IDENTIFICATION

PRODUCT CODE: RC-7883F-MC
PRODUCT NAME: CCMFAFO MSII, MFII, MAII-P MEM
DATE RELEASED: FEB 1978
MAINTAINER: DIAGNOSTIC GROUP

COPYRIGHT (C) 1972, 1978
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY
ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH
THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS
SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED
OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT
FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE
LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE
SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMIT-
MENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DEC.
1.0 ABSTRACT

THIS PROGRAM LOCATES THE PARITY MEMORY Registers FOR BOTH THE
CORE AND MOS PARITY Memories AND PREFORMS A CHECK OF THE BITS IN EACH.
IT THEN CREATES A MAP SHOWING THE MEMORY CONTROLLED BY EACH PARITY
REGISTER. THE PARITY Registers AND THE MEMORY ARE THEN TESTED USING
THE INFORMATION IN THE MAP.

2.0 REQUIREMENTS

2.1 EQUIPMENT

PDP-11 WITH MF11-LP OR MA11-P PARITY MEMORY (CORE), MS-11 (MOS) PARITY MEMORY

2.2 STORAGE

THE PROGRAM REQUIRES 4K OF MEMORY TO LOAD AND 8K TO RUN.

3.0 LOADING PROCEDURE

LOAD PROGRAM INTO MEMORY USING ABS LOADER OR XXD.

4.0 STARTING PROCEDURE

4.1 STARTING ADDRESSES

200= NORMAL (WORST CASE) TESTING
220= ROUTINE TO SCAN FOR BAD PARITY
230= RESTART OF NORMAL TESTING - USES PREVIOUS MAP OF PARITY MEMORY

4.2.1 PROGRAM AND/OR OPERATOR ACTION

LOAD STARTING ADDRESS.
SET DESIRED SWITCH REGISTER SETTINGS (SEE 5.1 - ALL DOWN FOR WORST CASE).
PRESS START.
IF SA 200 OR RESTART ADDRESS 230 IS USED, THE BELL WILL RING AT THE
COMPLETION OF EACH PASS AND END PASS= XXX WILL BE TYPED (WHERE XXX
IS THE NUMBER OF PASSES COMPLETED SINCE THE PROGRAM WAS LAST STARTED).

5.0 OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

THE DIAGNOSTIC IS DESIGNED TO USE HARDWARE SW FOR SYSTEMS HAVING THIS
REGISTER, HOWEVER FOR SYSTEMS NOT HAVING HARDWARE SWITCH REGISTER IT
WILL USE LOCATION 176 TO GIVE THE FOLLOWING OPTIONS:

SW 15=1 OR UP -- HALT ON ERROR
SW 14=1 OR UP -- SCOPE LOOP
SW 13=1 OR UP -- INHIBIT PRINTOUT
SW 11=1 OR UP -- INHIBIT ITERATIONS
SW 10=1 OR UP -- HALT AFTER LOCATING BAD PARITY BEFORE CORRECTING IT
(USED IN PARITY SCAN ROUTINE ONLY)
SW 09=1 OR UP -- HALT AFTER THE PARITY MEMORY MAP HAS BEEN PRINTED
(ALLOWS MANUAL CHANGES TO FORCE TESTING OF MEMORY)
5.2 SUBROUTINE ABSTRACTS

5.2.1 BEGIN SA 200, RESTART 230

5.2.2 SCOPE

THIS SUBROUTINE CALL IS PLACED BETWEEN EACH SUBTEST IN THE INSTRUCTION SECTION. IT RECORDS THE STARTING ADDRESS OF EACH SUBTEST AS IT IS ENTERED. IF A SCOPE LOOP IS REQUESTED, IT WILL JUMP TO THE START OF THE SUBTEST THAT THE SCOPE LOOP IS REQUESTED FOR. IF SCOPE LOOP IS NOT REQUESTED, THERE WILL BE 64 ITERATIONS OF THAT SUBTEST BEFORE THE NEXT SUBTEST IS ENTERED (EXCEPT IN THOSE ROUTINES WHERE IMAX IS CHANGED). SWITCH II ON A ONE INHIBITS ITERATION OF SUBTESTS.

5.2.3 ERROR HANDLERS ERRST, ERRR, ERR

THESE ROUTINES ARE CALLED VIA EMTS TO PRINT OUT ERROR INFORMATION. (SEE 6.0 FOR DESCRIPTION OF ERROR INFORMATION)

5.2.4 PSCAN (SCAN MEMORY FOR BAD PARITY)

THIS ROUTINE READS ALL LOCATIONS IN MEMORY AND PRINTS OUT THE PHYSICAL ADDRESS (16 BITS) OF THOSE LOCATIONS CONTAINING BAD PARITY. IT IS UTILIZED WITHIN THE PROGRAM WHILE EXERCISING MEMORY IF A PARITY ERROR OCCURS UNEXPECTEDLY, AND MAY ALSO BE CALLED USING STARTING ADDRESS 220.

5.2.5 $TYPE (ASCII MESSAGE TYPEOUT ROUTINE)

THIS IS THE STANDARD TYPEOUT ROUTINE, ALLOWING PATCHING TO UTILIZE OUTPUT DEVICES OTHER THAN THE KSR 33. $NULL CONTAINS THE VALUE TO BE USED AS A FILLER CHARACTER, AND $FILLS CONTAINS A NUMBER INDICATING THE NUMBER OF FILLER CHARACTERS REQUIRED. $PS AND $TPB CONTAIN THE STATUS AND BUFFER REGISTER ADDRESSES OF THE OUTPUT DEVICE.

5.2.6 TRAPCATCHER

THIS IS A SERIES OF INSTRUCTIONS STARTING AT LOCATION U DESIGNED TO DETECT AND ISOLATE UNEXPECTED TRAPS AND INTERRUPTS TO THE TRAP AND INTERRUPT VECTOR AREA OF MEMORY.

EACH VECTOR ENTRANCE ADDRESS IS LOADED WITH THE ADDRESS OF THE NEXT LOCATION. THE NEXT LOCATION IS LOADED WITH A HALT (000000). THUS AN ILLEGAL TRAP OR INTERRUPT WILL CAUSE A HALT AT THE TRAP LOCATION PLUS TWO.

IF A HALT OCCURS IN THE TRAP OR INTERRUPT AREA, EXAMINE REGISTER SIX. IT WILL CONTAIN THE CURRENT STACK ADDRESS. THE CONTENTS OF THE CURRENT STACK ADDRESS IS THE VALUE OF THE LOCATION COUNTER WHEN THE TRAP
5.3 PROGRAM AND/OR OPERATOR ACTION

5.3.1 altering the parity memory map

IF THE MAP TYPED AT RUN TIME DOES NOT AGREE WITH THE HARDWARE PRESENT
THE MAP CAN MANUALLY BE CHANGED TO ALLOW TESTING OF PARITY MEMORY.
THAT THE MAPPER DID NOT FIND SETTING SWITCH 9 TO A 1 WILL CAUSE THE
PROGRAM TO HALT. AFTER THE HALT IS MODIFIED THE MAP
AS DESIRED (SEE THE DESCRIPTION IN THE LISTING - THE MAP BEGINS
AT LOCATION 600). THEN PRESS CONTINUE. THE NEW MAP WILL BE
PRINTED, AND IF SW9 IS STILL SET THE PROCESS WILL BE REPEATED. IF
SW9 IS NOT SET, THE PROGRAM WILL TEST PARITY MEMORY USING THE
NEW MAP.

5.3.2 STOPPING THE PROGRAM

BECAUSE THE PROGRAM RELOCATES ITSELF TO BANK 1 WHILE TESTING BANK
O, A SWITCH IS PROVIDED TO HALT THE PROGRAM AT THE END OF A PASS.
SETTING THIS SWITCH (SW9) WILL CAUSE THE PROGRAM TO HALT IN BANK
O AT THE END OF THE CURRENT PASS (AFTER OUTPUTING THE END OF PASS
MESSAGE).

6.0 ERRORS

6.1 ERROR PRINTOUTS

THERE ARE THREE TYPES OF ERROR MESSAGES USI'G COMBINATIONS OF THE
FOLLOWING ERROR TYPE ROUTINES.

