

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40

.REM &

IDENTIFICATION

PRODUCT CODE: AC-T710B-MC  
PRODUCT NAME: CZKDMB0 KDJ11 CACHE MEMORY DIAGNOSTIC  
PRODUCT DATE: 15-MAR-84  
MAINTAINER: DIAGNOSTIC ENGINEERING  
AUTHORS: HENRY ENMAN, JIM PITTMAN, BARRY IRRGANG

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1984 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

&

```

616 .SBTTL GLOBAL DATA SECTION
617
618
619 ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
620 ; IN MORE THAN ONE TEST.
621
622 .SBTTL APT MAILBOX-ETABLE
623
624 ; *****
625 .EVEN
626 001000 $MAIL: ; APT MAILBOX
627 001000 000000 $MSGTY: .WORD AMSTY ; MESSAGE TYPE CODE
628 001002 000000 $FATAL: .WORD AFATAL ; FATAL ERROR NUMBER
629 001004 000000 $TESTN: .WORD ATESTN ; TEST NUMBER
630 001006 000000 $PASS: .WORD APASS ; PASS COUNT
631 001010 000000 $DEVCT: .WORD ADEVCT ; DEVICE COUNT
632 001012 000000 $UNIT: .WORD AUNIT ; I/O UNIT NUMBER
633 001014 000000 $MSGAD: .WORD AMSGAD ; MESSAGE ADDRESS
634 001016 000000 $MSGLG: .WORD AMISGLG ; MESSAGE LENGTH
635 001020 $ETABLE: ; APT ENVIRONMENT TABLE
636 001020 000 $ENV: .BYTE AENV ; ENVIRONMENT BYTE
637 001021 000 $ENVM: .BYTE AENVM ; ENVIRONMENT MODE BITS
638 001022 002000 $SWREG: .WORD ASWREG ; APT SWITCH REGISTER
639 001024 000000 $USWR: .WORD AUSWR ; USER SWITCHES
640 001026 000000 $CPUOP: .WORD ACPUOP ; CPU TYPE, OPTIONS
641 ;* BITS 15-11-CPU TYPE
642 ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
643 ;* 11/70=06,PDQ=07,Q=10
644 ;* BIT 10-REAL TIME CLOCK
645 ;* BIT 9-FLOATING POINT PROCESSOR
646 ;* BIT 8-MEMORY MANAGEMENT
647 001030 $ETEND:
648 .MEXIT
649
650 ; THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA
651 ; WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY
652 ; FROM THE DEFAULT RESPONSE.
653 001030 000000 SLOC00: .WORD 0
654 001032 000000 SLOC01: .WORD 0
655
656 ; THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.
657 001034 000000 EXPDAT: .WORD 0 ; STORES EXPECTED (GOOD) DATA FOR COMPARISONS
658 001036 000000 RECDAT: .WORD 0 ; STORES RECEIVED DATA TO BE VERIFIED
659 001040 000000 COUNT: .WORD 0 ; ERROR INDICATOR FOR FLOATING POINT TESTS
660 001042 000000 FLAG: .WORD 0 ; USED TO STORE "FLAG" CONDITIONS
661 001044 000000 ERRCNT: .WORD 0 ; STORAGE FOR ERROR COUNT
662 001046 177570 SWR: .WORD DSWR ; STORAGE FOR SWITCH REGISTER ADDRESS
663 001050 177570 DISPLAY: .WORD DDISP ; STORAGE FOR DISPLAY REGISTER ADDRESS
664 001052 000000 $ERFLG: .WORD 0 ; ERROR FLAG
665 ; THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
666 001054 000000 DCOUNT: .WORD 0
667 001056 000000 ALLCTR: .WORD 0
668 001060 000000 LOOPIN: .WORD 0
669 001062 000000 SAVSP1: .WORD 0 ; STORAGE FOR UNEXPECTED TRAP DATA
670 001064 000000 SAVSP2: .WORD 0
671

```

672	001066	000000	FWDSEQ: .WORD	0	USED TO INDICATE DIRECTION OF ADDRESSING
673	001070	000000	ADDLSB: .WORD	0	STORES LEAST SIGNIFICANT BIT FOR RAM TESTS
674	001072	000000	RITEDA: .WORD	0	STORES WRITE DATA FOR RAM TESTS
675	001074	000000	NEWDAT: .WORD	0	DATA STORE FOR RAM TESTS
676	001076	000000	CURDAT: .WORD	0	DATA STORE FOR RAM TESTS
677	001100	000000	FSTADD: .WORD	0	STORES FIRST ADDRESS IN ADDRESSING SEQUENCE
678	001102	000000	LSTADD: .WORD	0	STORES LAST ADDRESS IN ADDRESSING SEQUENCE
679	001104	000000	CURADD: .WORD	0	STORES CURRENT ADDRESS FOR RAM TESTS
680	001106	000000	LOWADD: .WORD	0	STORES LOW ADDRESS FOR RAM TESTS
681	001110	000000	GOODAD: .WORD	0	STORES GOOD ADDRESS FOR RAM TESTS
682	001112	000000	TSTADD: .WORD	0	ADDRESS STORE FOR RAM TESTS
683	001114	000000	NEWADD: .WORD	0	ADDRESS STORE FOR RAM TEST
684	001116	000000	SOFTER: .WORD	0	USED TO STORE SOFT ERROR COUNT
685	001120	000000	SOFTRE: .WORD	0	USED TO HOLD RETRY ADDRESS ON SOFT ERRORS

686  
687  
688  
689  
690  
691  
692  
693

!!!!!!THIS IS IT. THE PROGRAM TEST LOCATION AND WRITE BUFFER!!!!!!!!!!!!!!!!!!!!!!  
TSTLOC: .BLKW 20

694  
695 001122  
696 001122 000020

```

697          .SBTTL  GLOBAL TEXT SECTION
698
699          ;++
700          ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
701          ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
702          ; MORE THAN ONE TEST.
703          ;--
704
705          ;
706          ; FORMAT STATEMENTS USED IN PRINT CALLS
707          ;
708
709 001162 005015 042523 020124 OPMSG2: .ASCIIZ <CR><LF>/SET BIT 8 = 1 FOR 18 BIT SYSTEM/
710 001170 044502 020124 020070
711 001176 020075 020061 047506
712 001204 020122 034061 041040
713 001212 052111 051440 051531
714 001220 042524 000115
715 001224 005015 042523 020124 OPMSG3: .ASCIIZ <CR><LF>/SET BIT 9 = 1 FOR CACHE RAM AND TAG RAM DATA RELIABILITY TESTS/
716 001232 044502 020124 020071
717 001240 020075 020061 047506
718 001246 020122 040503 044103
719 001254 020105 040522 020115
720 001262 047101 020104 040524
721 001270 020107 040522 020115
722 001276 040504 040524 051040
723 001304 046105 040511 044502
724 001312 044514 054524 052040
725 001320 051505 051524 000
726 001325 015 041412 041501 ERRMSG:: .ASCIIZ <CR><LF>/CACHE SYSTEM ERROR/
727 001332 047510 051440 051531
728 001340 042524 020115 051105
729 001346 047522 000122
730 001352 005015 051105 047522 ERR1: .ASCIIZ <CR><LF>/ERROR 0 -/
731 001360 020122 020043 000075
732 001366 005015 051105 047522 ERR2: .ASCIIZ <CR><LF>/ERROR PC -/
733 001374 020122 041520 036440
734 001402 000
735 001403 015 020012 020040 $CRLF: .ASCIIZ <CR><LF>/ /
736 001410 000
737          001412          .EVEN

```

THE FOLLOWING ASSIGNMENTS HAVE BEEN MADE FOR THE KDJ11-A  
DIAGNOSTIC SWITCH REGISTER BITS:

BIT#	15	14	13	12	11	10	9	8
						DON'T TEST BEVENT	18 BIT ADDRESS ONLY	EXTENDED CACHE TESTS

DEFAULT SETTINGS ARE TO TEST 22 BIT ADDRESSES AND NOT DO THE  
EXTENDED CACHE TESTS.

PRIOR TO EXECUTING THE FIRST PASS OF THE DIAGNOSTIC THE OPERATOR  
WILL BE DIRECTED TO SET THE SWITCH REGISTER TO INDICATE WHETHER  
THE KDJ11-A UNDER TEST IS IN A SYSTEM CONFIGURED FOR 18 OR 22 BIT  
ADDRESSING AND WHETHER CACHE RAM DATA RELIABILITY TESTS ARE TO BE  
EXECUTED. AN 18 BIT ADDRESS CONFIGURATION SHOULD BE INDICATED IF  
ANY 18 ADDRESS BIT ONLY MEMORY BOARDS RESIDE IN THE SYSTEM OR IF  
THE SYSTEM BACKPLANE DOES NOT SUPPORT 22 ADDRESS BITS. THE CACHE  
RAM DATA RELIABILITY TESTS REQUIRE APPROXIMATELY 35 MINUTES TO  
EXECUTE AND SHOULD ONLY BE SELECTED IF PROBLEMS WITH CACHE DATA  
CORRUPTION OR RETENTION ARE SUSPECTED.

TO CHANGE THE SWITCH REGISTER; HALT THE PROGRAM AND RESTART IT  
AT 200 ANSWERING THE INITIAL QUESTIONS.

### 2.3 OPERATION UNDER APT

OPERATION IN THE APT ENVIRONMENT REQUIRES SOME SPECIAL CONSIDERA-  
TIONS DUE TO THE ASYNCHRONOUS HALTS OF THE DIAGNOSTIC BY THE APT  
MONITOR. IF THE EFFECTS OF THESE HALTS ARE NOT ANTICIPATED, FALSE  
ERRORS MAY BE REPORTED. THEREFORE, WHEN OPERATING IN THE APT ENVIRON-  
MENT THE FOLLOWING DIFFERENCES IN THE EXECUTION OF THE PROGRAM SHOULD  
BE NOTED:

1. CERTAIN CACHE TESTS WHICH RELY ON THE INTEGRITY OF DATA IN THE  
HIT/MISS REGISTER OR THE WAY IN WHICH THE CACHE IS ALLOCATED WILL  
JUMP TO A SUB-ROUTINE IF AN ERROR IS ENCOUNTERED. THE SUBROUTINE  
DETERMINES IF THE DIAGNOSTIC IS OPERATING IN APT ENVIRONMENT. IF  
IT IS IN APT ENVIRONMENT IT WILL RETRY THE FAILING TEST ONE TIME.  
IF THE TEST PASSES ON THE SECOND ATTEMPT THE FIRST ERROR WILL BE  
CONSIDERED TO BE APT INDUCED. IF IT FAILS ON THE SECOND ATTEMPT  
THE ERROR ROUTINE WILL BE CALLED.

### 2.4 EXECUTION TIMES

#### A. THE LONGEST TEST

THE LONGEST TEST IS THE TAG RAM DATA RELIABILITY TEST WHICH TAKES  
APP. OXIMAEY 35 MINUTES TO EXECUTE. FOR THIS REASON THE DATA RELIABILITY  
TESTS ARE NORMALLY DE-SELECTED.

#### B. FULL PASS TIME

THE TIME FOR A FULL PASS WITH THE DATA RELIABILITY TESTS DE-SELECTED

196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215

IS APPROXIMATELY 5 SECONDS. WITH ALL TESTS SELECTED A FULL PASS  
TAKES ABOUT 35 MINUTES.

3.0 ERROR INFORMATION

ERRORS WILL CAUSE THE FOLLOWING ERROR MESSAGE TO BE PRINTED:

CACHE SYSTEM ERROR  
ERROR # = (UNIQUE ERROR NUMBER)  
ERROR PC = (PC AT TIME OF ERROR)

THE ERROR WILL THEN BE REPORTED TO APT AND THE PROGRAM  
WILL HALT.

4.0 PROGRESS REPORT

AT THE END OF EACH PASS THE DIAGNOSTIC NAME AND PASS COUNT ARE PRINTED.

```

792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812 001554 012702 000006
813 001560 005003
814 001562 010321
815 001564 062703 000200
816 001570 077204
817 001572 012721 002000
818 001576 012711 177600
819 001602 000207

```

```

***
; FUNCTIONAL DESCRIPTION:
;   SUBROUTINE TO INITIALIZE ALL THE MMU PAGE ADDRESS REGISTERS (PARS).
;   THIS ROUTINE WILL INITIALIZE 8 PARS STARTING AT A BASE ADDRESS
;   SUPPLIED BY THE CALLING ROUTINE. PARS 0-5 WILL BE MAPPED FROM
;   ADDRESS 0 TO ADDRESS 137777 (0-24K). PAR 6 WILL BE MAPPED FROM
;   ADDRESS 200000 TO 217777 AND PAR 7 WILL BE MAPPED TO THE I/O
;   PAGE.
;
; INPUTS:
;   R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PARS TO BE INITIALIZED
;
; OUTPUTS: NONE
;
; SUBORDINATE ROUTINES USED: NONE
;
; FUNCTIONAL SIDE EFFECTS: NONE
;
; CALLING SEQUENCE:      JSR      PC,LDPARS
LDPARS: MOV      #6,      R2          ;LET LOOP COUNTER COUNT FIRST 6 PARS
        CLR      R3          ;INITIALIZE INDEX VALUE
1$:     MOV      R3,      (R1)+    ;LOAD PARS
        ADD      #200,     R3     ;INDEX IN 4K INCREMENTS
        SOB     R2,      1$      ;LOAD FIRST SIX PARS
        MOV     #2000,    (R1)+   ;LET PAR6 MAP TO 200000
        MOV     #177600,(R1)    ;LET PAR7 MAP TO I/O PAGE
        RTS     PC            ;RETURN

```

GLOBAL AREAS MACY11 30A(1052) 04-APR-84 10:38 PAGE 21  
 KDJ11A.MAC 03-APR-84 11:36 GLOBAL SUBROUTINES SECTION

SEQ 0021

```

820      ;**
821      ; FUNCTIONAL DESCRIPTION:
822      ;   SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DESCRIPTOR REGISTERS (PDRS).
823      ;   THIS ROUTINE WILL INITIALIZE 8 PDRS STARTING AT A BASE ADDRESS
824      ;   SUPPLIED BY THE CALLING ROUTINE. PDRS 0-5 WILL BE INITIALIZED TO
825      ;   4K READ/WRITE BYPASS AND PDRS 6 AND 7 WILL BE INITIALIZED TO
826      ;   4K READ/WRITE NO BYPASS.
827      ;   NOTE:   THERE IS NO NEED TO BYPASS ON I/O PAGE REFERENCES BECAUSE
828      ;           THE CACHE DOES NOT ALLOCATE ANY OF THESE REFERENCES.
829
830      ; INPUTS:
831      ;   R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PDRS TO BE INITIALIZED
832
833      ; OUTPUTS: NONE
834
835      ; SUBORDINATE ROUTINES USED: NONE
836
837      ; FUNCTIONAL SIDE EFFECTS: NONE
838
839      ; CALLING SEQUENCE:   JSR   PC,LDPARS
840
841      LDPDRS: MOV   #6,   R2           ;LET LOOP COUNTER COUNT FIRST 6 PARS
842      1$:   MOV   #177406,(R1)+      ;LOAD PDR5 WITH 4K READ/WRITE BYPASS
843           SOB   R2,   1$          ;LOAD FIRST SIX PDRS
844           MOV   #77406,(R1)+      ;LET PAR6 BE 4K READ/WRITE NO BYPASS
845           MOV   #77406,(R1)      ;LET PAR7 BE 4K READ/WRITE NO BYPASS ALSO
846           RTS   PC              ;RETURN

```



```

847      ;**
848      ; FUNCTIONAL DESCRIPTION:
849      ;   SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.
850
851      ; INPUTS:
852      ;   MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE
853
854      ; OUTPUTS: NONE
855
856      ; SUBORDINATE ROUTINES USED: NONE
857
858      ; FUNCTIONAL SIDE EFFECTS: NONE
859
860      ; CALLING SEQUENCE: CALLED BY PARITY ABORT
861      ;   MOV     @0114, SLOC00 ;SAVE CONTENTS OF PARITY ABORT VECTOR
862      ;   MOV     @DSPAR, @0114 ;LET VECTOR POINT TO PARITY ABORT ROUTINE
863      ;
864      ;   (CACHE PARITY ERROR OCCURS)
865
866      RAMPAR:
867      ERROR      ;ALL ERRORS TO TRAP TO FMT VECTOR
868      .WORD      1 ;UNIQUE ERROR NUMBER
869      .WORD      ERRMSG ;ADDRESS OF ERROR MESSAGE
870
871      RTI      ;RETURN ;CACHE SYSTEM ERROR

```

```

367      000030      EMTVEC= 30          ;;EMULATOR TRAP (EMT) **ERROR**
368      000034      TRAPVEC=34        ;; "TRAP" TRAP
369      000060      TKVEC= 60          ;; TTY KEYBOARD VECTOR
370      000064      TPVEC= 64          ;; TTY PRINTER VECTOR
371      000240      PIRQVEC=240       ;;PROGRAM INTERRUPT REQUEST VECTOR
372
373      .SBTTL. MEMORY MANAGEMENT DEFINITIONS
374
375      ;*KT11 VECTOR ADDRESS
376      000250      MMVEC= 250
377
378      ;*KT11 STATUS REGISTER ADDRESSES
379
380      177572      SR0= 177572
381      177574      SR1= 177574
382      177576      SR2= 177576
383      172516      SR3= 172516
384
385      ;*USER "I" PAGE DESCRIPTOR REGISTERS
386
387      177600      UIPDR0= 177600
388      177602      UIPDR1= 177602
389      177604      UIPDR2= 177604
390      177606      UIPDR3= 177606
391      177610      UIPDR4= 177610
392      177612      UIPDR5= 177612
393      177614      UIPDR6= 177614
394      177616      UIPDR7= 177616
395
396      ;*USER "D" PAGE DESCRIPTOR REGISTERS
397
398      177620      UDPDR0= 177620
399      177622      UDPDR1= 177622
400      177624      UDPDR2= 177624
401      177626      UDPDR3= 177626
402      177630      UDPDR4= 177630
403      177632      UDPDR5= 177632
404      177634      UDPDR6= 177634
405      177636      UDPDR7= 177636
406
407      ;*USER "I" PAGE ADDRESS REGISTERS
408
409      177640      UIPAR0= 177640
410      177642      UIPAR1= 177642
411      177644      UIPAR2= 177644
412      177646      UIPAR3= 177646
413      177650      UIPAR4= 177650
414      177652      UIPAR5= 177652
415      177654      UIPAR6= 177654
416      177656      UIPAR7= 177656
417
418      ;*USER "D" PAGE ADDRESS REGISTERS
419
420      177660      UDPAR0= 177660
421      177662      UDPAR1= 177662
422      177664      UDPAR2= 177664

```

423	177666	UDPAR3= 177666
424	177670	UDPAR4= 177670
425	177672	UDPAR5= 177672
426	177674	UDPAR6= 177674
427	177676	UDPAR7= 177676
428		
429		; *SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
430		
431	172200	SIPDR0= 172200
432	172202	SIPDR1= 172202
433	172204	SIPDR2= 172204
434	172206	SIPDR3= 172206
435	172210	SIPDR4= 172210
436	172212	SIPDR5= 172212
437	172214	SIPDR6= 172214
438	172216	SIPDR7= 172216
439		
440		; *SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
441		
442	172220	SDPDR0= 172220
443	172222	SDPDR1= 172222
444	172224	SDPDR2= 172224
445	172226	SDPDR3= 172226
446	172230	SDPDR4= 172230
447	172232	SDPDR5= 172232
448	172234	SDPDR6= 172234
449	172236	SDPDR7= 172236
450		
451		; *SUPERVISOR "I" PAGE ADDRESS REGISTERS
452		
453	172240	SIPAR0= 172240
454	172242	SIPAR1= 172242
455	172244	SIPAR2= 172244
456	172246	SIPAR3= 172246
457	172250	SIPAR4= 172250
458	172252	SIPAR5= 172252
459	172254	SIPAR6= 172254
460	172256	SIPAR7= 172256
461		
462		; *SUPERVISOR "D" PAGE ADDRESS REGISTERS
463		
464	172260	SDPAR0= 172260
465	172262	SDPAR1= 172262
466	172264	SDPAR2= 172264
467	172266	SDPAR3= 172266
468	172270	SDPAR4= 172270
469	172272	SDPAR5= 172272
470	172274	SDPAR6= 172274
471	172276	SDPAR7= 172276
472		
473		; *KERNEL "I" PAGE DESCRIPTOR REGISTERS
474		
475	172300	KIPDR0= 172300
476	172302	KIPDR1= 172302
477	172304	KIPDR2= 172304
478	172306	KIPDR3= 172306

```

479      172310      KIPDR4= 172310
480      172312      KIPDR5= 172312
481      172314      KIPDR6= 172314
482      172316      KIPDR7= 172316
483
484      ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
485
486      172320      KDPDR0= 172320
487      172322      KDPDR1= 172322
488      172324      KDPDR2= 172324
489      172326      KDPDR3= 172326
490      172330      KDPDR4= 172330
491      172332      KDPDR5= 172332
492      172334      KDPDR6= 172334
493      172336      KDPDR7= 172336
494
495      ;*KERNEL "I" PAGE ADDRESS REGISTERS
496
497      172340      KIPAR0= 172340
498      172342      KIPAR1= 172342
499      172344      KIPAR2= 172344
500      172346      KIPAR3= 172346
501      172350      KIPAR4= 172350
502      172352      KIPAR5= 172352
503      172354      KIPAR6= 172354
504      172356      KIPAR7= 172356
505
506      ;*KERNEL "D" PAGE ADDRESS REGISTERS
507
508      172360      KDPAR0= 172360
509      172362      KDPAR1= 172362
510      172364      KDPAR2= 172364
511      172366      KDPAR3= 172366
512      172370      KDPAR4= 172370
513      172372      KDPAR5= 172372
514      172374      KDPAR6= 172374
515      172376      KDPAR7= 172376
516
517      ;THESE ARE FLOATING POINT ACCUMULATOR EQUATES
518      000000      AC0= *0
519      000001      AC1= *1
520      000002      AC2= *2
521      000003      AC3= *3
522      000004      AC4= *4
523      000005      AC5= *5
524      000006      AC6= *6
525      000007      AC7= *7
526
527      000244      FPVEC= 244
528
529      ;THESE ARE CACHE REGISTER EQUATES
530      177746      CCR= 177746 ;CACHE CONTROL REGISTER
531      177744      MSER= 177744 ;MEMORY SYSTEM ERROR REGISTER
532      177752      HITMIS= 177752 ;HIT/MISS REGISTER
533      177766      CPREG= 177766 ;CPU ERROR REGISTER
534

```

```

535 ;MISCELLANEOUS DEFINITIONS
536 177546 BEVENT= 177546 ;BEVENT CONTROL REGISTER
537 177560 RCSR= 177560
538 177562 RBUF= 177562
539 177564 XCSR= 177564
540 177566 XBUF= 177566
541 000000 ERRIN= HALT
542 000001 $TSTNU=1
543 000001 ERRNUM= 1 ;INITIALIZE ERROR NUMBER COUNTER
544 002000 ASWREG= 2000 ;SWR FOR APT--NO BEVENT TESTING
545
546
547 ;THIS EQUATE DEFINES THE BOTTOM OF THE PROGRAM STACK POINTER
548 001000 STBOT= 1000
549 000000 .ASECT
550 .SBTTL TRAP CATCHER
551
552 000000 .*=0
553 ;*ALL UNUSED LOCATIONS OF THE VECTOR AREA CONTAIN
554 ;*A ".+2, IOT" SEQUENCE TO CATCH AND PROCESS ILLEGAL
555 ;*TRAPS AND INTERRUPTS THAT MIGHT OCCUR.
556 ;*THE IOT TRAP WHICH IS TAKEN ON THE ILLEGAL TRAP/INT
557 ;*TRAPS TO THE $SCOPE ROUTINE WHICH (IF THE RETURN PC IS
558 ;*LESS THAN 1002) JUMPS TO THE $ERROR ROUTINE.
559 ;*THE $ERROR ROUTINE WILL REPORT THE ERROR AS FOLLOWS:
560 ;* PC=YYYYYY UNEXPECTED TRAP TO XXX
561 ;*AND RETURN TO THE PROGRAM AT PC=YYYYYY+2
562 ;*WHERE XXX=LOCATION OF ILLEGAL TRAP
563 ;* YYYYYY=PC AT TIME OF TRAP
564 ;*NOTE: IF THE PROCESSOR IS NOT AN 11/05 THE PROGRAM
565 ;* CAN BE STARTED AT ADDRESS 0 AS WELL AS ADDRESS 200.
566
567 000000 000000 $40CAT: HALT ;:HALT
568 000002 000737 BR .-100 ;:BRANCH TO 177700 & TIME OUT (NOT ON
569 ;:11/05)
570 000004 001716 .WORD START ;:VECTOR TO STARTING ADDRESS.
571 000006 000340 .WORD 340 ;:WITH PRIORITY LEVEL 7
572 .*=174
573 000174 000000 DISPRG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
574 000176 000000 SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER
575 .SBTTL STARTING ADDRESS(ES)
576 000200 000137 001716 JMP @START ;:GO TO START OF PROGRAM
577 .SBTTL ACT11 HOOKS
578
579 ;*****
580 ;HOOKS REQUIRED BY ACT11
581 000204 $SVPC . ;:SAVE PC
582 000046 .-46 ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
583 000046 011034 $ENDAD ;:2)SET LOC.52 TO ZERO
584 000052 000052 .-52 ;:RESTORE PC
585 000052 000000 .WORD 0
586 000204 .-$SVPC
587 .SBTTL APT PARAMETER BLOCK
588
589 ;*****
590 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
    
```

```

591
592      000204      ;*****
593      000024      .,$X=,    ;SAVE CURRENT LOCATION
594 000024 000200    .-24     ;SET POWER FAIL TO POINT TO START OF PROGRAM
595      000044      200      ;FOR APT START UP
596 000044 000204    .-44     ;POINT TO APT INDIRECT ADDRESS PNTR,
597      000204      $APTHDR ;POINT TO APT HEADER BLOCK
598      000204      .-,$X    ;RESET LOCATION COUNTER
599      ;*****
600      ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
601      ;INTERFACE SPEC.
602 000204 $APTHD:
603 000204 000000 $HIBTS: .WORD 0      ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
604 000206 001000 $MBADR: .WORD $MAIL  ;ADDRESS OF APT MAILBOX (BITS 0-15)
605 000210 000001 $TSTM:  .WORD 1      ;RUN TIM OF LONGEST TEST
606 000212 000004 $PASTM: .WORD 4      ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
607 000214 000000 $UNITM: .WORD 0      ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
608 000216 000014 .WORD $ETEND-$MAIL/2 ;LENGTH MAILBOX-ETABLE(WORDS)
609      000204      .-,$X    ;SAVE CURRENT LOCATION COUNT
610      000002      .-2
611 000002 000000 0
612 000004 000006 6
613 000006 000004 4      ;SET UP SOME VECTORS
614      000204      .-,$X    ;RESTORE LOCATION COUNT
615      001000      .-1000

