THE PS
13733 044330 005066 000004      CLR     4(SP)          ;; GET READY FOR A CHARACTER
13734 044334 005046      CLR     -(SP)         ;; PUT NEW PS ON STACK
13735 044336 012746 044344      MOV     #64$,-(SP)    ;; PUT NEW PC ON STACK
13736 044342 000002      RTI                    ;; POP NEW PC AND PS
13737 044344      64$:
13738 044344 005737 044030 1$:      TST     $TKCNT        ;; WAIT ON A CHARACTER
13739 044350 001775      BEQ     1$           ;;
13740 044352 005337 044030      DEC     $TKCNT        ;; DECREMENT THE COUNTER
13741 044356 117766 177452 000004      MOVVB  0$TKQOUT,4(SP) ;; GET ONE CHARACTER
13742 044364 005237 044034      INC     $TKQOUT       ;; UPDATE THE POINTER
13743 044370 023727 044034 044047      CMP     $TKQOUT,#$TKQEND ;; DID IT GO OFF OF THE END?
13744 044376 001003      BNE     2$           ;; BRANCH IF NO
13745 044400 012737 044036 044034      MOV     #$TKQ$RT,$TKQOUT ;; RESET THE POINTER
13746 044406 000002      RTI                    ;; RETURN
13747
13748      ;*****
13749      ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
13750      ;CALL:
13751      ;      RDLIN                    ;; INPUT A STRING FROM THE TTY
13752      ;      RETURN HERE              ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
13753      ;                               ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
13754 044410 010346      $RDLIN: MOV     R3, -(SP) ;; SAVE R3
13755 044412 012703 044516 1$:      MOV     #$TTYIN,R3    ;; GET ADDRESS
13756 044416 022703 044527 2$:      CMP     #$TTYIN+9.,R3 ;; BUFFER FULL?
13757 044422 101405      BLOS   4$           ;; BR IF YES
13758 044424 104406      RDCHR                    ;; GO READ ONE CHARACTER FROM THE TTY
13759 044426 112613      MOVVB  (SP)+,(R3)     ;; GET CHARACTER
13760 044430 122713 000177 10$:     CMPB   #177,(R3)     ;; IS IT A RUBOUT
13761 044434 001003      BNE   3$           ;; SKIP IF NOT
13762 044436 104401 001222 4$:      TYPE  $QJES         ;; TYPE A '?'
13763 044442 000763      BR     1$           ;; CLEAR THE BUFFER AND LOOP
13764 044444 111337 044514 3$:      MOVVB  (R3),9$       ;; ECHO THE CHARACTER
13765 044450 104401 044514      TYPE  9$           ;;
13766 044454 122723 000015      CMPB   #15,(R3)+    ;; CHECK FOR RETURN
13767 044460 001356      BNE   2$           ;; LOOP IF NOT RETURN
13768 044462 105063 177777      CLRB  -1(R3)        ;; CLEAR RETURN (THE 15)
13769 044466 104401 001224      TYPE  $LF           ;; TYPE A LINE FEED
13770 044472 012603      MOV   (SP)+,R3      ;; RESTORE R3
13771 044474 011646      MOV   (SP),-(SP)   ;; ADJUST THE STACK AND PUT ADDRESS OF THE
13772 044476 016666 000004 000002      MOV   4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
13773 044504 012766 044516 000004      MOV   #$TTYIN,4(SP)
13774 044512 000002      RTI                    ;; RETURN
13775 044514      9$:      .BYTE  0            ;; STORAGE FOR ASCII CHAR. TO TYPE
13776 044516      .BYTE  0            ;; TERMINATOR
13777 044516 000011      .BLKB  9.          ;; RESERVE 9. BYTES FOR TTY INPUT
13778 044527 136 006503 000012 $CNTLC: .ASCIZ  /?C<15><12> ;; CONTROL "C"
13779 044534 052536 005015 000 $CNTLU: .ASCIZ  /?U<15><12> ;; CONTROL "U"
13780 044541 136 006507 000012 $CNTLG: .ASCIZ  /?G<15><12> ;; CONTROL "G"
13781 044546 005015 053523 020122 $MSWR:  .ASCIZ  <15><12> $SWR = /
13782 044554 020075 000
13783 044557 040 047040 053505 $MNEW:  .ASCIZ  NEW = /
13784 044564 036440 000040
13785
13786
13787
;FROM THE TTY.

```

```

13788
13789
13790
13791
13792
13793
13794
13795
13796
13797
13798
13799
13800
13801
13802
13803 044570 011646
13804 044572 016666 000004 000002
13805 044600 010046
13806 044602 010146
13807 044604 010246
13808 044606 104407
13809 044610 012600
13810 044612 010037 044716
13811 044616 005001
13812 044620 005002
13813 044622 112046
13814 044624 001420
13815 044626 122716 000060
13816 044632 003026
13817 044634 122716 000067
13818 044640 002423
13819 044642 006301
13820 044644 006102
13821 044646 006301
13822 044650 006102
13823 044652 006301
13824 044654 006102
13825 044656 042716 177770
13826 044662 062601
13827 044664 000756
13828 044666 005726
13829 044670 010166 000012
13830 044674 010237 044726
13831 044700 012602
13832 044702 012601
13833 044704 012600
13834 044706 000002
13835 044710 005726
13836 044712 105010
13837 044714 104401
13838 044716 000000
13839 044720 104401 001222
13840 044724 000730
13841 044726 000000

.SBTTL READ AN OCTAL NUMBER FROM THE TTY
:*****
:THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
:CHANGE IT TO BINARY.
:THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
:OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
:FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
:THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
:CALL:
:*      RDOCT          ;; READ AN OCTAL NUMBER
:*      RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK
:*                  ;; HIGH ORDER BITS ARE IN $HIOCT

$RDOCT: MOV      (SP), -(SP)      ;; PROVIDE SPACE FOR THE
MOV      4(SP), 2(SP)          ;; INPUT NUMBER
MOV      R0, -(SP)            ;; PUSH R0 ON STACK
MOV      R1, -(SP)            ;; PUSH R1 ON STACK
MOV      R2, -(SP)            ;; PUSH R2 ON STACK
1$:      RDLIN          ;; READ AN ASCII LINE
MOV      (SP)+, R0            ;; GET ADDRESS OF 1ST CHARACTER
MOV      R0, 5$              ;; AND SAVE IT
CLR      R1                  ;; CLEAR DATA WORD
CLR      R2

2$:      MOVB      (R0)+, -(SP)  ;; PICKUP THIS CHARACTER
BEQ      3$                  ;; IF ZERO GET OUT
CMPB    #'0, (SP)           ;; MAKE SURE THIS CHARACTER
BGT      4$                  ;; IS AN OCTAL DIGIT
CMPB    #'7, (SP)

4$:      BLT      4$
ASL      R1                  ;; *2
ROL      R2
ASL      R1                  ;; *4
ROL      R2
ASL      R1                  ;; *8
ROL      R2

3$:      BIC      #'C7, (SP)    ;; STRIP THE ASCII JUNK
ADD      (SP)+, R1          ;; ADD IN THIS DIGIT
BR       2$                ;; LOOP
TST      (SP)+              ;; CLEAN TERMINATOR FROM STACK
MOV      R1, 12(SP)         ;; SAVE THE RESULT
MOV      R2, $HIOCT
MOV      (SP)+, R2          ;; POP STACK INTO R2
MOV      (SP)+, R1          ;; POP STACK INTO R1
MOV      (SP)+, R0          ;; POP STACK INTO R0
RTI

4$:      TST      (SP)+
CLRB    (R0)                ;; CLEAN PARTIAL FROM STACK
TYPE    0                    ;; SET A TERMINATOR
5$:      .WORD    0            ;; TYPE UP THRU THE BAD CHAR.
TYPE    $QUES                ;; "?" "CR" & "LF"
BR       1$                  ;; TRY AGAIN
$HIOCT: .WORD    C            ;; HIGH ORDER BITS GO HERE

```


ERROR HANDLER ROUTINE

.SBTTL ERROR HANDLER ROUTINE

13842
13843
13844
13845
13846
13847
13848
13849
13850
13851
13852
13853
13854
13855
13856 044730
13857 044730 105237 001103
13858 044734 001775
13859 044736 013777 001102 134176
13860 044744 032777 002000 134166
13861 044752 001402
13862 044754 104401 001216
13863 044760 005237 001112
13864 044764 011637 001116
13865 044770 162737 000002 001116
13866 044776 117737 134114 001114
13867 045004 032777 020000 134126
13868 045012 001004
13869 045014 004737 045066
13870 045020 104401 001223
13871 045024
13872 045024 777 134110
13873 045030 000001
13874 045032 000000
13875 045034 032777 001000 134076
13876 045042 001402
13877 045044 013716 001110
13878 045050 005737 001214
13879 045054 001402
13880 045056 013716 001214
13881 045062
13882 045062 000002
13883
13884
13885
13886
13887
13888
13889
13890
13891
13892
13893
13894
13895 045064 000000
13896 045066 013746 001140
13897 045072 042716 177277

```
*****  
*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  
*SW09=1 LOOP ON ERROR  
*CALL  
* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER  
  
$ERROR:  
7$: INCB $ERFLG ;; SET THE ERROR FLAG  
BEQ 7$ ;; DON'T LET THE FLAG GO TO ZERO  
MOV $TSTNM, @DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG  
BIT #BIT10, @SWR ;; BELL ON ERROR?  
BEQ 1$ ;; NO - SKIP  
TYPE $BELL ;; RING BELL  
1$: INC $ERTTL ;; COUNT THE NUMBER OF ERRORS  
MOV (SP), $ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION  
SUB #2, $ERRPC  
MOVB @ERRPC, $ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE  
BIT #BIT13, @SWR ;; SKIP TYPEOUT IF SET  
BNE 20$ ;; SKIP TYPEOUTS  
JSR PC, $ERRTYP ;; GO TO USER ERROR ROUTINE  
TYPE , $CRLF  
  
20$:  
2$: TST @SWR ;; HALT ON ERROR  
BPL 3$ ;; SKIP IF CONTINUE  
HALT ;; HALT ON ERROR!  
3$: BIT #BIT09, @SWR ;; LOOP ON ERROR SWITCH SET?  
BEQ 4$ ;; BR IF NO  
MOV $LPERR, (SP) ;; FUDGE RETURN FOR LOOPING  
4$: TST $ESCAPE ;; CHECK FOR AN ESCAPE ADDRESS  
BEQ 5$ ;; BR IF NONE  
MOV $ESCAPE, (SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE  
  
5$: RTI ;; RETURN  
*****  
.SBTTL ERROR MESSAGE TYPEOUT ROUTINE  
*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH  
*ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" ($ERRTB),  
*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.  
*IT IS A COPY OF THE $ERRTYP SUBROUTINE FROM SYSMAC.  
*WITH ONLY MINOR CHANGES  
*FIRST IF SWITCH 6 IS SET AND SWITCH 8 RESET THEN  
*ALL REGISTER CONTENTS WILL BE TYPED BEFORE REPORTING THE ERROR  
*SECOND IF THE CURRENT ERROR HAS THE SAME ITEM NUMBER  
*AS THE PREVIOUS ERROR THEN ONLY THE DATA WILL BE TYPED  
*AND NOT THE ERROR MESSAGE AND HEADER.  
PRITEM: 0 ;; PREVIOUS ITEM NO. LOCATION  
$ERRTYP: MOV @SWR, -(SP) ;; GET SWITCH SETTING  
BIC #10500, (SP) ;; KEEP ONLY SWITCH 8 AND 6
```

F05

CERHADO MACY11 301046, 21-DEC-77 13:06 PAGE 265
 CERHAD.F11 21-DEC-77 12:48 ERROR MESSAGE TYPEOUT ROUTINE

FE1 0264

13898	045076	022726	000100	CMP	#SW06,(SP)+	; IS 6 SET AND 8 RESET
13899	045102	001001		BNE	1\$; IF NOT BRANCH
13900	045104	000402		BR	2\$; BRANCH IF SW 6 IS SET AND 8 RESET
13901	045106	000137	046006	1\$: JMP	3\$TYPERR	; JUMP IF SW 8 IS SET
13902						; OR IF SW 8 IS RESET AND SW 6 IS RESET
13903	045112	104401	000200	2\$: TYPE	,CRLF	
13904						
13905	045116	104401	045124	TYPE	65\$:: TYPE ASCIZ STRING
13906	045122	000405		BR	64\$:: GET OVER THE ASCIZ
13907				::65\$: .ASCIZ	/ RHWC= /	
13908	045136			64\$: MOV	3RHWC,-(SP)	; GET READY TO TYPE RHWC CONTENTS
13909	045136	017746	136720	TYPOC		
13910	045142	104402				
13911						
13912						
13913	045144	104401	045152	TYPE	67\$:: TYPE ASCIZ STRING
13914	045150	000405		BR	66\$:: GET OVER THE ASCIZ
13915				::67\$: .ASCIZ	/ RHBA= /	
13916	045164			66\$: MOV	3RHBA,-(SP)	; GET READY TO TYPE RHBA CONTENTS
13917	045164	017746	136674	TYPOC		
13918	045170	104402				
13919						
13920						
13921	045172	104401	045200	TYPE	69\$:: TYPE ASCIZ STRING
13922	045176	000406		BR	68\$:: GET OVER THE ASCIZ
13923				::69\$: .ASCIZ	/ RHCS2= /	
13924	045214			68\$: MOV	3RHCS2,-(SP)	; GET READY TO TYPE RHCS2 CONTENTS
13925	045214	017746	136650	TYPOC		
13926	045220	104402				
13927						
13928						
13929	045222	104401	045230	TYPE	71\$:: TYPE ASCIZ STRING
13930	045226	000406		BR	70\$:: GET OVER THE ASCIZ
13931				::71\$: .ASCIZ	/ RHCS1= /	
13932	045244			70\$: MOV	3RHCS1,-(SP)	; GET READY TO TYPE RHCS1 CONTENTS
13933	045244	017746	136610	TYPOC		
13934	045250	104402				
13935						
13936						
13937	045252	104401	045260	TYPE	73\$:: TYPE ASCIZ STRING
13938	045256	000406		BR	72\$:: GET OVER THE ASCIZ
13939				::73\$: .ASCIZ	/ RHDS1= /	
13940	045274			72\$: MOV	3RHDS1,-(SP)	; GET READY TO TYPE RHDS1 CONTENTS
13941	045274	017746	136572	TYPOC		
13942	045300	104402				
13943						
13944						
13945	045302	104401	045310	TYPE	75\$:: TYPE ASCIZ STRING
13946	045306	000406		BR	74\$:: GET OVER THE ASCIZ
13947				::75\$: .ASCIZ	/ RHER1= /	
13948	045324			74\$: MOV	3RHER1,-(SP)	; GET READY TO TYPE RHER1 CONTENTS
13949	045324	017746	136544	TYPOC		
13950	045330	104402				
13951						
13952						
13953	045332	104401	045340	TYPE	.77\$:: TYPE ASCIZ STRING

G05

CERHADO MACY11 301046, 21-DEC-77 13:06 PAGE 266
 CERHAD.P11 21-DEC-77 12:48

ERROR MESSAGE TYPEOUT ROUTINE

SEQ 0265

13954	045336	000406			BR 76\$::GET OVER THE ASCIZ
13955					77\$: .ASCIZ / RHER2= /	
13956	045354			76\$:		
13957	045354	017746	136540		MOV 78\$:GET READY TO TYPE RHER2 CONTENTS
13958	045360	104402			TYPOC	
13959						
13960						
13961	045362	104401	045370		TYPE 79\$::TYPE ASCIZ STRING
13962	045366	000406			BR 78\$::GET OVER THE ASCIZ
13963				79\$:	.ASCIZ / RHER3= /	
13964	045404			78\$:		
13965	045404	017746	136512		MOV 80\$:GET READY TO TYPE RHER3 CONTENTS
13966	045410	104402			TYPOC	
13967						
13968						
13969	045412	104401	045420		TYPE 81\$::TYPE ASCIZ STRING
13970	045416	000406			BR 80\$::GET OVER THE ASCIZ
13971				81\$:	.ASCIZ / RHDST= /	
13972	045434			80\$:		
13973	045434	017746	136426		MOV 82\$:GET READY TO TYPE RHDST CONTENTS
13974	045440	104402			TYPOC	
13975						
13976						
13977	045442	104401	045450		TYPE 83\$::TYPE ASCIZ STRING
13978	045446	000405			BR 82\$::GET OVER THE ASCIZ
13979				83\$:	.ASCIZ / RHCA= /	
13980	045462			82\$:		
13981	045462	017746	136426		MOV 84\$:GET READY TO TYPE RHCA CONTENTS
13982	045466	104402			TYPOC	
13983						
13984						
13985	045470	104401	045476		TYPE 85\$::TYPE ASCIZ STRING
13986	045474	000405			BR 84\$::GET OVER THE ASCIZ
13987				85\$:	.ASCIZ / RHAS= /	
13988	045510			84\$:		
13989	045510	017746	136362		MOV 86\$:GET READY TO TYPE RHAS CONTENTS
13990	045514	104402			TYPOC	
13991						
13992						
13993	045516	104401	045524		TYPE 87\$::TYPE ASCIZ STRING
13994	045522	000405			BR 86\$::GET OVER THE ASCIZ
13995				87\$:	.ASCIZ / RHOF= /	
13996	045536			86\$:		
13997	045536	017746	136350		MOV 88\$:GET READY TO TYPE RHOF CONTENTS
13998	045542	104402			TYPOC	
13999						
14000						
14001	045544	104401	045552		TYPE 89\$::TYPE ASCIZ STRING
14002	045550	000405			BR 88\$::GET OVER THE ASCIZ
14003				89\$:	.ASCIZ / RHMR= /	
14004	045564			88\$:		
14005	045564	017746	136314		MOV 90\$:GET READY TO TYPE RHMR CONTENTS
14006	045570	104402			TYPOC	
14007						
14008						
14009	045572	104401	045600		TYPE 91\$::TYPE ASCIZ STRING