PC=ZZZZZZZ PC OF FAILING ERROR CALL. REFER TO
THIS ADDRESS IN THE LISTING FOR AN
EXPLANATION OF THE ERROR.

INT=YYYYYY CURRENT ITERATION COUNT OF FAILING TEST.

MPR=XXXXXXX ADDRESS OF PARITY REGISTER UNDER TEST.

MPR DATA=VVVVVV CONTENTS OF PARITY REGISTER UNDER TEST.

TEST LOC=XXXXXX MEMORY LOCATION UNDER TEST

S/B: XXXXXX CONTENTS OF MEMORY LOCATION SHOULD BE.

W BS: XXXXXX CONTENTS OF MEMORY LOCATION WAS.

6.2 DETERMINING ADDRESS OF TEST LOCATION WHEN KT11 IS PRESENT

IN MOST OF THE SUBTESTS, IF A KT11 IS PRESENT IT IS USED. IN ALL
CASES IN THIS PROGRAM, WHEN THE KT11 IS ON, KERNEL PAGE O IS USED
TO REFERENCE BANK O AND KERNEL PAGE 7 IS USED TO REFERENCE THE
EXTERNAL BANK. IN MOST CASES KERNEL PAGE 1 IS USED TO REFERENCE
THE MEMORY CURRENTLY UNDER TEST AND THE USE OF THE MEMORY MANAGEMENT
OPTION IS SIMILAR THROUGHOUT THE PROGRAM AND IT IS EASY TO DETERMINE
THE ACTUAL (PHYSICAL) MEMORY ADDRESS BEING TESTED.

TO CALCULATE A PHYSICAL ADDRESS, ADD THE STARTING ADDRESS OF THE
BANK BEING TESTED TO THE OFFSET WHICH GIVES THE ADDRESS WITHIN THE
BANK. SINCE IN THIS PROGRAM ALL RELOCATED MEMORY TESTING IS DONE
THROUGH KERNEL PAGE 1, KERNEL PAGE ADDRESS REGISTER 1 (ADDRESS 777392).


6.3 ERROR RECOVERY

IN GENERAL, TEST FAILURES WILL PRINTOUT AN ERROR MESSAGE AND CONTINUE. IF THE HALT ON ERROR SWITCH IS SET, HITTING CONTINUE WILL RECOVER. IF THE PROGRAM HANGS UP IN A LOOP, THE ERROR IS LIKELY TO BE A SIGNAL WHICH WAS NEVER RECEIVED. IF A HALT OCCURS IN THE TRAP AND VECTOR AREA THE PROGRAM MUST BE RESTARTED. IF THE PROGRAM HALTS IN THE MAIN FLOW, CONSULT THE LISTING IF NO MESSAGE IS TYPED OUT.

6.4 ERRORS WHILE TESTING BANK ZERO (ERROR PC VALUES ABOVE 20000)

TEST 20 AND TEST 21 CHECK BANK 0 IF IT HAS PARITY MEMORY. TO DO THIS, THE CODE IS RELOCATED TO AND EXECUTED FROM BANK 1. THE ERROR PRINTOUTS WILL THEN GIVE THE PC IN BANK 1 OF THE ERROR CALL. SINCE ALL LOCATIONS HAVE BEEN MOVED UP 20000, SUBTRACT 20000 FROM THE ERROR PC TO GET THE ADDRESS IN THE LISTING WHICH CORRESPONDS TO THE PRINTOUT.

7.0 RESTRICTIONS

THE PROGRAM REQUIRES A MINIMUM OF BK MEMORY TO RUN. XXDP CHAINING IS POSSIBLE FOR SYSTEMS GREATER THAN BK.

7.1 STARTING PROCEDURE

PROGRAM MUST BE LOADED INTO LOWER 4K OF MEMORY.

7.2 OPERATING RESTRICTION- AVOID USING THE "HALT" SWITCH

IF THE PROGRAM IS HALTED AT A RANDOM POINT DURING EXECUTION, SEVERAL PROBLEMS MAY ARISE. THE PROGRAM MAY BE RELOCATED TO BANK 1 AT THE TIME IT IS STOPPED, IN WHICH CASE NONE OF THE STANDARD STARTING ADDRESSES WILL WORK. WRITE WRONG PARITY MAY BE SET, IN WHICH CASE
YOU MAY ENTER BAD PARITY WHILE PATCHING AND MEMORY MAY CONTAIN BAD
PARITY SINCE YOU MAY BE IN THE MIDDLE OF A TEST WHICH UTILIZES
WRITE WRONG PARITY. IT IS THEREFORE STRONGLY RECOMMENDED THAT YOU
HALT THE PROGRAM VIA THE "HALT AT END OF PASS" SWITCH (SW8) OR THE
"HALT ON ERROR" SWITCH (SW15) RATHER THAN VIA THE HALT/ENABLE SWITCH.

8.0 MISCELLANEOUS

8.1 EXECUTION TIME

EXECUTION TIME DEPENDS ON THE AMOUNT OF PARITY MEMORY UNDER TEST.
IT TAKES ABOUT 1 MINUTE TO TEST 24K OF PARITY MEMORY (1 PASS).

8.2 STACK POINTERS

THE KERNEL STACK POINTER IS INITIALIZED TO 510.

9.0 PROGRAM DESCRIPTION

THIS PROGRAM FIRST LOCATES MA11 & MF11 CORE PARITY AND MS-11 MOS PARITY CONTROL
REGISTERS BY ADDRESSING EACH POSSIBLE REGISTER ADDRESS AND CHECKING THOSE WHICH
DO NOT TIME OUT, ON DETECTING THE PRESENCE OF A PARITY REGISTER
THE PROGRAM CHECKS IF IT IS A CORE PARITY OR A MOS PARITY REGISTER
AND ACCORDINGLY STORES THIS INFORMATION IN AN INDICATOR (INDC-INDC15)
THE ADDRESSES OF THE REGISTERS ARE RECORDED
AND OUTPUT TO THE CONSOLE DEVICE, AND THEN THE REGISTERS ARE
CHECKED TO SEE THAT THE CORRECT BITS ARE R/W. RESET IS USED TO TEST
THE EFFECT OF INIT, PARITY MEMORY IS THEN LOCATED BY SETTING WRITE
WRONG PARITY IN ALL REGISTERS AND WAITING AND READING THE FIRST 4
ADDRESSES IN EACH 4K. EACH TIME A PARITY REGISTER RECORDS A PARITY
ERROR, THE MAP IS ALTERED TO INDICATE THAT REGISTER
CONTROLS THE MEMORY BEING ADDRESSED. THE FINAL MAP IS PRINTED AND
THEN THE PARITY CONTROL LOGIC IS CHECKED USING THE PARITY MEMORY
FOUND. SEVERAL PATTERNS ARE WRITTEN INTO EACH PARITY MEMORY
LOCATION TO SEE THAT NO PARITY ERRORS ARE CREATED. FINALLY, EACH
BYTE OF PARITY MEMORY IS WRITTEN WITH BOTH GOOD AND BAD PARITY TO
SHOW THAT THE PARITY BITS CAN BE TOGGLED AND SENSED.
SINCE THIS IS A COMBINED DIAGNOSTIC, AS FAR AS POSSIBLE
COMMON TESTS ARE USED FOR BOTH CORE AND MOS. ONLY WHERE THE
MOS CONTROLLER DEFERS FUNCTIONALLY FROM THE CORE, THE
INDICATOR IS CHECKED FOR MOS OR CORE AND THE MEMORY
IN QUESTION IS TESTED ACCORDINGLY.
A DETAILED EXPLANATION OF THE MAP IS GIVEN IN THE LISTING
(PAGE 9-12).
THE DISPLAY REGISTER CONTAINS THE NUMBER OF THE TEST BEING
EXECUTED.
MEMORY PARITY TEST

COPYRIGHT 1973, 1977, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

AUTHOR: JIM KAPADIA

; REGISTER SWITCH OPTIONS (SWITCH SET TO A 1)
; SR16 - HALT ON ERROR
; SCPE - SCOPE
; SR13 - INHIBIT PRINTOUT
; SR11 - INHIBIT IERATIONS
; SR10 - HALT AFTER LOCATING BAD PARITY BEFORE CORRECTING IT
; SR9 - HALT AFTER TYPING PARITY MEMORY MAP (ALLOWS MANUAL
; CHANGES TO BE MADE TO THE MAP TO FORCE TESTING OF
; MEMORY THAT WAS NOT LOCATED.
; SR8 - HALT AT END OF PASS (IF HALTED ELSEWHERE THE PROGRAM
; MAY BE RELOCATED TO BANK 1 WRITE WRONG PARITY MAY BE SET.
; AND/OR BAD PARITY MAY EXIST IN THE PARITY MEMORY).