```

```

616 .SBTTL GLOBAL DATA SECTION
617
618
619
620
621
622
623
624
625
626 001000 $MAIL:
627 001000 000000 $MSGTY: .WORD AMSGTY
628 001002 000000 $FATAL: .WORD AFATAL
629 001004 000000 $TESTN: .WORD ATESTN
630 001006 000000 $PASS: .WORD APASS
631 001010 000000 $DEVCT: .WORD ADEVCT
632 001012 000000 $UNIT: .WORD AUNIT
633 001014 000000 $MSGAD: .WORD AMSGAD
634 001016 000000 $MSGLG: .WORD AMSGLG
635 001020 $ETABLE:
636 001020 000 $ENV: .BYTE AENV
637 001021 000 $ENVM: .BYTE AENVM
638 001022 002000 $SWREG: .WORD ASWREG
639 001024 000000 $USWR: .WORD AUSWR
640 001026 000000 $CPUOP: .WORD ACPUOP
641
642
643
644
645
646
647 001030 $ETEND:
648 .MEXIT
649
650
651
652
653 001030 000000 SLOC00: .WORD 0
654 001032 000000 SLOC01: .WORD 0
655
656
657 001034 000000 EXPDAT: .WORD 0
658 001036 000000 RECDAT: .WORD 0
659 001040 000000 COUNT: .WORD 0
660 001042 000000 FLAG: .WORD 0
661 001044 000000 ERRCNT: .WORD 0
662 001046 177570 SWR: .WORD DSWR
663 001050 177570 DISPLAY: .WORD DDISP
664 001052 000000 $ERFLG: .WORD 0
665
666 001054 000000 DCOUNT: .WORD 0
667 001056 000000 ALLCTR: .WORD 0
668 001060 000000 LOOPIN: .WORD 0
669 001062 000000 SAVSP1: .WORD 0
670 001064 000000 SAVSP2: .WORD 0
671

```

; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED  
 ; IN MORE THAN ONE TEST.  
 ;  
 ; SBTTL APT MAILBOX-ETABLE  
 ;  
 ; \*\*\*\*\*  
 ;  
 ; APT MAILBOX  
 ; MESSAGE TYPE CODE  
 ; FATAL ERROR NUMBER  
 ; TEST NUMBER  
 ; PASS COUNT  
 ; DEVICE COUNT  
 ; I/O UNIT NUMBER  
 ; MESSAGE ADDRESS  
 ; MESSAGE LENGTH  
 ; APT ENVIRONMENT TABLE  
 ; ENVIRONMENT BYTE  
 ; ENVIRONMENT MODE BITS  
 ; APT SWITCH REGISTER  
 ; USER SWITCHES  
 ; CPU TYPE, OPTIONS  
 ; BITS 15-11-CPU TYPE  
 ; 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05  
 ; 11/70=06,POQ=07,Q=10  
 ; BIT 10-REAL TIME CLOCK  
 ; BIT 9-FLOATING POINT PROCESSOR  
 ; BIT 8-MEMORY MANAGEMENT  
 ;  
 ; THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA  
 ; WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY  
 ; FROM THE DEFAULT RESPONSE.  
 ;  
 ; THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA,  
 ; STORES EXPECTED (GOOD) DATA FOR COMPARISONS  
 ; STORES RECEIVED DATA TO BE VERIFIED  
 ; ERROR INDICATOR FOR FLOATING POINT TESTS  
 ; USED TO STORE "FLAG" CONDITIONS  
 ; STORAGE FOR ERROR COUNT  
 ; STORAGE FOR SWITCH REGISTER ADDRESS  
 ; STORAGE FOR DISPLAY REGISTER ADDRESS  
 ; ERROR FLAG  
 ; THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS  
 ; STORAGE FOR UNEXPECTED TRAP DATA  
 ; " " " " " "

D2

672	001066	000000	FWDSEQ: .WORD	0	USED TO INDICATE DIRECTION OF ADDRESSING
673	001070	000000	ADDLSB: .WORD	0	STORES LEAST SIGNIFICANT BIT FOR RAM TESTS
674	001072	000000	RITEDA: .WORD	0	STORES WRITE DATA FOR RAM TESTS
675	001074	000000	NEWDAT: .WORD	0	DATA STORE FOR RAM TESTS
676	001076	000000	CURDAT: .WORD	0	DATA STORE FOR RAM TESTS
677	001100	000000	FSTADD: .WORD	0	STORES FIRST ADDRESS IN ADDRESSING SEQUENCE
678	001102	000000	LSTADD: .WORD	0	STORES LAST ADDRESS IN ADDRESSING SEQUENCE
679	001104	000000	CURADD: .WORD	0	STORES CURRENT ADDRESS FOR RAM TESTS
680	001106	000000	LOWADD: .WORD	0	STORES LOW ADDRESS FOR RAM TESTS
681	001110	000000	GOODAD: .WORD	0	STORES GOOD ADDRESS FOR RAM TESTS
682	001112	000000	TSTADD: .WORD	0	ADDRESS STORE FOR RAM TESTS
683	001114	000000	NEWADD: .WORD	0	ADDRESS STORE FOR RAM TEST
684	001116	000000	SOFTERR: .WORD	0	USED TO STORE SOFT ERROR COUNT
685	001120	000000	SOFTRE: .WORD	0	USED TO HOLD RETRY ADDRESS ON SOFT ERRORS

686  
687  
688  
689  
690  
691  
692  
693

!!!!!!THIS IS IT. THE PROGRAM TEST LOCATION AND WRITE BUFFER!!!!!!!!!!!!!!!!!!!!!!

694			TSTLOC:		
695	001122				
696	001122	000020	.BLKW	20	



```

697          .SBTTL  GLOBAL TEXT SECTION
698
699          ;
700          ; THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
701          ; MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
702          ; MORE THAN ONE TEST.
703          ;
704          ;
705          ;
706          ; FORMAT STATEMENTS USED IN PRINT CALLS
707          ;
708
709 001162 005015 042523 020124 OPMSG2: .ASCIIZ <CR><LF>/SET BIT 8 = 1 FOR 18 BIT SYSTEM/
710 001170 044502 020124 020070
711 001176 020075 020061 047506
712 001204 020122 034061 041040
713 001212 052111 051440 051531
714 001220 042524 000115
715 001224 005015 042523 020124 OPMSG3: .ASCIIZ <CR><LF>/SET BIT 9 = 1 FOR CACHE RAM AND TAG RAM DATA RELIABILITY TESTS/
716 001232 044502 020124 020071
717 001240 020075 020061 047506
718 001246 020122 040503 044103
719 001254 020105 040522 020115
720 001262 047101 020104 040524
721 001270 020107 040522 020115
722 001276 040504 040524 051040
723 001304 046105 040511 044502
724 001312 044514 054524 052040
725 001320 051505 051524 000
726 001325 015 041412 041501 ERRMSG: .ASCIIZ <CR><LF>/CACHE SYSTEM ERROR/
727 001332 047510 051440 051531
728 001340 042524 020115 051105
729 001346 047522 000122
730 001352 005015 051105 047522 ERR1: .ASCIIZ <CR><LF>/ERROR # #/
731 001360 020122 020043 000075
732 001366 005015 051105 047522 ERR2: .ASCIIZ <CR><LF>/ERROR PC #/
733 001374 020122 041520 036440
734 001402 000
735 001403 015 020012 020040 $CRLF: .ASCIIZ <CR><LF>/ /
736 001410 000
737          001412          .EVEN

```

GLOBAL AREAS MACY11 30A(1052) 04-APR-84 10:38 PAGE 18  
KDJ11A.MAC 03-APR-84 11:36

GLOBAL ERROR REPORT SECTION

SEQ 0018

738  
739  
740  
741  
742  
743  
744

.SBTTL GLOBAL ERROR REPORT SECTION

; \*\*  
; THE GLOBAL ERROR REPORT SECTION CONTAINS MESSAGE PRINTING AREAS  
; USED BY MORE THAN TEST TO OUTPUT ADDITIONAL ERROR INFORMATION.  
; \*\*

GLOBAL AREAS  
KDJ11A.MACMAC111 30A(1052)  
03-APR-84 11:3604-APR-84 10:38 PAGE 19  
GLOBAL SUBROUTINES SECTION

SEQ 0019

```

745          .SBYTL  GLOBAL SUBROUTINES SECTION
746
747          ;**
748          ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
749          ; THAT ARE USED IN MORE THAN ONE TEST.
750          ;--
751          ;**
752          ; FUNCTIONAL DESCRIPTION:
753          ;   SUBROUTINE TO INITIALIZE ALL THE MMU REGISTERS
754
755
756          ; INPUTS: NONE
757
758          ; OUTPUTS: NONE
759
760          ; SUBORDINATE ROUTINES USED: LOAD PARS
761          ;                               LOAD PDRS
762
763          ; FUNCTIONAL SIDE EFFECTS: NONE
764
765          ; CALLING SEQUENCE:      JSR      PC,INITMM
766
767          INITMM: MOV      #172240,R1          ;BASE ADDRESS OF SIPARS
768                  JSR      PC,      LDPARS
769                  MOV      #172260,R1          ;BASE ADDRESS OF SDPARS
770                  JSR      PC,      LDPARS
771                  MOV      #172340,R1          ;BASE ADDRESS OF KIPARS
772                  JSR      PC,      LDPARS
773                  MOV      #172360,R1          ;BASE ADDRESS OF KDPARS
774                  JSR      PC,      LDPARS
775                  MOV      #177640,R1          ;BASE ADDRESS OF UIPARS
776                  JSR      PC,      LDPARS
777                  MOV      #177660,R1          ;BASE ADDRESS OF UOPARS
778                  JSR      PC,      LDPARS
779                  MOV      #177600,R1          ;BASE ADDRESS OF UIPDRS
780                  JSR      PC,      LDPDRS
781                  MOV      #177620,R1          ;BASE ADDRESS OF UOPDRS
782                  JSR      PC,      LDPDRS
783                  MOV      #172300,R1          ;BASE ADDRESS OF KIPDRS
784                  JSR      PC,      LDPDRS
785                  MOV      #172320,R1          ;BASE ADDRESS OF KDPDRS
786                  JSR      PC,      LDPDRS
787                  MOV      #172200,R1          ;BASE ADDRESS OF SIPDRS
788                  JSR      PC,      LDPDRS
789                  MOV      #172220,R1          ;BASE ADDRESS OF SDPDRS
790                  JSR      PC,      LDPDRS
791                  RTS      PC          ;RETURN

```

```

792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812 001554 012702 000006
813 001560 005003
814 001562 010321
815 001564 062703 000200
816 001570 077204
817 001572 012721 002000
818 001576 012711 177600
819 001602 000207

```

```

***
; FUNCTIONAL DESCRIPTION:
;   SUBROUTINE TO INITIALIZE ALL THE MMU PAGE ADDRESS REGISTERS (PARS),
;   THIS ROUTINE WILL INITIALIZE 8 PARS STARTING AT A BASE ADDRESS
;   SUPPLIED BY THE CALLING ROUTINE. PARS 0-5 WILL BE MAPPED FROM
;   ADDRESS 0 TO ADDRESS 137777 (0-24K), PAR 6 WILL BE MAPPED FROM
;   ADDRESS 200000 TO 217777 AND PAR 7 WILL BE MAPPED TO THE I/O
;   PAGE.
;
; INPUTS:
;   R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PARS TO BE INITIALIZED
;
; OUTPUTS: NONE
;
; SUBORDINATE ROUTINES USED: NONE
;
; FUNCTIONAL SIDE EFFECTS: NONE
;
; CALLING SEQUENCE:      JSR      PC,LDPARS
LDPARS: MOV      #6,      R2          ;LET LOOP COUNTER COUNT FIRST 6 PARS
          CLR      R3           ;INITIALIZE INDEX VALUE
1$:      MOV      R5,      (R1)+    ;LOAD PARS
          ADD      #200,    R3      ;INDEX IN 4K INCREMENTS
          SOB     R2,      1$      ;LOAD FIRST SIX PARS
          MOV      #2000,   (R1)+  ;LET PAR6 MAP TO 200000
          MOV      #177600,(R1)   ;LET PAR7 MAP TO I/O PAGE
          RTS      PC           ;RETURN

```

GLOBAL AREAS MACY11 30A(1052) 04-APR-84 10:38 PAGE 21  
 KDJ11A.MAC 03-APR-84 11:36 GLOBAL SUBROUTINES SECTION

SEQ 0021

```

820
821      ;++
822      ; FUNCTIONAL DESCRIPTION:
823      ;   SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DESCRIPTOR REGISTERS (PDRS).
824      ;   THIS ROUTINE WILL INITIALIZE 8 PDRS STARTING AT A BASE ADDRESS
825      ;   SUPPLIED BY THE CALLING ROUTINE. PDRS 0-5 WILL BE INITIALIZED TO
826      ;   4K READ/WRITE BYPASS AND PDRS 6 AND 7 WILL BE INITIALIZED TO
827      ;   4K READ/WRITE NO BYPASS.
828      ;   NOTE:   THERE IS NO NEED TO BYPASS ON I/O PAGE REFERENCES BECAUSE
829      ;           THE CACHE DOES NOT ALLOCATE ANY OF THESE REFERENCES.
830
831      ; INPUTS:
832      ;   R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PDRS TO BE INITIALIZED
833
834      ; OUTPUTS: NONE
835
836      ; SUBORDINATE ROUTINES USED: NONE
837
838      ; FUNCTIONAL SIDE EFFECTS: NONE
839
840      ; CALLING SEQUENCE:      JSR      PC,LDPARS
841      LDPDRS: MOV      #6,      R2           ;LET LOOP COUNTER COUNT FIRST 6 PARS
842      1$:   MOV      #177406,(R1)+       ;LOAD PDR5 WITH 4K READ/WRITE BYPASS
843      SOB      R2,      1$           ;LOAD FIRST SIX PDRS
844      MOV      #77406,(R1)+          ;LET PAR6 BE 4K READ/WRITE NO BYPASS
845      MOV      #77406,(R1)          ;LET PAR7 BE 4K READ/WRITE NO BYPASS ALSO
846      RTS      PC                   ;RETURN

```

```

847      ;**
848      ; FUNCTIONAL DESCRIPTION:
849      ;   SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.
850
851      ; INPUTS:
852      ;   MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE
853
854      ; OUTPUTS: NONE
855
856      ; SUBORDINATE ROUTINES USED: NONE
857
858      ; FUNCTIONAL SIDE EFFECTS: NONE
859
860      ; CALLING SEQUENCE: CALLED BY PARITY ABORT
861      ;   MOV     @0114, SLOC00 ;SAVE CONTENTS OF PARITY ABORT VECTOR
862      ;   MOV     @DSPAR, @0114 ;LET VECTOR POINT TO PARITY ABORT ROUTINE
863      ;
864      ;   (CACHE PARITY ERROR OCCURS)
865
866      RAMPAR;
867      ERROR      ;ALL ERRORS TO TRAP TO EMT VECTOR
868      .WORD     1 ;UNIQUE ERROR NUMBER
869      .WORD     ERRMSG ;ADDRESS OF ERROR MESSAGE
870
871      RTI        ;RETURN ;CACHE SYSTEM ERROR

```



```

909 001716          START:
910 001716 012737 000014 177746  MOV    #14,#CCR          ;SET CACHE TO FORCE MISS
911          .SBTTL INITIALIZE THE COMMON TAGS
912 001724 012706 001000  MOV    #STACK,SP        ;SETUP THE STACK POINTER
913          ;;INITIALIZE A FEW VECTORS
914 001730 012737 013314 000030  MOV    #ERROR,#EMTVEC   ;EMT VECTOR FOR ERROR ROUTINE
915 001736 012737 000340 000032  MOV    #340,#EMTVEC+2   ;LEVEL 7
916 001744 012737 012766 000034  MOV    #TRAP,#TRAPVEC  ;TRAP VECTOR FOR TRAP CALLS
917 001752 012737 000340 000036  MOV    #340,#TRAPVEC+2 ;LEVEL 7
918 001760 005067 177022          CLR    $PASS           ;CLEAR THE PASS COUNT
919 001764 016767 007012 007002  MOV    $ENDCT,$EOPCT   ;SETUP END-OF-PROGRAM COUNTER
920 001772 105067 177054          CLR    $ERFLG         ;CLEAR THE ERROR FLAG
921          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
922          ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
923 001776 013746 000004          MOV    #ERRVEC,-(SP)   ;SAVE ERROR VECTOR
924 002002 012737 002036 000004  MOV    #64#,#ERRVEC   ;SET UP ERROR VECTOR
925 002010 012767 177570 177030  MOV    #DSWP,#SWR      ;SETUP FOR A HARDWARE SWICH REGISTER
926 002016 012767 177570 177024  MOV    #DDISP,#DISPLAY ;AND A HARDWARE DISPLAY REGISTER
927 002024 022777 177777 177014  CMP    #-1,#SWR        ;TRY TO REFERENCE HARDWARE SWR
928 002032 001012          BNE    66$           ;BRANCH IF NO TIMEOUT TRAP OCCURRED
929          ;AND THE HARDWARE SWR IS NOT = -1
930 002034 000403          BR    65$           ;BRANCH IF NO TIMEOUT
931 002036 012716 002044          64$: MOV    #65#,(SP)    ;SET UP FOR TRAP RETURN
932 002042 000002          RTI
933 002044 012767 000176 176774  65$: MOV    #SWREG,#SWR   ;POINT TO SOFTWARE SWR
934 002052 012767 000174 176770  MOV    #DISPREG,#DISPLAY
935 002060 012637 000004          66$: MOV    (SP)+,#ERRVEC ;RESTORE ERROR VECTOR
936
937          .MACRO  $$SETMAIL    ?$ARG1
938          CLR    $PASS           ;CLEAR PASS COUNT
939          BITB   #APTSIZE,$ENVM   ;TEST USER SIZE UNDER APT
940          BEQ    $ARG1           ;YES,USE NON-APT SWITCH
941          MOV    #SWREG,#SWR     ;NO,USE APT SWITCH REGISTER
942          $ARG1:
943          .ENDM
944 002064 005067 176716          CLR    $PASS           ;CLEAR PASS COUNT
945 002070 132767 000200 176723  BITB   #APTSIZE,$ENVM   ;TEST USER SIZE UNDER APT
946 002076 001403          BEQ    67$           ;YES,USE NON-APT SWITCH
947 002100 012767 001022 176740  MOV    #SWREG,#SWR     ;NO,USE APT SWITCH REGISTER
948 002106          67$:
949 002106 012737 013314 000020  MOV    #ERROR,#IOTVEC   ;SET UP IOT VECTORS
950 002114 012737 000340 000022  MOV    #340,#IOTVEC+2   ;TO GO TO ERROR ROUTINE
951 002122 005037 177766          CLR    #177766        ;CLEAR CPU ERROR REGISTER
952 002126 104401 001162          TYPE   ,OPMSG2       ;OPERATOR MESSAGE 2
953 002132 104401 001224          TYPE   ,OPMSG3       ;OPERATOR MESSAGE 3
954          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
955 002136 005737 000042          TST    #04$           ;ARE WE RUNNING UNDER XXDP/ACT?
956 002142 001012          BNE    68$           ;BRANCH IF YES
957 002144 126727 176650 000001  CMPB   $ENV,#1         ;ARE WE RUNNING UNDER APT?
958 002152 001406          BEQ    68$           ;BRANCH IF YES
959 002154 026727 176666 000176  CMP    SWR,#SWREG      ;SOFTWARE SWITCH REG SELECTED?
960 002152 001005          BNE    69$           ;BRANCH IF NO
961 002164 104406          GETSWR                ;GET SOFT-SWR SETTINGS
962 002166 000403          BR    69$
963 002170 112767 000001 010566  68$: MOV    #1,$AUTOB      ;SET AUTO-MODE INDICATOR
964 002176          69$:

```



```

965 002176 005067 176602          RESTART: CLR      $TESTN          ;RESET $TESTN TO ZERO
966 002202 012737 000014 177746    MOV      #14,$CCR          ;SET CACHE TO FORCE MISS
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 002210
997 002210 005267 176570          TST1:  INC      $TESTN          ;INCREMENT TEST NUMBER
998 002214 013767 000114 176606    MOV      @0114,$D000       ;SAVE PARITY INTERRUPT VECTORS
999 002222 012737 001630 000114    MOV      @RAMPAR,@0114     ;SET UP TRAP CATCHER FOR PARITY ERRORS.
1000 002230 013704 000004          MOV      @04,R4            ;SAVE CONTENTS OF VECTOR 4
1001 002234 012737 002302 000004    MOV      @ERRROUT,@04      ;LET VECTOR 4 POINT TO ERROR ROUTINE
1002 002242 005037 177746          CLR      @CCR              ;TEST ACCESS TO CACHE CONTROL REG
1003 002246 005037 177744          CLR      @MSER             ;TEST ACCESS TO MEMORY SYSTEM ERR REG
1004 002252 005037 177752          CLR      @HITMIS           ;TEST ACCESS TO HIT MISS REGISTER
1005 002256 010437 000004          MOV      R4,@04            ;RESTORE VECTOR
1006 002262 005037 177744          CLR      @MSER             ;CLEAR THE MEM SYS ERROR REG
1007 002266 005037 177766          CLR      @CPCPEREG         ;CLEAR THE CPU ERROR REGISTER
1008 002272 005037 001116          CLR      @SOFTER           ;CLEAR THE SOFT ERROR COUNT
1009 002276 000167 000010          JMP      TST2              ;JUMP OVER DATA TABLES OR SUBROUTINES
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020

```

```

;*****
; *TEST 1      CACHE REGISTER ACCESS TEST
;*****
;CHECK REGISTER ACCESS - THIS TEST WILL VERIFY EACH OF THE THREE
;CACHE MEMORY SYSTEM REGISTERS CAN BE ACCESSED WITHOUT A TRAP TO
;LOCATION 4(NON-EXISTANT ADDRESS TRAP) OCCURRING THE REGISTERS TO
;BE TESTED ARE:
;   CACHE CONTROL      17777746
;   MEMORY SYSTEM ERROR 17777744
;   HIT/MISS           17777752
;EACH REGISTER WILL BE ACCESSED WITH A WRITE CYCLE.
;
;BGNST
;SAVE CONTENT OF LOCATION 4
;LET LOCATION 4 POINT TO ERROR ROUTINE
;TEST THE CACHE REGISTERS
;RESTORE CONTENTS OF LOCATION 4
;EXIT  TST
;
;CACHE REGISTER ADDRESSES
;   CCR= 17777746
;   MSER= 17777744
;   HITMIS= 17777752
;ERROR ROUTINE: ERROR IN CACHE SYSTEM
;          RETURN
;ENDTST
;*****

```

```

TST1:
INC      $TESTN          ;INCREMENT TEST NUMBER
MOV      @0114,$D000       ;SAVE PARITY INTERRUPT VECTORS
MOV      @RAMPAR,@0114     ;SET UP TRAP CATCHER FOR PARITY ERRORS.
MOV      @04,R4            ;SAVE CONTENTS OF VECTOR 4
MOV      @ERRROUT,@04      ;LET VECTOR 4 POINT TO ERROR ROUTINE
CLR      @CCR              ;TEST ACCESS TO CACHE CONTROL REG
CLR      @MSER             ;TEST ACCESS TO MEMORY SYSTEM ERR REG
CLR      @HITMIS           ;TEST ACCESS TO HIT MISS REGISTER
MOV      R4,@04            ;RESTORE VECTOR
CLR      @MSER             ;CLEAR THE MEM SYS ERROR REG
CLR      @CPCPEREG         ;CLEAR THE CPU ERROR REGISTER
CLR      @SOFTER           ;CLEAR THE SOFT ERROR COUNT
JMP      TST2              ;JUMP OVER DATA TABLES OR SUBROUTINES
;TO NEXT TEST

```

```

ERRROUT:
ERROR
.WORD 2
.WORD ERRMSG
;ALL ERRORS TO TRAP TO EMT VECTOR
;UNIQUE ERROR NUMBER
;ADDRESS OF ERROR MESSAGE
;CACHE SYSTEM ERROR
RTI
;CONTINUE FROM ERROR

```

1021  
1022 002312  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049 002312  
1050 002312 005267 176466  
1051 002316 012701 000001  
1052 002322 005067 175420  
1053 002326 010167 175414  
1054 002332 026701 175410  
1055 002336 001415  
1056 002340 005767 175402  
1057 002344 001007  
1058 002346 032701 174400  
1059 002352 001007  
1060 002354 104000  
1061 002356 000003  
1062 002360 001325  
1063  
1064 002362 000403  
1065 002364  
1066 002364 104000  
1067 002366 000004  
1068 002370 001325  
1069  
1070 002372 022737 002000 177746 4\$  
1071 002400 001003  
1072 002402 012737 000400 177746  
1073  
1074 002410 004101  
1075 002412 103345  
1076

```

CTST2:
;*****
;TEST 2      TEST CACHE CNTRL REG BITS
;*****
;CCR REGISTER BIT TEST - THIS TEST WILL VERIFY THAT EACH READ/WRITE BIT OF
;THE CACHE CONTROL REGISTER CAN BE SET AND CLEARED INDIVIDUALLY AND THAT
;BITS 15 - 11 AND BIT 8 ARE ALWAYS READ AS ZEROS.
;
;BGNTST
;INITIALIZE GOOD DATA TO #1
;CLEAR CCR
;DO UNTIL ALL BITS TESTED
;.      WRITE GOOD DATA TO CCR
;.      READ CCR
;.      IF CCR NOT EQUAL TO GOOD DATA THEN
;.          IF CCR EQUAL TO ZERO THEN
;.              IF GOOD DATA NOT EQUAL TO BITS<15-11> OR BIT 8 THEN
;.                  ERROR IN CACHE SYSTEM
;.              ENDIF
;.          ELSE
;.              ERROR IN CACHE SYSTEM
;.          ENDIF
;.      ENDIF
;.      UPDATE TO NEXT BIT
;ENDDO
;ENDTST
;*****
TST2:
      INC      $TESTN      ; INCREMENT TEST NUMBER
      MOV      #1,      R1  ; INITIALIZE GOOD DATA TO #1
      CLR      CCR        ; CLEAR CCR
1$:   MOV      R1,      CCR  ; WRITE GOOD DATA TO CCR
      CMP      CCR,      R1  ; IF CCR NOT EQUAL GOOD DATA
      BFE      4$        ; THEN
      TST      CCR        ; IF CCR EQUAL TO ZERO
      BNE      3$        ; THEN
      BIT      #174400,R1  ; IF GOOD DATA NOT BITS<15-11> OR 8
      BNE      4$        ; THEN
      ERROR    3        ; ALL ERRORS TO TRAP TO EMT VECTOR
      .WORD    ERRMSG    ; UNIQUE ERROR NUMBER
      .WORD    ERRMSG    ; ADDRESS OF ERROR MESSAGE
                          ; CACHE SYSTEM ERROR
      BR      4$        ; ENDIF
3$:   ERROR    4        ; ALL ERRORS TO TRAP TO EMT VECTOR
      .WORD    ERRMSG    ; UNIQUE ERROR NUMBER
      .WORD    ERRMSG    ; ADDRESS OF ERROR MESSAGE
                          ; CACHE SYSTEM ERROR
4$:   CMP      #2000,#CCR  ; WAS LAST FUNCTION WRITE WRONG TAG PART
      BNE      5$        ; IF NOT GO SHIFT LEFT AND DO NEXT TEST.
      MOV      #400,#CCR ; FLUSH CACHE TO GET RID OF WRONG
                          ; TAG PARITY
5$:   ROL      R1        ; UPDATE TO NEXT BIT
      ROR      1$        ; DO UNTIL ALL BITS TESTED