```

14010 045576 000405          BR      90$          ;;GET OVER THE ASCIZ
14011          ;;91$: .ASCIZ / RHLA= /
14012 045612          90$:          ;
14013 045612 017746 136262  MOV     2RHLA,-(SP)  ;GET READY TO TYPE RHLA CONTENTS
14014 045616 104402          TYPOC
14015
14016
14017 045620 104401 045626          TYPE     93$          ;;TYPE ASCIZ STRING
14018 045624 000405          BR      92$          ;;GET OVER THE ASCIZ
14019          ;;93$: .ASCIZ / RHCC= /
14020 045640          92$:          ;
14021 045640 017746 136234  MOV     2RHCC,-(SP) ;GET READY TO TYPE RHCC CONTENTS
14022 045644 104402          TYPOC
14023
14024
14025 045646 104401 045654          TYPE     95$          ;;TYPE ASCIZ STRING
14026 045652 000406          BR      94$          ;;GET OVER THE ASCIZ
14027          ;;95$: .ASCIZ / RHEC1= /
14028 045670          94$:          ;
14029 045670 017746 136230  MOV     2RHEC1,-(SP);GET READY TO TYPE RHEC1 CONTENTS
14030 045674 104402          TYPOC
14031
14032
14033 045676 104401 045704          TYPE     97$          ;;TYPE ASCIZ STRING
14034 045702 000406          BR      96$          ;;GET OVER THE ASCIZ
14035          ;;97$: .ASCIZ / RHEC2= /
14036 045720          96$:          ;
14037 045720 017746 136202  MOV     2RHEC2,-(SP);GET READY TO TYPE RHEC2 CONTENTS
14038 045724 104402          TYPOC
14039
14040
14041 045726 104401 045734          TYPE     99$          ;;TYPE ASCIZ STRING
14042 045732 000405          BR      98$          ;;GET OVER THE ASCIZ
14043          ;;99$: .ASCIZ / RHDT= /
14044 045746          98$:          ;
14045 045746 017746 136134  MOV     2RHDT,-(SP) ;GET READY TO TYPE RHDT CONTENTS
14046 045752 104402          TYPOC
14047
14048
14049 045754 104401 045762          TYPE     101$         ;;TYPE ASCIZ STRING
14050 045760 000405          BR      100$         ;;GET OVER THE ASCIZ
14051          ;;101$: .ASCIZ / RHSN= /
14052 045774          100$:          ;
14053 045774 017746 136110  MOV     2RHSN,-(SP) ;GET READY TO TYPE RHSN CONTENTS
14054 046000 104402          TYPOC
14055
14056 046002 005037 045064          CLR     2#PRITEM    ;CLEAR PREVIOUS ERROR ITEM
14057 046006          TYPERR:
14058 046006 104401 001223          TYPE     $CRLF        ;"CARRIAGE RETURN" & "LINE FEED"
14059 046012 010046          MOV     RO,-(SP)     ;SAVE RO
14060 046014 005000          CLR     RO           ;PICKUP THE ITEM INDEX
14061 046016 153700 001114          BISB   2#SITEMB,RO
14062 046022 001004          BNE    1$           ;IF ITEM NUMBER IS ZERO, JUST
14063
14064 046024 013746 001116          MOV     $ERRPC,-(SP);TYPE THE PC OF THE ERROR
14065
14066
    
```

```

14066 046030 104402          TYPOC          ;GO TYPE--OCTAL ASCII ALL DIGITS.
14067 046032 000454          BR          10$          ;GET OUT
14068 046034 005300          1$: DEC      RO          ;ADJUST THE INDEX SO THAT IT WILL
14069 046036 006300          ASL      RO          ;      WORK FOR THE ERROR TABLE
14070 046040 006300          ASL      RO
14071 046042 006300          ASL      RO
14072 046044 062700 001226    ADD      #SEHRTB,RO      ;FORM TABLE POINTER
14073 046050 020037 045064    CMP      RO,#PRITEM     ;WAS PREVIOUS ERROR SAME
14074 046054 001002          BNE      13$           ;BRANCH IF NOT
14075 046056 022020          CMP      (RO)+,(RO)+    ;POP RO OVER EM AND DH
14076 046060 000420          BR          5$
14077 046062 010037 045064    13$: MOV     RO,#PRITEM    ;SAVE NEW ERROR ITEM
14078 046066 012037 046076    MOV     (RO)+,2$       ;PICKUP "ERROR MESSAGE" POINTER
14079 046072 001404          BEQ     3$            ;SKIP TYPEOUT IF NO POINTER
14080 046074 104401          TYPE    ;TYPE THE "ERROR MESSAGE"
14081 046076 000000          2$: .WORD 0           ;"ERROR MESSAGE" POINTER GOES HERE
14082 046100 104401 001223    TYPE    ,SCLF         ;"CARRIAGE RETURN" & "LINE FEED"
14083 046104 012037 046114    3$: MOV     (RO)+,4$     ;PICKUP "DATA HEADER" POINTER
14084 046110 001404          BEQ     5$            ;SKIP TYPEOUT IF 0
14085 046112 104401          TYPE    ;TYPE THE "DATA HEADER"
14086 046114 000000          4$: .WORD 0           ;"DATA HEADER" POINTER GOES HERE
14087 046116 104401 001223    TYPE    ,SCLF         ;"CARRIAGE RETURN" & "LINE FEED"
14088 046122 010146          5$: MOV     R1,-(SP)     ;SAVE R1
14089 046124 012001          MOV     (RO)+,R1      ;PICKUP "DATA TABLE" POINTER
14090 046126 001415          BEQ     9$            ;BR IF NO DATA TO BE TYPED
14091 046130 012000          MOV     (RO)+,RO      ;PICKUP "DATA FORMAT" POINTER
14092 046132 105720          6$: TSTB   (RO)+       ;"OCTAL" OR "DECIMAL"
14093 046134 001003          BNE     7$            ;BR IF DECIMAL
14094 046136 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
14095 046140 104402          TYPOC          ;GO TYPE--OCTAL ASCII(ALL DIGITS)
14096 046142 000402          BR          8$
14097 046144          7$:
14098 046144 013146          MOV     @ (R1)+,-(SP) ;SAVE @ (R1)+ FOR TYPEOUT
14099 046146 104405          TYPDS   ;GO TYPE--DECIMAL ASCII WITH SIGN
14100 046150 005711          8$: TST    (R1)        ;IS THERE ANOTHER NUMBER?
14101 046152 001403          BEQ     9$            ;BR IF NO
14102 046154 104401 046202    TYPE    ,11$         ;TYPE TWO(2) SPACES
14103 046160 000764          BR          6$        ;LOOP
14104
14105 046162 012601          9$: MOV     (SP)+,R1    ;RESTORE R1
14106 046164 012600          10$: MOV    (SP)+,RO    ;"CARRIAGE RETURN" & "LINE FEED"
14107 046166 023737 000042 000046  CMP     @#42,@#46     ;ARE WE IN QV OR AUTO ACCEPT MODE?
14108 046174 001001          BNE     .+4          ;BRANCH OVER HALT IF NOT
14109 046176 000000          HALT   ;HALT AFTER ERROR MESSAGE
14110 046200 000207          RTS    PC           ;RETURN
14111 046202 020040 000          11$: .ASCIZ  / /      ;TWO(2) SPACES
14112          .EVEN
14113          ;*****
14114          ;*****
14115          ;SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
14116
14117          ;*****
14118          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
14119          ;*OCTAL (ASCII) NUMBER AND TYPE IT.
14120          ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
14121          ;*CALL:

```

J05

CERHADO MACY11 30(1046) 21-DEC-77 13:06 PAGE 269
 CERHAD.P11 21-DEC-77 12:48 BINARY TO OCTAL (ASCII) AND TYPE

SEG 0268

```

14122      *      MOV      NUM, -(SP)      ;; NUMBER TO BE TYPED
14123      *      TYPOS      ;; CALL FOR TYPEOUT
14124      *      .BYTE      N      ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
14125      *      .BYTE      M      ;; M=1 OR 0
14126      *      ;; 1=TYPE LEADING ZEROS
14127      *      ;; 0=SUPPRESS LEADING ZEROS
14128
14129      *      *STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
14130      *      *STYPOS OR STYPOC
14131      *      *CALL:
14132      *      MOV      NUM, -(SP)      ;; NUMBER TO BE TYPED
14133      *      TYPON      ;; CALL FOR TYPEOUT
14134      *
14135      *      *STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
14136      *      *CALL:
14137      *      MOV      NUM, -(SP)      ;; NUMBER TO BE TYPED
14138      *      TYPOC      ;; CALL FOR TYPEOUT
14139
14140      046206 017646 000000      STYPOS: MOV      2(SP), -(SP)      ;; PICKUP THE MODE
14141      046212 116637 000001      046431  MOVB     1(SP), $OFILL      ;; LOAD ZERO FILL SWITCH
14142      046220 112637 046433      MOVB     (SP)+, $OMODE+1      ;; NUMBER OF DIGITS TO TYPE
14143      046224 062716 000002      ADD      #2, (SP)      ;; ADJUST RETURN ADDRESS
14144      046230 000406      BR      STYPON
14145      046232 112737 000001      046431  STYPOC: MOVB     #1, $OFILL      ;; SET THE ZERO FILL SWITCH
14146      046240 112737 000006      046433  MOVB     #6, $OMODE+1      ;; SET FOR SIX(6) DIGITS
14147      046246 112737 000005      046430  STYPON: MOVB     #5, $OCNT      ;; SET THE ITERATION COUNT
14148      046254 010346      MOV      R3, -(SP)      ;; SAVE R3
14149      046256 010446      MOV      R4, -(SP)      ;; SAVE R4
14150      046260 010546      MOV      R5, -(SP)      ;; SAVE R5
14151      046262 113704 046433      MOVB     $OMODE+1, R4      ;; GET THE NUMBER OF DIGITS TO TYPE
14152      046266 005404      NEG      R4
14153      046270 062704 000006      ADD      #6, R4      ;; SUBTRACT IT FOR MAX. ALLOWED
14154      046274 110437 046432      MOVB     R4, $OMODE      ;; SAVE IT FOR USE
14155      046300 113704 046431      MOVB     $OFILL, R4      ;; GET THE ZERO FILL SWITCH
14156      046304 016605 000012      MOV      12(SP), R5      ;; PICKUP THE INPUT NUMBER
14157      046310 005003      CLR      R3      ;; CLEAR THE OUTPUT WORD
14158      046312 006105      1$: ROL      R5      ;; ROTATE MSB INTO "C"
14159      046314 000404      BR      3$      ;; GO DO MSB
14160      046316 006105      2$: ROL      R5      ;; FORM THIS DIGIT
14161      046320 006105      ROL      R5
14162      046322 006105      ROL      R5
14163      046324 010503      MOV      R5, R3
14164      046326 006103      3$: ROL      R3      ;; GET LSB OF THIS DIGIT
14165      046330 105337 046432      DECB     $OMODE      ;; TYPE THIS DIGIT
14166      046334 100016      BPL      7$      ;; BR IF NO
14167      046336 042703 177770      BIC      #177770, R3      ;; GET RID OF JUNK
14168      046342 001002      BNE      4$      ;; TEST FOR 0
14169      046344 005704      TST      R4      ;; SUPPRESS THIS 0?
14170      046346 001403      BEQ      5$      ;; BR IF YES
14171      046350 005204      4$: INC      R4      ;; DON'T SUPPRESS ANYMORE 0'S
14172      046352 052703 000060      BIS      #'0, R3      ;; MAKE THIS DIGIT ASCII
14173      046356 052703 000040      5$: BIS      #' , R3      ;; MAKE ASCII IF NOT ALREADY
14174      046362 110337 046426      MOVB     R3, 8$      ;; SAVE FOR TYPING
14175      046366 104401 046426      TYPE     8$      ;; GO TYPE THIS DIGIT
14176      046372 105337 046430      7$: DECB     $OCNT      ;; COUNT BY 1
14177      046376 003347      BGT      2$      ;; BR IF MORE TO DO
  
```

K05

CERHADD MACY11 3001046 21-DEC-77 13:06 PAGE 270
CERHAD.P11 21-DEC-77 12:48 BINARY TO OCTAL (ASCII) AND TYPE

SEQ 0263

14178	046400	002402		BLT	6\$:: BR IF DONE
14179	046402	005204		INC	R4	:: INSURE LAST DIGIT ISN'T A BLANK
14180	046404	000744		BR	2\$:: GO DO THE LAST DIGIT
14181	046406	012605		MOV	(SP)+,R5	:: RESTORE R5
14182	046410	012604		MOV	(SP)+,R4	:: RESTORE R4
14183	046412	012603		MOV	(SP)+,R3	:: RESTORE R3
14184	046414	016666	000002 000004	MOV	2(SP),4(SP)	:: SET THE STACK FOR RETURNING
14185	046422	012616		MOV	(SP)+,(SP)	
14186	046424	000002		RTI		:: RETURN
14187	046426	000		.BYTE	0	:: STORAGE FOR ASCII DIGIT
14188	046427	000		.BYTE	0	:: TERMINATOR FOR TYPE ROUTINE
14189	046430	000		.BYTE	0	:: OCTAL DIGIT COUNTER
14190	046431	000		.BYTE	0	:: ZERO FILL SWITCH
14191	046432	000000		.WORD	0	:: NUMBER OF DIGITS TO TYPE