; SYMBOL DEFINITIONS

;BIT DEFINITIONS

; ACTION ENABLE
; WRITE WRONG PARITY
; ADDRESS OF ERROR
; PARITY ERROR BIT
; HARDWARE SWITCH REGISTER
; HARDWARE DISPLAY REGISTER
;MACRO DEFINITIONS

;TRAPCATCHER (.42,HALT) LOADED INTO LOCATIONS 000-576

;LOAD EMT VECTOR

;LOAD STARTING ADDRESS AREA

;GENERAL DATA AREA

;TITLE PRINTED = 1

;MAPPING - ADDRESS POINTER
;MAPPING - BIT POINTER INDICATING BANK
;MAPPING - TRANSITION FLAG
;MAPPING - TYPED FLAG
;MAPPING - K CORE ACCUMULATOR
;USED TO CHECK WHEN DONE TESTING A BANK
;LOADED WITH ADDRESS OF LOCATION UNDER TEST IN SOME SUBTESTS
MEMORY PARITY CONTROL REGISTER ADDRESSES

THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1).

IF THE CONTROL IS FOUND NOT TO BE PRESENT, THE MEMORY PRESENT UNDER
CONTROL OF EACH CONTROLLER IS REPRESENTED BY 2 OCTAL WORDS. EACH BIT
REPRESENTS A 4K BLOCK. I.E., BIT 0 = 0-4K, BIT 1 = 4-8K, BIT 2 = 8-16K.

THE LOW BYTE OF THE LAST WORD FOR EACH REGISTER INDICATES THE OFFSET (0, 2, 4, OR 6)
FOR THE FIRST ADDRESS THAT ACTUALLY CORRESPONDED TO THE REGISTER. THE HIGH BYTE GETS
SET TO 1 TO INDICATE THAT A MEMORY ADDRESS HAS BEEN FOUND FOR THAT REGISTER.

FOR EXAMPLE, SAY THAT MPRO AND MPRI EXIST, CONTROLLING INTERLEAVED MEMORY
FROM 0 TO 16K, AND THAT MPRO CONTROLS THE ADDRESSES ENDING IN 0 AND 4.

THE MAP WOULD THEN LOOK AS FOLLOWS:

MPRO: 172100

<Registers control 16K (4 banks)>

LOW BYTE SHOWS THAT FIRST ADDRESS
ENDS IN 0 (OCTAL)

HIGH BYTE CONTAINS A 1 TO INDICATE
THAT AN ADDRESS WAS FOUND

BIT 0 IS CLEAR SINCE REGISTER IS PRESENT

MPRO CONTROLS 16K

MPRI: 172102

<Registers control 16K>

LOW BYTE INDICATES THAT THE FIRST
MEMORY ADDRESS ENDS IN 2 (OCTAL)

HIGH BYTE CONTAINS A 1 TO INDICATE
THAT AN ADDRESS WAS FOUND

THE REST OF THE MAP WOULD APPEAR AS IN THE LISTING:

MPRO: 172100+1

<Parity status registers>

0-64K parity mem under this control

MPRI: 172102+1

<Parity status registers>

0-64K parity mem under this control

MPR2: 172104+1

<Parity status registers>

0-64K parity mem under this control
K01

CMFAF MEMORY PARITY TEST

CMFAF 411 13-JAN-78 12:13

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR3: 17210b+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR4: 172110+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR5: 172112+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR6: 172114+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR7: 172116+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR8: 172120+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR9: 172122+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR10: 172124+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR11: 172126+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR12: 172130+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR13: 172132+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR14: 172134+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

MPR15: 172136+1

ADDRESS RESPONSE THIS CONTROL (0, 2, 4, 6)

TREG: 0

PARITY REGISTER UNDER TEST

SEQ 0010
INDICATORS FOR CORE OR MOS PARITY REGISTER:

- EACH INDICATOR REFERS TO A PARTICULAR PARITY registers. IF IT IS
  A CORE PARITY REGISTER THEN A '1' IS STORED IN THE INDICATOR.
- IF IT IS A MOS PARITY REGISTER THEN '1' IS STORED.
- EX: IF MP20 (172100) IS FOR CORE AND MP11 (172102) IS FOR MOS
  THEN THE INDICATOR MAP WILL LOOK AS FOLLOWING:

  INDC0: 000001
  INDC1: 177777

Core-Mos Parity Indicator for MP6
Core-Mos Parity Indicator for MP1
Core-Mos Parity Indicator for MP2
Core-Mos Parity Indicator for MP3
Core-Mos Parity Indicator for MP4
Core-Mos Parity Indicator for MP5
Core-Mos Parity Indicator for MP6
Core-Mos Parity Indicator for MP7
Core-Mos Parity Indicator for MP8
Core-Mos Parity Indicator for MP9
Core-Mos Parity Indicator for MP10
Core-Mos Parity Indicator for MP11
Core-Mos Parity Indicator for MP12
Core-Mos Parity Indicator for MP13
Core-Mos Parity Indicator for MP14
Core-Mos Parity Indicator for MP15

Bit Positions which are reserved
for future use in parity registers
Core Parity
Mos Parity

Parity Patterns:

ParPat: 125325
026126
072527
102070
072527
177777
107030
015626
000000
0

This is a map of the total memory present in the system:

Mem: 0

0-64K Mem present in 4K contiguous blocks
64-124 Mem present in 4K contiguous blocks

0-64K Parity Memory present
MO1

CMFAFO, MEMORY PARITY TEST          MACYII 30A(1052) 13-JAN-78 12:27 PAGE 12
CMFAP11 13-JAN-78 12:13

PMEMH: 0
PMEMX: 0

(IN 4K CONTIGUOUS BLOCKS)
64-124K PARITY MEMORY PRESENT
(IN 4K CONTIGUOUS BLOCKS)
4MB TO HOLD CONTENTS OF EITHER LOW OR HIGH MAP

ROUTINE TO TYPE ASCII MESSAGES. MESSAGE MUST TERMINATE WITH A 0 BYTE.
THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
NOTE: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.

$TYPE: TSB $TPFLG
$BEQ: B $6S

ADDRESS OF THE SWITCH REGISTER
ADDRESS OF THE DISPLAY REGISTER
PRINTER STATUS REGISTER ADDRESS
PRINTER BUFFER REGISTER ADDRESS
Contains null character for fills
Contains # of filler characters required
"TERMINAL AVAILABLE" FLAG (0=YES)
RESERVED

6S: MOVF $RO,-($SP)
6S: MOV $RO,-($SP)

IS THERE A TERMINAL?
BR IF YES
HALT HERE IF NO TERMINAL
SAVE RO
GET ADDRESS OF ASCII STRING
PUSH CHARACTER TO BE TYPED ONTO STACK
BR IF IT ISN'T THE TERMINATOR
IF TERMINATOR POP IT OFF THE STACK
RESTORE RO
JUST RETURN PC
RETURN

4S: DECB 1($SP)
4S: BR 4S
4S: BR 4S

GO TYPE THE CHARACTER
CHECK IF THE CHARACTER TYPED
WAS A LINE FEED
GO GET NEXT CHARACTER IF NOT LINE FEED
GET # OF FILLER CHARACTERS NEEDED
AND THE NULL CHARACTER
DOES A NULL NEED TO BE TYPED?
BR IF NO--GO POP THE NULL OF THE STACK
GO TYPE A NULL