```

1077  
1078 002414  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115

```

CTST3:
;*****
;TEST 3      TEST FORCE MISS FUNCTION
;*****
;FORCE MISS TEST - THIS TEST WILL VERIFY THAT ALL REFERENCES MADE
;WITH EITHER BIT<3> OR BIT<2> OF THE CCR SET CAUSE A CACHE MISS AND
;LEAVE THE CACHE ENTRY UNCHANGED. FIRST WRITE A TEST ADDRESS WITH
;BITS<3;2> CLEARED TO ALLOCATE CACHE AND SET A KNOWN DATA PATTERN
;INTO THE CACHE, THEN SET BIT<2> AND REWRITE THE SAME ADDRESS WITH
;NEW DATA. NEXT CLEAR BITS<3;2> AND READ THE TEST ADDRESS, THE DATA
;SHOULD EQUAL TO PATTERN 1. FINALLY READ THE TEST ADDRESS WITH BIT<3>
;SET, THE DATA SHOULD EQUAL PATTERN 2. A LAST WRITE MUST BE DONE WITH
;BOTH FORCE BITS CLEARED BECAUSE THE CACHE AND MAIN MEMORY HAVE
;DIFFERENT DATA.
;
;
;BGNTST
;WRITE TEST ADDRESS WITH PATTERN 1
;SET CCR BITS<3;2> = 0,1
;WRITE TEST ADDRESS WITH PATTERN 2
;CLEAR CCR BIT<3>
;READ TEST ADDRESS
;SAVE HIT/MISS REGISTER DATA
;COMPARE RECEIVED DATA TO PATTERN 1
;IF DATA NOT EQUAL THEN
;.    ERROR IN CACHE SYSTEM
;ENDIF
;SET CCR<3;2> = 1,0
;READ TEST ADDRESS
;SAVE HIT/MISS REGISTER DATA
;COMPARE RECEIVED DATA TO PATTERN 2
;IF DATA NOT EQUAL THEN
;.    ERROR IN CACHE SYSTEM
;ENDIF
;CLEAR CCR<3;2>
;WRITE TEST ADDRESS
;ENDTST
;
;*****

```

1116  
1117 002414  
1118 002414 005267 176364  
1119 002420 012737 002550 000114  
1120 002426 005037 001122  
1121  
1122 002432 012767 000004 175306  
1123 002440 012737 177777 001122  
1124  
1125 002446 012767 000200 175272  
1126  
1127 002454 013701 001122  
1128 002460 005701  
1129 002462 001403  
1130 002464 104000  
1131 002466 000005  
1132 002470 001325

```

TST3:      INC      $TESTN      ; INCREMENT TEST NUMBER
           MOV      @MPARR,@114  ; SETUP PARITY VECTOR TO POINT TO HANDLER
           CLR      @TSTLOC      ; WRITE TEST LOCATION WITH PATTERN 1
           ; ALLOCATES CACHE AND MEMORY WITH ZEROS
           MOV      @BIT02,CCR    ; SET CCR BITS<3;2> = 0,1(FORCE CACHE MISS)
           MOV      @177777,@TSTLOC ; WRITE TEST LOCATION WITH PATTERN 2
           ; WRITES MEMORY WITH ONES, CACHE STILL ZEROS
           MOV      @200,CCR      ; CLEAR CCR BIT 3 (CLEAR FORCE MISS)
           ; AND SET PARITY ABORTS
           TST      R1            ; READ TEST ADDRESS (SHOULD READ ZEROS IN CACHE)
           BEQ      11           ; COMPARE RECEIVED DATA TO PATTERN 1
           ; IF DATA NOT EQUAL THEN
           ; ALL ERRORS TO TRAP TO FMT VECTOR
           .WORD    5            ; UNIQUE ERROR NUMBER
           .WORD    ERRMSG      ; ADDRESS OF ERROR MESSAGE

```

```

1133
1134 002472 052767 000010 175246 1$: BIS @BIT03,CCR ;CACHE SYSTEM ERROR
1135 002500 013701 001122 MOV @TSTLOC,R1 ;SET CCR BITS <3:2> = 1,0(FORCE CACHE MISS)
1136 002504 020127 177777 CMP R1,@177777 ;READ TEST LOCATION (SHOULD READ ONES IN MEMORY)
1137 002510 001403 BEQ 2$ ;COMPARE RECEIVED DATA TO PATTERN 2
1138 002512 104000 ERROR ;IF DATAS NOT EQUAL THEN
1139 002514 000006 .WORD 6 ;ALL ERRORS TO TRAP TO EMT VECTOR
1140 002516 001325 .WORD ERRMSG ;UNIQUE ERROR NUMBER
1141 ;ADDRESS OF ERROR MESSAGE
1142 002520 005067 175222 2$: CLR CCR ;CACHE SYSTEM ERROR
1143 002524 005037 001122 CLR @TSTLOC ;CLEAR CCR BITS<3:2>
1144 002530 005037 177744 CLR @MSER ;WRITE TEST ADDRESS
1145 002534 005037 177766 CLR @CPEREG ;CLEAR THE MEM SYS ERROR REG
1146 002540 005037 001116 CLR @SOFTER ;CLEAR THE CPU ERROR REGISTER
1147 002544 000167 000006 JMP TST4 ;CLEAR THE SOFT ERROR COUNT
1148 ;JUMP OVER DATA TABLES OR SUBROUTINES
1149 ;TO NEXT TEST
1150 002550 FMPARR: ;PARITY ERROR ON FORCE MISS
1151 002550 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1152 002552 000007 .WORD 7 ;UNIQUE ERROR NUMBER
1153 002554 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1154
1155
1156 002556 CTST4:
1157 ;*****
1158 ;*TEST 4 TEST HIT MISS REGISTER
1159 ;*****
1160 ;HIT/MISS REGISTER TEST PART 1 - THIS TEST WILL VERIFY THAT THE HIT/MISS
1161 ;REGISTER CORRECTLY LOGS HITS AND MISSES.FIRST WRITE A TEST ADDRESS WITH
1162 ;BITS<3:2> CLEARED TO ALLOCATE CACHE AND SET A KNOWN DATA PATTERN INTO
1163 ;THE CACHE. THEN SET BIT<3> AND REWRITE THE SAME ADDRESS WITH NEW DATA.
1164 ;NEXT CLEAR CCR BITS<3:2> AND READ THE TEST ADDRESS. THE HIT MISS REGISTER
1165 ;SHOULD HAVE LOGGED A HIT. FINALLY READ THE TEST ADDRESS PLUS 20000(8) THE
1166 ;HIT/MISS REGISTER SHOULD HAVE LOGGED A MISS.
1167 ;
1168 ;
1169 ;BGNTST
1170 ;WRITE TEST ADDRESS WITH PATTERN 1
1171 ;SET CCR BITS<3:2> = 0,1
1172 ;WRITE TEST ADDRESS WITH PATTERN 2
1173 ;CLEAR CCR BIT<3>
1174 ;READ TEST ADDRESS
1175 ;IF HIT/MISS REGISTER BIT 3 NOT SET THEN
1176 ;. ERROR IN CACHE SYSTEM
1177 ;ENDIF
1178 ;READ TEST ADDRESS + 20000(8)
1179 ;IF HIT/MISS REGISTER BIT 3 SET THEN
1180 ;. ERROR IN CACHE SYSTEM
1181 ;ENDIF
1182 ;ENDTST
1183 ;
1184 ;*****
1185 002556 TST4:
1186 002556 005267 176222 INC $TESTN ;INCREMENT TEST NUMBER
1187 002562 012737 002702 000114 MOV @HMPARR,@114 ;SETUP PARITY VECTOR TO POINT TO HANDLER
1188 002570 005037 001122 CLR @TSTLOC ;WRITE TEST LOCATION WITH PATTERN 1

```

```

1189
1190 002574 012767 000004 175144      MOV      #BIT02, CCR      ;WRITE CACHE AND MEMORY WITH ZEROS
1191 002602 012737 177777 001122      MOV      #177777,#@TSTLOC ;SET CCR BITS<3:2> = 0,1(FORCE CACHE MISS)
1192                                     ;WRITE TEST LOCATION WITH PATTERN 2
1193 002610 012767 000200 175130      MOV      #200,  CCR      ;MEMORY =-1, CACHE STILL ZERO
1194                                     ;CLEAR CCR BIT 3 (CLEAR FORCE MISS)
1195 002616 013701 001122                MOV      @TSTLOC,R1      ;AND SET PARITY ABORTS
1196 002622 032737 000010 177752      BIT      @BIT03, @@HITMIS ;READ TEST ADDRESS (SHOULD BE CACHE HIT)
1197 002630 001003                       BNE      1$              ;IF HIT/MISS REGISTER BIT 3 NOT SET
1198 002632 104000                       ERROR     ;THEN
1199 002634 000010                       .WORD    10              ;ALL ERRORS TO TRAP TO EMT VECTOR
1200 002636 001325                       .WORD    ERRMSG         ;UNIQUE ERROR NUMBER
1201                                     ;ADDRESS OF ERROR MESSAGE
1202 002640 013701 021122 1$:          MOV      @TSTLOC+8192.,R1 ;CACHE SYSTEM ERROR
1203 002644 032737 000010 177752      BIT      @BIT03, @@HITMIS ;READ TEST LOCATION + 20000(8) SHOULD MISS
1204 002652 001403                       BEQ      2$              ;IF HIT/MISS REGISTER BIT 3 SET
1205 002654 104000                       ERROR     ;THEN
1206 002656 000011                       .WORD    11              ;ALL ERRORS TO TRAP TO EMT VECTOR
1207 002660 001325                       .WORD    ERRMSG         ;UNIQUE ERROR NUMBER
1208                                     ;ADDRESS OF ERROR MESSAGE
1209                                     ;CACHE SYSTEM ERROR
1210 002662 005037 177744 2$:          CLR      @@MSER          ;CLEAR THE MEM SYS ERROR REG
1211 002666 005037 177766                CLR      @@CPEREG        ;CLEAR THE CPU ERROR REGISTER
1212 002672 005037 001116                CLR      @@SCFTR         ;CLEAR THE SOFT ERROR COUNT
1213 002676 000167 000006                JMP      TST5            ;JUMP OVER DATA TABLES OR SUBROUTINES
1214                                     ;TO NEXT TEST
1215
1216 002702                                     HMPARR:
1217 002702 104000                       ERROR     ;PARITY ERROR DURING HIT/MISS REGISTER TEST
1218 002704 000012                       .WORD    12              ;ALL ERRORS TO TRAP TO EMT VECTOR
1219 002706 001325                       .WORD    ERRMSG         ;UNIQUE ERROR NUMBER
1220                                     ;ADDRESS OF ERROR MESSAGE
1221
1222
1223 002710                                     CTST5:
1224                                     ;*****
1225                                     ;*TEST 5      TEST HIT MISS REGISTER BITS
1226                                     ;*****
1227                                     ;HIT/MISS REGISTER TEST - THIS TEST WILL VERIFY THAT THE HIT/MISS
1228 ;REGISTER CORRECTLY LOGS CACHE HITS AND MISSES. IT WILL ALSO VERIFY
1229 ;THAT EACH BIT OF THE REGISTER IS UNIQUE. THIS WILL BE DONE BY
1230 ;FLOATING A ZERO THROUGH A FIELD OF ONES. THE ROTATING WILL BE DONE
1231 ;BY EXECUTING A TST @@HM (WHERE HM IS THE ADDRESS OF THE HIT/MISS
1232 ;REGISTER) AT SUCCESSIVE POSITIONS IN A SET OF FOUR NOPS. THE READ
1233 ;OF THE I/O PAGE ADDRESS OF THE HIT/MISS REGISTER SHOULD CAUSE A
1234 ;MISS TO BE RECORDED.
1235 ;
1236 ;BGNTST
1237 ;INITIALIZE LOOP INDICATOR
1238 ;INITIALIZE EXPECTED DATA
1239 ;DO UNTIL LOOP INDICATOR = SEVEN
1240 ;.
1241 ;.      PUT BACKGROUND DATA INTO EXECUTION BUFFER
1242 ;.      PUT TST INSTRUCTION INTO TEST LOCATION
1243 ;.      JUMP TO EXECUTE BUFFER
1244 ;.      IF EXPECTED DATA NE RECEIVED DATA THEN
1245 ;.          .      ERROR IN CACHE SYSTEM

```



1301 003070 000077  
 1302 003072 000037  
 1303 003074 000057  
 1304 003076 000067  
 1305 003100 000073  
 1306 003102 000075  
 1307 003104 000076

EXPTBL: .WORD 77  
 .WORD 37  
 .WORD 57  
 .WORD 67  
 .WORD 73  
 .WORD 75  
 .WORD 76

1308  
 1309  
 1310

1311  
 1312 003106

```

CTST6:
;*****
;TEST 6 TEST BYTE ALLOCATION OF CACHE
;*****
;BYTE ALLOCATION TEST- THIS TEST WILL VERIFY THAT THE CACHE SYSTEM CORRECTLY
;HANDLES BYTE ACCESSES. THE TEST WILL FIRST CHECK THAT A WRITE ACCESS WHICH
;IS A MISS DOES NOT UPDATE THE CACHE. THIS WILL BE DONE BY FIRST ASSURING
;A MISS AT A TEST LOCATION. THEN A WRITE BYTE ACCESS WILL BE DONE. FINALLY
;THE LOCATION WILL BE READ. THE WRITE BYTE SHOULD NOT HAVE ALLOCATED THE
;ADDRESS. NEXT THE TEST WILL VERIFY THAT IF A WRITE BYTE ACCESS IS A HIT
;THAT THE CACHE LOCATION IS UPDATED. THE TEST WILL FIRST WRITE A TEST LOCATION
;WITH A PATTERN TO ALLOCATE THE ADDRESS. THEN A SECOND PATTERN WILL BE WRITTEN
;TO THE HIGH BYTE OF THE TEST LOCATION. THE TEST LOCATION WILL THEN BE READ.
;IF THE HIGH BYTE OF THE TEST LOCATION IS EQUAL TO THE SECOND PATTERN THEN THE
;LOCATION WAS UPDATED. THE SECOND TEST WILL BE REPEATED TO TEST LOW BYTE
;ACCESSES.
;
;BGNTST
;SAVE VECTOR 114
;LET VECTOR 114 POINT TO BYTE PARITY ROUTINE
;SET PARITY ABORT BIT IN CACHE CONTROL REGISTER
;READ TEST LOCATION + 4K
;WRITE LOW BYTE TO TEST ADDRESS
;READ LOW BYTE OF TEST ADDRESS
;IF HIT/MISS REGISTER BIT 3 SET THEN
;. ERROR IN CACHE SYSTEM
;ENDIF
;WRITE PATTERN 1 TO TEST LOCATION
;WRITE BYTE PATTERN 2 TO TEST LOCATION+1
;IF BIT1 NOTSET IN HIT/MISS REGISTER THEN
;. ERROR IN CACHE SYSTEM
;ELSE
;. READ TEST LOCATION
;. IF RECEIVED DATA NOT EQUAL TO EXPECTED DATA THEN
;. ERROR IN CACHE SYSTEM
;ENDIF
;WRITE PATTERN 1 TO TEST LOCATION
;WRITE BYTE PATTERN 2 TO TEST LOCATION LOW BYTE
;IF BIT1 NOTSET IN HIT/MISS REGISTER THEN
;. ERROR IN CACHE SYSTEM
;ELSE
;. READ TEST LOCATION
;. IF RECEIVED DATA NOT EQUAL TO EXPECTED DATA THEN
;. ERROR IN CACHE SYSTEM
;ENDIF
    
```

1313  
 1314  
 1315  
 1316  
 1317  
 1318  
 1319  
 1320  
 1321  
 1322  
 1323  
 1324  
 1325  
 1326  
 1327  
 1328  
 1329  
 1330  
 1331  
 1332  
 1333  
 1334  
 1335  
 1336  
 1337  
 1338  
 1339  
 1340  
 1341  
 1342  
 1343  
 1344  
 1345  
 1346  
 1347  
 1348  
 1349  
 1350  
 1351  
 1352  
 1353  
 1354  
 1355  
 1356

```

1357 ;CLEAR CACHE CONTROL REGISTER
1358 ;RESTORE VECTOR 114
1359 ;EXIT TST
1360 ;
1361 ;BYTE PARITY ROUTINE: SAVE MSER
1362 ; RETURN
1363 ;ENDTST
1364 ;*****
1365 TST6:
1366 003106 005267 175672 INC $TESTN ;INCREMENT TEST NUMBER
1367 003112 012737 003374 000114 MOV $BYPARR,$0114 ;LET VECTOR 114 POINT TO ABORT ROUTINE
1368 003120 012767 000200 17462C MOV $BIT07,CCR ;SET PARITY ABORT BIT IN CCR
1369 003126 005767 015770 TST TSTLOC+8192. ;READ TEST LOCATION + 4K TO CAUSE MISS
1370 ;ON NEXT ACCESS OF TSTLOC
1371 003132 105067 175764 CLRB TSTLOC ;WRITE LOW BYTE OF TEST LOCATION
1372 003136 105767 175760 TSTB TSTLOC ;READ LOW BYTE OF TEST LOCATION
1373 003142 032737 000010 177752 BIT $BIT03,$0HITMIS ;IF BIT3 OF HIT/MISS REGISTER NOT SET
1374 003150 001410 BEQ 1$ ;THEN BRANCH
1375 003152 012737 003112 001120 MOV $TST6+4,$0SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1376 003160 004767 176454 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1377 003164 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1378 003166 000014 .WORD 14 ;UNIQUE ERROR NUMBER
1379 003170 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1380 ;CACHE SYSTEM ERROR
1381 003172 005067 175724 1$: CLR TSTLOC ;WRITE PATTERN 1 TO TEST LOCATION
1382 003176 152737 000377 001123 BISB $377,$0TSTLOC+1 ;WRITE PATTERN 2 TO HIGH BYTE
1383 003204 032767 000010 174540 BIT $BIT03,HITMIS ;IF BIT3 NOTSET IN HIT/MISS REGISTER
1384 003212 001011 BNE 2$ ;THEN
1385 003214 012737 003112 001120 MOV $TST6+4,$0SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1386 003222 004767 176412 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1387 003226 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1388 003230 000015 .WORD 15 ;UNIQUE ERROR NUMBER
1389 003232 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1390 ;CACHE SYSTEM ERROR
1391 003234 000414 BR 3$ ;ELSE
1392 003236 022737 177400 001122 2$: CMP $177400,$0TSTLOC ;IF TEST LOCATION NOT EQUAL TO
1393 003244 001410 BEQ 3$ ;PATTERN 2 THEN
1394 003246 012737 003112 001120 MOV $TST6+4,$0SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1395 003254 004767 176360 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1396 003260 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1397 003262 000016 .WORD 16 ;UNIQUE ERROR NUMBER
1398 003264 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1399 ;CACHE SYSTEM ERROR
1400 003266 005067 175630 3$: CLR TSTLOC ;WRITE PATTERN 1 TO TEST LOCATION
1401 003272 152737 000377 001122 BISB $377,$0TSTLOC ;WRITE PATTERN 2 TO LOW BYTE
1402 003300 032767 000010 174444 BIT $BIT03,HITMIS ;IF BIT03 SET IN HIT/MISS REGISTER
1403 003306 001011 BNE 4$ ;THEN BRANCH; WE GOT A HIT.
1404 ;ELSE
1405 003310 012737 003112 001120 MOV $TST6+4,$0SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1406 003316 004767 176316 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1407 003322 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1408 003324 000017 .WORD 17 ;UNIQUE ERROR NUMBER
1409 003326 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1410 ;CACHE SYSTEM ERROR
1411 003330 000407 BR 5$
1412 003332 022737 000377 001122 4$: CMP $377,$0TSTLOC ;IF TEST LOCATION NOT EQUAL TO
    
```



```

1413 003340 001403          BEQ      5$          ;PATTERN 2 THEN
1414 003342 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
1415 003344 000020          .WORD    20          ;UNIQUE ERROR NUMBER
1416 003346 001325          .WORD    ERRMSG      ;ADDRESS OF ERROR MESSAGE
1417                                     ;CACHE SYSTEM ERROR
1418 003350 005067 174372    5$;   CLR      CCR          ;CLEAR CCR BEFORE EXIT
1419 003354 005037 177744    CLR      @@MSER       ;CLEAR THE MEM SYS ERROR REG
1420 003360 005037 177766    CLR      @@CPEREG     ;CLEAR THE CPU ERROR REGISTER
1421 003364 005037 001116    CLR      @@SOFTER     ;CLEAR THE SOFT ERROR COUNT
1422 003370 000167 000006    JMP      TST7         ;JUMP OVER DATA TABLES OR SUBROUTINES
1423                                     ;TO NEXT TEST
1424
1425 003374          BYPARR;          ;PARITY ERROR DURING BYTE ALLOCATION TEST
1426 003374 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
1427 003376 000021          .WORD    21          ;UNIQUE ERROR NUMBER
1428 003400 001325          .WORD    ERRMSG      ;ADDRESS OF ERROR MESSAGE
1429
1430
1431
1432 003402          CTST7;
1433          ;*****
1434          ;*TEST 7      TEST CACHE BYPASS (MMU)
1435          ;*****
1436          ;PDR BIT15 (BYPASS) TEST - THIS TEST WILL VERIFY THAT WHEN BIT<15> IS SET
1437          ;IN A PDR AND AN ACCESS IS MADE TO A LOCATION MAPPED BY THE SELECTED PDR
1438          ;THAT WOULD NORMALLY CAUSE A CACHE ACCESS, THE CACHE IS BYPASSED (I.E. THE
1439          ;CACHE LOCATION IS INVALIDATED AND A MAIN MEMORY IS READ. THIS WILL BE DONE
1440          ;BY FIRST WRITING A TEST LOCATION WITH A KNOWN PATTERN TO ALLOCATE THE ADDRESS.
1441          ;THEN THE LOCATION WILL BE WRITTEN AGAIN WITH THE MMU ENABLED AND BIT <15> SET
1442          ;IN THE CONTROLLING PDR WITH A NEW PATTERN. THE LOCATION WILL THEN BE READ
1443          ;AND A MISS SHOULD BE LOGGED IN THE HIT/MISS REGISTER. THIS READ WILL ALSO
1444          ;BRING VALID DATA INTO CACHE FROM MEMORY. THE MMU WILL AGAIN BE ENABLED
1445          ;WITH BIT <15> SET AND A READ CYCLE DONE. THIS SHOULD INVALIDATE THE CACHE.
1446          ;NEXT THE MMU WILL BE DISABLED AND THE LOCATION READ FOR A THIRD TIME. THIS
1447          ;WILL BE DONE WITH BIT<15> STILL SET. A MISS SHOULD ALSO BE LOGGED. LASTLY
1448          ;BIT<15> WILL BE CLEARED AND THE MMU ENABLED AND THE LOCATION AGAIN READ.
1449          ;THIS TIME A CACHE HIT SHOULD BE LOGGED.
1450          ;
1451          ;
1452          ;BGNST
1453          ;SETUP MMU REGISTERS
1454          ;WRITE TEST LOCATION WITH PATTERN 1
1455          ;SET BIT<15> IN PDR FOR TEST LOCATION
1456          ;ENABLE MMU
1457          ;WRITE TEST LOCATION WITH PATTERN 2
1458          ;DISABLE MMU AND CLEAR BIT<15> IN PDR
1459          ;READ TEST LOCATION
1460          ;IF HIT/MISS REGISTER BIT<1> SET THEN
1461          ;.      ERROR IN CACHE SYSTEM
1462          ;ENDIF
1463          ;SET BIT<15> IN PDR FOR TEST LOCATION
1464          ;ENABLE MMU
1465          ;READ TEST LOCATION (SHOULD INVALIDATE CACHE)
1466          ;DISABLE MMU
1467          ;READ TEST LOCATION
1468          ;IF HIT/MISS REGISTER BIT<1> SET THEN
    
```

```

1469          ;.      ERROR IN CACHE SYSTEM
1470          ;ENDIF
1471          ;CLEAR BIT<15> IN PDR
1472          ;TURN ON MMU
1473          ;READ TEST LOCATON
1474          ;IF HIT/MISS REGISTER BIT<1> NOT SET THEN
1475          ;.      ERROR IN CACHE SYSTEM
1476          ;ENDIF
1477          ;TURN OFF MMU
1478          ;ENDTST
1479          ;
1480          ;*****
1481          TST7:
1482          003402 005267 175376          INC      $TESTN          ;INCREMENT TEST NUMBER
1483          003406 012737 001630 000114  MOV     WRAMPAR,00114    ;SET UP FOR UNEXPECTED PARITY ERROR
1484          003414 004767 175772          JSR     PC,INITMM       ;SETUP MEMORY MANAGEMENT REGISTERS
1485          003420 005037 001122          CLR     00TSTLOC       ;WRITE TEST LOCATION WITH PATTERN 1
1486          003424 052767 100000 166646  BIS     0BIT15,KIPDRO   ;SET BIT15 IN PDR FOR TEST LOCATION
1487          003432 052737 000001 177572  BIS     0BIT00,00SRO    ;TURN ON MEMORY MANAGEMENT
1488          003440 012737 177777 001122  MOV     0177777,00TSTLOC ;WRITE TEST LOCATION WITH PATTERN 2
1489          003446 005067 174120          CLR     SRO             ;TURN OFF MEMORY MANAGEMENT
1490          003452 042767 100000 166620  BIC     0BIT15,KIPDRO   ;CLEAR BIT15 IN PDR FOR TEST LOCATION
1491          003460 013701 001122          MOV     00TSTLOC,R1    ;READ TEST LOCATION
1492          003464 032767 000010 174260  BIT     0BIT03,HITMIS   ;IF HIT/MISS REGISTER BIT 3 SET
1493          003472 001410          BEQ     2$             ;THEN
1494          003474 012737 003406 001120  MOV     0TST7.4,00SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1495          003502 004767 176132          JSR     PC,APTSFT      ;GO TO SOFT ERROR SUBROUTINE
1496          003506 104000          ERROR   ;ALL ERRORS TO TRAP TO EMT VECTOR
1497          003510 000022          .WORD  22             ;UNIQUE ERROR NUMBER
1498          003512 001325          .WORD  ERRMSG         ;ADDRESS OF ERROR MESSAGE
1499
1500          003514 052767 100000 166556 2$:  BIS     0BIT15,KIPDRO   ;SET BIT15 IN PDR FOR TEST LOCATION
1501          003522 052737 000001 177572  BIS     0BIT00,00SRO    ;TURN ON MEMORY MANAGEMENT
1502          003530 005767 175366          TST     TSTLOC         ;READ TEST LOCATION, BYPASS CACHE
1503          003534 042767 100000 166536  BIC     0BIT15,KIPDRO   ;TURN OFF CACHE BYPASS ON KERNEL PAGE 0.
1504          003542 005067 174024          CLR     SRO            ;TURN OFF MMU
1505          003546 005737 001122          TST     00TSTLOC       ;READ TEST LOCATION-SHOULD BE CACHE MISS
1506          003552 032767 000010 174172  BIT     0BIT03,HITMIS   ;IF HIT/MISS REGISTER BIT 3 SET
1507          003560 001410          BEQ     3$             ;THEN
1508          003562 012737 003406 001120  MOV     0TST7.4,00SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1509          003570 004767 176044          JSR     PC,APTSFT      ;GO TO SOFT ERROR SUBROUTINE
1510          003574 104000          ERROR   ;ALL ERRORS TO TRAP TO EMT VECTOR
1511          003576 000023          .WORD  23             ;UNIQUE ERROR NUMBER
1512          003600 001325          .WORD  ERRMSG         ;ADDRESS OF ERROR MESSAGE
1513
1514          003602 052737 000001 177572 3$:  BIS     0BIT00,00SRO    ;TURN ON MEMORY MANAGEMENT
1515          003610 005737 001122          TST     00TSTLOC       ;READ TEST LOCATION ONE MORE TIME.
1516
1517          ;PREVIOUS READ SHOULD HAVE ALLOCATED
1518          ;CACHE SO WE'RE EXPECTING A CACHE HIT
1519          003614 032767 000010 174130  BIT     0BIT03,HITMIS   ;IF HIT/MISS REGISTER BIT 3 NOT SET
1520          003622 001012          BNE     4$             ;THEN
1521          003624 005067 173742          CLR     SRO            ;TURN OFF MMU BEFORE ERROR ROUTINE.
1522          003630 012737 003406 001120  MOV     0TST7.4,00SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1523          003636 004767 175776          JSR     PC,APTSFT      ;GO TO SOFT ERROR SUBROUTINE
1524          003642 104000          ERROR   ;ALL ERRORS TO TRAP TO EMT VECTOR
1524          003644 000024          .WORD  24             ;UNIQUE ERROR NUMBER