```

14192 .SBTTL TRAP DECODER
14193
14194 ::*****
14195 ::*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
14196 ::*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
14197 ::*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
14198 ::*GO TO THAT ROUTINE.
14199
14200 046434 010046 $TRAP: MOV R0, -(SP) ;; SAVE R0
14201 046436 016600 000002 MOV 2(SP), R0 ;; GET TRAP ADDRESS
14202 046442 005740 TST -(R0) ;; BACKUP BY 2
14203 046444 111000 MOV B (R0), R0 ;; GET RIGHT BYTE OF TRAP
14204 046446 006300 ASL R0 ;; POSITION FOR INDEXING
14205 046450 016000 046470 MOV $TRPAD(R0), R0 ;; INDEX TO TABLE
14206 046454 000200 RTS R0 ;; GO TO ROUTINE
14207
14208
14209 ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
14210
14211 046456 011646 $TRAP2: MOV (SP), -(SP) ;; MOVE THE PC DOWN
14212 046460 016666 000004 000002 MOV 4(SP), 2(SP) ;; MOVE THE PSW DOWN
14213 046466 000002 RTI ;; RESTORE THE PSW
14214
14215 .SBTTL TRAP TABLE
14216
14217 ::*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
14218 ::*BY THE "TRAP" INSTRUCTION.
14219
14220 : ROUTINE
14221 : -----
14222 046470 046456 $TRPAD: .WORD $TRAP2
14223 046472 043610 $TYPE ;; CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
14224 046474 046232 $TYPOC ;; CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
14225 046476 046206 $TYPOS ;; CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
14226 046500 046246 $TYPON ;; CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
14227 046502 043364 $TYPDS ;; CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
14228
14229
14230 046504 044320 $RDCHR ;; CALL=RDCHR TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
14231 046506 044410 $RDLIN ;; CALL=RDLIN TRAP+7(104407) TTY TYPEIN STRING ROUTINE
14232 046510 044570 $RDOCT ;; CALL=RDOCT TRAP+10(104410) READ AN OCTAL NUMBER FROM TT
14233 046512 040444 WAIT.T ;; CALL=WAT TRAP+11(104411) DONT ADD ABOVE THIS TRAP
14234
14235
14236

```


MOS

.SBTTL POWER DOWN AND UP ROUTINES

```

14237
14238
14239
14240
14241 046514 012737 046660 000024
14242 046522 012737 000340 000026
14243 046530 010046
14244 046532 010146
14245 046534 010246
14246 046536 010346
14247 046540 010446
14248 046542 010546
14249 046544 017746 132370
14250 046550 010637 046664
14251 046554 012737 046566 000024
14252 046562 000000
14253 046564 000776
14254
14255
14256
14257 046566 012737 046660 000024
14258 046574 013706 046664
14259 046600 005037 046664
14260 046604 005237 046664
14261 046610 001375
14262 046612 012677 132322
14263 046616 012605
14264 046620 012604
14265 046622 012603
14266 046624 012602
14267 046626 012601
14268 046630 012600
14269 046632 012737 046514 000024
14270 046640 012737 000340 000026
14271 046646 104401
14272 046650 046666
14273 046652 012716
14274 046654 005522
14275 046656 000002
14276 046660 000000
14277 046662 000776
14278 046664 000000
14279 046666 005015 047520 042527
14280 046674 000122
14281
    
```

;*****
 :POWER DOWN ROUTINE
 \$PWRDN: MOV \$SILLUP,@PWRVEC ;:SET FOR FAST UP
 MOV #340,@PWRVEC+2 ;:PRIO:7
 MOV RO,-(SP) ;:PUSH RO ON STACK
 MOV R1,-(SP) ;:PUSH R1 ON STACK
 MOV R2,-(SP) ;:PUSH R2 ON STACK
 MOV R3,-(SP) ;:PUSH R3 ON STACK
 MOV R4,-(SP) ;:PUSH R4 ON STACK
 MOV R5,-(SP) ;:PUSH R5 ON STACK
 MOV @JSWR,-(SP) ;:PUSH @JSWR ON STACK
 MOV SP,\$SAVR6 ;:SAVE SP
 MOV \$PWRUP,@PWRVEC ;:SET UP VECTOR
 HALT
 BR .-2 ;:HANG UP

;*****
 :POWER UP ROUTINE
 \$PWRUP: MOV \$SILLUP,@PWRVEC ;:SET FOR FAST DOWN
 MOV \$SAVR6,SP ;:GET SP
 CLR \$SAVR6 ;:WAIT LOOP FOR THE TTY
 1\$: INC \$SAVR6 ;:WAIT FOR THE INC
 BNE 1\$;:OF WORD
 MOV (SP)+,@JSWR ;:POP STACK INTO @JSWR
 MOV (SP)+,R5 ;:POP STACK INTO R5
 MOV (SP)+,R4 ;:POP STACK INTO R4
 MOV (SP)+,R3 ;:POP STACK INTO R3
 MOV (SP)+,R2 ;:POP STACK INTO R2
 MOV (SP)+,R1 ;:POP STACK INTO R1
 MOV (SP)+,R0 ;:POP STACK INTO R0
 MOV \$PWRDN,@PWRVEC ;:SET UP THE POWER DOWN VECTOR
 MOV #340,@PWRVEC+2 ;:PRIO:7
 TYPE ;:REPORT THE POWER FAILURE
 \$PWRMG: .WORD \$POWER ;:POWER FAIL MESSAGE POINTER
 MOV (PC)+,(SP) ;:RESTART AT BEGIN
 \$PWRAD: .WORD BEGIN ;:RESTART ADDRESS
 RTI
 \$SILLUP: HALT ;:THE POWER UP SEQUENCE WAS STARTED
 BR .-2 ;:BEFORE THE POWER DOWN WAS COMPLETE
 \$SAVR6: 0 ;:PUT THE SP HERE
 \$POWER: .ASCIZ <15><12>"POWER"
 .EVEN

```

14282
14283
14284
14285
14286
14287
14288 046676 044124 020105 044122
14289 046704 041040 051501 020105
14290 046712 042101 051104 051505
14291 046720 020123 040527 020123
14292 046726 033461 030062 030064
14293 046734 044440 042116 041511
14294 046742 052101 047111 020107
14295 046750 047101 051040 020123
14296 046756 052502 020124 044124
14297 046764 020105 051104 053111
14298 046772 020105 054524 042520
14299 047000 005015
14300 047002 044504 020104 047516
14301 047010 020124 047503 052116
14302 047016 044501 020116 020060
14303 047024 051117 030440 047440
14304 047032 020122 020062 051117
14305 047040 031440 047440 020122
14306 047046 000064
14307
14308 047050 044124 020105 044122
14309 047056 041040 051501 020105
14310 047064 042101 051104 051505
14311 047072 020123 040527 020123
14312 047100 033461 033466 030060
14313 047106 044440 042116 041511
14314 047114 052101 047111 020107
14315 047122 047101 051040 030120
14316 047130 026064 026065 020066
14317 047136 052502 020124 044124
14318 047144 020105 051104 053111
14319 047152 020105 054524 042520
14320 047160 005015
14321 047162 044504 020104 047516
14322 047170 020124 047503 052116
14323 047176 044501 020116 030062
14324 047204 031060 026060 032062
14325 047212 031060 026060 030062
14326 047220 031060 026061 032062
14327 047226 031060 026061 030062
14328 047234 031060 026062 047440
14329 047242 020122 032062 031060
14330 047250 000062
14331
14332 047252 044124 020105 044122
14333 047260 041040 051501 020105
14334 047266 042101 051104 051505
14335 047274 020123 040527 020123
14336 047302 033461 032062 030064
14337 047310 044440 042116 041511

```