5S: TSB $TPS
5S: BR 5S
5S: BR 5S

LOAD CHARACTER TO BE
_TYPED INTO DATA REGISTER

;GENERAL DATA AREA

;SCNLFQ: 0
;SCNLFQ GETS SET IF USING
;SCAN ROUTINE (SA=220)
NO1

CACHEFLG: 0
CACHE: 177746
KSTART: 0
ADRTYP: 0
PDRTAB: 177600

;KERNEL PAGE DESCRIPTOR REGISTER ADDRESSES

;KERNEL PAGE ADDRESS REGISTER ADDRESSES

;REGISTER SAVE LOCATIONS

;ROUTINE TO RESTART WITHOUT RETYPING MAP AFTER TEST HAS BEEN RUNNING

;SETUP STACK POINTER

;SET FLAG TO INDICATE MAP HAS BEEN TYPED

;INITIALIZE PASS COUNT

;CLEAR PROCESSOR STATUS REGISTER

;ROUTINE TO SCAN ALL MEMORY FOR BAD PARITY AND TYPE 18 BIT ADDRESSES OF BAD

LOCATIONS

;SETUP STACK POINTER

;IF TITLE HAS BEEN PRINTED, REGISTERS

;HAVE ALREADY BEEN LOCATED- GO

;AND LOCATE IF NOT ALREADY DONE

;BRANCH, REGISTERS HAVE ALREADY BEEN LOCAT\_D

;GO TO LOCATE THE REGISTERS

;RETURN HERE AFTER LOCATING THE REGISTERS

;SETUP MEMORY MAP

;SCAN FOR BAD PARITY

;TYPE MESSAGE "BAD PARITY SCAN COMPLETE"
NORMAL STARTUP

START: MOV #STKPT,SP
        CLR MTYFG
        CLR PSCNT
        MOV #PRDN, #24
        MOV #340, #26
        MOV #344, - (SP)
        MOV #166, (SP)
        MOV #25, #24
        TST SWR
        BR 48
        MOV #SWREG SWR
        MOV #DISPREG, DISPLAY
        CMP (SP) + (SP) +
        MOV #254, #24
        MOV #340, #26
        MOV #166, #24
        MOV #344, 177746
        BIS #14, 3177746
        BIS #200, CACHFL
        BR 59
        MOV (SP) +, (SP) +
        CMP (SP) + (SP) +
        MOV #167, (SP) +
        MOV #254, #24
        MOV #340, #26
        MOV #166, #24
        MOV #344, 177746
        BIS #14, 3177746
        BIS #200, CACHFL
        BR 59
        MOV (SP) +, (SP) +
        CMP (SP) + (SP) +
        MOV #167, (SP) +
        MOV #254, #24
        MOV #340, #26
        MOV #166, #24
        MOV #344, 177746
        BIS #14, 3177746
        BIS #200, CACHFL
        BR 59

SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
FAILURE TO LOCATE A REGISTER INDICATES THAT THE ADDRESS TIMED OUT OR THAT
BITS 5-7 IN THE REGISTER DID NOT SET
START: TYPE

; CLEAR MPAP REGISTER PRESENT ARE:
CLR MPAPK
CLR MPARO
CLR MPARO R2
MOV #INDC0, #23
MOV #MPARO, #24
CLR #26
BIC #1(2)
CLR 2(2)
CLR 4(2)
CLEAR FLAG BIT IN TABLE
INITIALIZE LOCATIONS IN THE TABLE
DOES THIS MPR EXIST? (IF NO TIMES OUT)
YES- IS IT AN MII-L-P OR MII-P CORE PARITY REG
NO- IS IT A MOS-11 PARITY REGISTER? BRANCH
YES- PRINT REGISTER ADDRESS
GET ASCII)

; (TYPE ADDRESS)
SET INDICATOR FOR CORE PARITY
IT IS A MOS REGISTER, PRINT ADDRESS
GET ASCII)

; (TYPE ADDRESS)
SET INDICATOR FOR MOS PARITY
SET MPR REGISTER PRESENT FLAG
SKIP MII
RESTORE STACK POINTER
SET -LAG INDICATING REGISTER NOT PRESENT
UPDATE POINTERS

DONE YET?
NO LOOP
YES, RESTORE TRAPCATCHER
ARE YOU IN THE ROUTINE TO SCAN MEMORY FOR BAD PARITY-(SCAN)
NO BRANCH TO CARRY ON NORMALLY
YES, GO BACK TO THE MEMORY
SCAN ROUTINE
ANY PARITY REGISTERS PRESENT?
YES- GO TEST CONTROLS PRESENT
NO- TYPE "NO PARITY REGISTER FOUND"

LOADED BY MONITOR?
NO, BRANCH
YES- EXIT, NO REGISTERS PRESENT
NO REGISTERS TO TEST
IF CONTINUED, TRY AGAIN

SETUP SCOPE RETURN
Cache?
BRANCH IF NO
DISABLE CACHE
SAVE MONITOR
MAXIMUM ITERATION COUNT
RESTORE TRAPCATCHER IN TIMEOUT VECTOR
;**********************************************************************
;SHOW THAT BIT 0, 2, 5, 11, AND 15 OF EACH PARITY REGISTER CAN BE SET
;AND CLEARED. Bits 0, 2, 15 OF EACH MOS PARITY REGISTER
;PRESENT CAN BE SET AND CLEARED
;**********************************************************************

TEST: SCOPE

1$: BIT $1, $2
    BNE 4$; ; NO- BRANCH TO GET NEXT ADDRESS
    MOV $3, R1
    CMP $1, (R4)
    ; NO- BRANCH TO GET NEXT ADDRESS
    BNE 5$; ; YES, LOAD R1 WITH ADDRESS OF PARITY REGISTER
    MOV $3, R1
    CMP $1, (R4)
    ; IS THIS REGISTER CORE?
    BNE 6$; ; YES, CORE. STORE RESERVED BITS
    MOV $3, R1
    CMP $1, (R4)
    ; MOS, STORE RESERVED BITS
    BNE 7$; ; LOAD R2 WITH VALUE OF FIRST BIT
    MOV $2, (R4)
    BNE 8$; ; TO BE TESTED
    MOV $2, R1
    CMP $2, R1
    ; INITIALIZE PARITY REGISTER
    BNE 9$; ; READ CONTENTS OF PARITY REGISTER
    MOV $2, R1
    CMP $2, R1
    ; CLEAR BITS WHICH ARE RESERVED
    BNE 10$; ; CLEAR INSTRUCTION DID NOT INITIALIZE
    MOV $2, R1
    CMP $2, R1
    ; ALL USED BITS IN PARITY REGISTER TO ZERO (R1 CONTAINING ADDRESS OF FAILING REGISTER)
    BNE 11$; ; IS THIS BIT RESERVED?
    MOV $2, R1
    CMP $2, R1
    ; YES- DON'T TEST IT SINCE IT MAY BE ZERO OR ONE
    BNE 12$; ; NO- SET THIS BIT IN THE PARITY REGISTER
    MOV $2, R1
    CMP $2, R1
    ; READ AND SAVE CONTENTS OF PARITY REGISTER
    BNE 13$; ; CLEAR PARITY REGISTER
    MOV $2, R1
    CMP $2, R1
    ; CLEAR BIT LOCATIONS THAT ARE RESERVED
    BNE 14$; ; CHECK REST
    MOV $2, R1
    CMP $2, R1
    ; BRANCH IF OK
    BNE 15$; ; PARITY REGISTER WHOSE ADDRESS IS IN R1
    MOV $2, R1
    CMP $2, R1
    ; WAS IN CORRECT AFTER THE VALUE IN R2 WAS WRITTEN INTO IT. ACTUAL CONTENTS
    BNE 16$; ; (WITH UNUSED BITS CLEARED) IS IN R3
    MOV $2, R1
    CMP $2, R1
    ; ROTATE BIT TO BE TESTED
    BNE 17$; ; IF NOT DONE WITH ALL BIT POSITIONS
    MOV $2, R1
    CMP $2, R1
    ; GO TEST THIS ONE
    BNE 18$; ; MOVE RD TO POINT TO NEXT POSSIBLE ADDRESS
    MOV $2, R1
    CMP $2, R1
    ; OF A PARITY REGISTER
    BNE 19$; ; AT END OF TABLE?
    MOV $2, R1
    CMP $2, R1
    ; NO, BRANCH
    BNE 20$; ;}