```

```

1525 003646 001325          .WORD  ERRMSG          ;ADDRESS OF ERROR MESSAGE
1526                                     ;CACHE SYSTEM ERROR
1527 003650 042767 000001 173714 4$: BIC  0BIT00, SR0    ;TURN OFF MMU
1528 003656 005037 177744          CLR  00MSER        ;CLEAR THE MEM SYS ERROR REG
1529 003662 005037 177766          CLR  00CPEREG      ;CLEAR THE CPU ERROR REGISTER
1530 003666 005037 001116          CLR  00SOFTER     ;CLEAR THE SOFT ERROR COUNT
1531 003672 000167 000000          JMP   TST10       ;JUMP OVER DATA TABLES OR SUBROUTINES
1532                                     ;TO NEXT TEST
1533
1534

```

```

1535 003676          CTST10:
1536          ;:*****^*****
1537          ;*TEST 10      TEST CACHE FLUSH
1538          ;:*****^*****
1539          ;FLUSH CACHE TEST - THIS TEST WILL VERIFY THAT WHEN CCR BIT<8> IS
1540          ;SET, THE ENTIRE CACHE IS INVALIDATED. FIRST 8K BYTES OF ADDRESSES
1541          ;WILL BE READ TO ALLOCATE CACHE. THEN THE CACHE WILL BE FLUSHED.
1542          ;THE SAME SET OF ADDRESS WILL THEN BE READ AND THE HIT/MISS REGISTER
1543          ;CHECKED AFTER EACH READ FOR A MISS TO BE LOGGED.
1544          ;
1545          ;BGNTST
1546          ;GET FIRST ADDRESS OF 8KB BUFFER
1547          ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
1548          ;DO UNTIL LOOP COUNTER = 0
1549          ;.   READ 0ADDRESS+
1550          ;ENDDO
1551          ;GET FIRST ADDRESS OF 8KB BUFFER
1552          ;INITIALIZE LOOP COUNTER TO 4KWORD COUNT
1553          ;INITIALIZE MISS COUNT
1554          ;FLUSH CACHE
1555          ;DO UNTIL LOOP COUNTER = 0
1556          ;.   READ 0ADDRESS+
1557          ;.   IF HIT/MISS REGISTER BIT<1> SET THEN
1558          ;.   .   DECREMENT MISS COUNT
1559          ;.   ENDDG
1560          ;IF MISS COUNT NOT EQUAL TO 4096. THEN
1561          ;.   ERROR IN CACHE SYSTEM
1562          ;ENDIF
1563          ;ENDTST
1564          ;
1565          ;:*****^*****

```

```

1567 003676          TST10:
1568 003676 005267 175102          INC  $TESTN        ;INCREMENT TEST NUMBER
1569 003702 004767 175504          JSR  PC, INLMM     ;INITIALIZE MMU
1570 003706 005067 166442          CLR  KIPAR6       ;LET PAR6 POINT TO LOW 4K
1571 003712 012701 140000          MOV  0140000,R1   ;GET ADDRESS OF 8K BYTE BUFFER
1572 003716 012702 010000          MOV  04096.,R2    ;INIT LOOP COUNTER TO 4K WORD COUNT
1573 003722 052737 000C01 177572  BIS  0BIT00,00SR0 ;ENABLE MMU
1574 003730 005721          1$: TST  (R1),      ;READ ADDRESS TO ALLOCATE CACHE
1575 003732 077202          SOB  R2, 1$       ;DO UNTIL ALL LOCATIONS ALLOCATED
1576 003734 012701 140000          MOV  0140000,R1   ;GET ADDRESS OF 8K BYTE BUFFER
1577 003740 012702 010000          MOV  04096.,R2    ;INIT LOOP COUNTER TO 4K WORD COUNT
1578 003744 012703 010000          MOV  04096.,R3    ;PRE-LOAD MISS COUNT WITH 4096.
1579 003750 012767 000400 173770  MOV  0400, CCR     ;FLUSH CACHE
1580 003756 005721          2$: TST  (R1),      ;READ ADDRESS

```



```

1637 ;ENDIF
1638 ;ENDTST
1639 ;
1640 ;*****
1641 TST11:
1642 004052 005267 174726 INC $TESTN ;INCREMENT TEST NUMBER
1643 004056 005737 001122 TST @TSTLOC ;ALLOCATE FIRST TEST LOCATION
1644 004062 005737 001124 TST @TSTLOC+2 ;ALLOCATE SECOND TEST LOCATION
1645 004066 052767 001000 173652 BIS @BIT09,CCR ;SET CCR BIT 9 (CACHE BYPASS)
1646 004074 005737 001122 1$: TST @TSTLOC ;READ FIRST TEST LOCATION
1647 004100 032767 000010 173644 BIT @BIT03,HITMIS ;IF HIT/MISS REG. BIT 3 NOT SET
1648 004106 001010 BNE 2$ ;THEN
1649 004110 012737 004056 001120 MOV @TST11+4,@SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1650 004116 004767 175516 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1651 004122 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1652 004124 000026 .WORD 26 ;UNIQUE ERROR NUMBER
1653 004126 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1654 ; ;CACHE SYSTEM ERROR
1655 004130 005037 001124 2$: CLR @TSTLOC+2 ;WRITE SECOND LOCATION
1656 004134 013703 177752 100$: MOV @HITMIS,R3 ;SAVE CONTENTS OF HIT MISS
1657 004140 012704 004134 MOV @100$,R4 ;MOV ADDR OF MOV INST TO R4
1658 004144 012702 001124 MOV @TSTLOC+2,R2 ;
1659 004150 042704 177701 BIC @177701,R4 ;STRIP OFF UNNEEDED BITS
1660 004154 042702 177701 BIC @177701,R2 ;
1661 004160 020204 CMP R2,R4 ;ARE THEY EQUAL?
1662 004162 001004 BNE 101$ ;BRANCH IF NOT
1663 004164 032703 000010 BIT @BIT03,R3 ;IF HIT/MISS REG. BIT 3 NOT SET
1664 004170 001014 BNE 3$ ;THEN
1665 004172 000403 BR 102$ ;IT WAS A MISS, GO TO ERROR
1666 004174 032703 000004 101$: BIT @BIT02,R3 ;TEST BIT02
1667 004200 001010 BNE 3$ ;BRANCH IFF IT WAS A HIT
1668 004202 102$:
1669 004202 012737 004056 001120 MOV @TST11+4,@SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1670 004210 004767 175424 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1671 004214 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1672 004216 000027 .WORD 27 ;UNIQUE ERROR NUMBER
1673 004220 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1674 ; ;CACHE SYSTEM ERROR
1675 004222 005737 001122 3$: TST @TSTLOC ;READ FIRST TEST LOCATION
1676 004226 032767 000010 173516 BIT @BIT03,HITMIS ;IF HIT/MISS REG. BIT 3 SET
1677 004234 001410 BEQ 4$ ;THEN
1678 004236 012737 004056 001120 MOV @TST11+4,@SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1679 004244 004767 175370 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1680 004250 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1681 004252 000030 .WORD 30 ;UNIQUE ERROR NUMBER
1682 004254 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1683 ; ;CACHE SYSTEM ERROR
1684 004256 005037 001124 4$: CLR @TSTLOC+2 ;WRITE SECOND LOCATION
1685 004262 016703 173464 200$: MOV HITMIS,R3 ;SAVE CONTENTS OF HIT MISS REG
1686 004266 012704 004262 MOV @200$,R4 ;
1687 004272 012702 001124 MOV @TSTLOC+2,R2 ;
1688 004276 042704 177701 BIC @177701,R4 ;
1689 004302 042702 177701 BIC @177701,R2 ;
1690 004306 020204 CMP R2,R4 ;ARE THEY EQUAL
1691 004310 001004 BNE 201$ ;BRANCH IF NOT
1692 004312 032703 000010 BIT @BIT03,R3 ;IF HIT/MISS REG. BIT 3 NOT SET
    
```

```

1693 004316 001414          BEQ      5$          ; THEN EXIT TEST
1694 004320 000403          BR       202$         ; ELSE, EXPECTED A MISS GO TO ERROR
1695 004322 032703 000004   201$:  BIT      @BIT02,R3 ; IF HIT/MISS REG. BIT 2 NOT SET
1696 004326 001410          BEQ      5$          ; THEN EXIT TEST, ELSE GO TO ERROR
1697 004330
1698 004330 012737 004056 001120 202$:  MOV      @TST11+4,@#SOFTRE ; MOVE RETRY ADDRESS TO SOFT RETURN
1699 004336 004767 175276      JSR      PC,APTSFT   ; GO TO SOFT ERROR SUBROUTINE
1700 004342 104000          ERROR     ; ALL ERRORS TO TRAP TO EMT VECTOR
1701 004344 000031          .WORD    31         ; UNIQUE ERROR NUMBER
1702 004346 001325          .WORD    ERRMSG     ; ADDRESS OF ERROR MESSAGE
1703
1704 004350 005067 173372   5$:    CLR      CCR          ; CLEAR BYPASS BIT BEFORE EXIT
1705 004354 005037 177744      CLR      @#MSER     ; CLEAR THE MEM SY: ERROR REG
1706 004360 005037 177766      CLR      @#CPEREG   ; CLEAR THE CPU ERROR REGISTER
1707 004364 005037 001116      CLR      @#SOFTER   ; CLEAR THE SOFT ERROR COUNT
1708 004370 000167 000000      JMP      TST12      ; JUMP OVER DATA TABLES OR SUBROUTINES
1709
1710
1711 004374          CTST12:
1712          ;*****
1713          ;*TEST 12      TEST WRITE WRONG DATA PARITY
1714          ;*****
1715          ;WRITE WRONG DATA PARITY TEST- THIS TEST WILL VERIFY THAT WHEN CCR
1716          ;BIT<6> = 1 AND CCR BITS<7,0> = 0,1, A READ MISS OCCURS AFTER A
1717          ;WRITE. THE WRITE WITH CCR BIT<6> = 1 TO A LOCATION WILL CAUSE A
1718          ;CACHE UPDATE AND WRONG PARITY TO BE WRITTEN SO WHEN THE LOCATION
1719          ;IS READ INSTEAD OF A HIT BEING RECORDED THE CACHE PARITY ERROR
1720          ;WILL CAUSE A CACHE MISS. THIS TEST WILL BE DONE A BYTE AT A TIME.
1721          ;NOTE: IF BOTH ERRORS OCCUR CCGA MOST LIKELY BAD.
1722          ;      IF ONLY ONE ERROR OCCURS CCGA LEAST LIKELY.
1723          ;
1724          ;BGNTST
1725          ;SAVE CONTENTS OF VECTOR 114
1726          ;LET VECTOR 114 POINT TO ABORT ROUTINE
1727          ;CLEAR MSER
1728          ;IF MSER NOT CLEAR THEN
1729          ;.      ERROR IN CACHE SYSTEM
1730          ;ENDIF
1731          ;WRITE TEST LOCATION
1732          ;SET BITS<6,0> IN CCR
1733          ;WRITE TEST LOCATION LOW BYTE
1734          ;CLEAR CCR BIT<6> (WRITE WRONG DATA PARITY)
1735          ;INITIALIZE ERROR INDICATORS
1736          ;READ TEST LOCATION LOW BYTE
1737          ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
1738          ;.      ERROR IN CACHE SYSTEM
1739          ;ELSE
1740          ;.      IF MSER BITS <7:5> ZERO THEN
1741          ;.      ERROR IN CACHE SYSTEM
1742          ;.      ENDIF
1743          ;ENDIF
1744          ;CLEAR MSER
1745          ;IF MSER NOT CLEAR THEN
1746          ;.      ERROR IN CACHE SYSTEM
1747          ;ENDIF
1748          ;SET CCR BIT<6>

```

```

1749 ;WRITE TEST LOCATION HIGH BYTE
1750 ;CLEAR CCR BIT<6>
1751 ;READ TEST LOCATION HIGH BYTE
1752 ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
1753 ;. ERROR IN CACHE SYSTEM
1754 ;ELSE
1755 ;. IF MSER BITS <7:5> ZERO THEN
1756 ;. ERROR IN CACHE SYSTEM
1757 ;. ENDIF
1758 ;ENDIF
1759 ;RESTORE CONTENTS OF VECTOR 114
1760 ;IF ERROR INDICATORS SET THEN
1761 ;. ERROR IN CACHE SYSTEM
1762 ;ENDIF
1763 ;EXIT TST
1764 ;
1765 ;DATA PARITY ABORT ROUTINE:
1766 ;. ERROR IN CACHE SYSTEM
1767 ;. RETURN
1768 ;
1769 ;ENDTST
1770 ;*****
1771 TST12:
1772 004374 005267 174404 INC $TESTN ;INCREMENT TEST NUMBER
1773 004400 013767 000114 174422 MOV @0114, SLOC00 ;SAVE CONTENTS OF VECTOR 114
1774 004406 012737 004706 000114 MOV @DAPAB0,@0114 ;LET VECTOR POINT TO ABORT ROUTINE
1775 004414 005067 173324 CLR MSER ;CLEAR MSER
1776 004420 005767 173320 TST MSER ;IF MSER NOT CLEAR
1777 004424 001403 BEQ 1$ ;THEN
1778 004426 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1779 004430 000032 .WORD 32 ;UNIQUE ERROR NUMBER
1780 004432 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1781 ;. CACHE SYSTEM ERROR
1782 004434 005037 001122 1$: CLR @0TSTLOC ;WRITE TEST LOCATION TO ALLOCATE CACHE
1783 004440 012767 000101 173300 MOV @101, CCR ;SET BITS<6,0> IN CCR
1784 ;WRITE WRONG DATA PARITY; IGNORE PARITY INTERRUPT
1785 004446 112767 000377 174446 MOVB @377, TS' 0C ;WRITE LOW BYTE WITH BAD PARITY
1786 004454 042767 000100 173204 BIC @BIT06, CCR ;CLEAR WRITE WRONG DATA PARITY BIT
1787 004462 105737 001122 TSTB @0TSTLOC ;READ LOW BYTE OF TEST LOCATION
1788 004466 032767 000010 173256 BIT @BIT03, HIT/MS ;EXPECT A MISS; IF BIT 3 SET IN HIT/MISS REGISTER
1789 004474 001410 BEQ 2$ ;THEN
1790 004476 012737 004400 001120 MOV @TST12+4,@0SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1791 004504 004767 175130 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1792 004510 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1793 004512 000033 .WORD 33 ;UNIQUE ERROR NUMBER
1794 004514 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1795 ;. CACHE SYSTEM ERROR
1796 004516 032767 000340 173220 2$: BIT @340, MSER ;IF MSER BITS <7:5> ZERO
1797 004524 001003 BNE 3$ ;THEN
1798 004526 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1799 004530 000034 .WORD 34 ;UNIQUE ERROR NUMBER
1800 004532 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1801 ;. CACHE SYSTEM ERROR
1802 004534 005067 173204 3$: CLR MSER ;CLEAR MSER
1803 004540 005767 173200 TST MSER ;IF MSER NOT CLEAR
1804 004544 001403 BEQ 4$ ;THEN

```

```

1805 004546 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
1806 004550 000035        .WORD 35      ;UNIQUE ERROR NUMBER
1807 004552 001325        .WORD ERRMSG  ;ADDRESS OF ERROR MESSAGE
1808                                     ;CACHE SYSTEM ERROR
1809 004554 052767 000100 173164 41:  BIS  #BIT06,CCR ;SET CCR BIT 6 (WRITE WRONG PARITY)
1810 004562 112737 000377 001123      MOVB #377, @TSTLOC+1 ;WRITE HIGH BYTE OF TEST LOCATION
1811 004570 042767 000100 173150      BIC  #BIT06,CCR ;CLEAR WRITE WRONG PARITY BIT
1812 004576 105737 001123              TSTB @TSTLOC+1 ;READ HIGH BYTE OF TEST LOCATION
1813 004602 032767 000010 173142      DIT  #BIT03,HITMIS ;IF BIT 3 SET IN HIT/MISS REGISTER
1814 004610 001411          BEQ  51          ;THEN
1815 004612 012737 004400 001120      MOV  @TST12+4,@SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1816 004620 004767 175014          JSR  PC,APTSFT    ;GO TO SOFT ERROR SUBROUTINE
1817 004624 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
1818 004626 000036        .WORD 36      ;UNIQUE ERROR NUMBER
1819 004630 001325        .WORD ERRMSG  ;ADDRESS OF ERROR MESSAGE
1820                                     ;CACHE SYSTEM ERROR
1821 004632 000407          BR  61          ;GO TO NEXT TEST IF NO HALT ON ERROR
1822 004634 032767 000340 173102 51:  BIT  #340,MSER ;IF BITS <7:5> ZERO
1823 004642 001003          BNE  61          ;THEN
1824 004644 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
1825 004646 000037        .WORD 37      ;UNIQUE ERROR NUMBER
1826 004650 001325        .WORD ERRMSG  ;ADDRESS OF ERROR MESSAGE
1827                                     ;CACHE SYSTEM ERROR
1828 004652 012767 000400 173066 61:  MOV  #BIT08,CCR ;FLUSH CACHE, CLEAR CCR BEFORE EXIT
1829 004660 016737 174144 000114      MOV  SLOC00,@0114 ;RESTORE CONTENTS OF VECTOR 114
1830 004666                                     ;
1831 004666 005037 177744          CLR  @MSER      ;CLEAR THE MEM SYS ERROR REG
1832 004672 005037 177766          CLR  @CPCEREG  ;CLEAR THE CPU ERROR REGISTER
1833 004676 005037 001116          CLR  @SOFTER   ;CLEAR THE SOFT ERROR COUNT
1834 004702 000167 000010          JMP  TST13     ;JUMP OVER DATA TABLES OR SUBROUTINES
1835                                     ;TO NEXT TEST
1836
1837 004706          DAPABU:
1838 004706 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
1839 004710 000040        .WORD 40      ;UNIQUE ERROR NUMBER
1840 004712 001325        .WORD ERRMSG  ;ADDRESS OF ERROR MESSAGE
1841                                     ;CACHE SYSTEM ERROR
1842 004714 000002          RTI
1843
1844
1845
1846 004716          CTST13:
1847                                     ;*****
1848 ;TEST 13      TEST WRITE WRONG TAG PARITY
1849                                     ;*****
1850 ;WRITE WRONG TAG PARITY - THIS TEST WILL VERIFY THAT A READ MISS
1851 ;OCCURS AFTER A WRITE WILL CCR<10> = 1 AND CCR<7,0> = C,1. THE WRITE
1852 ;TO THE LOCATION WILL CAUSE A CACHE UPDATE BUT THE TAG WILL BE
1853 ;WRITTEN WITH THE WRONG PARITY. WHEN THE LOCATION IS READ INSTEAD
1854 ;OF A CACHE HIT OCCURRING THE PARITY ERROR SHOULD CAUSE A CACHE
1855 ;MISS.
1856 ;
1857 ;BGNTST
1858 ;SAVE CONTENTS OF VECTOR 114
1859 ;LET VECTOR 114 POINT TO ABORT ROUTINE
1860 ;CLEAR MSER

```



```

1861 ;SET WRITE WRONG TAG PARITY BIT<10>
1862 ;WRITE TEST LOCATION
1863 ;CLEAR CCR BIT<10> AND SET ABORT DISABLE BIT<0>
1864 ;READ TEST LOCATION
1865 ;IF BIT<1> SET IN HIT/MISS REGISTER THEN
1866 ;. ERROR IN CACHE SYSTEM
1867 ;ELSE
1868 ;. ERROR IN CACHE SYSTEM
1869 ;ENDIF
1870 ;RESTORE VECTOR 114
1871 ;EXIT TST
1872 ;
1873 ;TAG PARITY ABORT ROUTINE:
1874 ;. ERROR IN CACHE SYSTEM
1875 ;RETURN
1876 ;ENDTST
1877 ;*****
1878 004716 TST13:
1879 004716 005267 174062 INC $TESTN ;INCREMENT TEST NUMBER
1880 ;***** NOTE *****
1881 ; DON'T SINGLE STEP THIS CODE USING THE GDT P COMMAND!!!
1882 ; PARITY ERRORS WILL OCCUR.
1883 ;*****
1884 004722 013767 000114 174100 MOV $0114, SLOCC0 ;SAVE CONTENTS OF VECTOR 114
1885 004730 012737 005062 000114 MOV $TAPAB0,$0114 ;LET VECTOR POINT TO ABORT ROUTINE
1886 004736 005067 173002 CLR MSER ;CLEAR MSER
1887 004742 012767 002001 172776 MOV $2001, CCH ;SET WRITE WRONG TAG PARITY & NO PARITY INTERRUPT
1888 004750 005037 001122 CLR $TSTLOC ;WRITE LOCATION WITH BAD TAG PARITY
1889 004754 012767 000001 172764 MOV $BIT00, CCR ;CLEAR BIT 10 AND SET NO PARITY INTERRUPTS
1890 004762 005737 001122 TST $TSTLOC ;READ TEST LOCATION
1891 004766 032767 000010 172756 BIT $BIT03, HITMIS ;IF BIT 3 SET IN HIT/MISS REGISTER
1892 004774 001414 BEQ 2$ ;THEN
1893 004776 012737 004722 001120 MOV $TST13+4,$SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
1894 005004 004767 174630 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
1895 005010 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1896 005012 000041 .WORD 41 ;UNIQUE ERROR NUMBER
1897 005014 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1898 ;. CACHE SYSTEM ERROR
1899 005016 000403 BR 2$ ;ELSE
1900 005020 1$:
1901 005020 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1902 005022 000042 .WORD 42 ;UNIQUE ERROR NUMBER
1903 005024 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
1904 ;. CACHE SYSTEM ERROR
1905 005026 012767 000400 172712 2$: MOV $BIT08,CCR ;FLUSH CACHE, CLEAR CCR BEFORE EXIT
1906 005034 016737 173770 000114 MOV SLOCC0, $0114 ;RESTORE CONTENTS OF VECTOR 114
1907 005042 005037 177744 CLR $MSER ;CLEAR THE MEM SYS ERROR REG
1908 005046 005037 177766 CLR $CPEREG ;CLEAR THE CPU ERROR REGISTER
1909 005052 005037 001116 CLR $SOFTER ;CLEAR THE SOFT ERROR COUNT
1910 005056 000167 000010 JMP TST14 ;JUMP OVER DATA TABLES OR SUBROUTINES
1911 ;TO NEXT TEST
1912
1913 005062 TAPAB0:
1914 005062 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
1915 005064 000042 .WORD 42 ;UNIQUE ERROR NUMBER
1916 005066 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE

```

```

1917                                     ;CACHE SYSTEM ERROR
1918 005070 000002                       RTI
1919
1920
1921 005072      CTST15:
1922      ;*****
1923      ;*TEST 14      TEST PARITY INTERRUPT
1924      ;*****
1925      ;PARITY INTERRUPT TEST - THIS TEST WILL VERIFY THAT WHEN CCR<7,0> =
1926      ;0,0, A PARITY INTERRUPT OCCURS AFTER EXECUTION OF AN INSTRUCTION
1927      ;THAT HAS BEEN WRITTEN WITH WRONG PARITY.
1928
1929      ;
1930      ;BGNIST
1931      ;SAVE CONTENTS OF 114
1932      ;SETUP VECTOR 114 TO POINT TO INTERRUPT ROUTINE
1933      ;CLEAR EXPECTING INTERRUPT FLAG
1934      ;WRITE TEST ADDRESS WITH BAD PARITY
1935      ;SET EXPECTING INTERRUPT FLAG
1936      ;READ TEST ADDRESS
1937      ;IF INTERRUPT FLAG NE 0 THEN
1938      ;.      ERROR IN CACHE SYSTEM
1939      ;ENDIF
1940      ;RESTORE CONTENTS OF VECTOR 114
1941      ;EXIT   TST
1942
1943      ; INTERRUPT ROUTINE:      IF EXPECTING INTERRUPT FLAG NE 1 THEN
1944      ;.                          ERROR IN CACHE SYSTEM
1945      ;.
1946      ;.      ELSE
1947      ;.                          CLEAR (EXPECTING) INTERRUPT FLAG
1948      ;.      ENDIF
1949      ;.      IF SAVED PC NE TO UPDATED PC THEN
1950      ;.                          ERROR IN CACHE SYSTEM
1951      ;.      ENDIF
1952      ;.      IF MSER NE 0340 THEN
1953      ;.                          ERROR IN CACHE SYSTEM
1954      ;.      ENDIF
1955      ;.      RETURN
1956      ;
1957      ;ENDTST
1958      ;*****
1959      TST14:
1960      INC      $TESTN      ;INCREMENT TEST NUMBER
1961      JSR      PC,INITMM   ;INIT MMU; PAGE ZERO REFERENCES WILL
1962      ;BYPASS CACHE.
1963      MOV      01600,KIPAR6 ;SET PAR6 TO RELOCATE TO BANK 7
1964      MOV      0140000,00TSTLOC ;SET UP TEST LOCATION TO OBTAIN
1965      ;PHYSICAL ADDR VIA KIPAR6.
1966      BIS      0BIT00,SRO  ;TURN ON MMU
1967      CLR      R1          ;CLEAR EXPECTING INTERRUPT FLAG
1968      BIS      0RIT06,CCR  ;SET WRITE WRONG DATA PARITY
1969      CLR      0TSTLOC    ;WRITE CACHE LOCATION 0 WITH BAD DATA PARITY
1970      CLR      CCR        ;CLEAR WRITE WRONG DATA PARITY
1971      CLR      00MSER     ;INIT MSER
1972      MOV      00114,SLOC00 ;SAVE CONTENTS OF VECTOR 114
1973      MOV      0INTERR,00114 ;LET VECTOR POINT TO INTERRUPT ROUTINE.
1974      COM      R1         ;SET EXPECTING INTERRUPT FLAG

```

```

1973 005166 005777 173730      INTRPC: TST      @TSTLOC
1974 005172 005701              TST      R1
1975                                ;READ TEST LOCATION; SHOULD GET PARITY INTERRUPT
1976 005174 001403              BEQ      1$
1977 005176 104000              ERROR
1978 005200 000044              .WORD   44
1979 005202 001325              .WORD   ERRMSG
1980                                ;RETURN HERE FROM INTERRUPT
1981 005204 016737 173620 000114 1$:  MOV      SLOC00, @0114
1982 005212 005067 172354              CLR      SRO
1983 005218 012737 000400 177746      MOV      @BIT08, @0CCR
1984 005224 005037 177744              CLR      @0MSER
1985 005230 005037 177766              CLR      @0CPEREG
1986 005234 005037 001116              CLR      @0SOFTER
1987 005240 000167 000052              JMP      TST15
1988                                ;RESTORE VECTOR 114
1989                                ;TURN OFF MMU
1990 005244 005701              INTERR: TST      R1
1991 005246 001004              BNE     1$
1992 005250 104000              ERROR
1993 005252 000045              .WORD   45
1994 005254 001325              .WORD   ERRMSG
1995                                ;IF EXPECTING INTERRUPT FLAG NOT SET
1996 005256 000401              BR      2$
1997 005260 005001              1$:  CLR      R1
1998 005262 021627 005172              2$:  CMP      @SP,    @INTRPC+4
1999 005266 001403              BEQ     4$
2000 005270 104000              ERROR
2001 005272 000046              .WORD   46
2002 005274 001325              .WORD   ERRMSG
2003                                ;IF SAVED PC NOT EQUAL TO UPDATED PC
2004 005276 022767 000340 172440 4$:  CMP      @540,   MSER
2005 005304 001403              BEQ     5$
2006 005306 104000              ERROR
2007 005310 000047              .WORD   47
2008 005312 001325              .WORD   ERRMSG
2009                                ;IF MSER NOT EQUAL TO EXPECTED VALUE
2010 005314 000002              5$:  RTI
2011                                ;THEN
2012                                ;ALL ERRORS TO TRAP TO EMT VECTOR
2013 005316                                ;UNIQUE ERROR NUMBER
2014                                ;ADDRESS OF ERROR MESSAGE
2015                                ;CACHE SYSTEM ERROR
2016                                ;ELSE
2017                                ;CLEAR (EXPECTING) INTERRUPT FLAG
2018                                ;IF SAVED PC NOT EQUAL TO UPDATED PC
2019                                ;THEN
2020                                ;ALL ERRORS TO TRAP TO EMT VECTOR
2021                                ;UNIQUE ERROR NUMBER
2022                                ;ADDRESS OF ERROR MESSAGE
2023                                ;CACHE SYSTEM ERROR
2024                                ;RETURN
2025
2026 CTST17:
2027 ;*****
2028 ;*TEST 15      TEST THAT PARITY ERRORS BLOCKED BY NXM ABORT
2029 ;*****
2030 ;CHECK PARITY ABORTS BLOCKED BY NON-EXISTENT MEMORY ABORT - THIS TEST WILL
2031 ;VERIFY THAT IF A PARITY ERROR OCCURS ON THE SAME ADDRESS REFERENCE AS A
2032 ;NON-EXISTENT MEMORY ERROR THAT THE CACHE DATA PATH GATE ARRAY BLOCKS THE
2033 ;PARITY ERROR TO THE J-11 CHIP SET. THIS WILL BE DONE BY USING THE DIAGNOSTIC
2034 ;BIT TO CAUSE A PARITY ERROR IN A CACHE REFERENCE THAT DOES NOT HAVE A
2035 ;CORRESPONDING ADDRESS IN MAIN MEMORY. THE ADDRESS WILL THEN BE READ CAUSING
2036 ;BOTH A CACHE PARITY ERROR AND A NON EXISTENT MEMORY ERROR. TO AVOID HAVING
2037 ;TO SIZE THE ENTIRE MEMORY TO FIND A NON-EXISTENT MEMORY ADDRESS THIS TEST
2038 ;WILL USE THE LARGEST NON I/O ADDRESS (1775776). THE TEST WILL FIRST READ
2039 ;THIS ADDRESS AND IF A NXM TRAP OCCURS THE TEST WILL BE DONE. IF THE ACCESS
2040 ;TO THIS ADDRESS DOES NOT TRAP THEN THE TEST WILL BE SKIPPED.
2041 ;

```

```

2085 005454 000051 .WORD 51 ;UNIQUE ERROR NUMBER
2086 005456 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
2087 ;CACHE SYSTEM ERROR
2088 005460 052706 000004 2$: ADD #4, SP ;RESET STACK AFTER TRAP
2089 005462 005067 172256 3$: CLR CCR ;CLEAR CCR FOR EXIT
2090 005470 010437 000004 MOV R4, #04 ;RESTORE VECTOR 4
2091 005474 016737 173330 000114 MOV SLOC00, #0114 ;RESTORE VECTOR 114
2092 005480 005067 172064 CLR SRO ;TURN OFF MMU BEFORE EXITING.
2093 005506 005037 177744 CLR #0MSER ;CLEAR THE MEM SYS ERROR REG
2094 005510 005037 177766 CLR #0CPEREG ;CLEAR THE CPU ERROR REGISTER
2095 005514 005037 001116 CLR #0SOFTER ;CLEAR THE SOFT ERROR COUNT
2096 005520 000167 000000 JMP TST16 ;JUMP OVER DATA TABLES OR SUBROUTINES
2097 ;TO NEXT TEST
2098
2099
2100
2101 005526

```

```

CTST20:
;*****
;*TEST 16 TEST MULTIPROCESSING INSTRUCTIONS
;*****
;MULTIPROCESSING INSTRUCTION TESTS - THIS TEST WILL VERIFY THAT THE MULTI-
;PROCESSING INSTRUCTIONS DO A BYPASS OF THE CACHE. THIS TEST WILL NOT VERIFY
;THE REST OF THE FUNCTIONALITY OF THESE INSTRUCTIONS BECAUSE THAT WILL ALREADY
;HAVE BEEN CHECKED IN THE BASE INSTRUCTION TESTS.
;THE TEST FOR TSTSET AND WRTLCK WILL FIRST ALLOCATE AN ADDRESS IN CACHE WITH
;KNOWN DATA, THEN FORCE CACHE MISS WILL BE ENABLED AND THE SAME ADDRESS WILL
;BE WRITTEN WITH WITH DIFFERENT DATA. THIS WILL SET UP CACHE WITH DATA 1 AND
;IT'S ASSOCIATED MEMORY LOCATION WITH DATA 2. FORCE CACHE MISS WILL THEN BE
;DISABLED AND THE TSTSET/WRTLCK INSTRUCTION WILL BE EXECUTED. THIS INSTRUCTION
;WILL CAUSE CACHE TO BE BYPASSED AND AN ACCESS TO MAIN MEMORY WILL BE MADE. IN
;ADDITION THIS INSTRUCTION WILL CAUSE CACHE TO BE RE-ALLOCATED WITH THE CORRECT DATA.
;THE TEST FOR ASRB INSTRUCTION DIFFERS SLIGHTLY. FIRST THE CACHE WILL BE
;ALLOCATED, THEN AN ASRB INSTRUCTION WILL BE DONE AT THAT ADDRESS.
;A HIT SHOULD BE RECORDED ON THE ACCESS. NEXT, THE ADDRESS WILL BE READ AND
;A MISS SHOULD BE RECORDED BECAUSE THE FORCED BYPASS ON THE ASRB INSTRUCTION
;SHOULD HAVE INVALIDATED THE CACHE ENTRY, AND BECAUSE IT'S A BYTE WRITE; CACHE
;WILL NOT BE RE-ALLOCATED.
; *****NOTE*****
; IF THE TSTSET OR WRTLCK ISTRUCTIONS ARE ASSEMBLED SO THAT BITS
; 5-0 OF THE ADDRESS OF THE INSTRUCTION FOLLOWING THE TSTSET/WRTLCK INSTRUCTION
; ARE EQUAL TO BITS 5-0 OF THE TSTLOC ADDRESS THE TEST WILL FAIL. THIS IS CAUSED
; BY THE NEED OF AN ADDITIONAL FETCH WHEN THE PRE-FETCH BUFFER IS INVALIDATED
; WHEN BITS 5-0 OF AN ADDRESS WRITTEN TO ARE EQUAL TO BITS 5-0 OF THE ADDRESS
; IN THE PREFETCH BUFFER. THE ADDITIONAL FETCH CAUSES THE REQUIRED DATA TO BE
; SHIFTED OUT OF THE RANGE OF THE HIT/MISS REGISTER. PADDING THE FRONT OF THE
; TEST WITH NOPS MAY BE REQUIRED TO REMEDY THIS PROBLEM.
;BGNST
;READ TEST LOCATION TO ALLOCATE CACHE
;DO TSTSET INSTRUCTION
;IF HIT/MISS REGISTER BIT 3 NOT SET THEN
;. ERROR IN CACHE SYSTEM
;ENDIF
;READ TEST LOCATION
;IF HIT/MISS REGISTER BIT 3 SET THEN
;. ERROR IN CACHE SYSTEM
;ENDIF

```

```

2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140

```

```

2141 ;READ TEST LOCATION TO ALLOCATE CACHE
2142 ;DO WRITLCK INSTRUCTION
2143 ;IF HIT/MISS REGISTER BIT 3 NOT SET THEN
2144 ;. ERROR IN CACHE SYSTEM
2145 ;ENDIF
2146 ;READ TEST LOCATION
2147 ;IF HIT/MISS REGISTER BIT 3 SET THEN
2148 ;. ERROR IN CACHE SYSTEM
2149 ;ENDIF
2150 ;READ TEST LOCATION TO ALLOCATE CACHE
2151 ;DO ASRB INSTRUCTION
2152 ;IF HIT/MISS REGISTER BIT 3 NOT SET THEN
2153 ;. ERROR IN CACHE SYSTEM
2154 ;ENDIF
2155 ;READ TEST LOCATION
2156 ;IF HIT/MISS REGISTER BIT 3 SET THEN
2157 ;. ERROR IN CACHE SYSTEM
2158 ;ENDIF
2159 ;ENDTST
2160 ;*****
2161 TST16:
2162 005526 005267 173252 INC $TESTN ;INCREMENT TEST NUMBER
2163 005532 005067 173364 CLR TSTLOC ;WRITE TEST LOCATION TO ALLOCATE CACHE
2164 005536 012767 000004 172202 MOV $BIT02,CCR ;SET CACHE TO FORCE MISS
2165 005544 012737 177777 001122 MOV $177777,$TSTLOC ;WRITE TO MEMORY BUT NOT TO CACHE.
2166 ;CACHE=000000; TSTLOC=177777
2167 005552 005037 177746 CLR $CCR ;CLEAR FORCE MISS
2168 ; TSTSET TSTLOC ;DO TSTSET INSTRUCTION
2169 005556 007237 .WORD 7237 ;THESE NEXT TWO LOCATIONS ARE THE TSTSET
2170 005560 001122 .WORD TSTLOC ;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY
2171 005562 001415 BEQ 100$ ;IF = 0 THEN CACHE WAS NOT BYPASSED
2172 005564 032767 000040 172160 BIT $BIT05,HITMIS ;IF HIT/MISS REGISTER BIT 5 IS SET
2173 005572 001014 BNE 1$ ;THEN GO TO NEXT TEST.
2174 005574 012737 005532 001120 MOV $TST16+4,$SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
2175 005582 004767 174032 JSR PC,APTSFT ;GO TO SOFT ERROR SUBROUTINE
2176 005606 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
2177 005610 000052 .WORD 52 ;UNIQUE ERROR NUMBER
2178 005612 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
2179
2180 ;CACHE SYSTEM ERROR
2181 005614 000403 BR 1$ ;ERROR! BIT 5 IN HIT MISS REG WAS NOT SET.
2182 005616 100$ ;RECOVER FROM ERROR
2183 005616 104000 ERROR ;ALL ERRORS TO TRAP TO EMT VECTOR
2184 005620 000053 .WORD 53 ;UNIQUE ERROR NUMBER
2185 005622 001325 .WORD ERRMSG ;ADDRESS OF ERROR MESSAGE
2186
2187 ;CACHE SYSTEM ERROR
2188 005624 032767 177777 173270 1$: CMP $177777,TSTLOC ;ERROR! CACHE WAS NOT BYPASSED BY TSTSET
2189 005632 001015 BNE 200$ ;READ TEST LOCATION (ALSO ALLOCATES CACHE FOR WR
2190 ;CACHE SHOULD BE RE-ALLOCATED WITH
2191 ;177777 AS A RESULT OF TSTSET INSTRUCTION.
2192 005642 001014 BNE 2$ ;IF HIT/MISS REGISTER BIT 4 SET (SHOULD BE)
2193 005644 012737 005532 001120 MOV $TST16+4,$SOFTRE ;THEN
2194 005652 004767 173762 JSR PC,APTSFT ;MOVE RETRY ADDRESS TO SOFT RETURN
2195 005656 104000 ERROR ;GO TO SOFT ERROR SUBROUTINE
2196 005660 000054 .WORD 54 ;ALL ERRORS TO TRAP TO EMT VECTOR
;UNIQUE ERROR NUMBER

```

GLOBAL AREAS MACY11 30A(1052) 04-APR-84 10:38 PAGE 47  
 KDJ11A.MAC 03-APR-84 11:36 T16 TEST MULTIPROCESSING INSTRUCTIONS

SEQ 0047

2197	005662	001325				.WORD	ERRMSG		;ADDRESS OF ERROR MESSAGE
2198									;CACHE SYSTEM ERROR
2199	005664	000403				BR	2\$		;RECOVER FROM ERROR
2200	005666				200\$:				
2201	005666	104000				ERROR			;ALL ERRORS TO TRAP TO EMT VECTOR
2202	005670	000055				.WORD	55		;UNIQUE ERROR NUMBER
2203	005672	001325				.WORD	ERRMSG		;ADDRESS OF ERROR MESSAGE
2204									;CACHE SYSTEM ERROR
2205									;ERROR! CACHE WAS NOT RE-ALLOCATED
2206	005674	012767	000004	172044	2\$:	MOV	0BIT02,CCR		;SET FORCE CACHE MISS.
2207	005702	005067	173214			CLR	TSTLOC		;CACHE=177777; TSTLOC=000000
2208	005706	005000				CLR	RO		;CLR RO
2209	005710	005067	172032			CLR	CCR		;CLEAR FORCE CACHE MISS.
2210						WRTLCK	TSTLOC		;DO WRTLCK INSTRUCTION
2211	005714	007337				.WORD	7337		;THESE NEXT TWO WORDS ARE THE WRTLCK
2212	005716	001122				.WORD	TSTLOC		;INSTR. BECAUSE THE ASSEMBLER WAS NOT READY
2213	005720	001015				BNE	300\$		
2214	005722	032767	000040	172022		BIT	0BIT05, HITMIS		;IF HIT/MISS REGISTER BIT 5 NOT SET
2215	005730	001014				BNE	3!		;THEN
2216	005732	012737	005532	001120		MOV	0TST16+4,00SOFTRE		;MOVE RETRY ADDRESS TO SOFT RETURN
2217	005740	004767	173674			JSR	PC,APTSFT		;GO TO SOFT ERROR SUBROUTINE
2218	005744	104000				ERROR			;ALL ERRORS TO TRAP TO EMT VECTOR
2219	005748	000056				.WORD	56		;UNIQUE ERROR NUMBER
2220	005750	001325				.WORD	ERRMSG		;ADDRESS OF ERROR MESSAGE
2221									;CACHE SYSTEM ERROR
2222	005752	000403				BR	3\$		;RECOVER FROM ERROR
2223	005754				300\$:				
2224	005754	104000				ERROR			;ALL ERRORS TO TRAP TO EMT VECTOR
2225	005756	000057				.WORD	57		;UNIQUE ERROR NUMBER
2226	005760	001325				.WORD	ERRMSG		;ADDRESS OF ERROR MESSAGE
2227									;CACHE SYSTEM ERROR
2228									;ERROR! CACHE WAS NOT BYPASSED BY WRTLCK.
2229	005762	007767	173134		3\$:	TSTLOC			;READ TEST LOCATION
2230	005766	001015				BNE	400\$		
2231	005770	032767	000020	171754		BIT	0BIT04, HITMIS		;IF HIT/MISS REGISTER BIT 4 SET
2232	005776	001014				BNE	4!		;THEN
2233	006000	012737	005532	001120		MOV	0TST16+4,00SOFTRE		;MOVE RETRY ADDRESS TO SOFT RETURN
2234	006006	004767	173626			JSR	PC,APTSFT		;GO TO SOFT ERROR SUBROUTINE
2235	006012	104000				ERROR			;ALL ERRORS TO TRAP TO EMT VECTOR
2236	006014	000060				.WORD	60		;UNIQUE ERROR NUMBER
2237	006016	001325				.WORD	ERRMSG		;ADDRESS OF ERROR MESSAGE
2238									;CACHE SYSTEM ERROR
2239	006020	000403				BR	4\$		;RECOVER FROM ERROR
2240	006022				400\$:				
2241	006024	104000				ERROR			;ALL ERRORS TO TRAP TO EMT VECTOR
2242	006024	000061				.WORD	61		;UNIQUE ERROR NUMBER
2243	006026	001325				.WORD	ERRMSG		;ADDRESS OF ERROR MESSAGE
2244									;CACHE SYSTEM ERROR
2245									;ERROR! CACHE WAS NOT RE-ALLOCATED
2246	006030	005767	173066		4\$:	TST	TSTLOC		;READ TEST LOCATION TO ALLOCATE CACHE
2247	006032	000240				NOP			;NOP PAD FOR ASRB HIT/MISS CALC
2248	006036	106267	173060			ASRB	TSTLOC		;DO ASRB INSTRUCTION
2249	006042	032767	000020	171702		BIT	0BIT04, HITMIS		;IF HIT/MISS REGISTER BIT 4 NOT SET
2250	006050	001010				BNE	5\$		;THEN
2251	006052	012737	005532	001120		MOV	0TST16+4,00SOFTRE		;MOVE RETRY ADDRESS TO SOFT RETURN
2252	006060	004767	173554			JSR	PC,APTSFT		;GO TO SOFT ERROR SUBROUTINE

```

2253 006061 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
2254 006066 000062          .WORD          62          ;UNIQUE ERROR NUMBER
2255 006070 001325          .WORD          ERRMSG       ;ADDRESS OF ERROR MESSAGE
2256                                     ;CACHE SYSTEM ERROR
2257 006072 005767 173024 5$:  TST          TSTLOC       ;READ TEST LOCATION
2258 006076 002767 000010 171646 BIT          @BIT03, HITMIS ;IF HIT/MISS REGISTER BIT 3 SET
2259 006104 001410          BEQ          6$          ;THEN
2260 006106 002737 005532 001120 MOV          @TST16+4,@SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
2261 006114 004767 173520          JSR          PC,APTSFT     ;GO TO SOFT ERROR SUBROUTINE
2262 006120 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
2263 006122 000062          .WORD          63          ;UNIQUE ERROR NUMBER
2264 006124 001325          .WORD          ERRMSG       ;ADDRESS OF ERROR MESSAGE
2265                                     ;CACHE SYSTEM ERROR
2266 006126          6$:          CLR          @MMSER       ;CLEAR THE MEM SYS ERROR REG
2267 006132 005037 177744          CLR          @CPEREG      ;CLEAR THE CPU ERROR REGISTER
2268 006136 005037 001116          CLR          @SOFTERR     ;CLEAR THE SOFT ERROR COUNT
2269 006142 000167 000000          JMP          TST17        ;JUMP OVER DATA TABLES OR SUBROUTINES
2270                                     ;TO NEXT TEST
    
```

```

2271
2272
2273
2274
2275 006146  CTST21:
2276          ;*****
2277          ;*TEST 17      TEST CACHE DATA RAMS
2278          ;*****
2279          ;DATA STORE RAM TESTS - THERE ARE TWO TESTS FOR THE DATA STORE RAM.
2280          ;THE FIRST TEST WILL BE A NO DUAL ADDRESSING TEST AND THE SECOND
2281          ;TEST WILL BE A DATA RELIABILITY TEST. THE NO DUAL ADDRESSING TEST
2282          ;WILL FIRST WRITE EACH WORD OF THE CACHE RAM WITH ITS WORD ADDRESS
2283          ;AND VERIFY THE CONTENTS WITH A READ OF EACH ADDRESS. THEN EACH
2284          ;BYTE WILL BE WRITTEN WITH ITS ADDRESS AND CHECKED. THE DATA
2285          ;RELIABILITY TEST THAT WILL BE USED IS A TEST CALLED MOVING INVERSIONS.
2286          ;
2287          ;BUNTST 1
2288          ;SETUP MMU REGISTERS TO HAVE BYPASS ON KERNAL SPACE AND NO
2289          ;      BYPASS ON USER SPACE
2290          ;LET PS EQUAL KERNAL FOR CURRENT MODE AND USER FOR PREVIOUS
2291          ;      MODE
2292          ;ENABLE MMU
2293          ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
2294          ;CLEAR DATA TO BE WRITTEN
2295          ;SET DIAGNOSTIC BIT (BIT<1>) IN CCR
2296          ;DO UNTIL DATA TO BE WRITTEN EQUALS 20000(8)
2297          ;.      WRITE DATA TO ADDRESS
2298          ;.      ADD 2 TO ADDRESS
2299          ;.      ADD 2 TO DATA TO BE WRITTEN
2300          ;ENDDO
2301          ;SAVE CONTENTS OF VECTOR 114
2302          ;LET VECTOR 114 POINT TO DATA STORE PARITY ABORT ROUTINE
2303          ;GET FIRST ADDRESS OF 4K WORD DATA BUFFER
2304          ;CLEAR EXPECTED DATA
2305          ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
2306          ;.      READ ADDRESS
2307          ;.      IF RECEIVED DATA NE EXPECTED DATA THEN
2308          ;.      ERROR IN CACHE SYSTEM
    
```

```

2309      ;;      ENDIF
2310      ;;      ADD 2 TO EXPECTED DATA
2311      ;;      ADD 2 TO ADDRESS
2312      ;ENDDO
2313      ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
2314      ;CLEAR DATA TO BE WRITTEN
2315      ;DO UNTIL ALL BYTES CHECKED
2316      ;;      WRITE DATA TO ADDRESS
2317      ;;      ADD 1 TO ADDRESS
2318      ;;      ADD 1 TO DATA TO BE WRITTEN
2319      ;ENDDO
2320      ;GET FIRST ADDRESS OF 8K BYTE DATA BUFFER
2321      ;CLEAR EXPECTED DATA
2322      ;DO UNTIL EXPECTED DATA EQUALS 20000(8)
2323      ;;      READ ADDRESS
2324      ;;      IF RECEIVED DATA NE EXPECTED DATA THEN
2325      ;;      ERROR IN CACHE SYSTEM
2326      ;;      ENDIF
2327      ;;      ADD 1 TO EXPECTED DATA
2328      ;;      ADD 1 TO ADDRESS
2329      ;ENDDO
2330      ;RESTORE VECTOR 114
2331      ;ENDTST
2332      ;
2333      ;*****
2334      TST17:
2335      006146      005267      172632      INC      $TESTN      ;INCREMENT TEST NUMBER
2336      006152      004767      173234      JSR      PC, INITMM ;SETUP MMU REGISTERS/FORCE CACHE BYPASS PAGES 0-
2337      006156      012767      001600      164170      MOV      $1600, KIPAR6 ;LET PAR6 MAP TO ADDRESS 160000
2338      006164      042767      000004      164324      BIC      $BIT02,SR3   ;DISABLE KERNEL D SPACE.
2339      006172      052737      000001      177572      BIS      $BIT00,$SRO  ;TURN ON MEMORY MANAGEMENT
2340      006200      012702      140000      MOV      $140000,R2   ;GET FIRST ADDRESS OF 4K WORD BUFFER
2341      006204      005001      CLR      R1           ;INIT DATA TO BE WRITTEN
2342      006206      052757      000002      171532      BIS      $BIT01,CCR   ;SET DIAGNOSTIC BIT
2343      006214      020127      020000      1$:      CMP      R1, $20000  ;DO UNTIL ALL ADDRESSES WRITTEN
2344      006220      001404      BEQ      2$          ;IF DONE GO CHECK DATA
2345      006222      010122      MOV      R1, (R2)+   ;WRITE DATA TO ADDRESS
2346      006224      062701      000002      ADD      $2, R1      ;UPDATE DATA BY 2 (WORD BOUNDARY)
2347      006230      000771      RR      1$          ;ENDDO
2348      006232      013767      000114      172570      2$:      MOV      $114, SLO00  ;SAVE CONTENTS OF VECTOR 114
2349      006240      012737      001630      000114      MOV      $RAMPAR,$114 ;LET VECTOR 114 POINT TO ABRCT ROUTINE
2350      006246      042767      000002      171472      BIC      $BIT01,CCR   ;CLEAR DIAGNOSTIC BIT. ALL FURTHER
2351      ;REFERENCES TO PAGE 6 SHOULD BE CACHED.
2352      006254      012702      140000      MOV      $140000,R2   ;GET FIRST ADDRESS OF 4K WORD BUFFER
2353      006260      005001      CLR      R1           ;INIT DATA TO BE CHECKED
2354      006262      020127      020000      3$:      CMP      R1, $20000  ;DO UNTIL ALL ADDRESSES CHECKED
2355      006266      001430      BEQ      5$          ;IF DONE GO DO BYTES
2356      006270      005712      TST      (R2)        ;CHECK THAT CACHE WAS
2357      006272      032737      000010      177752      BIT      $BIT03,$HITMIS ;ALLOCATED
2358      006300      001010      BNE      4$          ;SHOULD HAVE HIT
2359      006302      012737      006152      001120      MOV      $TST17+4,$SOFTRE ;MOVE RETRY ADDRESS TO SOFT RETURN
2360      006310      004767      173324      JSR      PC,APTSFT   ;GO TO SOFT ERROR SUBROUTINE
2361      ;
2362      ;
2363      006314      104000      ERROR   ;CACHE SYSTEM ERROR
2364      006316      000064      .WORD   64          ;EXPECTED A HIT; GOT A MISS
                       ;ALL ERRORS TO TRAP TO EMT VECTOR
                       ;UNIQUE ERROR NUMBER