```

:*****
:ERROR AND MESSAGE TABLE CONDIMENTS
:*****

```

EM1: .ASCII /THE RH BASE ADDRESS WAS 172040 INDICATING AN RS BUT THE DRIVE TYPE//15

.ASCIZ /DID NOT CONTAIN 0 OR 1 OR 2 OR 3 OR 4/

EM2: .ASCII /THE RH BASE ADDRESS WAS 176700 INDICATING AN RPO4.5.6 BUT THE DRIVE TYP

.ASCIZ /DID NOT CONTAIN 20020,24020,20021,24021,20022, OR 24022.

EM3: .ASCII /THE RH BASE ADDRESS WAS 172440 INDICATING A TMO2 BUT THE DRIVE TYPE//15

14338	047316	052101	047111	020107	
14339	047324	020101	046524	031060	
14340	047332	041040	052125	052040	
14341	047340	042510	042040	044522	
14342	047346	042526	052040	050131	
14343	047354	006505	012		
14344	047357	104	042111	047040	.ASCIZ DID NOT CONTAIN 142010
14345	047364	052117	041440	047117	
14346	047372	040524	047111	030440	
14347	047400	031064	030460	000060	
14348					
14349	047406	044124	020105	044122	EM4: .ASCII THE RH BASE ADDRESS WAS 176300 INDICATING MORE THAN ONE RH DEVICE BUT
14350	047414	041040	051501	020105	
14351	047422	042101	051104	051505	
14352	047430	020123	040527	020123	
14353	047436	033461	031466	030060	
14354	047444	044440	042116	041511	
14355	047452	052101	047111	020107	
14356	047460	047515	042522	052040	
14357	047466	040510	020116	047117	
14358	047474	020105	044122	042040	
14359	047502	053105	041511	020105	
14360	047510	052502	006524	012	
14361	047515	124	042510	042040	.ASCII THE DRIVE TYPE DID NOT CONTAIN 0,1,2,3,4,20020,24020,15,12
14362	047522	044522	042526	052040	
14363	047530	050131	020105	044504	
14364	047536	020104	047516	020124	
14365	047544	047503	052116	044501	
14366	047552	020116	026060	026061	
14367	047560	026062	026063	026064	
14368	047566	030062	031060	026060	
14369	047574	032062	031060	026060	
14370	047602	005015			
14371	047604	030062	031060	026061	.ASCIZ 20021,24021,20022,24022,142010,15,12
14372	047612	032062	031060	026061	
14373	047620	030062	031060	026062	
14374	047626	032062	031060	026062	
14375	047634	032061	030062	030061	
14376	047642	005015	000		
14377	047645	111	042116	041511	.ASCIZ INDICATING NO RH DEVICE PRESENT
14378	047652	052101	047111	020107	
14379	047660	047516	051040	020110	
14380	047666	042504	044526	042503	
14381	047674	050040	042522	042523	
14382	047702	052116	000		
14383					
14384	047705	124	042510	052440	EM5: .ASCII THE UNIBUS TIMED OUT FOR EACH OF THE FOLLOWING ADDRESSES, INDICATING
14385	047712	044516	052502	020123	
14386	047720	044524	042515	020104	
14387	047726	052517	020124	047506	
14388	047734	020122	040505	044103	
14389	047742	047440	020106	044124	
14390	047750	020105	047506	046114	
14391	047756	053517	047111	020107	
14392	047764	042101	051104	051505	
14393	047772	042523	026123	044440	

14394	050000	042116	041511	052101	
14395	050006	047111	006507	012	
14396	050013	116	020117	044122	.ASCII /NO RM ON THE SYSTEM SO ABORT PROGRAM/
14397	050020	047440	020116	044124	
14398	050026	020105	054523	052123	
14399	050034	046505	051440	020117	
14400	050042	041101	051117	020124	
14401	050050	051120	043517	040522	
14402	050056	000115			
14403					
14404	050060	043101	042524	020122	EM6: .ASCII /AFTER AN RM CLEAR (RHCS2-BIT 05) RHCS2 DOES NOT HAVE ONLY IR<15> 12
14405	050066	047101	051040	020110	
14406	050074	046103	040505	020122	
14407	050102	051050	041510	031123	
14408	050110	041055	052111	021440	
14409	050116	024465	051040	041510	
14410	050124	031123	042040	042517	
14411	050132	020123	047516	020124	
14412	050140	040510	042526	047440	
14413	050146	046116	020131	051111	
14414	050154	005015			
14415	050156	047101	020104	047125	.ASCII /AND UNIT NUMBER/
14416	050164	052111	047040	046525	
14417	050172	042502	000122		
14418					
14419	050176	043101	042524	020122	EM7: .ASCII /AFTER CLEARING THE RM AND WRITING ONE WORD INTO RHDB AND<15><12>
14420	050204	046103	040505	044522	
14421	050212	043516	052040	042510	
14422	050220	051040	020110	047101	
14423	050226	020104	051127	052111	
14424	050234	047111	020107	047117	
14425	050242	020105	047527	042122	
14426	050250	044440	052116	020117	
14427	050256	044122	041104	040440	
14428	050264	042116	005015		
14429	050270	042522	042101	047111	.ASCII /READING IT BACK, CAUSED RHDB TO HAVE WRONG VALUE GIVEN IN "BAD"/
14430	050276	020107	052111	041040	
14431	050304	041501	026113	041440	
14432	050312	052501	042523	020104	
14433	050320	044122	041104	052040	
14434	050326	020117	040510	042526	
14435	050334	053440	047522	043516	
14436	050342	053040	046101	042525	
14437	050350	043440	053111	047105	
14438	050356	044440	020116	041042	
14439	050364	042101	000042		
14440					
14441	050370	043101	042524	020122	EM10: .ASCII /AFTER CLEARING THE RM AND WRITING ONE WORD INTO RHDB AND<15><12>
14442	050376	046103	040505	044522	
14443	050404	043516	052040	042510	
14444	050412	051040	020110	047101	
14445	050420	020104	051127	052111	
14446	050426	047111	020107	047117	
14447	050434	020105	047527	042122	
14448	050442	044440	052116	020117	
14449	050450	044122	041104	040440	

14450	050456	042116	005015	
14451	050462	042522	042101	047111
14452	050470	020107	052111	041040
14453	050476	041501	026113	041440
14454	050504	052501	042523	020104
14455	050512	044122	051503	020062
14456	050520	047524	044040	053101
14457	050526	020105	051127	047117
14458	050534	020107	040526	052514
14459	050542	020105	044507	042526
14460	050550	020116	047111	021040
14461	050556	040502	021104	000
14462				
14463	050563	101	052106	051105
14464	050570	041440	042514	051101
14465	050576	047111	020107	044124
14466	050604	020105	044122	040440
14467	050612	042116	053440	044522
14468	050620	044524	043516	047440
14469	050626	042516	053440	051117
14470	050634	020104	047111	047524
14471	050642	051040	042110	020102
14472	050650	047101	006504	012
14473	050655	122	040505	044504
14474	050662	043516	044440	020124
14475	050670	040502	045503	020054
14476	050676	040503	051525	042105
14477	050704	051040	041510	030523
14478	050712	052040	020117	040510
14479	050720	042526	053440	047522
14480	050726	043516	053040	046101
14481	050734	042525	043440	053111
14482	050742	047105	044440	020116
14483	050750	041042	042101	000042
14484				
14485	050756	043101	042524	020122
14486	050764	046103	040505	044522
14487	050772	043516	052040	042510
14488	051000	051040	020110	047101
14489	051006	020104	051127	052111
14490	051014	047111	020107	047117
14491	051022	020105	047527	042122
14492	051030	044440	052116	020117
14493	051036	044122	041104	040440
14494	051044	042116	005015	
14495	051050	042522	042101	047111
14496	051056	020107	052111	041040
14497	051064	041501	026113	041440
14498	051072	052501	042523	020104
14499	051100	044122	051503	020063
14500	051106	047524	044040	053101
14501	051114	020105	051127	047117
14502	051122	020107	040526	052514
14503	051130	020105	044507	042526
14504	051136	020116	047111	021040
14505	051144	040502	021104	000

.ASCIZ /READING IT BACK, CAUSED RHCS2 TO HAVE WRONG VALUE GIVEN IN "BAD"/

EM11: .ASCII /AFTER CLEARING THE RH AND WRITING ONE WORD INTO RHDB AND /15/12/

.ASCIZ /READING IT BACK, CAUSED RHCS1 TO HAVE WRONG VALUE GIVEN IN "BAD"/

EM12: .ASCII /AFTER CLEARING THE PH AND WRITING ONE WORD INTO PHDB AND /15/12/

.ASCIZ /READING IT BACK, CAUSED RHCS3 TO HAVE WRONG VALUE GIVEN IN "BAD"/

E06

14506					
14507	051151	101	052106	051105	EM13: .ASCII /AFTER CLEARING THE RH AND WRITING ONE WORD INTO RHDB AND <<15><12>
14508	051156	041440	042514	051101	
14509	051164	047111	020107	044124	
14510	051172	020105	044122	040440	
14511	051200	042116	053440	044522	
14512	051206	044524	043516	047440	
14513	051214	042516	053440	051117	
14514	051222	020104	047111	047524	
14515	051230	051040	042110	020102	
14516	051236	047101	006504	012	
14517	051243	122	040505	044504	.ASCIIZ READING IT BACK, CAUSED RHBA TO HAVE WRONG VALUE GIVEN IN "BAD"
14518	051250	043516	044440	020124	
14519	051256	040502	045503	020054	
14520	051264	040503	051525	042105	
14521	051272	051040	041110	020101	
14522	051300	047524	044040	053101	
14523	051306	020105	051127	047117	
14524	051314	020107	040526	052514	
14525	051322	020105	044507	042526	
14526	051330	020116	047111	021040	
14527	051336	040502	021104	000	
14528					
14529	051343	101	052106	051105	EM14: .ASCII /AFTER CLEARING THE RH AND WRITING ONE WORD INTO RHDB AND <<15><12>
14530	051350	041440	042514	051101	
14531	051356	047111	020107	044124	
14532	051364	020105	044122	040440	
14533	051372	042116	053440	044522	
14534	051400	044524	043516	047440	
14535	051406	042516	053440	051117	
14536	051414	020104	047111	047524	
14537	051422	051040	042110	020102	
14538	051430	047101	006504	012	
14539	051435	122	040505	044504	.ASCIIZ /READING IT BACK, CAUSED RHBAE TO HAVE WRONG VALUE GIVEN IN "BAD"
14540	051442	043516	044440	020124	
14541	051450	040502	045503	020054	
14542	051456	040503	051525	042105	
14543	051464	051040	041110	042501	
14544	051472	052040	020117	040510	
14545	051500	042526	053440	047522	
14546	051506	043516	053040	046101	
14547	051514	042525	043440	053111	
14548	051522	047105	044440	020116	
14549	051530	041042	042101	000042	
14550					
14551	051536	043101	042524	020122	EM15: .ASCII /AFTER CLEARING THE RH AND WRITING ONE WORD INTO RHDB AND <<15><12>
14552	051544	046103	040505	044522	
14553	051552	043516	052040	042510	
14554	051560	051040	020110	047101	
14555	051566	020104	051127	052111	
14556	051574	047111	020107	047117	
14557	051602	020105	047527	042122	
14558	051610	044440	052116	020117	
14559	051616	044122	041104	040440	
14560	051624	042116	005015		
14561	051630	042522	042101	047111	.ASCIIZ READING IT BACK, CAUSED RHWC TO HAVE WRONG VALUE GIVEN IN "BAD"

14562	051636	020107	052111	041040	
14563	051644	041501	026113	041440	
14564	051652	052501	042523	020104	
14565	051660	044122	041527	052040	
14566	051666	020117	040510	042526	
14567	051674	053440	047522	043516	
14568	051702	053040	046101	042525	
14569	051710	043440	053111	047105	
14570	051716	044440	020116	041042	
14571	051724	042101	000042		
14572					
14573	051730	043101	042524	020122	EM16: .ASCII AFTER CLEARING THE RH AND WRITING ONE WORD INTO RHDB AND <<15>>12
14574	051736	046103	040505	044522	
14575	051744	043516	052040	042510	
14576	051752	051040	020110	047101	
14577	051760	020104	051127	052111	
14578	051766	047111	020107	047117	
14579	051774	020105	047527	042122	
14580	052002	044440	052116	020117	
14581	052010	044122	041104	040440	
14582	052016	042116	005015		
14583	052022	042522	042101	047111	.ASCIZ /READING IT BACK, CAUSED RHCS2 TO HAVE WRONG VALUE GIVEN IN "BAD"
14584	052030	020107	052111	041040	
14585	052036	041501	026113	041440	
14586	052044	052501	042523	020104	
14587	052052	044122	051503	020062	
14588	052060	047524	044040	053101	
14589	052066	020105	051127	047117	
14590	052074	020107	040526	052514	
14591	052102	020105	044507	042526	
14592	052110	020116	047111	021040	
14593	052116	040502	021104	000	
14594					
14595	052123	101	052106	051105	EM17: .ASCII /AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND <<15>>12
14596	052130	041440	042514	051101	
14597	052136	047111	020107	044124	
14598	052144	020105	044122	040440	
14599	052152	042116	053440	044522	
14600	052160	044524	043516	052040	
14601	052166	047527	053440	051117	
14602	052174	020104	047111	047524	
14603	052202	051040	042110	020102	
14604	052210	047101	006504	012	
14605	052215	122	040505	044504	.ASCIZ /READING IT BACK, CAUSED RHDB TO HAVE WRONG VALUE GIVEN IN "BAD"
14606	052222	043516	044440	020124	
14607	052230	040502	045503	020054	
14608	052236	040503	051525	042105	
14609	052244	051040	042110	020102	
14610	052252	047524	044040	053101	
14611	052260	020105	051127	047117	
14612	052266	020107	040526	052514	
14613	052274	020105	044507	042526	
14614	052302	020116	047111	021040	
14615	052310	040502	021104	000	
14616					
14617	052315	101	052106	051105	EM20: .ASCII AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND <<15>>12

G06

14618	052322	041440	042514	051101
14619	052330	047111	020107	044124
14620	052336	020105	044122	040440
14621	052344	042116	053440	044522
14622	052352	044524	043516	052040
14623	052360	047527	053440	051117
14624	052366	020104	047111	047524
14625	052374	051040	042110	020102
14626	052402	047101	006504	012
14627	052407	122	040505	044504
14628	052414	043516	047440	042516
14629	052422	041040	041501	026113
14630	052430	041440	052501	042523
14631	052436	020104	044122	051503
14632	052444	020062	047524	044040
14633	052452	053101	020105	051127
14634	052460	047117	020107	040526
14635	052466	052514	020105	044507
14636	052474	042526	020116	047111
14637	052502	021040	040502	021104
14638	052510	000		
14639				
14640	052511	101	052106	051105
14641	052516	041440	042514	051101
14642	052524	047111	020107	044124
14643	052532	020105	044122	040440
14644	052540	042116	053440	044522
14645	052546	044524	043516	052040
14646	052554	047527	053440	051117
14647	052562	020104	047111	047524
14648	052570	051040	042110	020102
14649	052576	047101	006504	012
14650	052603	122	040505	044504
14651	052610	043516	044440	020124
14652	052616	040502	045503	052040
14653	052624	044527	042503	020054
14654	052632	040503	051525	042105
14655	052640	051040	042110	020102
14656	052646	047524	044040	053101
14657	052654	020105	051127	047117
14658	052662	020107	040526	052514
14659	052670	020105	044507	042526
14660	052676	020116	047111	021040
14661	052704	040502	021104	000
14662				
14663	052711	101	052106	051105
14664	052716	041440	042514	051101
14665	052724	047111	020107	044124
14666	052732	020105	044122	040440
14667	052740	042116	053440	044522
14668	052746	044524	043516	052040
14669	052754	047527	053440	051117
14670	052762	020104	047111	047524
14671	052770	051040	042110	020102
14672	052776	047101	006504	012
14673	053003	122	040505	044504

.ASCIZ READING ONE BACK, CAUSED RHCS2 TO HAVE WRONG VALUE GIVEN IN "BAD"/

EM21: .ASCII /AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND /15/12/

.ASCIZ /READING IT BACK TWICE, CAUSED RHDB TO HAVE WRONG VALUE GIVEN IN "BAD"

EM22: .ASCII AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND /15/12/

.ASCIZ READING IT BACK TWICE, CAUSED RHCS2 TO HAVE WRONG VALUE GIVEN IN "BAD"

14674	053010	043516	044440	020124
14675	053016	040502	045503	052040
14676	053024	044527	042503	020054
14677	053032	040503	051525	042105
14678	053040	051040	041510	031123
14679	053046	052040	020117	040510
14680	053054	042526	053440	047522
14681	053062	043516	053040	046101
14682	053070	042525	043440	053111
14683	053076	047105	044440	020116
14684	053104	041042	042101	000042
14685				
14686	053112	043101	042524	020122
14687	053120	046103	040505	044522
14688	053126	043516	052040	042510
14689	053134	051040	020110	047101
14690	053142	020104	051127	052111