;**********************************************************************
;SHOW THAT RESET CLEARS BITS 0, 2, AND 15 OF EACH PARITY REGISTER
;**********************************************************************
LOAD THE TEST NUMBER INTO THE DISPLAY
DON'T IRRITER TEST
LOAD POINTER
POINTER TO INDICATOR
IS THIS PARITY REGISTER PRESENT?
NO BRANCH
IS THIS CORE OR MOS PARITY REGISTER?
NO - BRANCH
MOS - SET ALL DEFINED BITS TO 1
CORE - SET ALL DEFINED BITS TO 1
MOVE POINTER TO POINT TO NEXT MPR ADDRESS
INCREMENT POINTER TO INDICATOR
OF A PARITY REGISTER
AT END OF TABLE?
NO - CONTINUE
YES - TERMINAL AVAILABLE?
NO - BRANCH
YES - WAIT FOR TERMINAL TO FINISH
TH: ADDRESS OF THE TABLE
POINTER TO INDICATOR
CACHE
BRANCH IF NO
DISABLE CACHE
IS THIS PARITY REGISTER PRESENT?
NO - BRANCH
IS THIS A CORE PAR REGISTER?
NO - BRANCH
YES - GET CONTENTS OF REGISTER
MAKE SURE THAT WMP AND AE ARE CLEAR
MASK RESERVED BITS FOR CORE PAR
-ITY REGISTER, BITS 51 (ADDR
- BIT) ARE ALSO MASKED
CHECK, IF REST WERE CLEARED
CORE PARITY REGISTER WHOSE ADDRESS IS
POINDED TO BY RD WAS INCORRECT
AFTER A RESET WAS ISSUED - CONTENTS
SAVED IN R2 WITH UNUSED BITS MASKED
MOS - GET CONTENTS OF REGISTER
MAKE SURE THAT WMP AND AE ARE CLEAR
MASK RESERVED BITS FOR MOS PAR REG
CHECK RESET
RESET DID CLEAR ALL BITS
MOS PARITY REGISTER WHOSE ADDRESS IS
POINDED TO BY RD WAS INCORRECT, AFTER
ISSUING RESET CONTENTS OF PAR REG
HERE AS SHOWN IN R2 (UNUSED BITS
HAVE BEEN MASKED)
**NOTE:** IF PARITY MEMORY IS NOT LOCATED CORRECTLY BY THIS SUBTEST

IT IS DUE TO ONE OF THE FOLLOWING FAILURES:

- SETTING WRITE ARMOR PARITY DID NOT CAUSE BAD PARITY TO BE WRITTEN
- PARITY GENERATE OR DETECT LOGIC FAILED
- PARITY ERROR BIT FAILED TO SET
- PARITY BITS IN MEMORY LOCATION FAILED (I.E., BIT STUCK AT GOOD PARITY VALUE)

NOTE THAT SETTING SWITCH REGISTER SWITCH 9 WILL CAUSE A HALT AFTER THE MAP IS TYPED. IF YOU WISH TO CHANGE THE MAP TO ISOLATE THE CAUSE OF A MAPPING FAILURE, YOU CAN DO THIS ONCE THE PROCESSOR IS HALTED. SEE THE DESCRIPTION IN THE LISTING (PRECEDING THE MAP TAG "MAPS" AT LOCATION 600) FOR THE MEANING OF THE MAP CONTENTS. AFTER MAKING THE DESIRED CHANGES, PRESS CONTINUE, THE NEW MAP WILL BE TYPED AND IF SWITCH 9 IS LEFT SET THE PROCESS WILL BE REPEATED.

*NOTE:* IF SWITCH 9 IS NOT LEFT SET THE PROGRAM WILL PROCEED TO TEST THE PARITY MEMORY AND REGISTERS AS RECORDED IN THE NEW MAP.

---

**TEST 3:**

LOAD THE TEST NUMBER INTO THE DISPLAY

IF MAPPING HAS ALREADY BEEN DONE

SKIP SUBTEST

MAP MEMORY

FIND PARITY MEMORY AND CORRESPONDING REGISTERS USING WRITE WRONG PARITY

WITHOUT ACTION ENABLE SET

INITIALIZE LOCATIONS INDICATING

TOTAL PARITY MEMORY PRESENT

FLAG EXISTING PARITY MEMORY (LOW 64K)

FLAG EXISTING PARITY MEMORY (HIGH 64K)

TYPE MAP

INDICATE MAPPING DONE

SWITCH 9 SET?

YES - SWITCH 9 SET INDICATING HALT

AFTER TYPING PARITY MEMORY MAP

GO TYPE NEW MAP TO VERIFY USER'S INTENT
SHOW THAT SETTING AE WITH ERROR ALREADY SET DOESN'T CAUSE A TRAP
NOTE THAT IF A KT11 IS PRESENT, IT IS USED DURING THIS SUBTEST

TEST4: SCOPE

LOAD THE TEST NUMBER INTO THE DISPLAY
CLEAR ALL PARITY REGISTERS
KT11 PRESENT?
NO-BRANCH
YES-MAP ALL PAGES NON RESIDENT
THEN MAP KERNEL 0 TO BANK 0, KERN 1 TO EXTERNAL BANK, SET KERN 1, AND 7 RW, AND TURN ON KT11
SETUP TO FIND REGISTERS PRESENT

IS THIS REGISTER PRESENT?
YES-BRANCH TO TEST IF
NO-CHECK FOR ANOTHER ONE
INCREMENT PTRAP

BRANCH WHEN ALL REGISTERS HAVE BEEN TESTED
LOCATE MEMORY CORRESPONDING TO THIS REGISTER- IF NO KT11, R1 SHOULD BE RETURNED CONTAINING THE ADDRESS OF THE 1ST LOCATION CONTROLLED BY THIS REGISTER (INCLUDING EXTERNAL INTERLEAVE OFFSET IF NEEDED)
IF KT11 IS PRESENT, R1 SHOULD BE RETURNED POINTING TO THE 1ST LOCATION CONTROLLED BY THIS REGISTER, MAPPED THRU KERNEL PAGES. KERN 1 SHOULD BE MAPPED TO THE CORRECT BANK
IS ERROR RETURN INDICATED?
NO-BRANCH
MAP INDICATES NO PARITY MEMORY
IS CONTROLLED BY THIS REGISTER
R0 POINTS TO THE ADDRESS OF THE PARITY REGISTER

SETUP PARITY TRAP RETURN
SET WRITE WRONG PARITY
WRITE CONTENTS OF LOCATION WITH WRONG PARITY
CLEAR PARITY REGISTER
READ BAD PARITY WITH ACTION ENABLE
CLEARED- NO TRAP EXPECTED
CHANGE PARITY TRAP RETURN
SET ACTION ENABLE WITH PARITY ERROR ALREADY SET- SHOULDN'T TRAP YET
CHANGE PARITY TRAP RETURN
READ LOCATION AGAIN- SHOULD GET A PARITY TRAP DUE TO READING BAD PARITY WITH ACTION ENABLE SET
NO PARITY TRAP AFTER READING LOCATION WHICH SHOULD CONTAIN BAD PARITY- R1 CONTAINS ADDRESS OF MEMORY LOCATION
0149 003074 005070 000000  CONT4: CLR 2(R0)
0150 003100 005511  ADC 2(R0)
0151 003102 005070 000000  CLR 2(R0)
0152 003106 014002  LRP4
0153 003110 104002  TRP4A: ERROR
0154 003112 022626  CMP (SP)+,(SP)+
0155 003114 003762  BR CONT4
0156 003116 104002  TRP4B: ERROR
0157 003120 022626  CMP (SP)+,(SP)+
0158 003124 007670 000000  BR CONT4
0159 003128 104002  TRP4C: TST 2(R0)
0160 003130 104001  BRI .4
0161 003132 104002  ERROR
0162 003134 022626  CMP (SP)+,(SP)+
0163 003138 007670 000000  BR CONT4
0164 00313C 102737 000116 000114  DONE4: MOV #PARVEC+2,#PARVEC
0165 003140 005767 175376  TST NOKT
0166 003144 010002  BNE +6
0167 003148 005037 177572  CLR @#5RO
0168 ;TURN OFF KT11 IF PRESENT