```



GLOBAL AREAS MACY11 30A(1052) 04-APR-84 10:38 PAGE 50  
 KDJ11A,MAC 03-APR-84 11:36 T17 TEST CACHE DATA RAMS

SEQ 0050

```

2365 006320 001325          .WORD  ERRMSG          ;ADDRESS OF ERROR MESSAGE
2366 006322 012267 172510 4$:  MOV    (R2)+, RECDAT    ;READ TEST LOCATION
2367 006326 020167 172504    CMP    R1,    RECDAT    ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
2368 006332 001403          BEQ    100$           ;THEN
2369 006334 104000          ERROR          ;ALL ERRORS TO TRAP TO EMT VECTOR
2370 006336 000065          .WORD  65           ;UNIQUE ERROR NUMBER
2371 006340 001325          .WORD  ERRMSG          ;ADDRESS OF ERROR MESSAGE
2372                                .WORD  65           ;UNIQUE ERROR NUMBER
2373 006342 062701 000002 100$: ADD    #2,R1          ;UPDATE EXPECTED DATA BY 2
2374 006346 000745          BR     3$           ;ENDDD
2375
2376                                ;***NOW CHECK BYTES***
2377                                ;SINCE CACHE IS ALREADY ALLOCATED; WRITE BYTES WILL HIT
2378                                ;AND WRITE THROUGH CACHE TO MEMORY.
2379
2380 006350 052737 000002 177746 5$:  BIS    #BIT01,#CCR          ;SET DIAGNOSTIC MODE IN CCR
2381 006356 012702 140000    MOV    #140000,R2        ;GET FIRST ADDRESS OF 4K WORD BUFFER
2382 006362 005001          CLR    R1           ;INIT DATA TO BE WRITTEN
2383 006364 020127 020000 6$:  CMP    R1,    #20000    ;DO UNTIL ALL ADDRESSES WRITTEN
2384 006370 001404          BEQ    7$           ;IF DONE GO CHECK DATA
2385 006372 110122          MOVB   R1,    (R2)+    ;WRITE DATA TO ADDRESS
2386 006374 062701 000001    ADD    #1,    R1      ;UPDATE DATA BY 1 (BYTE BOUNDARY)
2387 006400 000771          BR     6$           ;ENDDD
2388 006402 012702 140000 7$:  MOV    #140000,R2        ;GET FIRST ADDRESS OF 4K WORD BUFFER
2389 006406 005001          CLR    R1           ;INIT DATA TO BE CHECKED
2390
2391 006410 105712          TSTB   (R2)          ;DO UNTIL ALL ADDRESSES CHECKED
2392 006412 032737 000010 177752 8$:  BIT    #BIT03,#HITMIS    ;CHECK THAT CACHE WAS ALLOCATED
2393 006420 001010          BNE   101$          ;SHOULD BE
2394 006422 012737 006152 001120    MOV    #TST17+4,#SOFTRE ; A HIT
2395 006430 004767 173204    JSR    PC,APTSFT    ;MOVE RETRY ADDRESS TO SOFT RETURN
2396                                ;GO TO SOFT ERROR SUBROUTINE
2397                                ;CACHE SYSTEM ERROR
2398 006434 104000          ERROR          ;EXPECTED A HIT; GOT A MISS
2399 006436 000066          .WORD  66           ;ALL ERRORS TO TRAP TO EMT VECTOR
2400 006440 001325          .WORD  ERRMSG          ;UNIQUE ERROR NUMBER
2401 006442 112267 172370 101$: MOVB   (R2)+, RECDAT    ;ADDRESS OF ERROR MESSAGE
2402 006446 120167 172364    CMPB   R1,    RECDAT    ;READ TEST LOCATION
2403 006452 001410          BEQ    9$           ;IF RECEIVED DATA NOT EQUAL TO EXPECTED
2404 006454 012737 006152 001120    MOV    #TST17+4,#SOFTRE ;THEN
2405 006462 004767 173152    JSR    PC,APTSFT    ;MOVE RETRY ADDRESS TO SOFT RETURN
2406 006466 104000          ERROR          ;GO TO SOFT ERROR SUBROUTINE
2407 006470 000067          .WORD  67           ;ALL ERRORS TO TRAP TO EMT VECTOR
2408 006472 001325          .WORD  ERRMSG          ;UNIQUE ERROR NUMBER
2409                                .WORD  67           ;ADDRESS OF ERROR MESSAGE
2410                                .WORD  ERRMSG          ;ADDRESS OF ERROR MESSAGE
2411 006474 005201          INC    R1           ;CACHE SYSTEM ERROR
2412 006476 020127 020000 9$:  CMP    R1,#20000    ;UPDATE EXPECTED DATA BY 1
2413 006502 001342          BNE   8$           ;IF ALL ADDRESS HAVE NOT BEEN TESTED
2414                                ;THEN GET NEXT ADDRESS
2415 006504 005067 171256 10$: CLR    CCR          ;ELSE EXIT TEST
2416 006510 005067 171056    CLR    SRO          ;CLEAR THE CCR
2417 006514 016737 172310 000114    MOV    SLOCO0, #114   ;TURN OFF MMU
2418 006522 005037 177744    CLR    #MSER        ;RESTORE VECTOR 114
2419 006526 005037 177766    CLR    #CPEREG      ;CLEAR THE MEM SYS ERROR REG
2420 006532 005037 001116    CLR    #SOFTERR     ;CLEAR THE CPU ERROR REGISTER
2420 006536 000167 000000    JMP    TST20        ;CLEAR THE SOFT ERROR COUNT
2420                                ;JUMP OVER DATA TABLES OR SUBROUTINES

```

```

2421                                     ;TO NEXT TEST
2422
2423 006542 CTST22:
2424 ;*****
2425 ;*TEST 20      CACHE DATA RAM DATA RELIABILITY TEST
2426 ;*****
2427 ;MOVING INVERSIONS TEST FOR DATA RAMS - THE TEST IS STARTED AFTER LOADING THE
2428 ;RAM STORE WITH 0'S. EACH ADDRESS IS READ AND VERIFIED TO BE ALL 0'S. THEN A
2429 ;1 IS SUBSTITUTED IN A BIT POSITION AND THE NEW WORD IS WRITTEN. NEXT THE
2430 ;ADDRESS IS READ TO VERIFY THE NEW CONTENTS. THIS IS REPEATED FOR EACH BIT OF
2431 ;THE WORD LEAVING THE ARRAY FILLED WITH 1'S. THE WHOLE PROCESS IS REPEATED
2432 ;PLUGGING IN 0'S TO THE 1'S AND REPEATED TWICE MORE ADDRESSING IN THE DOWNWARD
2433 ;DIRECTION. FINALLY EVERYTHING IS REPEATED FOR EACH BIT POSITION BEING THE
2434 ;LSB. TO SAVE TIME AND KNOWING THE LAYOUT OF THE RAM CHIPS INSTEAD OF DOING
2435 ;ONLY A SINGLE BIT AT A TIME EVERY FOURTH BIT WILL BE DONE CONCURRENTLY.
2436 ;
2437 ;BGNTST
2438 ;SETUP AND ENABLE MMU
2439 ;SETUP CCR TO ABORT PARITY ERRORS
2440 ;CLEAR CACHE
2441 ;LET FWDSEQ = #1
2442 ;LET ADDLSB = #1
2443 ;DO UNTIL ADDLSB EQ #0
2444 ;.      LET CURDAT = 0
2445 ;.      LET RITEDA = #1
2446 ;.      LET NEWDAT = #1
2447 ;.      IF FWDSEQ = #1 THEN
2448 ;.          LET FSTADD EQUAL FIRST ADDRESS OF 8K BYTE BUFFER
2449 ;.          LET LASTAD EQUAL LAST ADDRESS OF 8K BYTE BUFFER
2450 ;.      ELSE
2451 ;.          LET FSTADD EQUAL LAST ADDRESS OF 8K BYTE BUFFER
2452 ;.          LET LASTAD EQUAL FIRST ADDRESS OF 8K BYTE BUFFER
2453 ;.      ENDIF
2454 ;.      LET CURADD = FSTADD
2455 ;.      LET DCOUNT = #0
2456 ;.      SAVE CONTENTS OF VECTOR 114
2457 ;.      LET VECTOR 114 POINT TO DATA STORE PARITY ABORT ROUTINE
2458 ;.      DO UNTIL DCOUNT EQ #10
2459 ;.          LET RECDAT = @CURADD
2460 ;.          IF RECDAT NE CURDAT THEN
2461 ;.              ERROR IN CACHE SYSTEM
2462 ;.          ENDIF
2463 ;.          IF DCOUNT GT #3 THEN
2464 ;.              LET @CURADD = @CURADD CLEARBY RITEDA
2465 ;.          ELSE
2466 ;.              LET @CURADD = @CURADD SETBY RITEDA
2467 ;.          ENDIF
2468 ;.          LET RECDAT = @CURADD
2469 ;.          IF RECDAT NE NEWDAT THEN
2470 ;.              ERROR IN CACHE SYSTEM
2471 ;.          ENDIF
2472 ;.          IF CURADD EQ LASTAD THEN
2473 ;.              IF RITEDA = #210 THEN
2474 ;.                  IF DCOUNT NE #7 THEN
2475 ;.                      LET CURDAT = #377
2476 ;.                      LET RITEDA = #21

```





```

2589 007076 000401          BR      7#
2590 007100 050304          BIS     R3,    R4
2591 007102 005267 171746   6#:    INC     DCOUNT
2592 007106 016700 171766   7#:    MOV     FSTADD, R0
2593 007112 000426          BR      10#
2594 007114 005767 171746   8#:    TST     FWDSEQ
2595 007120 001412          BEQ     9#
2596 007122 066700 171742          ADD     ADDLSB, R0
2597 007126 022700 160000          CMP     0160000, R0
2598 007132 003016          BGT     10#
2599 007134 162700 020000          SUB     020000, R0
2600 007140 062700 000001          ADD     01,    R0
2601 007144 000411          BR      10#
2602 007146 166700 171716   9#:    SUB     ADDLSB, R0
2603 007152 022700 140000          CMP     0140000, R0
2604 007156 003404          BLE     10#
2605 007160 062700 020000          ADD     020000, R0
2606 007164 162700 000001          SUB     01,    R0
2607 007170 000663          10#:   BR      BGNLTP
2608 007172 005767 171670   ENDTLP: TST     FWDSEQ
2609 007176 001404          BEQ     1#
2610 007200 005067 171662          CLR     FWDSEQ
2611 007204 000167 177446          JMP     TSTLUP
2612 007210 012767 000001 171650 1#:    MOV     01,    FWDSEQ
2613 007216 006167 171646          ROL     ADDLSB
2614 007222 022767 020000 171640 ENDLUP: CMP     020000, ADDLSB
2615 007230 001402          BEQ     EXTST
2616 007232 000167 177420          JMP     TSTLUP
2617 007236 016737 171566 000114 EXTST: MOV     SLOC00, 00114
2618 007244 005067 170476          CLR     CCR
2619 007250 005067 170316          CLR     SRO
2620 007254 005037 177744          CLR     00MSER
2621 007260 005037 177766          CLR     00CPEREG
2622 007264 005037 001116          CLR     00SOFTER
2623 007270 000167 000000          JMP     TST21
2624
2625
2626 007274
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644

```

```

;ELSE
;LET NEWDAT BE SET BY WRITE DATA
;UPDATE LOOP COUNTER
;REINIT FIRST ADDRESS FOR THIS PASS
;ELSE
;IF ADDRESSING UPWARD
;THEN
;CALCULATE NEXT HIGHER ADDRESS
;IF CURRENT ADDRESS HAS BEEN INCREASED
;ABOVE HIGHEST ADDRESS THEN
;ROLL ADDRESS BACK
;ELSE
;CALCULATE NEXT LOWER ADDRESS
;IF CURRENT ADDRESS HAS BEEN DECREASED
;BELOW LOWEST ADDRESS THEN
;ROLL ADDRESS BACK
;ENDDO
;IF ADDRESSING UPWARD FINISHED
;THEN
;DO ADDRESSING DOWNWARD
;SET ADDRESSING UPWARD INDICATOR
;UPDATE LSB TO NEXT POSITION
;WE ONLY GO AS FAR AS 10000
;ENDDO
;RESTORE VECTOR 114
;INIT CCR FOR EXIT
;TURN OFF MMU
;CLEAR THE MEM SYS ERROR REG
;CLEAR THE CPU ERROR REGISTER
;CLEAR THE SOFT ERROR COUNT
;JUMP OVER DATA TABLES OR SUBROUTINES
;TO NEXT TEST

```

```

CTST23:
;*****
;TEST 21 TAG RAM DUAL ADDRESS TEST
;*****
;TAG STORE RAM TESTS - THERE ARE TWO TESTS FOR THE TAG STORE RAMS. THE FIRST
;TEST IS AN ADDRESSING TEST TO ENSURE NO DUAL ADDRESSING OF THE TAG STORE.
;THE SECOND TEST IS A "GALLOPING ONES AND ZEROS" TEST THAT CHECKS THE DATA
;RELIABILITY OF THE TAG RAMS.
;
;DUAL ADDRESSING TEST - THIS WILL BE DONE BY FIRST FLUSHING THE CACHE, THEN
;THE FIRST 256(10) LOCATIONS (BLOCK 0) WILL BE WRITTEN WITH ADDRESSES THAT
;WILL CAUSE A INCREMENTING PATTERN TO BE STORED IN THOSE LOCATIONS OF THE TAG
;STORE RAM. NEXT THE ENTIRE 4K ADDRESS RANGE WILL BE READ, THE HIT/MISS
;REGISTER SHOULD ONLY REPORT HITS ON THE ADDRESSES THAT WERE ALLOCATED. THIS
;PROCESS WILL BE REPEATED FOR EACH BLOCK FOR A TOTAL OF 16 (10) BLOCKS.
;
;INITIALIZE LOW ADDRESS
;SETUP AND ENABLE MMU
;INITIALIZE BLOCK COUNTER

```

```

2645 ;DO UNTIL ALL BLOCKS TESTED (16 TIMES)
2646 ;. FLUSH CACHE AND SET DIAGNOSTIC BIT
2647 ;. LET CURRENT ADDRESS = LOW ADDRESS
2648 ;. INITIALIZE ALLOCATION COUNTER
2649 ;. DO UNTIL ENTIRE BLOCK ALLOCATED (256 TIMES)
2650 ;. . IF CURRENT ADDRESS < 32K THEN
2651 ;. . . READ CURRENT ADDRESS
2652 ;. . . ELSE
2653 ;. . . WRITE CURRENT ADDRESS
2654 ;. . . ENDF
2655 ;. . . UPDATE CURRENT ADDRESS (ALSO ADD 200 TO PAR)
2656 ;. ENDDO
2657 ;. INITIALIZE GOOD ADDRESS
2658 ;. INITIALIZE CURRENT ADDRESS
2659 ;. INITIALIZE LOCATION COUNTER
2660 ;. SAVE CONTENTS OF VECTOR 114
2661 ;. LET VECTOR 114 POINT TO TAG STORE PARITY ABORT ROUTINE
2662 ;. DO UNTIL ALL LOCATIONS CHECKED (256 TIMES)
2663 ;. . INITIALIZE CHECK COUNTER
2664 ;. . DO UNTIL ADDRESS IN EACH BLOCK CHECKED
2665 ;. . . READ CURRENT ADDRESS
2666 ;. . . IF HIT THEN
2667 ;. . . . IF CURRENT ADDRESS NE GOOD ADDRESS THEN
2668 ;. . . . . ERROR IN CACHE SYSTEM
2669 ;. . . . . ENDF
2670 ;. . . . ELSE
2671 ;. . . . . IF CURRENT ADDRESS EQUAL GOOD ADDRESS THEN
2672 ;. . . . . . ERROR IN CACHE SYSTEM
2673 ;. . . . . ENDF
2674 ;. . . . ENDF
2675 ;. . . . UPDATE GOOD ADDRESS (ADD 02)
2676 ;. . . . UPDATE CURRENT ADDRESS
2677 ;. . . ENDDO
2678 ;. . . UPDATE LOW ADDRESS (ADD 02000)
2679 ;. ENDDO
2680 ;ENDDO
2681 ;DISABLE MMU
2682 ;RESTORE VECTOR 114
2683 ;ENDTST
2684
2685 ;R3=GOODAD
2686 ;R5=CURADD
2687
2688 007274 *****
2689 007274 005267 171504 TST21: INC $TESTN ;INCREMENT TEST NUMBER
2690 007300 032777 000400 171540 BIT 081108,BSWR ;IS THIS AN 18 BIT SYSTEM?
2691 007306 001402 BEQ 300$ ;DO THIS TEST IF IT ISN'T
2692 007310 000167 000404 JMP 10$ ;ELSE; JUMP TO END OF TEST
2693 007314 012767 140000 171564 300$: MOV 014000,LOWADD ;INITIALIZE LOW ADDRESS (USE PAR6)
2694 007322 004767 172064 JSR PC, INITHM ;INITIALIZE MMU
2695 007326 012767 000020 163162 MOV 020,SR3 ;ENABLE 22 BIT ADDRESSING
2696 007334 052737 000001 177572 RIS 081100,00SR0 ;ENABLE MMU
2697 007342 012767 177760 171510 MOV 0-16,, LOOPIN ;DO UNTIL ALL BLOCKS TESTED
2698 007350 012737 002000 172354 1$: MOV 02000,00KIPAR6 ;SET KIPAR6 RELOCATION CONSTANT TO PAGE 8
2699 007356 012737 000402 177746 MOV 0402,, 00CCR ;FLUSH CACHE AND SET DIAG BIT
2700 007364 016705 171516 MOV LOWADD, R5 ;GET FIRST ADDRESS IN CURRENT BLOCK

```



```

2757
2758 007576 104000
2759 007600 000073
2760 007602 001325
2761 007604 062705 001000      8$:
2762 007610 005267 171240
2763 007614 002724
2764 007616 062703 000002
2765 007622 042705 017000
2766 007626 062705 000002
2767 007632 062737 000200      172354
2768 007640 005267 171212
2769 007644 002705
2770 007646 062767 001000      171232
2771 007654 005267 171200
2772 007660 002002
2773 007662 000167 177462
2774 007666 005067 167700
2775 007672 016737 171132      000114
2776 007700 005037 177744
2777 007704 005037 177766
2778 007710 005037 001116
2779 007714 000167 000000
2780
2781 007720      10$:
2782
    
```

```

ERROR
.WORD 75
.WORD ERRMSC
ADD #1000, R5
INC DCOUNT
BLT 6$
ADD #2, R3
BIC #17000, R5
ADD #2, R5
ADD #200, #KIPAR6
INC ALLCTR
BLT 5$
ADD #1000, LOWADD
INC LOOPIN
BGE 9$
JMP 1$
CLR SRO
MOV SLOC00, #114
CLR #MSER
CLR #CPEREG
CLR #SOFTER
JMP TST22
    
```

```

; GOT A CACHE HIT WHEN EXPECTING A MISS.
; ALL ERRORS TO TRAP TO EMT VECTOR
; UNIQUE ERROR NUMBER
; ADDRESS OF ERROR MESSAGE
; UPDATE TO NEXT BLOCK
; IF ADDRESS NOT CHECKED FOR EACH BLOCK
; ENDDO
; UPDATE GOOD ADDRESS
; UPDATE ADDRESS TO BE CHECKED

; IF ALL ADDRESSES CHECKED
; ENDDO
; UPDATE TO NEXT BLOCK TO BE ALLOCATED
; IF ALL BLOCKS WERE TESTED
; ENDDO
;
; DISABLE MMU
; RESTORE ABORT VECTOR
; CLEAR THE MEM SYS ERROR REG
; CLEAR THE CPU ERROR REGISTER
; CLEAR THE SOFT ERROR COUNT
; JUMP OVER DATA TABLES OR SUBROUTINES
; TO NEXT TEST
    
```



```

2783          .DSABL  AMA
2784
2785 007720    CTST24:
2786          ;*****
2787          ;*TEST 22      TAG RAM DATA RELIABILITY TEST
2788          ;*****
2789 007720    TST22:
2790 007720    005267 171060          INC      $TESTN          ;INCREMENT TEST NUMBER
2791 007724    032777 001000 171114    BIT      @BIT09,@SWR    ;ARE DATA RELIABILITY TESTS SELECTED?
2792 007732    001002          BNE     1000$          ;IF BIT09 IN SWR = 1 ;THEN DO TEST
2793 007734    000167 000772          JMP     XXX           ;ELSE GO TO END.
2794 007740    032777 000400 171100    1000$:  BIT      @BIT08,@SWR    ;IS THIS AN 18 BIT SYSTEM?
2795 007746    001404          BEQ     1001$          ;IF BIT08 IN SWR=0 BRANCH TO 1001$
2796 007750    012767 177777 171064    MOV     @177777,FLAG   ;SET UP INDICATOR FOR 18 BIT SYS
2797 007756    000402          BR     1002$          ;
2798 007760    005067 171056    1001$: CLR     FLAG          ;SET UP INDICATOR FOR 22 BIT SYSTEM
2799 007764    013767 000114 171036    1002$: MOV     @@114,SLOC00 ;SAVE OLD PARITY ERROR VECTOR
2800 007772    012737 000116 000114    MOV     @116,@@114    ;PUT TRAP CATCHER IN FOR CACHE PARITY ERRORS
2801 010000    005037 000116          CLR     @@116        ;
2802 010004    004767 171402          JSR     PC, INITMM    ;INITIALIZE MMU
2803 010010    042767 100000 162274    BIC     @BIT15,KIPDR5 ;CLEAR CACHE BYPASS ON PDR5
2804 010016    012767 000020 162472    MOV     @20,SR3       ;ENABLE 22 BIT ADDRESSING.
2805 010024    012767 001600 162316    MOV     @1600,KIPAR4  ;SET KIPAR4 TO RELOCATE TO BANK 7 OF MEM.
2806 010032    012701 100000          MOV     @100000,R1    ;SET UP R1 AS POINTER TO BANK 7 VIA KIPAR4
2807 010036    012702 000000          MOV     @0,R2         ;SET UP R2 AS POINTER TO BANK 0
2808 010042    012700 010000          MOV     @4096.,R0    ;SET UP R0 AS COUNTER
2809 010046    012767 000001 167516    MOV     @1,SR0        ;TURN ON MMU
2810 010054    012221    100$: MOV     (R2)+,(R1)+    ;MOVE ALL THE CODE IN FIRST 4KW TO BANK 7
2811 010056    077002          SOB    RC,100$       ;
2812 010060    000137 110064          JMP     @@101$+100000 ;START EXECUTING OUT OF BANK 7
2813          ;
2814          ;WE ARE NOW OPERATING FROM PHYSICAL MEMORY IN BANK 7 (160000-177776)
2815          ;
2816 010064    012701 140000    101$: MOV     @140000,R1  ;SETUP TO INIT CACHE TAG STORE TO ZEROS
2817 010070    005037 172354          CLR     @@KIPAR6     ;SET KIPAR6 TO RELOCATE TO PAGE 0
2818 010074    042737 100000 172300    BIC     @BIT15,@@KIPDR0 ;ALLOW CACHE REFERENCE ON PAGE ZERO
2819 010102    012702 010000          MOV     @10000,R2    ;INITIALIZE 4K WORDS
2820 010106    005721    1$:   TST     (R1)+      ;CLEAR TAG STORE
2821 010110    077202          SOB    R2, 1$       ;LOOP UNTIL ALL OF TAG STORE INITIALIZED
2822          ;
2823 010112    012737 000002 177746    MOV     @BIT101,@CCR  ;SET DIAGNOSTIC BIT IN CCR
2824 010120    012701 110672          MOV     @TAGAD1+100000,R1 ;MAKE R1 A POINTER TO TABLE OF DATA.
2825 010124    012706 120000          MOV     @120000,R6    ;SET UP TEST ADDRESS IN R6
2826          ;USING R6 TO SPEED UP TEST LOOP
2827 010130    012702 140002          MOV     @140002,R2    ;R2 WILL BE USED TO INDEX THRU THE CACHE.
2828          ;START TESTING
2829          ;THE OBJECT OF THIS TEST IS TO TRY EVERY POSSIBLE DATA COMBINATION AT A
2830          ;GIVEN ADDRESS WHILE ALL OTHER ADDRESSES ARE SET TO ALL ZEROS. ONCE A
2831          ;CHANGE IN THE DATA AT A PARTICULAR ADDRESS HAS BEEN MADE; ALL OTHER
2832          ;ADDRESSES ARE CHECKED TO ENSURE THAT DATA HAS NOT BEEN ALTERED. WHEN ALL
2833          ;DATA COMBINATIONS HAVE BEEN CHECKED AT A PARTICULAR ADDRESS; THAT ADDRESS
2834          ;IS RESET TO ZEROS AND THE NEXT CONSECUTIVE ADDRESS IS TESTED. THIS IS DONE
2835          ;UNTIL ALL 4096 LOCATIONS HAVE BEEN TESTED.
2836
2837
2838 010134    012137 172352    FLDO:  MOV     (R1)+,@@KIPAR5 ;MOVE RELOCATION OFFSET (DATA THAT WILL
    
```