14691	053150	047111	020107	044505
14692	053156	044107	020124	047527
14693	053164	042122	044440	052116
14694	053172	020117	044122	041104
14695	053200	040440	042116	005015
14696	053206	042522	042101	047111
14697	053214	020107	052111	041040
14698	053222	041501	026113	041440
14699	053230	052501	042523	020104
14700	053236	044122	051503	020062
14701	053244	047524	044040	053101
14702	053252	020105	051127	047117
14703	053260	020107	040526	052514
14704	053266	020105	044507	042526
14705	053274	020116	047111	021040
14706	053302	040502	021104	000
14707				
14708	053307	124	042510	051040
14709	053314	020110	040527	020123
14710	053322	046103	040505	042522
14711	053330	020104	047101	020104
14712	053336	020101	040520	052124
14713	053344	051105	020116	043117
14714	053352	034040	053440	051117
14715	053360	051504	053440	051105
14716	053366	020105	051127	052111
14717	053374	042524	020116	047111
14718	053402	047524	051040	042110
14719	053410	006502	012	
14720	053413	122	040505	044504
14721	053420	043516	051040	042110
14722	053426	020102	047506	020122
14723	053434	044124	020105	047042
14724	053442	020042	044124	020056
14725	053450	044524	042515	043440
14726	053456	053101	020105	051127
14727	053464	047117	020107	040526
14728	053472	052514	020105	047111
14729	053500	051040	042110	006502

EM23: .ASCII <AFTER CLEARING THE RH AND WRITING EIGHT WORD INTO RHDB AND<<15>>12>

.ASCIZ <READING IT BACK, CAUSED RHCS2 TO HAVE WRONG VALUE GIVEN IN "BAD">

EM24: .ASCII <THE RH WAS CLEARED AND A PATTERN OF 8 WORDS WERE WRITTEN INTO RHDB 15

.ASCII <READING RHDB FOR THE "N" TH. TIME GAVE WRONG VALUE IN RHDB<<15>>12>

14730	053506	012				
14731	053507	042	021116	044440	.ASCIZ	/"N" IS GIVEN IN WORD NO
14732	053514	020123	044507	042526		
14733	053522	020116	047111	053440		
14734	053530	051117	020104	047516		
14735	053536	000				
14736						
14737	053537	124	042510	051040	EM25:	.ASCII THE RH WAS CLEARED AND A PATTERN OF 8 WORDS WERE WRITTEN INTO RHDB//15
14738	053544	020110	040527	020123		
14739	053552	046103	040505	042522		
14740	053560	020104	047101	020104		
14741	053566	020101	040520	052124		
14742	053574	051105	020116	043117		
14743	053602	034040	053440	051117		
14744	053610	051504	053440	051105		
14745	053616	020105	051127	052111		
14746	053624	042524	020116	047111		
14747	053632	047524	051040	042110		
14748	053640	006502	012			
14749	053643	101	052106	051105	.ASCIZ	AFTER READING ALL 8 WORDS, FOLLOWING REGISTER CONTAINED WRONG VALUE 6IV
14750	053650	051040	040505	044504		
14751	053656	043516	040440	046114		
14752	053664	034040	053440	051117		
14753	053672	051504	020054	047506		
14754	053700	046114	053517	047111		
14755	053706	020107	042522	044507		
14756	053714	052123	051105	041440		
14757	053722	047117	040524	047111		
14758	053730	042105	053440	047522		
14759	053736	043516	053040	046101		
14760	053744	042525	043440	053111		
14761	053752	047105	044440	020116		
14762	053760	041042	042101	000042		
14763						
14764	053766	042523	052124	047111	EM33:	.ASCIZ /SETTING RH CLEAR (RHCS2-BIT #5) CAUSED ERROR REGISTER 1 TO HAVE WRONG V
14765	053774	020107	044122	041440		
14766	054002	042514	051101	024040		
14767	054010	044122	51503	026462		
14768	054016	044502	020124	032443		
14769	054024	020051	040503	051525		
14770	054032	042105	042440	051122		
14771	054040	051117	051040	043505		
14772	054046	051511	042524	020122		
14773	054054	020061	047524	044040		
14774	054062	053101	020105	051127		
14775	054070	047117	020107	040526		
14776	054076	052514	000105			
14777						
14778	054102	047101	051040	020110	EM34:	.ASCII /AN RH CLEAR WAS GIVEN RHER1 WAS CHECKED TO HAVE ZERO. PAT (RHCS2-BIT #4
14779	054110	046103	040505	020122		
14780	054116	040527	020123	044507		
14781	054124	042526	020116	044122		
14782	054132	051105	020061	040527		
14783	054140	020123	044103	041505		
14784	054146	042513	020104	047524		
14785	054154	044040	053101	020105		

14786	054162	042532	047522	020056
14787	054170	040520	020124	051050
14788	054176	041510	031123	041055
14789	054204	052111	021440	024464
14790	054212	053440	051501	051440
14791	054220	052105	005015	
14792	054224	047524	044440	053116
14793	054232	051105	020124	040520
14794	054240	044522	054524	041440
14795	054246	042510	045503	047111
14796	054254	027107	051040	042510
14797	054262	030522	053440	051501
14798	054270	051040	040505	020104
14799	054276	052502	020124	044504
14800	054304	020104	047516	020124
14801	054312	047503	052116	044501
14802	054320	020116	044127	052101
14803	054326	044440	020123	047111
14804	054334	021040	047507	042117
14805	054342	000042		
14806				
14807	054344	044122	041440	042514
14808	054352	051101	053440	051501
14809	054360	043440	053111	047105
14810	054366	020056	044122	051105
14811	054374	020061	040527	020123
14812	054402	044103	041505	042513
14813	054410	027104	050040	052101
14814	054416	024040	044122	051503
14815	054424	026462	044502	020124
14816	054432	032043	020051	040527
14817	054440	020123	042523	006524
14818	054446	012		
14819	054447	101	042116	051040
14820	054454	042510	030522	053440
14821	054462	051501	051040	040505
14822	054470	027104	051040	041510
14823	054476	030523	051440	047510
14824	054504	046125	020104	040510
14825	054512	042526	051040	054504
14826	054520	020054	041523	040440
14827	054526	042116	046440	050103
14828	054534	020105	042523	000124
14829				
14830	054542	044122	041440	042514
14831	054550	051101	053440	051501
14832	054556	043440	053111	047105
14833	054564	020056	044122	051105
14834	054572	020061	040527	020123
14835	054600	044103	041505	042513
14836	054606	027104	050040	052101
14837	054614	024040	044122	051503
14838	054622	026462	044502	020124
14839	054630	032043	020051	040527
14840	054636	020123	042523	006524
14841	054644	012		

.ASCIZ TO INVERT PARITY CHECKING. RHER1 WAS READ BUT DID NOT CONTAIN WHAT IS I

EM35: .ASCII /RH CLEAR WAS GIVEN. RHER1 WAS CHECKED. PAT (RHCS2-BIT #4) WAS SET <<15>>

.ASCIZ /AND RHER1 WAS READ. RHCS1 SHOULD HAVE RDY, SC AND MCPE SET

EM36: .ASCII /RH CLEAR WAS GIVEN. RHER1 WAS CHECKED. PAT (RHCS2-BIT #4) WAS SET <15

14842	054645	122	042510	030522	.ASCII	RHER1 WAS READ. "1" WAS WRITTEN INTO "TRE" IN RHCS1 (15) (12)
14843	054652	053440	051501	051040		
14844	054660	040505	027104	021040		
14845	054666	021061	053440	051501		
14846	054674	053440	044522	052124		
14847	054702	047105	044440	052116		
14848	054710	020117	052042	042522		
14849	054716	020042	047111	051040		
14850	054724	041510	030523	005015	.ASCII	ON CHECKING RHCS1 IT DID NOT CONTAIN WHAT IS IN GOOD
14851	054732	047117	041440	042510		
14852	054740	045503	047111	020107		
14853	054746	044122	051503	020061		
14854	054754	052111	042040	042111		
14855	054762	047040	052117	041440		
14856	054770	047117	040524	047111		
14857	054776	053440	040510	020124		
14858	055004	051511	044440	020116		
14859	055012	047507	042117	000		
14860						
14861	055017	122	020110	040527	EM37: .ASCII	AN RH WAS CLEARED. RHER1 WAS CHECKED. PAT (RHCS2-BIT #4) WAS SET (15) (12)
14862	055024	020123	046103	040505		
14863	055032	042522	027104	051040		
14864	055040	042510	030522	053440		
14865	055046	051501	041440	042510		
14866	055054	045503	042105	020056		
14867	055062	040520	020124	051050		
14868	055070	041510	031123	041055		
14869	055076	052111	021440	024464		
14870	055104	053440	051501	051440		
14871	055112	052105	005015			
14872	055116	044122	051105	020061	.ASCII	RHER1 WAS READ. "1" WAS WRITTEN IN "TRE" IN RHCS1 (15) (12)
14873	055124	040527	020123	042522		
14874	055132	042101	020056	030442		
14875	055140	020042	040527	020123		
14876	055146	051127	052111	042524		
14877	055154	020116	047111	021040		
14878	055162	051124	021105	044440		
14879	055170	020116	044122	051503		
14880	055176	006461	012			
14881	055201	101	020116	044122	.ASCII	AN RH CLEAR WAS GIVEN BUT FOLLOWING REGISTER DID NOT (15) (12)
14882	055206	041440	042514	051101		
14883	055214	053440	051501	043440		
14884	055222	053111	047105	041040		
14885	055230	052125	043040	046117		
14886	055236	047514	044527	043516		
14887	055244	051040	043505	051511		
14888	055252	042524	020122	044504		
14889	055260	020104	047516	006524		
14890	055266	012				
14891	055267	103	047117	040524	.ASCII	CONTAIN WHAT IS IN "GOOD"
14892	055274	047111	053440	040510		
14893	055302	020124	051511	044440		
14894	055310	020116	043442	047517		
14895	055316	021104	000			
14896						
14897	055321	101	052106	051105	EM45: .ASCII	AFTER AN RH CLEAR "1" WAS WRITTEN IN THE DISK (TAPE) ADDRESS (15) (12)

14898	055326	040440	020116	044122	
14899	055334	041440	042514	051101	
14900	055342	021040	021061	053440	
14901	055350	051501	053440	044522	
14902	055356	052124	047105	044440	
14903	055364	020116	044124	020105	
14904	055372	044504	045523	024040	
14905	055400	040524	042520	020051	
14906	055406	042101	051104	051505	
14907	055414	006523	012		
14908	055417	122	043505	051511	.ASCII REGISTER. PAT IN RHCS1 WAS SET. ON READING DISK (TAPE) ADDRESS 15 12
14909	055424	042524	027122	050040	
14910	055432	052101	044440	020116	
14911	055440	044122	051503	020061	
14912	055446	040527	020123	042523	
14913	055454	027124	047440	020116	
14914	055462	042522	042101	047111	
14915	055470	020107	044504	045523	
14916	055476	024040	040524	042520	
14917	055504	020051	042101	051104	
14918	055512	051505	006523	012	
14919	055517	122	043505	051511	.ASCIZ /REGISTER IT DID NOT CONTAIN "1"/
14920	055524	042524	020122	052111	
14921	055532	042040	042111	047040	
14922	055540	052117	041440	047117	
14923	055546	040524	047111	021040	
14924	055554	021061	000		
14925					
14926	055557	101	052106	051105	EM46: .ASCII /AFTER AN RH CLEAR "1" WAS WRITTEN IN THE DISK (TAPE) ADDRESS REGISTER/
14927	055564	040440	020116	044122	
14928	055572	041440	042514	051101	
14929	055600	021040	021061	053440	
14930	055606	051501	053440	044522	
14931	055614	052124	047105	044440	
14932	055622	020116	044124	020105	
14933	055630	044504	045523	024040	
14934	055636	040524	042520	020051	
14935	055644	042101	051104	051505	
14936	055652	020123	042522	044507	
14937	055660	052123	051105	005015	
14938	055666	040520	020124	047111	.ASCII /PAT IN RHCS1 WAS SET AFTER READING DISK (TAPE) ADDRESS REGISTER THE FOL
14939	055674	051040	041510	030523	
14940	055702	053440	051501	051440	
14941	055710	052105	040440	052106	
14942	055716	051105	051040	040505	
14943	055724	044504	043516	042040	
14944	055732	051511	020113	052050	
14945	055740	050101	024505	040440	
14946	055746	042104	042522	051523	
14947	055754	051040	043505	051511	
14948	055762	042524	020122	044124	
14949	055770	020105	047506	046114	
14950	055776	053517	047111	006507	
14951	056004	012			
14952	056005	122	043505	051511	.ASCIZ /REGISTER DID NOT CONTAIN WHAT IS IN "GOOD"/
14953	056012	042524	020122	044504	

14954	056020	020104	047516	020124
14955	056026	047503	052116	044501
14956	056034	020116	044127	052101
14957	056042	044440	020123	047111
14958	056050	021040	047507	042117
14959	056056	000042		
14960				
14961	056060	047101	051040	020110
14962	056066	046103	040505	020122
14963	056074	051050	041510	031123
14964	056102	041040	052111	021440
14965	056110	024465	053440	051501
14966	056116	043440	053111	047105
14967	056124	020056	030501	020066
14968	056132	051050	041510	030523
14969	056140	041040	052111	021440
14970	056146	024470	053440	051501
14971	056154	051440	052105	005015
14972	056162	047117	051040	040505
14973	056170	044504	043516	052040
14974	056176	042510	043040	046117
14975	056204	047514	044527	043516
14976	056212	051040	043505	051511
14977	056220	042524	020122	052111
14978	056226	042040	042111	047040
14979	056234	052117	041440	047117
14980	056242	040524	047111	053440
14981	056250	040510	020124	051511
14982	056256	044440	020116	043442
14983	056264	047517	021104	000
14984				
14985	056271	101	020116	044122
14986	056276	041440	042514	051101
14987	056304	024040	044122	051503
14988	056312	020062	044502	020124
14989	056320	032443	020051	040527
14990	056326	020123	044507	042526
14991	056334	027116	040440	033061
14992	056342	024040	044122	051503
14993	056350	020061	044502	020124
14994	056356	034043	020051	040527
14995	056364	020123	042523	006524
14996	056372	012		
14997	056373	101	046114	055040
14998	056400	051105	051517	053440
14999	056406	051105	020105	051127
15000	056414	052111	042524	020116
15001	056422	047111	051040	041110
15002	056430	042501	020056	044124
15003	056436	047105	052040	042510
15004	056444	043040	046117	047514
15005	056452	044527	043516	051040
15006	056460	043505	051511	042524
15007	056466	020122	044504	020104
15008	056474	047516	020124	040510
15009	056502	042526	053440	040510

EMS4: .ASCII /AN RH CLEAR (RHCS2 BIT #5) WAS GIVEN. A16 (RHCS1 BIT #8) WAS SET<<15>>1

.ASCIZ /ON READING THE FOLLOWING REGISTER IT DID NOT CONTAIN WHAT IS IN "GOOD"/

EMS6: .ASCII /AN RH CLEAR (RHCS2 BIT #5) WAS GIVEN. A16 (RHCS1 BIT #8) WAS SET<<15>>1

.ASCIZ /ALL ZEROS WERE WRITTEN IN RHBAE. THEN THE FOLLOWING REGISTER DID NOT HA

N06

CERHADO MACY11 301046, 21-DEC-77 13:06 PAGE 286
 CERHAD.P11 21-DEC-77 12:48 POWER DOWN AND UP ROUTINES

SEQ 0285

15010	056510	020124	051511	044440	
15011	056516	020116	043442	047517	
15012	056524	021104	000		
15013					
15014	056527	101	020116	044122	EM60: .ASCII AN RH CLEAR (RHCS2 BIT #5) WAS GIVEN. A17 (RHCS1 BIT #9) WAS SET/15/1
15015	056534	041440	042514	051101	
15016	056542	024040	044122	051503	
15017	056550	020062	044502	020124	
15018	056556	032443	020051	040527	
15019	056564	020123	044507	042526	
15020	056572	027116	040440	033461	
15021	056600	024040	044122	051503	
15022	056606	020061	044502	020124	
15023	056614	034443	020051	040527	
15024	056622	020123	042523	006524	
15025	056630	012			
15026	056631	117	020116	042522	.ASCIZ /ON READING THE FOLLOWING REGISTER IT DID NOT CONTAIN WHAT IS IN "GOOD"/
15027	056636	042101	047111	020107	
15028	056644	044124	020105	047506	
15029	056652	046114	053517	047111	
15030	056660	020107	042522	044507	
15031	056666	052123	051105	044440	
15032	056674	020124	044504	020104	
15033	056702	047516	020124	047503	
15034	056710	052116	044501	020116	
15035	056716	044127	052101	044440	
15036	056724	020123	047111	021040	
15037	056732	047507	042117	000042	
15038					
15039	056740	047101	051040	020110	EM62: .ASCII /AN RH CLEAR (RHCS2 BIT #8) WAS GIVEN. A17 (RHCS1 BIT #9) WAS SET/15/1
15040	056746	046103	040505	020122	
15041	056754	051050	041510	031123	
15042	056762	041040	052111	021440	
15043	056770	024470	053440	051501	
15044	056776	043440	053111	047105	
15045	057004	020056	030501	020067	
15046	057012	051050	041510	030523	
15047	057020	041040	052111	021440	
15048	057026	024471	053440	051501	