;***************
;SHOW THAT READING GOOD PARITY AFTER BAD PARITY DOESN'T CLEAR PARITY ERROR BIT
;***************

0169 003150 104001  TESTS: SCOPE
0170 003152 012777 000005 175712  MOV #5,DISP standout
0171 003154 004767 010616  JSR PC,CLEARPAR
0172 003156 005767 175350  TST NOKT
0173 003158 001004  BNE 12
0174 00315A 004767 010550  JSR PC,MAPALL
0175 00315C 004767 010560  JSR PC,MAP1
0176 00315E 010650 000000  THEN MAP KERNEL D TO BANK 0, MAP KERNEL
0177 003160 010650 000010  TO THE EXTERNAL BANK, SET KERNEL
0178 003162 010650 000010  A, AND 7 RW, AND TRAP ON KT11
0179 003164 010650 000010  SETUP TO FIND REGISTERS PRESENT
0180 003166 012700 000056  IS: MOV #MPRO,RO
0181 003168 012710 000001  LOOP5: BIT #1,RO
0182 00316A 012706 000010  BEQ TS15
0183 00316C 010700 000010  LOOPS: ADD #10,RO
0184 00316E 010796 000010  CMP RO,TRREG
0185 003170 013770 000010  BLO LOOP5
0186 003172 005431 000010  BR DONE
0187 003174 004767 010662  TS7S: JSR PC,LOCATM

;EXIT WHEN ALL REGISTERS HAVE BEEN TESTED
;LOCATION MEMORY CORRESPONDING TO THIS
REG-R1 WILL BE RETURNED CONTAINING
THE ADDRESS OF THE FIRST LOCATION
CONTROLLED BY THIS REGISTER (MAPPED
THRU KERNEL PAGE 1 IF KI1 IS PRESENT)
IS ERROR RETURN INDICATED?
NO-
BRANCH
MAP INDICATES NO PARITY MEMORY IS
CONTROLLED BY THIS REGISTER. RO
POINTS TO THE ADDRESS OF THE
PARITY REGISTER

SET WMP IN THIS REGISTER
WRITE CONTENTS OF LOCATION WITH
WRONG PARITY
DETECT WRONG PARITY
CLEAR WMP IN PARITY REGISTER
RESTORE GOOD PARITY
READ LOCATION
READ CONTENTS OF PARITY REGISTER
BRANCH IF PARITY ERROR IS STILL SET
PARITY ERROR BIT CLEARED BY READING
GOOD PARITY (OR POSSIBLY WHILE DOING
THE BIC TO CLEAR WMP). RO POINTS TO
THE PARITY REGISTER ADDRESS.
CLEAR THE PARITY REGISTER

TURN OFF XT11 IF PRESENT

SHOW THAT PARITY GENERATE AND DETECT LOGIC WORKS CORRECTLY FOR EACH BYTE
SHOW THAT WRITE WRONG PARITY WORKS FOR HIGH AND LOW BYTES
SHOW THAT WRITING INTO LOCATION WHEN WRITE WRONG PARITY IS NOT SET
RESTORES GOOD PARITY

TESTS:

LOAD THE TEST NUMBER INTO THE DISPLAY
CLEAR ALL PARITY REGISTERS
NO-
BRANCH
YES- MAP IT (KERNEL 0 TO BANK 0, RW;
AND BANK 2 TO EXTERNAL BANK, RW; KERNEL 1 RW)
SETUP TO FIND REGISTERS PRESENT
IS THIS REGISTER PRESENT?
YES-
BRANCH TO TEST IT
NO- CHECK FOR ANOTHER ONE
BRANCH TO DOME IF ALL REGISTERS
HAVE BEEN TESTED
LOCATE MEMORY CORRESPONDING TO
THIS REGISTER- R1 SHOULD BE RETURNED
CONTAINING THE ADDRESS OF THE FIRST
LOCATION CONTROLLED BY THIS REGISTER VIRTUAL THRU KERNAL PAGE 1 IF KILL PRESENT
IF NO MEMORY WAS FOUND TO CORRESPOND GOD ADDRESS IS RETURNED-BRANCH IF OK
MAP INDICATES NO PARITY MEMORY IS
CONTROLLED BY THIS REGISTER
RD POINTS TO THE ADDRESS OF THE
PARITY REGISTER
AFTER ERROR, CHECK FOR NEXT REGISTER

FIRST SHOW THAT IF THE PARITY REGISTER IS CLEARED INITIALLY,
PARITY ERROR DOESN'T SET

INIVILIZE LOCATION UNDER TEST
INITIALLY CLEAR PARITY REGISTER
R2 CONTAINS VALUE TO BE LOADED
WRITE VALUE INTO LOW BYTE
READ WORD TO CHECK PARITY
CHECK PARITY REGISTER
BRANCH IF ERROR NOT SET
PARITY ERROR SET WHICH VALUE IN R2 WAS
WRITTEN AND READ BACK FROM LOW BYTE OF
LOCATION WHOSE ADDRESS IS CONTAINED IN
R1 (MAP WAS NOT SET)
CLEAR ERROR BIT
REINITIALIZE TEST LOCATION
REINITIALIZE VALUE TO BE USED

INCREMENT VALUE TO BE LOADED
LOOP UNTIL ALL VALUES HAVE BEEN USED
WRITE ALL VALUES INTO HIGH BYTE
READ WORD TO CHECK PARITY
CHECK PARITY REGISTER
BRANCH IF ERROR NOT SET
PARITY ERROR SET WHEN VALUE IN R2
WAS WRITTEN AND READ BACK FROM HIGH BYTE
OF LOCATION WHOSE ADDRESS IS CONTAINED
IN R1 (MAP WAS NOT SET)
INCREMENT VALUE TO BE LOADED
LOOP UNTIL ALL VALUES HAVE BEEN USED

TEST PARITY GENERATE AND DETECT LOGIC BY SETTING WRITE WRONG PARITY AND
WRITING EACH POSSIBLE VALUE TO THE LOW BYTE, THEN TO THE HIGH BYTE

INCREMENT VALUE TO BE WRITTEN
SET WRITE WRONG PARITY
WRITE WRONG PARITY IN LOW BYTE
CLEAR WRITE WRONG PARITY, AND CLEAR
PARITY ERROR IF SET
READ BACK WRONG PARITY
PARITY ERROR SET?
YES-BRANCH
PARITY ERROR DID NOT SET WHEN THE
LOCATION UNDER TEST WAS WRITTEN
AND READ BACK WITH WRITE WRONG PARITY
SET. RD POINTS TO ADDRESS OF PARITY
REG 01 CONTAINS ADDRESS OF LOCATION
BEING TESTED (VIRTUAL THRU KERNEL
PAGE 1 IF KT11 IS PRESENT). REG 02
CONTAINS THE VALUE WRITTEN.
EXIT LOOP AFTER ERROR
INCREMENT DATA
LOOP TILL DONE WITH ALL VALUES
REINITIALIZE LOCATION TO CLEAR BAD PARITY
CLEAR ERROR IF SET
READ LOCATION, WHICH SHOULD NOW HAVE
GOOD PARITY
PARITY ERROR SET?
NO, BRANCH
GOOD PARITY WAS NOT RESTORED BY
WRITING INTO THE LOCATION WITH
WRITE WRONG PARITY CLEARED
SET WRITE WRONG PARITY
WRITE WRONG PARITY IN HIGH BYTE
CLEAR WRITE WRONG PARITY AND PARITY
ERROR IF SET
READ BACK WRONG PARITY
PARITY ERROR SET?
YES-BRANCH
PARITY ERROR DID NOT SET WHEN THE LOCATION
UNDER TEST WAS WRITTEN AND READ BACK WITH
WRITE WRONG PARITY SET. RD POINTS
TO THE ADDRESS OF THE PARITY REGISTER.
THE VALUE IN RD WAS WRITTEN INTO THE HIGH
BYTE OF THE LOCATION WHOSE ADDRESS IS IN RI
(VIRTUAL THRU KERNEL PAGE 1 IF KT11 PRESENT)
THEN THE PARITY REGISTER WAS
CLEARED AND THE WORD WAS READ BACK
EXIT LOOP AFTER ERROR
INCREMENT DATA
LOOP TILL DONE WITH ALL VALUES
CLEAR BAD PARITY IN TEST LOCATION
CLEAR PARITY ERROR BIT IF SET
READ LOCATION, WHICH SHOULD NOW
HAVE GOOD PARITY
CHECK PARITY ERROR BIT
BRANCH IF CLEAR
WRITING INTO LOCATION WHEN WRITE
WRONG PARITY WAS NOT SET. DID NOT
WRITE GOOD PARITY
GO CHECK FOR ANOTHER PARITY REGISTER
; TURN OFF KT11 IF PRESENT