GLOBAL AREAS MACY11 30A(1052) 04-APR-84 10:38 PAGE 60  
 KDJ11A.MAC 03-APR-84 11:36 T22 TAG RAM DATA RELIABILITY TEST

SEQ 0060

```

2895 010324 012701 110672      MOV      #TAGAD1+100000,R1      ;RESET DATA POINTER
2896 010330 000167 177600      JMP      FLD0                  ;DO IT AGAIN
2897
2898 010334 012701 140000      FLD1:  MOV      #140000,R1      ;INIT R1
2899 010340 012737 077600 172354  MOV      #77600,#KIPAR6      ;DATA IN KIPAR6 WILL CAUSE TAG RAMS
2900                                     ;TO BE SET TO ALL 1'S.
2901 010346 005767 170470      TST      FLAG                  ;IS THIS AN 18 BIT SYSTEM?
2902 010352 001406 120000      BEQ      100$                 ;BRANCH IF NOT
2903 010354 042737 170000 172354  BIC      #170000,#KIPAR6      ;MODIFY FOR 18 BIT SYSTEM
2904 010362 012737 000002 177746  MOV      #BIT01,#CCR          ;SET DIAGNOSTIC BIT IN CACHE CONTROL REG
2905 010370 012702 010000      100$:  MOV      #4096.,R2          ;SET UP TO FILL 4KW OF CACHE WITH ONES
2906 010374 005021 1$:      CLR      (R1)+                ;WRITE THE CACHE
2907 010376 077202 1$:      SUB      R2,1$                ;
2908 010400 012701 110704      MOV      #TAGAD2+100000,R1    ;SET UP R1 AS A POINTER TO DATA TABLE
2909 010404 012706 120000      MOV      #120000,R6           ;SET UP TESTADDRESS
2910 010410 012702 140002      MOV      #140002,R2           ;SET UP R2
2911 010414 012137 172352      FLD1A: MOV      (R1)+,#KIPAR5     ;DATA IN KIPAR5 WILL BE WRITTEN INTO TAG RAM
2912 010420 005767 170416      TST      FLAG                  ;IS THIS AN 18 BIT SYSTEM?
2913 010424 001403 1$:      BEQ      1$                   ;BRANCHIF NOT
2914 010426 042737 170000 172352  BIC      #170000,#KIPAR5      ;MODIFY FOR 18 BIT SYSTEM
2915 010434 005016 1$:      CLR      (R6)                 ;WRITE TAG RAM WITH TEST DATA
2916 010436 013703 177752      MOV      #HITMIS,R3           ;STORE CONTENTS OF HIS MISS REGISTER
2917 010442 005763 1$:      TST      R3                   ;HIT/MISS REGISTER SHOULD BE CLEAR
2918 010444 001403 1$:      BEQ      FLD1B                ;BRANCH IF IT WAS A MISS
2919 010446 104000 1$:      ERROR                      ;ALL ERRORS TO TRAP TO EMT VECTOR
2920 010450 000100 1$:      .WORD   100                   ;UNIQUE ERROR NUMBER
2921 010452 001325 1$:      .WORD   ERRMSG                ;ADDRESS OF ERROR MESSAGE
2922                                     ;CACHE SYSTEM ERROR
2923 010454 1$:      FLD1B:  TST      (R6)           ;TEST ADDRESS SHOULD NOW BE ALLOCATED IN CACHE
2924 010454 005716 1$:      BIT      #BIT03,#HITMIS      ;DID IT HIT?
2925 010456 032737 000010 177752  BNE      100$                 ;BRANCH IF YES
2926 010464 001003 1$:      ERROR                      ;ALL ERRORS TO TRAP TO EMT VECTOR
2927 010466 104000 1$:      .WORD   101                   ;UNIQUE EPROR NUMBER
2928 010470 000101 1$:      .WORD   ERRMSG                ;ADDRESS OF ERROR MESSAGE
2929 010472 001325 1$:      ERROR                      ;CACHE SYSTEM ERROR
2930                                     ;
2931 010474 012700 010000      100$:  MOV      #4096.,R0           ;INIT R0 TO COUNT 4 KWORDS
2932 010500 012705 160000      MOV      #160000,R5           ;STORE DATA IN R5 FOR FASTER CMP INST.
2933 010504 020502 2$:      CMP      R5,R2                ;IF THE TOP OF CACHE HAS BEEN REACHED
2934 010506 001412 2$:      BEQ      104$                 ;THEN GO RESET R2 TO 140000
2935 010510 005022 101$:  CLR      (R2)+                ;CHECK THE OTHER CACHE LOCATIONS TO
2936                                     ;SEE THAT THEY STILL HIT.
2937 010512 013703 177752      102$:  MOV      #HITMIS,R3           ;SAVE HIT MISS REG IN R3
2938 010516 032703 000004      BIT      #BIT02,R3           ;DID IT HIT?
2939 010522 001407 102$:  BEQ      105$                 ;KEEP TESTING UNTIL A MISS OCCURS
2940 010524 077011 103$:  SOB      R0,2$                ;TEST THE NEXT CACHE LOCATION
2941 010526 104000 103$:  ERROR                      ;ALL ERRORS TO TRAP TO EMT VECTOR
2942 010530 000102 103$:  .WORD   102                   ;UNIQUE ERROR NUMBER
2943 010532 001325 103$:  .WORD   ERRMSG                ;ADDRESS OF ERROR MESSAGE
2944                                     ;CACHE SYSTEM ERROR
2945 103$:  ERROR                      ;ERROR! SHOULD NOT HAVE FALLEN THRU
2946                                     ;SOB INST BEFORE GETTING A MISS
2947 010534 042702 037776      104$:  BIC      #37776,R2           ;RESET R2 TO 140000
2948 010540 000763 104$:  BR       101$                 ;
2949 010542 012704 010512 105$:  MOV      #102$,R4            ;SAVE ADDR OF MOV INSTRUCTION IN R4
2950 010546 010267 170332      MOV      R2,CURADD           ;SAVE R2 IN CURADD

```

```

2951 010552 162767 000002 170324      SUB      #2,CURADD      ;SUBTRACT 2 FROM R2
2952 010560 042767 177701 170316      BIC      #177701,CURADD ;ONLY INTERESTED IN BITS 1-5
2953 010566 042704 177701              BIC      #177701,R4     ;STRIP OFF UN-NEEDED BITS
2954 010572 020467 170306      CMP      R4,CURADD     ;IF BITS 5-11 IN R4 AND CURADD ARE EQUAL?
2955 010576 001004              BNE      3$            ;THEN
2956 010600 032703 000010      BIT      #BIT03,R3     ;TEST BIT03 IN HIT MISS REG
2957 010604 001401              BEQ      3$            ;IF ITS A HIT THEN DO THE NEXT ADDR
2958 010606 000746              BR       103$          ;
2959 010610 010203              3$:     MOV      R2,R3        ;STORE CONTENTS OF R2 IN R3
2960 010612 162703 020002      SUB      #20002,R3     ;SUBTRACT 20002 FROM R3
2961 010616 020306      CMP      R3,R6        ;IS R3= TO THE TEST ADDRESS.
2962 010620 001403      BEQ      4$            ;BRANCH IF THEY ARE EQUAL
2963 010622 104000      ERROR                    ;ALL ERRORS TO TRAP TO EMT VECTOR
2964 010624 000103      .WORD   103           ;UNIQUE ERROR NUMBER
2965 010626 001325      .WORD   ERRMSG        ;ADDRESS OF ERROR MESSAGE
2966                                ;
2967 010630 005711              4$:     TST      (R1)      ;CACHE SYSTEM ERROR
2968 010632 001400      BEQ      6$            ;WAS THIS THE LAST PATTERN FOR THIS LOCATION?
2969                                ;
2970                                ;CHECK TO SEE IF ALL LOCATIONS HAVE BEEN TESTED. IF NOT, GENERATE THE NEW
2971                                ;ADDRESS, RESET THE DATA POINTER AND TEST THE NEXT CACHE LOCATION.
2972                                ;
2973 010634 022706 137776      6$:     CMP      #137776,R6 ;HAVE WE TESTED ALL 4KW CACHE LOCATIONS?
2974 010640 001412      BEQ      8$            ;IF YES THEN GO TO END OF TEST
2975 010642 005062 177776      7$:     CLR      -2(R2)    ;ELSE WRITE ONES IN THE LOCATION LAST TESTED
2976 010646 062706 000002      ADD      #2,R6        ;GET THE NEXT ADDRESS...
2977 010652 062702 000002      ADD      #2,R2        ;BUMP R2
2978 010656 012701 110704      MOV      #TAGAD2+100000,R1 ;RESET DATA POINTER
2979 010662 000167 177526      JMP      FLD1A        ;DO IT AGAIN
2980 010666 000137 010716      8$:     JMP      #0TSEND  ;JUMP OUT OF BANK 7
2981                                ;
2982                                ;
2983 010672 104200      TAGAD1: .WORD   104200
2984 010674 010400      .WORD   010400
2985 010676 121000      .WORD   121000
2986 010700 042000      .WORD   042000
2987 010702 000000      .WORD   000000
2988 010704 073400      TAGAD2: .WORD   073400
2989 010706 167200      .WORD   167200
2990 010710 056600      .WORD   056600
2991 010712 135600      .WORD   135600
2992 010714 000000      .WORD   000000
2993                                ;
2994 010716 005037 177746      TSEND:  CLR      #0CCR   ;CLEAR THE CCR
2995 010722 005037 177572      CLR      #0SR0        ;TURN OFF MMU
2996 010726 012706 001000      MOV      #5TBOT,R6    ;RESTORE STACK POINTER
2997 010732 016737 170072 000114  xxx:    MOV      SLOC00, #0114 ;RESTORE VECTOR
2998 010740 016737 170066 000116      MOV      SLOC01, #0116
2999
3000
3001
    
```

```

3002 .MCALL IDMSG,ENDPAS
3003 .SBTTL END OF PASS ROUTINE
3004
3005 ;*****
3006 ;*INCREMENT THE PASS NUMBER ($PASS)
3007 ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
3008 ;*IF THERES A MONITOR GO TO IT
3009 ;*IF THERE ISN'T JUMP TO RESTART
3010
3011 $EOP:
3012 010746 005767 170034 TST $PASS ;ONLY TYPE MESSAGE AT END OF FIRST PASS
3013 010752 001002 BNE SKIPID ;IF >0 THEN SKIP THE ID MESSAGE
3014 010754 104401 011054 TYPE ,MSG1 ;ELSE TYPE THE ID MESSAGE
3015 010760
3016 010760 005267 170022 SKIPID: INC $PASS ;;INCREMENT THE PASS NUMBER
3017 010764 042767 100000 170014 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
3018 010772 005327 DEC (PC)+ ;;LOOP?
3019 010774 000001 $EOPCT: .WORD 1
3020 010776 003022 BGT $DOAGN ;;YES
3021 011000 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
3022 011002 000001 $ENDCT: .WORD 1
3023 011004 010774 $EOPCT
3024 011006 104401 011135 TYPE ,MSG2
3025 011012 016746 167770 MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
3026 011016 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
3027 011020 104401 011050 TYPE , $ENULL
3028 011024 013700 000042 $GET42: MOV @42,R0 ;;GET MONITOR ADDRESS
3029 011030 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
3030 011032 000005 RESET ;;CLEAR THE WORLD
3031 011034 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
3032 011036 000240 NOP ;;SAVE ROOM
3033 011040 000240 NOP ;;FOR
3034 011042 000240 NOP ;;ACT11
3035 011044
3036 011044 000137 $DOAGN:
3037 011046 002176 JMP @ (PC)+ ;;RETURN
3038 011050 377 377 000 $RTNAD: .WORD RESTART
3039 011054 011054 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
3040 011054 005015 055103 042113 MSG1: .ASCIZ <CR><LF>/CZKDM-B-0 KDJ11 CACHE MEMORY SYSTEM DIAGNOSTIC/
3041 011062 026515 026502 020060
3042 011070 042113 030512 020061
3043 011076 040503 044103 020105
3044 011104 042515 047515 054522
3045 011112 051440 051531 042524
3046 011120 020115 044504 043501
3047 011126 047516 052123 041511
3048 011134 000
3049 011135 015 041412 045532 MSG2: .ASCIZ <CR><LF>/CZKDMB END PASS #/
3050 011142 046504 020102 047105
3051 011150 020104 040520 051523
3052 011156 021440 000
3053 011162 .EVEN
3054 .SBTTL TYPE ROUTINE
3055
3056 ;*****
3057 ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.

```

```

3058 ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
3059 ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
3060 ;*NOTE2: $FILLC CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
3061 ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
3062 ;*
3063 ;*CALL:
3064 ;*1) USING A TRAP INSTRUCTION
3065 ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
3066 ;*OR
3067 ;* TYPE
3068 ;* MESADR
3069 ;*
3070
3071 011162 105767 000343 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
3072 011166 100002 BPL 1$ ;;BR IF YES
3073 011170 000000 HALT ;;HALT HERE IF NO TERMINAL
3074 011172 000430 BR 3$ ;;LEAVE
3075 011174 010046 1$: MOV RO, -(SP) ;;SAVE RO
3076 011176 017600 000002 MOV @2(SP),RO ;;GET ADDRESS OF ASCIZ STRING
3077 011202 122767 000001 167610 CMPB @APTENV,$ENV ;;RUNNING IN APT MODE
3078 011210 001011 BNE 62$ ;;NO,GO CHECK FOR APT CONSOLE
3079 011212 132767 000100 167601 BITB @APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
3080 011220 001405 BEQ 62$ ;;NO,GO CHECK FOR CONSOLE
3081 011222 010067 000004 MOV RO,61$ ;;SETUP MESSAGE ADDRESS FOR APT
3082 011226 004767 001622 JSR PC,$ATY3 ;;SPOOL MESSAGE TO APT
3083 011232 000000 61$: .WORD 0 ;;MESSAGE ADDRESS
3084 011234 132767 000040 167557 62$: BITB @APTCSUP,$ENVM ;;APT CONSOLE SUPPRESSED
3085 011242 001003 BNE 60$ ;;YES,SKIP TYPE OUT
3086 011244 112046 2$: MOVB (RO)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
3087 011246 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
3088 011250 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
3089 011252 012600 60$: MOV (SP)+,RO ;;RESTORE RO
3090 011254 062716 000002 3$: ADD @2,(SP) ;;ADJUST RETURN PC
3091 011260 000002 RTI ;;RETURN
3092 011262 122716 000011 4$: CMPB @HT,(SP) ;;BRANCH IF <HT>
3093 011266 001430 BEQ 8$
3094 011270 122716 000200 CMPB @CRLF,(SP) ;;BRANCH IF NOT <CRLF>
3095 011274 001006 BNE 5$
3096 011276 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
3097 011300 104401 TYPE ;;TYPE A CR AND LF
3098 011302 001403 $CRLF
3099 011304 105067 000202 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
3100 011310 000755 BR 2$ ;;GET NEXT CHARACTER
3101 011312 004767 000056 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
3102 011316 126726 000206 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
3103 011322 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
3104 011324 016746 000176 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
3105 ;;AND THE NULL CHAR.
3106 011330 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
3107 011334 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
3108 011336 004767 000032 JSR PC,$TYPEC ;;GO TYPE A NULL
3109 011342 105367 000144 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
3110 011346 000770 BR 7$ ;;LOOP
3111
3112 ;HORIZONTAL TAB PROCESSOR
3113

```

```

3114 011350 112716 000040      8$:   MOVB   #' ,(SP)      ;;REPLACE TAB WITH SPACE
3115 011354 004767 000014      9$:   JSR    PC,$TYPEC    ;;TYPE A SPACE
3116 011360 132767 000007 000124  BITB   #7,$CHARCNT    ;;BRANCH IF NOT AT
3117 011366 001372          BNE    9$              ;;TAB STOP
3118 011370 005726          TST   (SP)+           ;;POP SPACE OFF STACK
3119 011372 000724          BR     2$              ;;GET NEXT CHARACTER
3120 011374
3121 011374 105777 000116      $TYPEC: TSTB   @TKS           ;;CHAR IN KYBD BUFFER?      ;MJ0001
3122 011400 100022          BPL   10$             ;;BR IF NOT                ;MJ0001
3123 011402 017746 000112      MOV    @TKB,-(SP)     ;;GET CHAR                  ;MJ0001
3124 011406 042716 177600      BIC   #177600,(SP)   ;;STRIP EXTRANEIOUS BITS  ;MJ0001
3125 011412 122716 000023      CMPB  #XOFF,(SP)     ;;WAS CHAR XOFF            ;MJ0001
3126 011416 001012          BNE   102$           ;;BR IF NOT                ;MJ0001
3127 011420
3128 011420 105777 000072      101$: TSTB   @TKS           ;;WAIT FOR CHAR            ;MJ0001
3129 011424 100375          BPL   101$           ;;BR IF NOT                ;MJ0001
3130 011426 117716 000066      MOVB  @TKB,(SP)      ;;GET CHAR                  ;MJ0001
3131 011432 042716 177600      BIC   #177600,(SP)   ;;STRIP IT                  ;MJ0001
3132 011436 122716 000021      CMPB  #XON,(SP)      ;;WAS IT XON?             ;MJ0001
3133 011442 001366          BNE   101$           ;;BR IF NOT                ;MJ0001
3134 011444
3135 011444 005726          102$: TST   (SP)+           ;;FIX STACK                ;MJ0001
3136 011446
3137 011446 105777 000050      10$:  TSTB   @TPS           ;;WAIT UNTIL PRINTER IS READY ;MJ0001
3138 011452 100375          BPL   10$             ;;BR IF NOT                ;MJ0001
3139 011454 116677 000002 000042  MOVB  2(SP),@TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
3140 011462 122766 000015 000002  CMPB  #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
3141 011470 001003          BNE   1$              ;;BRANCH IF NO
3142 011472 105067 000014      CLRB  $CHARCNT       ;;YES -CLEAR CHARACTER COUNT
3143 011476 000406          BR    $TYPEX         ;;EXIT
3144 011500 122766 000012 000002  1$:  CMPB  #LF,2(SP)     ;;IS CHARACTER A LINE FEED?
3145 011506 001402          BNE   $TYPEX         ;;BRANCH IF YES
3146 011510 105227          INCB  (PC)+          ;;COUNT THE CHARACTER
3147 011512 000000          $CHARCNT: .WORD 0    ;;CHARACTER COUNT STORAGE
3148 011514 000207          $TYPEX: RTS         PC
3149
3150 011516 177560          $TKS: .WORD 177560   ;;TTY KDB STATUS          ;MJ0001
3151 011520 177562          $TKB: .WORD 177562   ;;TTY KBD BUFFER         ;MJ0001
3152 011522 177564          $TPS: .WORD 177564   ;;TTY PRINTER STATUS REG. ADDRESS
3153 011524 177566          $TPB: .WORD 177566   ;;TTY PRINTER BUFFER REG. ADDRESS
3154 011526 000          $NULL: .BYTE 0      ;;CONTAINS NULL CHARACTER FOR FILLS
3155 011527 002          $FILLS: .BYTE 2     ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
3156 011530 012          $FILLC: .BYTE 12    ;;INSERT FILL CHARS. AFTER A "LINE FEED"
3157 011531 000          $TPFLG: .BYTE 0     ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
3158 011532 077          $QUES: .ASCII "?"   ;;QUESTION MARK
3159 011533 012 000          $LF: .ASCIZ <12>   ;;LINEFEED
3160 011536
3161 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
3162
3163 ;;*****
3164 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
3165 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
3166 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
3167 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
3168 ;*REPLACED WITH SPACES.
3169 ;*CALL:

```

```

3170          ;*      MOV      NUM, -(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
3171          ;*      TYPDS          ;;GO TO THE ROUTINE
3172
3173          $TYPDS:
3174          011536 010046      MOV      R0, -(SP)          ;;PUSH R0 ON STACK
3175          011540 010146      MOV      R1, -(SP)          ;;PUSH R1 ON STACK
3176          011542 010246      MOV      R2, -(SP)          ;;PUSH R2 ON STACK
3177          011544 010346      MOV      R3, -(SP)          ;;PUSH R3 ON STACK
3178          011546 010546      MOV      R5, -(SP)          ;;PUSH R5 ON STACK
3179          011550 012746 020200      MOV      #20200, -(SP)      ;;SET BLANK SWITCH AND SIGN
3180          011554 016605 000020      MOV      20(SP), R5        ;;GET THE INPUT NUMBER
3181          011560 100004      BPL          1$           ;;BR IF INPUT IS POS.
3182          011562 005405      NEG      R5                ;;MAKE THE BINARY NUMBER POS.
3183          011564 112766 000055 000001      MOVVB   #'-, 1(SP)        ;;MAKE THE ASCII NUMBER NEG.
3184          011572 005000      1$:      CLR      R0                ;;ZERO THE CONSTANT'S INDEX
3185          011574 012703 011752      MOV      #DBLK, R5         ;;SETUP THE OUTPUT POINTER
3186          011600 112723 000040      MOVVB   #' , (R3)+        ;;SET THE FIRST CHARACTER TO A BLANK
3187          011604 005002      2$:      CLR      R2                ;;CLEAR THE BCD NUMBER
3188          011606 016001 011742      MOV      $DTBL(R0), R1     ;;GET THE CONSTANT
3189          011612 160105      3$:      SUB      R1, R5           ;;FORM THIS BCD DIGIT
3190          011614 002402      BLT          4$           ;;BR IF DONE
3191          011616 005202      INC      R2                ;;INCREASE THE BCD DIGIT BY 1
3192          011620 000774      BR          3$
3193          011622 060105      4$:      ADD      R1, R5           ;;ADD BACK THE CONSTANT
3194          011624 005702      TST      R2                ;;CHECK IF BCD DIGIT=0
3195          011626 001002      BNE          5$           ;;FALL THROUGH IF 0
3196          011630 105716      TSTB   (SP)                ;;STILL DOING LEADING 0'S?
3197          011632 100407      BMI          7$           ;;BR IF YES
3198          011634 106316      5$:      ASLH   (SP)                ;;MSD?
3199          011636 103003      BCC          6$           ;;BR IF NO
3200          011640 116663 000001 177777      MOVVB   1(SP), -(R3)      ;;YES--SET THE SIGN
3201          011646 052702 000060      BIS      #'0, R2           ;;MAKE THE BCD DIGIT ASCII
3202          011652 052702 000040      7$:      BIS      #' , R2           ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
3203          011656 110223      MOVVB   R2, (R3)+         ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
3204          011660 005720      TST      (R0)+            ;;JUST INCREMENTING
3205          011662 020027 000010      CMP      R0, #10          ;;CHECK THE TABLE INDEX
3206          011666 002746      BLT          2$           ;;GO DO THE NEXT DIGIT
3207          011670 003002      BGT          8$           ;;GO TO EXIT
3208          011672 010502      MOV      R5, R2           ;;GET THE LSD
3209          011674 000764      BR          6$
3210          011676 105726      8$:      TSTB   (SP)+            ;;GO CHANGE TO ASCII
3211          011700 100003      BMI          9$           ;;WAS THE LSD THE FIRST NON-ZERO?
3212          011702 116663 177777 177776      MOVVB   -1(SP), -(R3)    ;;BR IF NO
3213          011710 105013      9$:      CLRB   (R3)                ;;YES--SET THE SIGN FOR TYPING
3214          011712 012605      MOV      (SP)+, R5        ;;SET THE TERMINATOR
3215          011714 012603      MOV      (SP)+, R5        ;;POP STACK INTO R5
3216          011716 012602      MOV      (SP)+, R3        ;;POP STACK INTO R3
3217          011720 012601      MOV      (SP)+, R2        ;;POP STACK INTO R2
3218          011722 012600      MOV      (SP)+, R1        ;;POP STACK INTO R1
3219          011724 104401 011752      MOV      (SP)+, R0        ;;POP STACK INTO R0
3220          011730 016666 000002 000004      TYPE   , $DBLK           ;;NOW TYPE THE NUMBER
3221          011736 017616      MOV      2(SP), 4(SP)     ;;ADJUST THE STACK
3222          011740 000002      MOV      (SP)+, (SP)
3223          011742 023420      RTI          ;;RETURN TO USER
3224          011744 001750      $DTBL: 10000,
3225          011746 000144      1000,
3226          100,
3227          100.
    
```



```

3226 011750 000012          10,
3227 011752 000004      $DBLK: .BLKW 4
3228                                .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
3229
3230      ;*****
3231      ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
3232      ;OCTAL (ASCII) NUMBER AND TYPE IT.
3233      ;$TYPOS--ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
3234      ;*CALL:
3235      ;*   MOV     NUM, -(SP)          ;NUMBER TO BE TYPED
3236      ;*   TYPOS          ;CALL FOR TYPEOUT
3237      ;*   .BYTE  N          ;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
3238      ;*   .BYTE  M          ;M=1 OR 0
3239      ;*                                   ;1=TYPE LEADING ZEROS
3240      ;*                                   ;0=SUPPRESS LEADING ZEROS
3241      ;*
3242      ;*$TYPON--ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
3243      ;*$TYPOS OR $TYPOC
3244      ;*CALL:
3245      ;*   MOV     NUM, (SP)          ;NUMBER TO BE TYPED
3246      ;*   TYPON          ;CALL FOR TYPEOUT
3247      ;*
3248      ;*$TYPOC--ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
3249      ;*CALL:
3250      ;*   MOV     NUM, -(SP)          ;NUMBER TO BE TYPED
3251      ;*   TYPOC          ;CALL FOR TYPEOUT
3252
3253 011762 017646 000000      $TYPOS: MOV     0(SP), -(SP)          ;PICKUP THE MODE
3254 011766 116667 000001 000211  MOVB  1(SP), $OFILL          ;LOAD ZERO FILL SWITCH
3255 011774 112667 000207      MOVB  (SP)+, $OMODE+1        ;NUMBER OF DIGITS TO TYPE
3256 012000 062716 000002      ADD     02, (SP)          ;ADJUST RETURN ADDRESS
3257 012004 000406      BR     $TYPON
3258 012006 112767 000001 000171 $TYPOC: MOVB  01, $OFILL          ;SET THE ZERO FILL SWITCH
3259 012014 112767 000006 000165  MOVB  06, $OMODE+1        ;SET FOR SIX(6) DIGITS
3260 012022 112767 000005 000154 $TYPON: MOVB  05, $OCNT          ;SET THE ITERATION COUNT
3261 012030 010346      MOV     R3, -(SP)          ;SAVE R3
3262 012032 010446      MOV     R4, -(SP)          ;SAVE R4
3263 012034 010546      MOV     R5, -(SP)          ;SAVE R5
3264 012036 116704 000145      MOVB  $OMODE+1, R4        ;GET THE NUMBER OF DIGITS TO TYPE
3265 012042 005404      NEG     R4
3266 012044 062704 000006      ADD     06, R4          ;SUBTRACT IT FOR MAX. ALLOWED
3267 012050 110467 000132      MOVB  R4, $OMODE        ;SAVE IT FOR USE
3268 012054 116704 000125      MOVB  $OFILL, R4        ;GET THE ZERO FILL SWITCH
3269 012060 016605 000012      MOV     12(SP), R5        ;PICKUP THE INPUT NUMBER
3270 012064 005003      CLR     R3          ;CLEAR THE OUTPUT WORD
3271 012066 006105      1$:   ROTL  R5          ;ROTATE MSB INTO "C"
3272 012070 000404      BR     3$          ;GO DO MSB
3273 012072 006105      2$:   ROTL  R5          ;FORM THIS DIGIT
3274 012074 006105      ROTL  R5
3275 012076 006105      ROTL  R5
3276 012100 010503      MOV     R5, R3
3277 012102 006103      3$:   ROTL  R5          ;GET LSB OF THIS DIGIT
3278 012104 105367 000076      DECH  $OMODE          ;TYPE THIS DIGIT?
3279 012110 100016      BPL  7$          ;IF NO
3280 012112 042705 177770      BIC  017770, R3        ;GET RID OF JUNK
3281 012116 001002      BNE  4$          ;TEST FOR 0