15049	057034	051440	052105	005015	
15050	057042	046101	020114	042532	.ASCIZ /ALL ZEROS WERE WRITTEN IN RHBAE THEN THE FOLLOWING REGISTER DID NOT HAV
15051	057050	047522	020123	042527	
15052	057056	042522	053440	044522	
15053	057064	052124	047105	044440	
15054	057072	020116	044122	040502	
15055	057100	020105	044124	047105	
15056	057106	052040	042510	043040	
15057	057114	046117	047514	044527	
15058	057122	043516	051040	043505	
15059	057130	051511	042524	020122	
15060	057136	044504	020104	047516	
15061	057144	020124	040510	042526	
15062	057152	053440	040510	020124	
15063	057160	051511	044440	020116	
15064	057166	047507	042117	005015	
15065	057174	000			

15066						
15067	057175	101	020116	044122	EM64:	.ASCII /AN RH CLEAR (RHCS2-BIT #5) WAS GIVEN. IE (RHCS1 BIT #6) WAS SET/'15'12
15068	057202	041440	042514	051101		
15069	057210	024040	044122	051503		
15070	057216	026462	044502	020124		
15071	057224	032443	020051	040527		
15072	057232	020123	044507	042526		
15073	057240	027116	044440	020105		
15074	057246	051050	041510	030523		
15075	057254	041040	052111	021440		
15076	057262	024466	053440	051501		
15077	057270	051440	052105	005015		
15078	057276	047117	051040	040505	.ASCII	/ON READING THE FOLLOWING REGISTER IT DID NOT CONTAIN WHAT IS IN "GOOD"
15079	057304	044504	043516	052040		
15080	057312	042510	043040	046117		
15081	057320	047514	044527	043516		
15082	057326	051040	043505	051511		
15083	057334	042524	020122	052111		
15084	057342	042040	042111	047040		
15085	057350	052117	041440	047117		
15086	057356	040524	047111	053440		
15087	057364	040510	020124	051511		
15088	057372	044440	020116	043442		
15089	057400	047517	021104	000		
15090						
15091	057405	101	020116	044122	EM66:	.ASCII /AN RH CLEAR (RHCS2-BIT #5) WAS GIVEN. IE (RHCS1 BIT #6) WAS SET/'15'12
15092	057412	041440	042514	051101		
15093	057420	024040	044122	051503		
15094	057426	026462	044502	020124		
15095	057434	032443	020051	040527		
15096	057442	020123	044507	042526		
15097	057450	027116	044440	020105		
15098	057456	051050	041510	030523		
15099	057464	041040	052111	021440		
15100	057472	024466	053440	051501		
15101	057500	051440	052105	005015		
15102	057506	046101	020114	042532	.ASCII	/ALL ZEROS WERE WRITTEN IN RHCS3 THEN THE FOLLOWING REGISTER DID NOT CON
15103	057514	047522	020123	042527		
15104	057522	042522	053440	044522		
15105	057530	052124	047105	044440		
15106	057536	020116	044122	051503		
15107	057544	020063	044124	047105		
15108	057552	052040	042510	043040		
15109	057560	046117	047514	044527		
15110	057566	043516	051040	043505		
15111	057574	051511	042524	020122		
15112	057602	044504	020104	047516		
15113	057610	020124	047503	052116		
15114	057616	044501	020116	044127		
15115	057624	052101	044440	020123		
15116	057632	047111	021040	047507		
15117	057640	042117	000042			
15118						
15119	057644	044122	041440	042514	EM70:	.ASCII /RH CLEAR WAS GIVEN. TWO SUCCESSIVE "GO" (RHCS1-BIT #0) WAS GIVEN/'15'11
15120	057652	051101	053440	051501		
15121	057660	043440	053111	047105		

15122	057666	020056	053524	020117
15123	057674	052523	041503	051505
15124	057702	044523	042526	021040
15125	057710	047507	020042	051050
15126	057716	041510	030523	041055
15127	057724	052111	021440	024460
15128	057732	043440	051501	043440
15129	057740	053111	047105	005015
15130	057746	044527	044124	052517
15131	057754	020124	040527	052111
15132	057762	047111	020107	047506
15133	057770	020122	044124	020105
15134	057776	044506	051522	020124
15135	060004	043442	021117	052040
15136	060012	020117	047503	050115
15137	060020	042514	042524	020056
15138	060026	044124	020105	047506
15139	060034	046114	053517	047111
15140	060042	020107	042522	044507
15141	060050	052123	051105	005015
15142	060056	044504	020104	047516
15143	060064	020124	047503	052116
15144	060072	044501	020116	044127
15145	060100	052101	044440	020123
15146	060106	047111	021040	047507
15147	060114	042117	000042	
15148				
15149	060120	044122	041440	042514
15150	060126	051101	053440	051501
15151	060134	043440	053111	047105
15152	060142	040440	031040	053440
15153	060150	051117	020104	051127
15154	060156	052111	020105	040527
15155	060164	020123	047504	042516
15156	060172	043040	047522	020115
15157	060200	020101	047514	040503
15158	060206	044524	047117	005015
15159	060214	040524	042507	020104
15160	060222	051127	051106	046517
15161	060230	040440	042116	053440
15162	060236	052111	020110	040502
15163	060244	020111	051050	041510
15164	060252	031123	041040	052111
15165	060260	021440	024463	051440
15166	060266	052105	040440	020124
15167	060274	044124	020105	047105
15168	060302	006504	012	
15169	060305	117	020106	044124
15170	060312	020105	051127	052111
15171	060320	020105	044124	020105
15172	060326	047506	046114	053517
15173	060334	047111	020107	042522
15174	060342	044507	052123	051105
15175	060350	042040	042111	047040
15176	060356	052117	041440	047117
15177	060364	040524	047111	053440

.ASCII WITHOUT WAITING FOR THE FIRST "GO" TO COMPLETE. THE FOLLOWING REGISTER/

.ASCII /DID NOT CONTAIN WHAT IS IN "GOOD"/

EM76: .ASCII /RH CLEAR WAS GIVEN A 2 WORD WRITE WAS DONE FROM A LOCATION/ 15/ 12

.ASCII /TAGED WRFROM AND WITH BAI (R4CS2 BIT #3) SET AT THE END/ 15 / 12/

.ASCII /OF THE WRITE THE FOLLOWING REGISTER DID NOT CONTAIN WHAT IS IN "GOOD" /

15178	060372	040510	020124	051511
15179	060400	044440	020116	043442
15180	060406	047517	021104	000
15181				
15182	060413	0405122	020110	046103
15183	060420	040505	020122	040527
15184	060426	020123	044507	042526
15185	060434	020116	047101	044440
15186	060442	041520	020113	044502
15187	060450	020124	044123	053517
15188	060456	020116	047111	021040
15189	060464	050111	045503	020042
15190	060472	040527	020123	042523
15191	060500	006524	012	
15192	060503	132	051105	051517
15193	060510	053440	051105	020105
15194	060516	047515	042526	020104
15195	060524	047111	047524	051040
15196	060532	042110	027102	047440
15197	060540	020116	042522	042101
15198	060546	047111	020107	04222
15199	060554	041104	052040	042510
15200	060562	043040	046117	047514
15201	060570	044527	043516	005015
15202	060576	042522	044507	052123
15203	060604	051105	042040	042111
15204	060612	047040	052117	041440
15205	060620	047117	040524	047111
15206	060626	053440	040510	020124
15207	060634	051511	044440	020116
15208	060642	043442	047517	021104
15209	060650	000		
15210				
15211	060651	122	020110	046103
15212	060656	040505	020122	040527
15213	060664	020123	044507	042526
15214	060672	020116	047101	044440
15215	060700	041520	020113	044502
15216	060706	020124	044123	053517
15217	060714	020116	047111	021040
15218	060722	050111	045503	020042
15219	060730	040527	020123	042523
15220	060736	006524	012	
15221	060741	132	051105	051517
15222	060746	053440	051105	020105
15223	060754	047515	042526	020104
15224	060762	047111	047524	051040
15225	060770	042110	027102	051040
15226	060776	042110	020102	040527
15227	061004	020123	042522	042101
15228	061012	020056	047101	051040
15229	061020	020110	046103	040505
15230	061026	020122	040527	020123
15231	061034	044507	042526	006516
15232	061042	012		
15233	061043	124	042510	043040

EM104: .ASCII RH CLEAR WAS GIVEN AN IPCK BIT SHOWN IN "IPCK" WAS SET / 15 / 12

.ASCII ZEROS WERE MOVED INTO RHDB. ON READING RHDB THE FOLLOWING / 15 / 12 /

.ASCII /REGISTER DID NOT CONTAIN WHAT IS IN "GOOD"/

EM106: .ASCII /RH CLEAR WAS GIVEN AN IPCK BIT SHOWN IN "IPCK" WAS SET / 15 / 12 /

.ASCII /ZEROS WERE MOVED INTO RHDB. RHDB WAS READ. AN RH CLEAR WAS GIVEN / 15 / 1 /

.ASCII THE FOLLOWING REGISTER DID NOT CONTAIN WHAT IS GIVEN IN "GOOD" /

15234	061050	046117	047514	044527
15235	061056	043516	051040	043505
15236	061064	051511	042524	020122
15237	061072	044504	020104	047516
15238	061100	020124	047503	052116
15239	061106	044501	020116	044127
15240	061114	052101	044440	020123
15241	061122	044507	042526	020116
15242	061130	047111	021040	047507
15243	061136	042117	000042	
15244				
15245	061142	044122	041440	042514
15246	061150	051101	053440	051501
15247	061156	043440	053111	047105
15248	061164	020056	040440	053440
15249	061172	044522	042524	041440
15250	061200	042510	045503	005015
15251	061206	040527	020123	047504
15252	061214	042516	020056	044124
15253	061222	020105	047506	046114
15254	061230	053517	047111	020107
15255	061236	042522	044507	052123
15256	061244	051105	042040	042111
15257	061252	047040	052117	041440
15258	061260	047117	040524	047111
15259	061266	053440	040510	020124
15260	061274	051511	044440	020116
15261	061302	043442	047517	021104
15262	061310	000		
15263				
15264	061311	122	020110	046103
15265	061316	040505	020122	040527
15266	061324	020123	044507	042526
15267	061332	027116	020040	020101
15268	061340	051127	052111	020105
15269	061346	044103	041505	006513
15270	061354	012		
15271	061355	127	051501	042040
15272	061362	047117	020105	044124
15273	061370	047105	040440	047516
15274	061376	044124	051105	051040
15275	061404	020110	046103	040505
15276	061412	020122	040527	020123
15277	061420	044507	042526	027116
15278	061426	052040	042510	043040
15279	061434	046117	047514	044527
15280	061442	043516	051040	043505
15281	061450	051511	042524	006522
15282	061456	012		
15283	061457	104	042111	047040
15284	061464	052117	041440	047117
15285	061472	040524	047111	053440
15286	061500	040510	020124	051511
15287	061506	044440	020116	043442
15288	061514	047517	021104	000
15289				

EM114: .ASCII RH CLEAR WAS GIVEN. A WRITE CHECK /15 /12

.ASCIZ /WAS DONE. THE FOLLOWING REGISTER DID NOT CONTAIN WHAT IS IN "GOOD"

EM116: .ASCII /RH CLEAR WAS GIVEN. A WRITE CHECK /15 /12

.ASCII /WAS DONE THEN ANOTHER RH CLEAR WAS GIVEN. THE FOLLOWING REGISTER /15 /1

.ASCIZ /DID NOT CONTAIN WHAT IS IN "GOOD".

15290	061521	117	020116	020101	EM124: .ASCII	ON A SILO TEST TO TEST DBL RMCS3 BIT #10 THE FOLLOWING/15/12
15291	061526	044523	047514	052040		
15292	061534	051505	020124	047524		
15293	061542	052040	051505	020124		
15294	061550	041104	020114	044122		
15295	061556	051503	020063	044502		
15296	061564	020124	030443	020060		
15297	061572	044124	020105	047506		
15298	061600	046114	053517	047111		
15299	061606	006507	012			
15300	061611	122	043505	051511	.ASCIZ	REGISTER DID NOT CONTAIN WHAT IS IN "GOOD"/
15301	061616	042524	020122	044504		
15302	061624	020104	047516	020124		
15303	061632	047503	052116	044501		
15304	061640	020116	044127	052101		
15305	061646	044440	020123	047111		
15306	061654	021040	047507	042117		
15307	061662	000042				
15308	061664	042502	047506	042522	EM132: .ASCII	BEFORE DATA TRANSFER COMMAND WAS TO/15/12
15309	061672	042040	052101	020101		
15310	061700	051124	047101	043123		
15311	061706	051105	041440	046517		
15312	061714	040515	042116	053440		
15313	061722	051501	052040	006517		
15314	061730	012				
15315	061731	102	020105	044507	.ASCII	BE GIVEN THE RP DRIVE STATUS REGISTER DID/15/12
15316	061736	042526	020116	044124		
15317	061744	020105	050122	042040		
15318	061752	044522	042526	051440		
15319	061760	040524	052524	020123		
15320	061766	042522	044507	052123		
15321	061774	051105	042040	042111		
15322	062002	005015				
15323	062004	047516	020124	047503	.ASCIZ	NOT CONTAIN WHAT IS IN "GOOD"/
15324	062012	052116	044501	020116		
15325	062020	044127	052101	044440		
15326	062026	020123	047111	021040		
15327	062034	047507	042117	000042		
15328	062042	042502	047506	042522	EM133: .ASCII	BEFORE DATA TRANSFER COMMAND WAS TO/15/12
15329	062050	042040	052101	020101		
15330	062056	051124	047101	043123		
15331	062064	051105	041440	046517		
15332	062072	040515	042116	053440		
15333	062100	051501	052040	006517		
15334	062106	012				
15335	062107	102	020105	044507	.ASCII	BE GIVEN THE RS DRIVE STATUS REGISTER DID/15/12
15336	062114	042526	020116	044124		
15337	062122	020105	051522	042040		
15338	062130	044522	042526	051440		
15339	062136	040524	052524	020123		
15340	062144	042522	044507	052123		
15341	062152	051105	042040	042111		
15342	062160	005015				
15343	062162	047516	020124	047503	.ASCIZ	NOT CONTAIN WHAT IS IN "GOOD"/
15344	062170	052116	044501	020116		
15345	062176	044127	052101	044440		

15346	062204	020123	047111	021040	
15347	062212	047507	042117	000042	
15348	062220	042502	047506	042522	EM134: .ASCII BEFORE DATA TRANSFER COMMAND WAS TO BE GIVEN <15> <12>
15349	062226	042040	052101	020101	
15350	062234	051124	047101	043123	
15351	062242	051105	041440	046517	
15352	062250	040515	042116	053440	
15353	062256	051501	052040	020117	
15354	062264	042502	043440	053111	
15355	062272	047105	005015		
15356	062276	044124	020105	040515	.ASCII THE MAG TAPE DRIVE STATUS REGISTER DID NOT <15> <12>
15357	062304	020107	040524	042520	
15358	062312	042040	044522	042526	
15359	062320	051440	040524	052524	
15360	062326	020123	042522	044507	
15361	062334	052123	051105	042040	
15362	062342	042111	047040	052117	
15363	062350	005015			
15364	062352	047503	052116	044501	.ASCIIZ /CONTAIN WHAT IS IN RHDS1-GOOD/
15365	062360	020116	044127	052101	
15366	062366	044440	020123	047111	
15367	062374	051040	042110	030523	
15368	062402	043455	047517	000104	
15369	062410	040527	020123	040527	EM135: .ASCII /WAS WAITING FOR A BIT TO SET IN A REGISTER <15> <12>
15370	062416	052111	047111	020107	
15371	062424	047506	020122	020101	
15372	062432	044502	020124	047524	
15373	062440	051440	052105	044440	
15374	062446	020116	020101	042522	
15375	062454	044507	052123	051105	
15376	062462	005015			
15377	062464	044502	020124	047111	.ASCIIZ /BIT IN QUESTION IS IN "BIT WAITED" <15> <12>
15378	062472	050440	042525	052123	
15379	062500	047511	020116	051511	
15380	062506	044440	020116	041042	
15381	062514	052111	053440	044501	
15382	062522	042524	021104	005015	
15383	062530	000			
15384	062531	122	043505	051511	.ASCIIZ /REGISTER ADDRESS IN QUESTION IS IN "REG. ADDR" /
15385	062536	042524	020122	042101	
15386	062544	051104	051505	020123	
15387	062552	047111	050440	042525	
15388	062560	052123	047511	020116	
15389	062566	051511	044440	020116	
15390	062574	051042	043505	020056	
15391	062602	042101	051104	000042	
15392	062610	040527	020123	040527	EM136: .ASCII /WAS WAITING FOR A BIT TO RESET IN A REGISTER <15> <12>
15393	062616	052111	047111	020107	
15394	062624	047506	020122	020101	
15395	062632	044502	020124	047524	
15396	062640	051040	051505	052105	
15397	062646	044440	020116	020101	
15398	062654	042522	044507	052123	
15399	062662	051105	005015		
15400	062666	044502	020124	047111	.ASCIIZ BIT IN QUESTION IS IN "BIT WAITED" <15> <12>
15401	062674	050440	042525	052123	

15402	062702	047511	020116	051511	
15403	062710	044440	020116	041042	
15404	062716	052111	053440	044501	
15405	062724	042524	021104	005015	