; SHOW THAT SETTING PARITY ERROR AFTER SETTING ACTION ENABLE WON'T CAUSE A TRAP

; LOAD THE TEST NUMBER INTO THE DISPLAY
Initially clear all parity registers

Setup parity trap return

Setup to get address of register present

Branch to test register

Branch if done testing all registers

Set action enable

Set parity error

Should not trap

Clear parity register

Go check next register

Trap occurred when parity error bit

Was set via a bis instruction

With action enable already set

Ro points to the address of the

Parity register

Restore stack pointer

Clear parity register

Show that repeated parity errors will cause repeated traps if action

Enable is set and parity error is left set

Show that the error address bits (11-5) track (only for core parity registers)

**********************************************************

L02
90 LOAD THE TEST NUMBER INTO THE DISPLAY
91 INITIALLY CLEAR ALL PARITY REGISTERS
92 KTI1 PRESENT?
93 NO- BRANCH
94 YES- INITIALLY MAP ALL PAGES NR
95 MAP KERNEL D TO BANK D, RW
96 KERNEL 7 TO THE EXTERNAL BANK, RW
97 MAKE KERNEL PAGE 1 RW AND TURN ON THE KT11
98 
99 ;BRANCH TO TEST REGISTER IF PRESENT
100 
101 ;BRANCH IF ALL REGISTERS HAVE BEEN TESTED
102 ERROR RETURN INDICATED?
103 BRANCH IF NO
104 MAP INDICATES THERE IS NO MEMORY
105 CORRESPONDING TO THIS REGISTER
106 RO POINTS TO THE ADDRESS OF
107 THE PARITY REGISTER
108 
109 ;SET WRITE WRONG PARITY
110 WRITE WRONG PARITY IN FIRST LOCATION
111 WRITE WRONG PARITY IN SECOND LOCATION
112 SET ACTION ENABLE AND CLEAR HCST
113 TRAP PARITY TRAP RETURN
114 SETUP COUNTER TO EXECUTE INSTRUCTION 1 (INSTE EM) TEN TIMES
115 READ WRONG PARITY WITH AE SET- SHOUL
116 TRAP TO TRGPIO
117 NO PARITY TRAP OCCURRED. RO POINTS TO
118 ADDRESS OF THE PARITY REGISTER BEING
119 TESTED.
120 
121 ;READ WRITE PARITY FROM SECOND ADDRESS
122 WITH AE SET- SHOULD TRAP TO TRGPIO
123 NO PARITY TRAP OCCURRED. RO POINTS TO
124 THE ADDRESS OF THE PARITY REGISTER
125 BEING TESTED
126 
127 ;HAS PARITY TRAP OCCURRED TEN TIMES?
128 YES- BRANCH
129 IS THIS A CORE PAR SEG?
130 NO- BRANCH (NO ERROR
131 ADDRESS BITS FOR MOS PAR
132 IF ERROR ADDRESS BITS ARE TRACKING
133 BIT S SHOULD BE CLEAR (ONLY FOR CORE PARITY)
134 ;PARITY ERROR ADDRESS BITS INCORRECT
135 RO POINTS TO THE ADDRESS OF THE PARITY
136 REGISTER. RI CONTAINS THE ADDRESS
137
2$: MOV #INST!,@SP
RTI
2$: MOV #TRPIOR, #PARVEC
MOV #INST!,@SP
RTI
000000
TRPIOR: CMP $1, (R2)
BNE $+4
BNE .+4
ERROR

; REFERENCED TO CAUSE A PARITY TRAP
; (VIRTUAL IF KT11 IS PRESENT)
; GO EXECUTE INSTRUCTION 1 AGAIN
; CHANGE PARITY TRAP RETURN
; GO EXECUTE INSTRUCTION 2
; IS THIS A CORE REG?
; NO BRANCH
; PARITY TRAP OCCURRED- CHECK PARITY
; ERROR ADDRESS BITS
; BRANCH IF OK (IF THE PARITY ERROR
; ADDRESS BITS TRACKED, BIT 5 WILL BE SET)
; PARITY ERROR ADDRESS BITS INCORRECT
; RD POINTS TO THE ADDRESS OF THE
; PARITY REGISTER. THE ADDRESS REFERENCED
; TO CAUSE THE ERROR WAS THAT IN
; RI PLUS 4000 (OCTAL).
; RESTORE TRAPCATCHER
; CLEAR PARITY REGISTER
; CLEAR BAD PARITY
; CLEAR PARITY ERROR BIT IF SET
; TURN OFF KT11 IF PRESENT

;***********************************************************************
; IF MULTIPLE PARITY ERRORS OCCUR DURING ONE INSTRUCTION (WITH ACTION ENABLE
; NOT SET) THE ERROR ADDRESS BITS WILL RECORD THE LAST ERROR (ONLY FOR CORE PARITY
; REGISTERS)
;***********************************************************************

TEST11: SCOPE

; LOAD THE TEST NUMBER INTO THE DISPLAY
; INITIALLY CLEAR ALL PARITY REGISTERS
; KT11 PRESENT?
; NO- BRANCH
; YES- MAP KERNEL PAGE 0 TO BANK 0 RW
; MAP KERNEL PAGE 7 TO THE EXTERNAL BANK, RW
; SET KERNEL PAGE 1 RW AND TURN ON KT11
; SETUP TO GET ADDRESSES OF REGISTERS.PRESENT

; IF REG NOT PRESENT, SKIIP
; IS THIS A CORE PAR REG?
; YES THEN TEST IT
; IF NOT CORE, SKIP THIS REGISTER.
004430 103764          BLC LUP11
004439 004439          BR DONE11
00443f 00443f          ;BRANCH OUT IF ALL REGISTERS HAVE BEEN TESTED
004440 004440          TST11: JSR #7,LOCATM
004445 004445          ;GET THE ADDRESS OF A MEMORY LOCATION
00444a 00444a          CORRESPONDING TO THIS PARITY REGISTER
00444e 00444e          ERROR RETURN INDICATED?
004452 004452          ;BRANCH IF NOT
004457 004457          ;NO MEMORY IN MAP CORRESPONDING TO
00445c 00445c          ;THIS PARITY REGISTER, RD POINTS TO
004460 004460          ;THE ADDRESS OF THE PARITY REGISTER
00446a 00446a          ;SETUP SECOND TEST ADDRESS LOCATION
004470 004470          ;WRITE WRONG PARITY
004475 004475          ;WRITE WRONG PARITY IN FIRST TEST LOCATION
00447a 00447a          ;WRITE WRONG PARITY IN SECOND TEST LOCATION
00447f 00447f          ;CLEAR PARITY REGISTER
004484 004484          ;READ FIRST TEST LOCATION, AND
004489 004489          ;THEN READ SECOND TEST LOCATION
00448e 00448e          ;MAKE SURE PARITY ERROR SET
004493 004493          ;PARITY ERROR NOT SET AFTER
004498 004498          ;READING TWO LOCATIONS WHICH
00449d 00449d          ;SHOULD HAVE BAD PARITY
0044a2 0044a2          ;CHECK ERROR ADDRESS- IF THE LAST
0044a7 0044a7          ;ADDRESS WAS RECORDED, BIT 6 WILL
0044ae 0044ae          ;BE SET
0044b4 0044b4          ;PARITY ERROR ADDRESS BITS INCORRECT
0044ba 0044ba          ;RD POINTS TO ADDRESS OF PARITY REGISTER
0044be 0044be          ;RE CONTAINS ADDRESS OF LAST BAD PARITY
0044c1 0044c1          ;LOCATION REFERENCED (IF KT11 PRESENT)
0044c6 0044c6          ;ADDRESS IS VIRTUAL THRU KERNEL PAGE 1
0044c8 0044c8          ;CLEAR PARITY REGISTER
0044cb 0044cb          ;CLEAR BAD PARITY
0044d0 0044d0          ;CLEAR PARITY ERROR BIT
0044de 0044de          ;TURN OFF KT11 IF PRESENT