```

```

3282 012120 005704          TST      R4          ;;SUPPRESS THIS 0?
3283 012122 001405          BEQ      5$          ;;BR IF YES
3284 012124 005204          4$: INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
3285 012126 052703 000060    BIS      0'0,R3      ;;MAKE THIS DIGIT ASCII
3286 012132 052703 000040    5$: BIS      0' ,R3      ;;MAKE ASCII IF NOT ALREADY
3287 012136 110367 000040    MOVB     R3,R$       ;;SAVE FOR TYPING
3288 012142 104401 012202    TYPE     ,R$        ;;GO TYPE THIS DIGIT
3289 012146 105367 000032    7$: DECB    $OCNT     ;;COUNT BY 1
3290 012152 003347          BGT      2$          ;;BR IF MORE TO DO
3291 012154 002402          BLT      6$          ;;BR IF DONE
3292 012156 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
3293 012160 000744          BR       2$          ;;GO DO THE LAST DIGIT
3294 012162 012605          6$: MOV      (SP)+,R5  ;;RESTORE R5
3295 012164 012604          MOV      (SP)+,R4  ;;RESTORE R4
3296 012166 012603          MOV      (SP)+,R3  ;;RESTORE R3
3297 012170 0166.S 000002 000004  MOV      2(SP),4(SP) ;;SET THE STACK FOR RETURNING
3298 012176 012616          MOV      (SP)+,(SP)
3299 012200 000002          RTI          ;;RETURN
3300 012202 000          8$: .BYTE   0          ;;STORAGE FOR ASCII DIGIT
3301 012203 000          .BYTE   0          ;;TERMINATOR FOR TYPE ROUTINE
3302 012204 000          $OCNT: .BYTE  0          ;;OCTAL DIGIT COUNTER
3303 012205 000          $OFILL: .BYTE 0          ;;ZERO FILL SWITCH
3304 012206 000000          $OMODE: .WORD  0        ;;NUMBER OF DIGITS TO TYPE
3305          .SBTTL  TTY INPUT ROUTINE
3306
3307          ;;*****
3308          .ENABL  LSB
3309
3310          ;;*****
3311          ;;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
3312          ;;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
3313          ;;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
3314          ;;WHEN OPERATING IN TTY FLAG MODE.
3315 012210 022767 000176 166630  $CKSWR: CMP      0$SWREG,SWR  ;;IS THE SOFT-SWR SELECTED?
3316 012216 001074          GNE      15$          ;;BRANCH IF NO
3317 012220 105777 177272          TSTB     0$TKS          ;;CHAR THERE?
3318 012224 100071          BPL      15$          ;;IF NO, DON'T WAIT AROUND
3319 012226 117746 177266          MOVB     0$TKB,(SP)     ;;SAVE THE CHAR
3320 012232 042716 177600          BIC      0$C177,(SP)   ;;STRIP OFF THE ASCII
3321 012236 022726 000007          CMP      07,(SP)+      ;;IS IT A CONTROL G?
3322 012242 001062          BNE      15$          ;;NO, RETURN TO USER
3323 012244 126727 000514 000001  CMPB     $AUTOB,01      ;;ARE WE RUNNING IN AUTO-MODE?
3324 012252 001456          BEQ      15$          ;;BRANCH IF YES
3325
3326 012254 104401 012735          TYPE     , $CNTLG      ;;ECHO THE CONTROL-G (%G)
3327 012260 104401 012742          1GTSWR: TYPE     , $MSWR  ;;TYPE CURRENT CONTENTS
3328 012264 016746 165706          MOV      SWREG,(SP)   ;;SAVE SWREG FOR TYPEOUT
3329 012270 104402          TYPEOC  ;;GO TYPE --OCTAL ASCII(ALL DIGITS)
3330 012272 104401 012753          TYPE     , $MNEW      ;;PROMPT FOR NEW SWR
3331 012276 005046          19$: CLR      -(SP)     ;;CLEAR COUNTER
3332 012300 005046          CLR      (SP)        ;;THE NEW SWR
3333 012302 105777 177210          7$: TSTB     0$TKS          ;;CHAR THERE?
3334 012306 100375          BPL      7$          ;;IF NOT TRY AGAIN
3335
3336 012310 117746 177204          MOVB     0$TKB,(SP)   ;;PICK UP CHAR
3337 012314 042716 177600          BIC      0$C177,(SP)  ;;MAKE IT 7-BIT ASCII

```

```

3338
3339
3340
3341 012320 021627 000025      9$:    CMP      (SP),025      ;;IS IT A CONTROL-U?
3342 012324 001005              BNE      10$          ;;BRANCH IF NOT
3343 012326 104401 012730      TYPE    ,#CNLU      ;;YES, ECHO CONTROL-U (U)
3344 012332 062706 000006      20$:   ADD      #6,SP      ;;IGNORE PREVIOUS INPUT
3345 012336 000757              BR       19$          ;;LET'S TRY IT AGAIN
3346
3347
3348 012340 021627 000015      10$:   CMP      (SP),015      ;;IS IT A <CR>?
3349 012344 001002              BNE      16$          ;;BRANCH IF NO
3350 012346 005706 000004      TST     4(SP)        ;;YES, IS IT THE FIRST CHAR?
3351 012352 001403              BEQ      11$          ;;BRANCH IF YES
3352 012354 016677 000002 166464  MOV     2(SP),#SWR    ;;SAVE NEW SWR
3353 012362 062706 000006      11$:   ADD      #6,SP      ;;CLEAR UP STACK
3354 012366 104401 001403      14$:   TYPE    ,#CRLF     ;;ECHO <CR> AND <LF>
3355 012372 126727 000367 000001  CMPB   #INTAG,01     ;;RE-ENABLE TTY KBD INTERRUPTS?
3356 012400 001003              BNE      15$          ;;BRANCH IF NOT
3357 012402 012777 000100 177106  MOV     #100,#TKS    ;;RE-ENABLE TTY KBD INTERRUPTS
3358 012410 000002              RTI                      ;;RETURN
3359 012412 004767 176756      16$:   JSR     PC,#TYPEC    ;;ECHO CHAR
3360 012416 021627 000060      CMP     (SP),#60     ;;CHAR < 0?
3361 012422 002420              BLT     19$          ;;BRANCH IF YES
3362 012424 021627 000067      CMP     (SP),#67     ;;CHAR > 7?
3363 012430 003015              BGT     18$          ;;BRANCH IF YES
3364 012432 042726 000060      BIC     #60,(SP)+    ;;STRIP-OFF ASCII
3365 012436 005766 000002      TST     2(SP)        ;;IS THIS THE FIRST CHAR
3366 012442 001403              BEQ     17$          ;;BRANCH IF YES
3367 012444 006316              ASL     (SP)         ;;NO, SHIFT PRESENT
3368 012446 006316              ASL     (SP)         ;; CHAR OVER TO MAKE
3369 012450 006316              ASL     (SP)         ;; ROOM FOR NEW ONE.
3370 012452 005266 000002      17$:   INC     2(SP)        ;;KEEP COUNT OF CHAR
3371 012456 056616 177776      BIS     -2(SP),(SP)  ;;SET IN NEW CHAR
3372 012462 000707              BR      7$           ;;GET THE NEXT ONE
3373 012464 104401 011532      18$:   TYPE    ,#QUES     ;;TYPE ?<CR><LF>
3374 012470 000720              BR      20$          ;;SIMULATE CONTROL-U
3375      .DSABL  LSB
3376
3377
3378
3379      ;;*****
3380      ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
3381      ;*CALL:
3382      ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
3383      ;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
3384      ;*                   ;;WITH PARITY BIT STRIPPED OFF
3385      ;
3386 012472 011646      $RDCHR: MOV     (SP), (SP)    ;;PUSH DOWN THE PC
3387 012474 016666 000004 000002  MOV     4(SP),2(SP)  ;;SAVE THE PS
3388 012502 105777 177010      1$:   TSTB   #TKS        ;;WAIT FOR
3389 012506 100375              BPL     1$           ;;A CHARACTER
3390 012510 117766 177004 000004  MOVB   #TKB,4(SP)   ;;READ THE TTY
3391 012516 042766 177600 000004  BIC   #C<177>,4(SP) ;;GET RID OF JUNK IF ANY
3392 012524 026627 000004 000023  CMP     4(SP),#23   ;;IS IT A CONTROL-S?
3393 012532 001013              BNE     3$           ;;BRANCH IF NO

```

```

3450 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3451 ;*OF THE DESIRED ROUTINE, THEN USING THE ADDRESS OBTAINED IT WILL
3452 ;*GO TO THAT ROUTINE.
3453
3454 012766 010046 ;$TRAP: MOV RO,-(SP) ;;SAVE RO
3455 012770 016600 000002 MOV 2(SP),RO ;;GET TRAP ADDRESS
3456 012774 005740 TST -(RO) ;;BACKUP BY 2
3457 012776 111000 MOV8 (RO),RO ;;GET RIGHT BYTE OF TRAP
3458 013000 006300 ASL RO ;;POSITION FOR INDEXING
3459 013002 016000 013022 MOV $TRPAD(RO),RO ;;INDEX TO TABLE
3460 013006 000200 RTS RO ;;GO TO ROUTINE
3461
3462
3463 ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
3464
3465 013010 011646 ;$TRAP2: MOV (SP),-(SP) ;;MOVE THE PC DOWN
3466 013012 016666 000004 000002 MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
3467 013020 000002 RTI ;;RESTORE THE PSW
3468
3469 .MACRO SETTRAP A,B,MSG
3470 $$SET A,B,\<TRAP+$TRP>,\$TRP,<MSG>
3471 .NLIST
3472 $TRP=$TRP+1
3473 .LIST
3474 .ENDM SETTRAP
3475 .MACRO $$SET A,B,C,D,COMNT
3476 .IF EQ $TRP-1
3477 .SBTTL TRAP TABLE
3478
3479 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3480 ;*BY THE "TRAP" INSTRUCTION.
3481
3482 ; ROUTINE
3483 ; -----
3484 $TRPAD: .WORD $TRAP2
3485 .ENDC
3486 .IIF NDF GNS,.NLIST
3487 A= C
3488 .IIF NDF GNS,.LIST
3489 B ;;CALL=A TRAP+D(C) COMNT
3490 .ENDM $$SET
3491 .MACRO TRMTRP
3492 $TERM=-$TRPAD
3493 .ENDM TRMTRP
3494 .SBTTL TRAP TABLE
3495
3496 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3497 ;*BY THE "TRAP" INSTRUCTION.
3498
3499 ; ROUTINE
3500 ; -----
3501 013022 013010 ;$TRPAD: .WORD $TRAP2
3502 013024 011162 $TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
3503 013026 012006 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3504 013030 011762 $TYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3505 013032 012022 $TYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
    
```

3506	013034	011536			\$TYPDS	;;CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
3507								
3508	013036	012260			\$GTSWR	;;CALL=GTSWR	TRAP+6(104406)	GET SOFT-SWR SETTING
3509								
3510	013040	012210			\$CKSWR	;;CALL=CKSWR	TRAP+7(104407)	TEST FOR CHANGE IN SOFT-SWR
3511	013042	012472			\$RDCHR	;;CALL=RDCHR	TRAP+10(104410)	TTY TYPEIN CHARACTER ROUTINE
3512	013044	012612			\$RDLIN	;;CALL=RDLIN	TRAP+11(104411)	TTY TYPEIN STRING ROUTINE
3513								
3514					.SBTTL			APT COMMUNICATIONS ROUTINE
3515								
3516	013046	112767	000001	000236	***** \$ATY1:	MOVB	\$1,\$FFLG	;;TO REPORT FATAL ERROR
3517	013054	112767	000001	000226	\$ATY3:	MOVB	\$1,\$MFLG	;;TO TYPE A MESSAGE
3518	013062	000403				BR	\$ATYC	
3519	013064	112767	000001	000220	\$ATY4:	MOVB	\$1,\$FFLG	;;TO ONLY REPORT FATAL ERROR
3520	013072				\$ATYC:			
3521	013072	010046				MOV	RO,-(SP)	;;PUSH RO ON STACK
3522	013074	010146				MOV	R1,-(SP)	;;PUSH R1 ON STACK
3523	013076	105767	000206			TSTB	\$MFLG	;;SHOULD TYPE A MESSAGE?
3524	013102	001450				BEQ	5\$	;;IF NOT: BR
3525	013104	122767	000001	165706		CMPB	\$APTENV,\$ENV	;;OPERATING UNDER APT?
3526	013112	001031				BNE	3\$	;;IF NOT: BR
3527	013114	132767	000100	165677		BITB	\$APTPOOL,\$ENVM	;;SHOULD SPOOL MESSAGES?
3528	013122	001425				BEQ	3\$	;;IF NOT: BR
3529	013124	017600	000004			MOV	\$4(SP),RO	;;GET MESSAGE ADDR.
3530	013130	062766	000002	000004		ADD	\$2,4(SP)	;;BUMP RETURN ADDR.
3531	013136	005767	165636		1\$:	TST	\$MSGTYPE	;;SEE IF DONE W/ LAST XMISSION?
3532	013142	001375				BNE	1\$	;;IF NOT: WAIT
3533	013144	010067	165644			MOV	RO,\$MSGAD	;;PUT ADDR IN MAILBOX
3534	013150	105720			2\$:	TSTB	(RO)+	;;FIND END OF MESSAGE
3535	013152	001376				BNE	2\$	
3536	013154	166700	165634			SUB	\$MSGAD,RO	;;SUB START OF MESSAGE
3537	013160	006200				ASR	RO	;;GET MESSAGE LNTH IN WORDS
3538	013162	010067	165630			MOV	RO,\$MSGLGT	;;PUT LENGTH IN MAILBOX
3539	013166	012767	000004	165604		MOV	\$4,\$MSGTYPE	;;TELL APT TO TAKE MSG.
3540	013174	000413				BR	5\$	
3541	013176	017667	000004	000016	3\$:	MOV	\$4(SP),4\$	;;PUT MSG ADDR IN JSR LINKAGE
3542	013204	062766	000002	000004		ADD	\$2,4(SP)	;;BUMP RETURN ADDRESS
3543	013212	016746	164560			MOV	177776,-(SP)	;;PUSH 177776 ON STACK
3544	013216	004767	175740			JSR	PC,\$TYPE	;;CALL TYPE MACRO
3545	013222	000000			4\$:		.WORD	0
3546	013224				5\$:			
3547	013224	105767	000062		10\$:	TSTB	\$FFLG	;;SHOULD REPORT FATAL ERROR?
3548	013230	001416				BEQ	12\$	;;IF NOT: BR
3549	013232	005767	165562			TST	\$ENV	;;RUNNING UNDER APT?
3550	013236	001413				BEQ	12\$	;;IF NOT: BR
3551	013240	005767	165534		11\$:	TST	\$MSGTYPE	;;FINISHED LAST MESSAGE?
3552	013244	001375				BNE	11\$	;;IF NOT: WAIT
3553	013246	017667	000004	165526		MOV	\$4(SP),\$FATAL	;;GET ERROR #
3554	013254	062766	000002	000004		ADD	\$2,4(SP)	;;BUMP RETURN ADDR.
3555	013262	005267	165512			INC	\$MSGTYPE	;;TELL APT TO TAKE ERROR
3556	013266	105067	000020		12\$:	CLRB	\$FFLG	;;CLEAR FATAL FLAG
3557	013272	105067	000013			CLRB	\$LFLG	;;CLEAR LOG FLAG
3558	013276	105067	000006			CLRB	\$MFLG	;;CLEAR MESSAGE FLAG
3559	013302	012601				MOV	(SP)+,R1	;;POP STACK INTO R1
3560	013304	012600				MOV	(SP)+,RO	;;POP STACK INTO RO
3561	013306	000207				RTS	PC	;;RETURN

```

3562 013310      000      $MFLG: .BYTE 0      ;;MESSG. FLAG
3563 013311      000      $LFLG: .BYTE 0      ;;LOG FLAG
3564 013312      000      $FFLG: .BYTE 0      ;;FATAL FLAG
3565          013314      .EVEN
3566          000200      APTSIZE=200
3567          000001      APTENV=001
3568          000100      APTSPOOL=100
3569          000040      APTCSUP=040
3570          ;;*****
3571          ;THIS ROUTINE WILL INCREMENT THE ERROR COUNT AND THEN PASS THE UNIQUE
3572          ;ERROR NUMBER TO THE APT ERROR ROUTINE TO BE REPORTED TO THE APT SYSTEM.
3573
3574 013314 005267 165532      $ERROR: INC      $ERFLG      ;INCREMENT ERROR FLAG
3575 013320 001775          BEQ      $ERROR      ;DON'T LET IT GO TO ZERO
3576 013322 005267 165516      INC      ERRCNT      ;INCREMENT THE ERROR COUNT
3577 013326 021627 001002      CMP      (SP), #1002 ;IS ERROR FROM VECTOR AREA
3578 013332 101010          BHI      1$          ;IF YES THEN
3579 013334 012767 007777 000106      MOV      #7777, 3$   ;REPORT AN UNEXPECTED TRAP
3580 013342 012637 001062      MOV      (SP)+, @SAVSP1 ;SAVE UNEXPECTED TRAP DATA
3581 013346 012637 001064      MOV      (SP)+, @SAVSP2 ;AND RESTORE SP
3582 013352 000430          BR      2$          ;ELSE
3583 013354 017667 000000 000066 1$:      MOV      @SP, 3$     ;REPORT UNIQUE ERROR NUMBER TO APT
3584 013362 011667 000072      MOV      (SP), 101$  ;SAVE ERROR PC
3585 013366 062716 000002          ADD      #2, (SP)    ;GET OVER UNIQUE ERROR NUMBER FOR RETURN
3586 013372 017637 000000 013402 100$:      MOV      @SP, @102$
3587 013400 104401          TYPE          ;TYPE ERROR MESSAGE
3588 013402 000000      102$:      .WORD 0
3589 013404 062716 000002      ADD      #2, (SP)   ;GET OVER ERROR MESSAGE
3590 013410 104401 001352      TYPE      ,ERR1    ;
3591 013414 016746 000030      MOV      3$, -(SP) ;PUSH UNIQUE ERROR NUMBER ON THE STACK
3592 013420 104402          TYPOC          ;TYPE OCTAL ERROR NUMBER
3593 013422 104401 001366      TYPE      ,ERR2    ;
3594 013426 016746 000026      MOV      101$, -(SP) ;PUSH ERROR PC ON THE STACK
3595 013432 104402          TYPOC          ;TYPE THE ERROR PC
3596 013434 122767 000001 165356 2$:      CMPB     @APTENV, $ENV ;CHECK TO MAKE SURE WE'RE IN APT MODE
3597 013442 001004          BNE      5$          ;IF YES THEN
3598 013444 004767 177414      JSR      PC, $ATY4  ;GO REPORT ERROR TO APT
3599 013450 000000      3$:      .WORD 0          ;STORAGE FOR ERROR NUMBER
3600 013452 000777      4$:      BR      4$          ;LOOP HERE AFTER REPORTING ERROR TO APT
3601 013454 000000      5$:      HALT          ;IF NOT APT THEN HALT
3602 013456 000002          RTI          ;ALLOW RECOVERY FROM HALT
3603 013460 000000      101$:      .WORD 0
3604 013462          $PATCH:
3605 013462 000010          .BLKW 10
3606          000001      .END
    
```



















ERRF18	229*	1649
ERRF19	229*	1668
ERRF2	227*	1060
ERRF20	229*	1678
ERRF21	229*	1697
ERRF22	229*	1778
ERRF23	229*	1790
ERRF24	230*	1798
ERRF25	230*	1805
ERRF26	230*	1815
ERRF27	230*	1824
ERRF28	230*	
ERRF29	230*	1837
ERRF3	227*	1065
ERRF30	230*	1893
ERRF31	230*	1900
ERRF32	231*	1913
ERRF33	231*	
ERRF34	231*	
ERRF35	231*	
ERRF36	231*	
ERRF37	231*	1977
ERRF38	231*	1992
ERRF39	231*	2000
ERRF4	227*	1130
ERRF40	232*	2006
ERRF41	232*	
ERRF42	232*	
ERRF43	232*	
ERRF44	232*	
ERRF45	232*	
ERRF46	232*	2078
ERRF47	232*	2084
ERRF48	233*	2182
ERRF49	233*	2200
ERRF5	227*	1138
ERRF50	233*	2223
ERRF51	233*	2240
ERRF52	233*	2251
ERRF53	233*	2260
ERRF54	233*	2369
ERRF55	233*	2404
ERRF56	234*	2558
ERRF57	234*	2570
ERRF58	234*	2745
ERRF59	234*	2754
ERRF6	227*	1198
ERRF60	234*	2847
ERRF61	234*	2856
ERRF62	234*	2871
ERRF63	234*	2834
ERRF64	235*	2919
ERRF65	235*	2927
ERRF66	235*	2941
ERRF67	235*	2963
ERRF7	227*	1205







J7

GLOBAL AREAS MACY11 30A(1052) 04-APR-84 10:38 PAGE 89  
KDJ11A.MAC 03-APR-84 11:36 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0087

\$\$SET	3475*	3494	3503	3504	3505	3506	3508	3510	3511	3512
\$\$SETM	937*	944								
\$\$SKIP	372*									
.EQUAT	237*	262								
.HEADE	239*	244								
.KT11	237*	372								
.SETUP	239*	549								
.\$ACT1	239*	577								
.\$APT8	237*	622								
.\$APTH	239*	587								
.\$APTY	240*	3513								
.\$EOP	237*	3003								
.\$ERRO	240*									
.\$READ	240*	3305								
.\$TRAP	239*	3446								
.\$TYPD	238*	3161								
.\$TYPE	238*	3054								
.\$TYPD	240*	3228								
.\$40CA	237*	550								

. ABS. 013502 000

ERRORS DETECTED: 0

CZKDMB/EN:ABS,CZKDMB.SEQ/DOC/SOL/CRF/NL:TOC=SYSMAC.SML/ML,CZKDMB.MAC/ML,KDJ11A.MAC  
RUN-TIME: 748 109 5 SECONDS  
RUN-TIME RATIO: 1286/864=1.4  
CORE USED: 58K (115 PAGES)

DOCUMENT PAGES: 87

KDJ11A.MAC	03-APR....B1	KDJ11A.MAC	03-APR....B5
KDJ11A.MAC	03-APR....C1	KDJ11A.MAC	03-APR....C5
KDJ11A.MAC	03-APR....D1	KDJ11A.MAC	03-APR....D5
KDJ11A.MAC	03-APR....E1	KDJ11A.MAC	03-APR....E5
KDJ11A.MAC	03-APR....F1	KDJ11A.MAC	03-APR....F5
KDJ11A.MAC	03-APR....G1	KDJ11A.MAC	03-APR....G5
KDJ11A.MAC	03-APR....H1	KDJ11A.MAC	03-APR....H5
KDJ11A.MAC	03-APR....I1	KDJ11A.MAC	03-APR....I5
KDJ11A.MAC	03-APR....J1	KDJ11A.MAC	03-APR....J5
KDJ11A.MAC	03-APR....K1	KDJ11A.MAC	03-APR....K5
KDJ11A.MAC	03-APR....L1	KDJ11A.MAC	03-APR....L5
KDJ11A.MAC	03-APR....M1	KDJ11A.MAC	03-APR....M5
KDJ11A.MAC	03-APR....N1	KDJ11A.MAC	03-APR....N5

KDJ11A.MAC	03-APR....B2	KDJ11A.MAC	03-APR....B6
KDJ11A.MAC	03-APR....C2	KDJ11A.MAC	03-APR....C6
KDJ11A.MAC	03-APR....D2	KDJ11A.MAC	03-APR....D6
KDJ11A.MAC	03-APR....E2	KDJ11A.MAC	03-APR....E6
KDJ11A.MAC	03-APR....F2	KDJ11A.MAC	03-APR....F6
KDJ11A.MAC	03-APR....G2	KDJ11A.MAC	03-APR....G6
KDJ11A.MAC	03-APR....H2	KDJ11A.MAC	03-APR....H6
KDJ11A.MAC	03-APR....I2	KDJ11A.MAC	03-APR....I6
KDJ11A.MAC	03-APR....J2	KDJ11A.MAC	03-APR....J6
KDJ11A.MAC	03-APR....K2	KDJ11A.MAC	03-APR....K6
KDJ11A.MAC	03-APR....L2	KDJ11A.MAC	03-APR....L6
KDJ11A.MAC	03-APR....M2	KDJ11A.MAC	03-APR....M6
KDJ11A.MAC	03-APR....N2	KDJ11A.MAC	03-APR....N6

KDJ11A.MAC	03-APR....B3	KDJ11A.MAC	03-APR....B7
KDJ11A.MAC	03-APR....C3	KDJ11A.MAC	03-APR....C7
KDJ11A.MAC	03-APR....D3	KDJ11A.MAC	03-APR....D7
KDJ11A.MAC	03-APR....E3	KDJ11A.MAC	03-APR....E7
KDJ11A.MAC	03-APR....F3	KDJ11A.MAC	03-APR....F7
KDJ11A.MAC	03-APR....G3	KDJ11A.MAC	03-APR....G7
KDJ11A.MAC	03-APR....H3	KDJ11A.MAC	03-APR....H7
KDJ11A.MAC	03-APR....I3	KDJ11A.MAC	03-APR....I7
KDJ11A.MAC	03-APR....J3	KDJ11A.MAC	03-APR....J7
KDJ11A.MAC	03-APR....K3		
KDJ11A.MAC	03-APR....L3		
KDJ11A.MAC	03-APR....M3		
KDJ11A.MAC	03-APR....N3		

KDJ11A.MAC	03-APR....B4		
KDJ11A.MAC	03-APR....C4		
KDJ11A.MAC	03-APR....D4		
KDJ11A.MAC	03-APR....E4		
KDJ11A.MAC	03-APR....F4		
KDJ11A.MAC	03-APR....G4		
KDJ11A.MAC	03-APR....H4		
KDJ11A.MAC	03-APR....I4		
KDJ11A.MAC	03-APR....J4		
KDJ11A.MAC	03-APR....K4		
KDJ11A.MAC	03-APR....L4		
KDJ11A.MAC	03-APR....M4		
KDJ11A.MAC	03-APR....N4		