15406	062732	000			
15407	062733	122	043505	051511	.ASCIZ REGISTER ADDRESS IN QUESTION IS IN "REG. ADDR"
15408	062740	042524	020122	042101	
15409	062746	051104	051505	020123	
15410	062754	047111	050440	042525	
15411	062762	052123	047511	020116	
15412	062770	051511	044440	020116	
15413	062776	051042	043505	020056	
15414	063004	042101	051104	000042	
15415	063012	047111	053440	044522	EM137: .ASCII /ON WRITING AND READING THE REGISTER IN "REG. ADDR"/15/12
15416	063020	044524	043516	040440	
15417	063026	042116	051040	040505	
15418	063034	044504	043516	052040	
15419	063042	042510	051040	043505	
15420	063050	051511	042524	020122	
15421	063056	047111	021040	042522	
15422	063064	027107	040440	042104	
15423	063072	021122	005015		
15424	063076	052111	042040	042111	.ASCIZ /IT DID NOT CONTAIN EXPECTED VALUE/
15425	063104	047040	052117	041440	
15426	063112	047117	040524	047111	
15427	063120	042440	050130	041505	
15428	063126	042524	020104	040526	
15429	063134	052514	000105		
15430	063140	044122	041440	042514	EM140: .ASCII /RH CLEAR WAS GIVEN AN IPCK BIT SHOWN IN "IPCK" WAS SET /15/12
15431	063146	051101	053440	051501	
15432	063154	043440	053111	047105	
15433	063162	040440	020116	050111	
15434	063170	045503	041040	052111	
15435	063176	051440	047510	047127	
15436	063204	044440	020116	044442	
15437	063212	041520	021113	053440	
15438	063220	051501	051440	052105	
15439	063226	005015			
15440	063230	042532	047522	020123	.ASCII /ZEROS WERE MOVED INTO RHDB. ON READING THE FOLLOWING/15/12
15441	063236	042527	042522	046440	
15442	063244	053117	042105	044440	
15443	063252	052116	020117	044122	
15444	063260	041104	020056	047117	
15445	063266	051040	040505	044504	
15446	063274	043516	020040	044124	
15447	063302	020105	047506	046114	
15448	063310	053517	047111	006507	
15449	063316	012			
15450	063317	122	043505	051511	.ASCIZ /REGISTER DID NOT CONTAIN WHAT IS IN "GOOD"/
15451	063324	042524	020122	044504	
15452	063332	020104	047516	020124	
15453	063340	047503	052116	044501	
15454	063346	020116	044127	052101	
15455	063354	044440	020123	047111	
15456	063362	021040	047507	042117	
15457	063370	000042			

15458	063372	043101	042524	020122
15459	063400	046103	040505	044522
15460	063406	043516	052040	042510
15461	063414	051040	020110	047101
15462	063422	020104	051127	052111
15463	063430	047111	020107	053524
15464	063436	020117	047527	042122
15465	063444	044440	052116	020117
15466	063452	044122	041104	040440
15467	063460	042116	005015	
15468	063464	042522	042101	047111
15469	063472	020107	052111	041040
15470	063500	041501	020113	053524
15471	063506	041511	026105	041440
15472	063514	052501	042523	020104
15473	063522	044122	051503	020061
15474	063530	047524	044040	053101
15475	063536	020105	051127	047117
15476	063544	020107	040526	052514
15477	063552	020105	044507	042526
15478	063560	020116	047111	021040
15479	063566	040502	021104	000
15480	063573	101	052106	051105
15481	063600	041440	042514	051101
15482	063606	047111	020107	044124
15483	063614	020105	044122	040440
15484	063622	042116	053440	044522
15485	063630	044524	043516	052040
15486	063636	047527	053440	051117
15487	063644	020104	047111	047524
15488	063652	051040	042110	020102
15489	063660	047101	006504	012
15490	063665	122	040505	044504
15491	063672	043516	044440	020124
15492	063700	040502	045503	052040
15493	063706	044527	042503	020054
15494	063714	040503	051525	042105
15495	063722	051040	041510	031523
15496	063730	052040	020117	040510
15497	063736	042526	053440	047522
15498	063744	043516	053040	046101
15499	063752	042525	043440	053111
15500	063760	047105	044440	020116
15501	063766	041042	042101	000042
15502	063774	043101	042524	020122
15503	064002	046103	040505	044522
15504	064010	043516	052040	042510
15505	064016	051040	020110	047101
15506	064024	020104	051127	052111
15507	064032	047111	020107	053524
15508	064040	020117	047527	042122
15509	064046	044440	052116	020117
15510	064054	044122	041104	040440
15511	064062	042116	005015	
15512	064066	042522	042101	047111
15513	064074	020107	052111	041040

EM141: .ASCII AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND (15)(12)

.ASCIZ /READING IT BACK TWICE, CAUSED RHCS1 TO HAVE WRONG VALUE GIVEN IN "BAD"

EM142: .ASCII /AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND (15)(12)

.ASCIZ /READING IT BACK TWICE, CAUSED RHCS3 TO HAVE WRONG VALUE GIVEN IN "BAD"

EM143: .ASCII AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND (15)(12)

.ASCIZ /READING IT BACK TWICE, CAUSED RHBA TO HAVE WRONG VALUE GIVEN IN "BAD"

15514	064102	041501	020113	053524
15515	064110	041511	026105	041440
15516	064116	052501	042523	020104
15517	064124	044122	040502	052040
15518	064132	020117	040510	042526
15519	064140	053440	047522	043516
15520	064146	053040	046101	042525
15521	064154	043440	053111	047105
15522	064162	044440	020116	041042
15523	064170	042101	000042	
15524	064174	043101	042524	020122
15525	064202	046103	040505	044522
15526	064210	043516	052040	042510
15527	064216	051040	020110	047101
15528	064224	020104	051127	052111
15529	064232	047111	020107	053524
15530	064240	020117	047527	042122
15531	064246	044440	052116	020117
15532	064254	044122	041104	040440
15533	064262	042116	005015	
15534	064266	042522	042101	047111
15535	064274	020107	052111	041040
15536	064302	041501	020113	053524
15537	064310	041511	026105	041440
15538	064316	052501	042523	020104
15539	064324	044122	040502	020105
15540	064332	047524	044040	053101
15541	064340	020105	051127	047117
15542	064346	020107	040526	052514
15543	064354	020105	044507	042526
15544	064362	020116	047111	021040
15545	064370	040502	021104	000
15546	064375	101	052106	051105
15547	064402	041440	042514	051101
15548	064410	047111	020107	044124
15549	064416	020105	044122	040440
15550	064424	042116	053440	044522
15551	064432	044524	043516	052040
15552	064440	047527	053440	051117
15553	064446	020104	047111	047524
15554	064454	051040	042110	020102
15555	064462	047101	006504	012
15556	064467	122	040505	044504
15557	064474	043516	044440	020124
15558	064502	040502	045503	052040
15559	064510	044527	042503	020054
15560	064516	040503	051525	042105
15561	064524	051040	053510	020103
15562	064532	047524	044040	053101
15563	064540	020105	051127	047117
15564	064546	020107	040526	052514
15565	064554	020105	044507	042526
15566	064562	020116	047111	021040
15567	064570	040502	021104	000
15568				
15569	064575	101	042040	053105

EM144: .ASCII <AFTER CLEARING THE PH AND WRITING TWO WORD INTO RHDB AND<<15>>12>

.ASCIZ <READING IT BACK TWICE, CAUSED RHBAE TO HAVE WRONG VALUE GIVEN IN "BAD">

EM145: .ASCII <AFTER CLEARING THE RH AND WRITING TWO WORD INTO RHDB AND<<15>>12>

.ASCIZ <READING IT BACK TWICE, CAUSED RHWC TO HAVE WRONG VALUE GIVEN IN "BAD">

EM146: .ASCII <A DEVICE BASE ADDRESS DID NOT TIME OUT BUT <15>12>

15570	064602	041511	020105	040502				
15571	064610	042523	040440	042104				
15572	064616	042522	051523	042040				
15573	064624	042111	047040	052117				
15574	064632	052040	046511	020105				
15575	064640	052517	020124	052502				
15576	064646	006524	012					
15577	064651	124	042510	041440	.ASCII	THE CORRESPONDING VECTOR DID TIME OUT.	<15	<12
15578	064656	051117	042522	050123				
15579	064664	047117	044504	043516				
15580	064672	053040	041505	047524				
15581	064700	020122	044504	020104				
15582	064706	044524	042515	047440				
15583	064714	052125	006456	012				
15584	064721	110	052111	041440	.ASCIIZ	HIT CONTINUE TO REPEAT VECTOR TIMECUT		
15585	064726	047117	044524	052516				
15586	064734	020105	047524	051040				
15587	064742	050105	040505	020124				
15588	064750	042526	052103	051117				
15589	064756	052040	046511	047505				
15590	064764	052125	000					
15591	064767	101	042040	053105	EM147:	.ASCII	A DEVICE BASE ADDRESS DID NOT TIME OUT	<15><12
15592	064774	041511	020105	040502				
15593	065002	042523	040440	042104				
15594	065010	042522	051523	042040				
15595	065016	042111	047040	052117				
15596	065024	052040	046511	020105				
15597	065032	052517	006524	012				
15598	065037	102	052125	047040	.ASCIIZ	BUT NO UNIT NUMBERS HAD APPROPRIATE DRIVE TYPES		
15599	065044	020117	047125	052111				
15600	065052	047040	046525	042502				
15601	065060	051522	044040	042101				
15602	065066	040440	050120	047522				
15603	065074	051120	040511	042524				
15604	065102	042040	044522	042526				
15605	065110	052040	050131	051505				
15606	065116	000						
15607								
15608								
15609	065117	120	004503	044122	DH1:	.ASCII	PC	RHDT
15610	065124	052104						
15611								
15612	065126	041520	052411	044516	DH5:	.ASCII	PC	UNIBUS ADDRESS ON WHICH TIME OUT OCCURRED
15613	065134	052502	020123	042101				
15614	065142	051104	051505	020123				
15615	065150	047117	053440	044510				
15616	065156	044103	052040	046511				
15617	065164	020105	052517	020124				
15618	065172	041517	052503	051122				
15619	065200	042105						
15620								
15621	065202	041520	052011	051505	DH6:	.ASCII	PC	TEST NO RHCS2 RHCS2 <15 12
15622	065210	020124	047516	051040				
15623	065216	041510	031123	051011				
15624	065224	041510	031123	005015				
15625	065232	004411	047507	042117	.ASCIIZ		GOOD	BAD

15626	065240	041011	042101	000							
15627											
15628	065245	120	004503	042524	DH7:	.ASCII	/PC	TEST	RHDB	RHDB/<15><12>	
15629	065252	052123	051011	042110							
15630	065260	004502	044122	041104							
15631	065266	005015									
15632	065270	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15633	065276	042117	041011	042101							
15634	065304	000									
15635											
15636	065305	120	004503	042524	DH11:	.ASCII	/PC	TEST	RHCS1	RHCS1/<15><12>	
15637	065312	052123	051011	041510							
15638	065320	030523	051011	041510							
15639	065326	030523	005015								
15640	065332	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15641	065340	042117	041011	042101							
15642	065346	000									
15643											
15644	065347	120	004503	042524	DH12:	.ASCII	/PC	TEST	RHCS3	RHCS3/<15><12>	
15645	065354	052123	051011	041510							
15646	065362	031523	051011	041510							
15647	065370	031523	005015								
15648	065374	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15649	065402	042117	041011	042101							
15650	065410	000									
15651											
15652	065411	120	004503	042524	DH13:	.ASCII	/PC	TEST	RHBA	RHBA/<15><12>	
15653	065416	052123	051011	041110							
15654	065424	004501	044122	040502							
15655	065432	005015									
15656	065434	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15657	065442	042117	041011	042101							
15658	065450	000									
15659											
15660	065451	120	004503	042524	DH14:	.ASCII	/PC	TEST	RHBAE	RHBAE/<15><12>	
15661	065456	052123	051011	041110							
15662	065464	042501	051011	041110							
15663	065472	042501	005015								
15664	065476	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15665	065504	042117	041011	042101							
15666	065512	000									
15667											
15668	065513	120	004503	042524	DH15:	.ASCII	/PC	TEST	RHWC	RHWC/<15><12>	
15669	065520	052123	051011	053510							
15670	065526	004503	044122	041527							
15671	065534	005015									
15672	065536	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15673	065544	042117	041011	042101							
15674	065552	000									
15675											
15676	065553	120	004503	042524	DH24:	.ASCII	/PC	TEST	WORD	RHDB	RHDB/<15><12>
15677	065560	052123	053411	051117							
15678	065566	004504	044122	041104							
15679	065574	051011	042110	006502							
15680	065602	012									
15681	065603	011	047516	047011		.ASCIZ	/	NO	NO	GOOD	BAD/

15682	065610	004517	047507	042117							
15683	065616	041011	042101	000							
15684											
15685	065623	120	004503	042524	DH33:	.ASCII	/PC	TEST	RHER1	RHER1	<15><12>
15686	065630	052123	051011	042510							
15687	065636	030522	051011	042510							
15688	065644	030522	005015								
15689	065650	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15690	065656	042117	041011	042101							
15691	065664	000									
15692											
15693	065665	120	004503	042524	DH45:	.ASCII	/PC	TEST	RHDA	RHDA	<15><12>
15694	065672	052123	051011	042110							
15695	065700	004501	044122	040504							
15696	065706	005015									
15697	065710	047011	004517	047507		.ASCIZ	/	NO	GOOD	BAD/	
15698	065716	042117	041011	042101							
15699	065724	000									
15700											
15701	065725	120	004503	042524	DH104:	.ASCII	/PC	TEST	IPCK	RHCS3	RHCS3/<15><12>
15702	065732	052123	044411	041520							
15703	065740	004513	044122	051503							
15704	065746	004463	044122	051503							
15705	065754	006463	012								
15706	065757	011	047516	004411		.ASCIZ	/	NO	GOOD	BAD/	
15707	065764	047507	042117	041011							
15708	065772	042101	000								
15709											
15710	065775	120	004503	042524	DH105:	.ASCII	/PC	TEST	IPCK	RHCS2	RHCS2/<15><12>
15711	066002	052123	044411	041520							
15712	066010	004513	044122	051503							
15713	066016	004462	044122	051503							
15714	066024	006462	012								
15715	066027	011	047516	004411		.ASCIZ	/	NO	GOOD	BAD/	
15716	066034	047507	042117	041011							
15717	066042	042101	000								
15718											
15719	066045	120	004503	042524	DH106:	.ASCII	/PC	TEST	IPCK	RHCS1	RHCS1/<15><12>
15720	066052	052123	044411	041520							
15721	066060	004513	044122	051503							
15722	066066	004461	044122	051503							
15723	066074	006461	012								
15724	066077	011	047516	004411		.ASCIZ	/	NO	GOOD	BAD/	
15725	066104	047507	042117	041011							
15726	066112	042101	000								
15727											
15728	066115	120	004503	042524	DH111:	.ASCII	/PC	TEST	IPCK	RHBA	RHBA/<15><12>
15729	066122	052123	044411	041520							
15730	066130	004513	044122	040502							
15731	066136	051011	041110	006501							
15732	066144	012									
15733	066145	011	047516	004411		.ASCIZ	/	NO	GOOD	BAD/	
15734	066152	047507	042117	041011							
15735	066160	042101	000								
15736											
15737	066163	120	004503	042524	DH112:	.ASCII	/PC	TEST	IPCK	RHBAE	RHBAE/<15><12>

15738	066170	052123	044411	041520						
15739	066176	004513	044122	040502						
15740	066204	004505	044122	040502						
15741	066212	006505	012							
15742	066215	011	047516	004411	.ASCIZ	NO	GOOD	BAD/		
15743	066222	047507	042117	041011						
15744	066230	042101	000							
15745										
15746	066233	120	004503	042524	DH113:	.ASCII	/PC	TEST	IPCK	RHWC RHWC/1512
15747	066240	052123	044411	041520						
15748	066246	004513	044122	041527						
15749	066254	051011	053510	006503						
15750	066262	012								
15751	066263	011	047516	004411	.ASCIZ	/	NO	GOOD	BAD/	
15752	066270	047507	042117	041011						
15753	066276	042101	000							
15754	066301	120	004505	042524	DH132:	.ASCII	/PE	TEST	RHDS1	RHDS1/1512
15755	066306	052123	051011	042110						
15756	066314	030523	051011	042110						
15757	066322	030523	005015							
15758	066326	047011	004517	047507	.ASCIZ	/	NO	GOOD	BAD/	
15759	066334	042117	041040	042101						
15760	066342	000								
15761	066343	120	004503	042524	DH135:	.ASCII	/PC	TEST	PC OF	BIT WAIT REG/1512
15762	066350	052123	050011	020103						
15763	066356	043117	041011	052111						
15764	066364	053440	04450	020124						
15765	066372	051040	043505	005015						
15766	066400	047011	004517	040527	.ASCIZ	/	NO	WAT	FOR	ADDR/
15767	066406	004524	047506	004522						
15768	066414	020040	042101	051104						
15769	066422	000								
15770	066423	120	004503	042524	DH137:	.ASCII	/PC	TEST	REG.	GOOD BAD/1512
15771	066430	052123	051011	043505						
15772	066436	004456	047507	042117						
15773	066444	041011	042101	005015						
15774	066452	047011	004517	042101	.ASCIZ	/	NO	ADDR	DATA	DATA/
15775	066460	051104	042011	052101						
15776	066466	004501	040504	040524						
15777	066474	000								
15778										
15779	066475	120	020103	020040	DH146:	.ASCIZ	/PC	BASE	VECTOR/	
15780	066502	020040	041040	051501						
15781	066510	020105	020040	053040						
15782	066516	041505	047524	000122						
15783	066524	041520	020040	020040	DH147:	.ASCIZ	/PC	BASE	ADDRESS	
15784	066532	020040	040502	042523						
15785	066540	040440	042104	042522						
15786	066546	051523	000							
15787										
15788										
15789		066552			.EVEN					
15790										
15791	066552	001116	001126	000000	DT1:	.WORD	\$ERRPC,\$BDDAT,0			
15792	066560	001116	004046	004050	DT5:	.WORD	\$ERRPC,\$RCSI,\$PCSI,\$TMCSI,\$MIXCSI,0			
15793	066566	004052	004054	000000						

15794	066574	001116	004656	001124	DT6:	.WORD	SERRPC,TSTNM,\$GDDAT,\$BDDAT,0
15795	066602	001126	000000				
15796	066606	001116	004656	004664	DT24:	.WORD	SERRPC,TSTNM,SILONM,\$GDDAT,\$BDDAT,0
15797	066614	001124	001126	000000			
15798	066622	001116	004656	004662	DT104:	.WORD	SERRPC,TSTNM,IP,\$GDDAT,\$BDDAT,0
15799	066630	001124	001126	000000			
15800	066636	001116	004656	004662	DT105:	.WORD	SERRPC,TSTNM,IP,\$GDDAT,\$BDDAT,0
15801	066644	001124	001126	000000			
15802	066652	001116	004656	004662	DT106:	.WORD	SERRPC,TSTNM,IP,\$GDDAT,\$BDDAT,0
15803	066660	001124	001126	000000			
15804	066666	001116	004656	004672	DT135:	.WORD	SERRPC,TSTNM,WAITBT,WAITPC,WAITRE,0
15805	066674	004666	004670	000000			
15806	066702	001116	004656	001122	DT137:	.WORD	SERRPC,TSTNM,\$BDADR,\$GDDAT,\$BDDAT,0
15807	066710	001124	001126	000000			
15808	066716	001116	005324	005326	DT146:	.WORD	SERRPC,TESTDV,TESTVC,0
15809	066724	000000					
15810	066726	001116	005324	000000	DT147:	.WORD	SERRPC,TESTDV,0
15811							
15812							
15813							
15814	066734	000	000		DF1:	.BYTE	0,0
15815	066736	000	000	000	DF5:	.BYTE	0,0,0,0,0
15816	066741	000	000				
15817	066743	000	000	000	DF6:	.BYTE	0,0,0,0
15818	066746	000					
15819	066747	000	000	000	DF24:	.BYTE	0,0,0,0,0
15820	066752	000	000				
15821	066754	000	000	000	DF104:	.BYTE	0,0,0,0,0
15822	066757	000	000				
15823	066761	000	000	000	DF105:	.BYTE	0,0,0,0,0
15824	066764	000	000				
15825	066766	000	000	000	DF106:	.BYTE	0,0,0,0,0
15826	066771	000	000				
15827	066773	000	000	000	DF135:	.BYTE	0,0,0,0,0
15828	066776	000	000				
15829	067000	000	000	000	DF137:	.BYTE	0,0,0,0,0
15830	067003	000	000				
15831	067005	000	000	000	DF146:	.BYTE	0,0,0
15832	067010	000	000		DF147:	.BYTE	0,0
15833						.EVEN	
15834		00000:				.END	

USEVEC	005004	2975*												
US1 =	000001	2553*												
US2 =	000002	2554*												
JS4 =	000004	2555*												
JWR =	000010	2772*												
VECPNT	005310	3047*	3252*	3260										
VECTOR	005512	3088*	3666*											
VUF =	000002	2771*												
VU30 =	010000	2707*												
VV =	000100	2620*	12928	12943	13181									
WAITBT	004672	2937*	12977*	12985	12990	13006	15804							
WAITPC	004666	2935*	12974*	12975*	15804									
WAITRE	004670	2936*	12976*	12985	12990	12994	13006	13012	13016	15804				
WAIT.T	040444	12970*	14233											
WAT =	104411	4212	4402	4578	4614	4851	5246	5668	6090	6513	13153	13207	13247	13258
		13269	13285	13306	13317	13328	13353	13364	14233*					
WATO	004646	2921*	12982	13257*	13267*	13316*	13326*	13458*						
WC =	000002	2799*	13105											
WCE =	040000	2568*	9076	9309	9542	9775								
WCEEW =	004000	2579*	9565	9798										
WCEOW =	010000	2580*	9099	9332										
WCF =	000040	2637*												
WCU =	000001	2695*												
WLE =	004000	2643*												
WORKBS	004770	2969*												
WORKNM	004772	2970*												
WORKVC	004774	2971*												
WORUNT	005510	3085*	3537*	3580	12759	12786*	12789*							
WRCHDT	004152	2897*												
WRCHK	004150	2896*	9070	9303	9536	9769								
WRFROM	004210	2914*	7725	7834	7898	7977*	7978	8213*	8214	8450*	8451	8452	8752*	8753
		8754	9056*	9058	9289*	9291	9522*	9524	9755*	9757				
WRIDAT	004154	2898*	7730	7836	9061	9294	9527	9760	9994	10107	10221	10334	10449	10566
WRIFOR	004156	2899*												
WRL =	004000	2625*												
WRTBIT	004650	2924*	3706*	3786*	3864*	3944*	4022*	4103*						
WRU =	000400	2703*												
WSL =	000004	2697*												
\$AUTOB	001134	1467*												
\$BDADP	001122	1462*	3725*	3751*	3803*	3828*	3883*	3909*	3961*	3986*	4041*	4067*	4122*	4148*
		12938*	12945*	15806										
\$BDDAT	001126	1464*	3284*	3288	3299*	3300	3720*	3723	3746*	3749	3799*	3801	3824*	3826
		3878*	3881	3904*	3907	3957*	3959	3982*	3984	4036*	4039	4062*	4065	4117*
		4120	4143*	4146	4192*	4193	4226*	4227	4245*	4246	4264*	4265	4282*	4283
		4299*	4300	4317*	4318	4335*	4336	4382*	4383	4416*	4417	4434*	4435	4453*
		4454	4471*	4472	4489*	4490	4507*	4508	4557*	4558	4593*	4594	4627*	4628
		4645*	4646	4667*	4668	4686*	4687	4706*	4707	4725*	4726	4743*	4744	4761*
		4762	4779*	4780	4825*	4826	4865*	4866	4889*	4890	4911*	4912	4933*	4934
		4955*	4956	4977*	4978	4999*	5000	5021*	5022	5043*	5044	5066*	5067	5088*
		5089	5109*	5110	5129*	5130	5149*	5150	5169*	5170	5220*	5221	5261*	5262
		5286*	5287	5310*	5311	5334*	5335	5358*	5359	5382*	5383	5406*	5407	5430*
		5431	5454*	5455	5478*	5479	5501*	5502	5523*	5524	5544*	5545	5565*	5566
		5586*	5587	5640*	5641	5683*	5684	5709*	5710	5733*	5734	5757*	5758	5781*
		5782	5805*	5806	5829*	5830	5853*	5854	5877*	5878	5901*	5902	5924*	5925
		5946*	5947	5967*	5968	5988*	5989	6009*	6010	6062*	6063	6105*	6106	6130*
		6131	6154*	6155	6178*	6179	6202*	6203	6226*	6227	6250*	6251	6274*	6275

CROSS REFERENCE TABLE -- USER SYMBOLS

6298*	6299	6322*	6323	6345*	6346	6367*	6368	6388*	6389	6409*	6410	6430*
6431	6483*	6484	6528*	6529	6554*	6555	6579*	6580	6604*	6605	6623*	6637*
6654*	6655	6679*	6680	6704*	6705	6729*	6730	6754*	6755	6778*	6779*	6800*
6802	6923*	6824	6845*	6846	6867*	6868	6919*	6920	6941*	6942	6962*	6963*
6987*	6988	7018*	7019	7044*	7045	7070*	7071	7094*	7095	7118*	7119	7142*
7143	7201*	7202	7222*	7223	7243*	7244	7264*	7265	7283*	7284	7302*	7303*
7321*	7322	7370*	7371	7350*	7391	7414*	7415	7437*	7438	7490*	7491	7510*
7511	7534*	7535	7557*	7558	7610*	7611	7630*	7631	7654*	7655	7677*	7678*
7741*	7742	7761*	7762	7781*	7782	7842*	7843	7862*	7863	7882*	7883	7900*
7901	7918*	7919	7936*	7937	7983*	7984	8004*	8005	8026*	8027	8054*	8055*
8078*	8079	8102*	8103	8124*	8125	8146*	8147	8168*	8169	8219*	8220	8240*
8241	8262*	8263	8290*	8291	8314*	8315	8338*	8339	8360*	8361	8382*	8383*
8404*	8405	8457*	8458	8479*	8480	8501*	8502	8524*	8525	8545*	8546	8566*
8567	8592*	8593	8616*	8617	8640*	8641	8662*	8663	8684*	8685	8706*	8707*
8759*	8760	8781*	8782	8803*	8804	8826*	8827	8847*	8848	8868*	8869	8894*
8895	8918*	8919	8942*	8943	8964*	8965	8986*	8987	9008*	9009	9079*	9080*
9101*	9102	9128*	9129	9152*	9153	9176*	9177	9198*	9199	9220*	9221	9242*
9243	9312*	9313	9334*	9335	9361*	9362	9385*	9386	9409*	9410	9431*	9432*
9453*	9454	9475*	9476	9545*	9546	9567*	9568	9594*	9595	9618*	9619	9642*
9643	9664*	9665	9686*	9687	9708*	9709	9778*	9779	9800*	9801	9827*	9828*
9851*	9852	9875*	9876	9897*	9898	9919*	9920	9941*	9942	10000*	10001	10019*
10020	10039*	10040	10059*	10060	10113*	10114	10133*	10134	10153*	10154	10173*	10174*
10227*	10228	10246*	10247	10266*	10267	10286*	10287	10340*	10341	10359*	10360	10379*
10380	10399*	10400	10455*	10456	10475*	10476	10496*	10497	10517*	10518	10572*	10573*
10592*	10593	10612*	10613	10632*	10633	10687*	10688	10706*	10707	10726*	10727	10746*
10747	10800*	10801	10820*	10821	10840*	10841	10860*	10861	10914*	10915	10933*	10934*
10953*	10954	10973*	10974	11027*	11028	11046*	11047	11066*	11067	11086*	11087	11140*
11141	11160*	11161	11180*	11181	11200*	11201	11256*	11257	11276*	11277	11297*	11298*
11318*	11319	11374*	11375	11393*	11394	11413*	11414	11433*	11434	11487*	11488	11506*
11507	11526*	11527	11546*	11547	11600*	11601	11620*	11621	11640*	11641	11660*	11661*
11714*	11715	11734*	11735	11754*	11755	11774*	11775	11828*	11829	11847*	11848	11867*
11868	11887*	11888	11943*	11944	11963*	11964	11984*	11985	12005*	12006	12066*	12067*
12086*	12087	12106*	12107	12126*	12127	12185*	12186	12204*	12205	12224*	12225	12244*
12245	12303*	12304	12323*	12324	12343*	12344	12363*	12364	12422*	12423	12442*	12443*
12462*	12463	12482*	12483	12541*	12542	12560*	12561	12580*	12581	12600*	12601	12661*
12662*	12681*	12682	12702*	12703	12723*	12724	12994*	13012*	13016*	13131*	13132	13183*
13184	13232*	13233*	13234	15791	15794	15796	15798	15800	15802	15806		
1496*	13692	13778	13862	13883								
13611*	13621*	13628	13637*	13642*								
14230												
\$BEL =	001216											
\$CHARC	044024											
\$CKSWR =	*****											
\$CNTAG	001100											
\$CM1 =	000006											
\$CM2 =	000014											
\$CM3 =	000006											
\$CM4 =	000006											
\$CNTLC	044527											
\$CNTLG	044541											
\$CNTLU	044534											
\$CRLF	001223											
\$DBLK	043600											
\$DOAGN	037550											
\$DTBL	043570											
\$ENDAD	037540											
\$ENDCT	037506											
\$ENDMG	037557											
\$ENULL	037554											

\$BEL = 001216
\$CHARC 044024
\$CKSWR = *****
\$CNTAG 001100
\$CM1 = 000006
\$CM2 = 000014
\$CM3 = 000006
\$CM4 = 000006
\$CNTLC 044527
\$CNTLG 044541
\$CNTLU 044534
\$CRLF 001223
\$DBLK 043600
\$DOAGN 037550
\$DTBL 043570
\$ENDAD 037540
\$ENDCT 037506
\$ENDMG 037557
\$ENULL 037554

\$EOP	037452	12799#												
\$EOPCT	037500	12806#	12810											
\$ERFLG	001103	1453#	13448	13479	13481	13487*	13508	13857*	13883					
\$ERMAX	001115	1459#	3117*	13481	13503*	13508								
\$ERROP	044730	3109	13856#											
\$ERRPC	001116	1460#	13864*	13865*	13866	13883	14064	15791	15792	15794	15796	15798	15800	15802
		15804	15806	15808	15810									
\$ERRTB	001226	1515#	14072											
\$ERRTY	045066	13869	13896#											
\$ERTTL	001112	1457#	12775	12785*	13863*	13883								
\$ESCAP	001214	1495#	3116*	13502*	13878	13880	13883							
\$FILLC	001156	1478#	13614	13645										
\$FILLS	001155	1477#	13645											
\$GDADR	001120	1461#												
\$GDADR*	001124	1463#	3713*	3714*	3719	3721*	3722*	3723	3739*	3740*	3745	3747*	3748*	3749
		3792*	3793*	3798	3800*	3801	3817*	3818*	3823	3825*	3826	3871*	3872*	3877*
		3879*	3880*	3881	3897*	3898*	3903	3905*	3906*	3907	3950*	3951*	3956	3958*
		3959	3975*	3976*	3981	3983*	3984	4029*	4030*	4035	4037*	4038*	4039	4055*
		4056*	4061	4063*	4064*	4065	4110*	4111*	4116	4118*	4119*	4120	4136*	4137*
		4142	4144*	4145*	4146	4189*	4190*	4193	4224*	4227	4242*	4243*	4246	4262*
		4265	4280*	4283	4297*	4300	4315*	4318	4333*	4336	4379*	4380*	4383	4414*
		4417	4431*	4432*	4435	4451*	4454	4469*	4472	4487*	4490	4505*	4508	4554*
		4555*	4558	4590*	4591*	4594	4625*	4628	4642*	4643*	4646	4665*	4668	4683*
		4684*	4687	4704*	4707	4723*	4726	4741*	4744	4759*	4762	4777*	4780	4822*
		4823*	4826	4862*	4863*	4866	4887*	4890	4909*	4912	4931*	4934	4953*	4956
		4975*	4978	4997*	5000	5019*	5022	5041*	5044	5063*	5064*	5067	5086*	5089
		5107*	5110	5127*	5130	5147*	5150	5167*	5170	5217*	5218*	5221	5258*	5259*
		5262	5284*	5287	5308*	5311	5332*	5335	5356*	5359	5380*	5383	5404*	5407
		5428*	5431	5452*	5455	5475*	5478	5479	5499*	5502	5521*	5524	5542*	5545
		5563*	5566	5584*	5587	5637*	5638*	5641	5680*	5681*	5684	5707*	5710	5731*
		5734	5755*	5758	5779*	5782	5803*	5806	5827*	5830	5851*	5854	5875*	5878
		5898*	5899*	5902	5922*	5925	5944*	5947	5965*	5968	5986*	5989	6007*	6010
		6059*	6060*	6063	6102*	6103*	6106	6128*	6131	6152*	6155	6176*	6179	6200*
		6203	6224*	6227	6248*	6251	6272*	6275	6296*	6299	6319*	6320*	6323	6343*
		6346	6365*	6368	6386*	6389	6407*	6410	6428*	6431	6480*	6481*	6484	6525*
		6526*	6529	6552*	6555	6577*	6580	6602*	6605	6627*	6630	6652*	6655	6677*
		6680	6702*	6705	6727*	6730	6751*	6752*	6755	6776*	6779	6799*	6802	6821*
		6824	6843*	6846	6865*	6868	6917*	6920	6939*	6942	6960*	6963	6985*	6988
		7016*	7019	7041*	7042*	7045	7068*	7071	7092*	7095	7116*	7119	7140*	7143
		7199*	7202	7220*	7223	7240*	7241*	7244	7262*	7265	7281*	7284	7300*	7303
		7319*	7322	7368*	7371	7388*	7391	7412*	7415	7435*	7438	7488*	7491	7508*
		7511	7532*	7535	7555*	7558	7608*	7611	7628*	7631	7652*	7655	7675*	7678
		7739*	7742	7758*	7759*	7762	7779*	7782	7840*	7843	7859*	7860*	7863	7880*
		7883	7898*	7901	7916*	7919	7934*	7937	7981*	7984	8002*	8005	8023*	8024*
		8027	8052*	8055	8075*	8076*	8079	8100*	8103	8122*	8125	8144*	8147	8166*
		8169	8217*	8220	8238*	8241	8259*	8260*	8263	8288*	8291	8311*	8312*	8315
		8336*	8339	8358*	8361	8380*	8383	8402*	8405	8455*	8458	8477*	8480	8498*
		8499*	8502	8522*	8525	8542*	8543*	8546	8564*	8567	8590*	8593	8613*	8614*
		8617	8638*	8641	8660*	8663	8682*	8685	8704*	8707	8757*	8760	8779*	8782
		8800*	8801*	8804	8824*	8827	8844*	8845*	8848	8866*	8869	8892*	8895	8915*
		8916*	8919	8940*	8943	8962*	8965	8984*	8987	9006*	9009	9076*	9077*	9080
		9099*	9102	9126*	9129	9149*	9150*	9153	9174*	9177	9196*	9199	9218*	9221
		9240*	9243	9309*	9310*	9313	9332*	9335	9359*	9362	9382*	9383*	9386	9407*
		9410	9429*	9432	9451*	9454	9473*	9476	9542*	9543*	9546	9565*	9568	9592*
		9595	9615*	9616*	9619	9640*	9643	9662*	9665	9684*	9687	9706*	9709	9725*
		9776*	9779	9798*	9801	9825*	9828	9848*	9849*	9852	9873*	9876	9895*	9898

M09

CERHAD0 MACY11 3001046, 21-DEC-77 13:06 PAGE 326

CERHAD.P11 21-DEC-77 12:48

CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0323

.STYPO 10 12480 14115
.S40CA 10
.1170 10

. ABS. 067012 000

ERRORS DETECTED: 0

CERHAD.BIN,CERHAD.LST/CRF SOL NL:TOC=CERHAD.SML,CERHAD.P11

RUN-TIME: 34 50 3 SECONDS

RUN-TIME RATIO: 343787=3.9

CORE USED: 37K 173 PAGES

N09

EOJICERHADSE0

00010000

780330

POP10 411