**-----------------------------**

;SHOW THAT IF AN INSTRUCTION DOING A DARTIPS GETS A PARITY ERROR,
;THE ORIGINAL DATA IS REWRITTEN IF ACTION ENABLE IS SET, AND IS
;ALTERED IF ACTION ENABLE IS CLEAR

**-----------------------------**

TEST12: SNEEPE

00454e 010000          MOV #12,DISPLAY
004557 072222          JSR #7,CLRPAR
004560 173754          TST NOKT
004565 001004          BNE 1S
00456a 007110          JSR #7,RA
00456f 001074          JSR #7,MAP1

**-----------------------------**

;LOAD THE TEST NUMBER INTO THE DISPLAY

;KT11 PRESENT?
;NO- BRANCH
;YES, MAP KERNEL 0 TO BANK 0, RW
;MAP KERNEL 7 TO THE EXTERNAL BANK, RW
;SET KERNEL 1 RW AND TURN ON KT11
1474 004606 012700 000000
1475 004612 012706 000070
1476 004618 002710 000000
1477 004624 001003
1478 004624 002712 000001
1479 004630 001047
1480 004632 062700 000010
1481 004636 005722
1482 004640 020027 00076b
1483 004644 103764
1484 004646 000474
1485 004650 007677 007252
1486 004651 002701 000001
1487 004660 001403
1488 004662 104002
1489 004664 00167 000000
1490 004666 00167 177742
1491 004670 012737 004724 000114
1492 004670 012770 000000 000000
1493 004670 125252 125252
1494 004670 012770 000000 000000
1495 004670 000471
1496 004716 005211
1497 004720 104002
1498 004722 000400
1499 004724 005070 000000
1500 004730 021127 125252
1501 004734 001401
1502 004736 104002
1503 004736 000400
1504 004739 005070 000000
1505 004740 021127 125252
1506 004740 005720 000000
1507 004744 104001
1508 004746 104002
1509 004750 022626
1510 004752 012770 000004 000000
1511 004756 125252
1512 004760 012711 125252

; SETUP TO GET ADDRESSES OF REGISTERS PRESENT
; SKIP, IF THIS REG NOT PRESENT
; REG PRESENT, IS IT CORE?
; YES, DO THIS TEST
; SKIP, IF THIS REG IS NOT CORE

; BRANCH TO DONE, IF ALL REGISTERS
; HAVE BEEN TESTED
; LOCATE MEMORY CORRESPONDING TO THIS
; REGISTER
; ERROR RETURN INDICATED?
; NO-BRANCH
; NO MEMORY IN MAP CORRESPONDING TO
; THIS REGISTER. RD POINTS TO
; THE ADDRESS OF THE PARITY REGISTER

; SET UP PARITY TRAP RETURN
; SET WRITE WRONG PARITY
; WRITE WRONG PARITY IN TEST LOCATION
; SET ACTION ENABLE AND CLEAR
; WRITE WRONG PARITY
; DO DATIP, DATP WITH ACTION ENABLE
; SET SHOULDB ABORT ON DATIP AND
; RESTORE ORIGINAL DATA
; NO ABORT OCCURRED ON READING LOCATION
; WHICH SHOULD CONTAIN BAD PARITY
; (WITH AE SET)
; RD POINTS TO ADDRESS OF PARITY REGISTER.
; R1 CONTAINS ADDRESS OF TEST LOCATION
; (VIRTUAL THRU KERNEL PAGE 1 IF R1 IS
; PRESENT)

; PARITY TRAP OCCURRED - CLEAR PARITY REGISTER
; ORIGINAL DATA RESTORED?
; YES, BRANCH
; NO - DATIP WHICH GOT A PARITY ERROR
; TRAP ALTERED CONTENTS OF LOCATION
; READ. ADDRESS OF TEST LOCATION IS IN R1
; (IF R1 IS PRESENT, ADDRESS IN R1
; IS VIRTUAL THRU KERNEL PAGE 1)
; RD POINTS TO ADDRESS OF PARITY REGISTER
; MAKE SURE PARITY ERROR SET WHEN
; DATA WAS REBOW IN THE ABOVE CMP
; DATIP WHICH GOT A PARITY ERROR TRAP
; ALTERED THE PARITY OF THE LOCATION READ
; R1 CONTAINS ADDRESS OF TEST LOCATION
; (VIRTUAL THRU KERNEL 1 IF R1 IS PRESENT)

; RESTORE STACK POINTER
; SET WRITE WRONG PARITY AND CLEAR
; PARITY ERROR
; REWRITE DATA WITH WRONG PARITY
LOCATE MEMORY CORRESPONDING TO THIS
REGISTER AND IF KTI1 IS PRESENT
MAP KERNEL 1 TO THAT MEMORY
ERROR RETURN INDICATED?
NO- BRANCH
MAP INDICATES NO MEMORY WAS FOUND
CORRESPONDING TO THIS REGISTER
NO POINTS TO THE ADDRESS OF THE
PARITY REGISTER

SETUP PARITY TRAP RETURN
SET WRITE WRONG PARITY
WRITE WRONG PARITY
SET ACTION ENABLE AND CLEAR
WRITE WRONG PARITY
DATA WITH ACTION ENABLE SET SHOULD
ABORT LEAVING DATA UNCHANGED
NO ABORT ON READING BAD PARITY
WITH ACTION ENABLE SET. RD POINTS
TO THE ADDRESS OF THE PARITY REGISTER.
RI CONTAINS THE ADDRESS OF THE TEST
LOCATION (VIRTUAL THRU KERNEL PAGE
1 IF KTI1 IS PRESENT).

ABORT OCCURRED AS EXPECTED- CLEAR REGISTER
ORIGINAL DATA RESTORED?
YES- BRANCH
DATA WHICH GOT A PARITY ERROR ALTERED
THE CONTENTS OF THE LOCATION ADDRESSED.
(RI CONTAINS THE ADDRESS OF
MEMORY BEING TESTED- IF KTI1 IS
PRESENT, ADDRESS IN RI IS VIRTUAL)
NO POINTS TO THE ADDRESS OF THE
PARITY REGISTER
CHECK PARITY REGISTER
BRANCH IF PARITY ERROR SET
PARITY ERROR NOT SET AFTER READING
DATA WITH BAD PARITY
NO POINTS TO THE ADDRESS OF THE PARITY
REGISTER. RI CONTAINS THE ADDRESS
OF THE TEST LOCATION (VIRTUAL THRU
KERNEL PAGE 1 IF KTI1 PRESENT)
RESTORE STACK POINTER
SET WRITE WRONG PARITY, CLEAR PARITY ERROR
REWRITE DATA WITH WRONG PARITY
CLEAR PARITY REGISTER
RESTORE TRAPCATCHER
DATA TO LOCATION WITH BAD PARITY
NO SET-INSTRUCTION SHOULD COMPLETE
CLEAR PARITY ERROR BIT
CHECK DATA

DATA TO LOCATION WITH BAD PARITY
WITHOUT ACTION ENABLE SET LEFT INCORRECT DATA
RI CONTAINS THE ADDRESS OF THE TEST
LOCATION (VIRTUAL THRU KERNEL PAGE 1
IF K11 IS PRESENT, RD POINTS TO THE
ADDRESS OF THE PARITY REGISTER.
CLEAR PARITY REGISTER
CLEAR LOCATION
CLEAR PARITY ERROR IF SET.
GO CHECK FOR ANOTHER PARITY
REGISTER
RESTORE TRAPCATCHER
;TURN OFF K11 IF PRESENT

CHECK PARITY MEMORY WITH SERIES OF PATTERNS FROM 4K TO 28K
ENABLE PARITY TRAP

TEST14: SCOPE
TEST16: JMP
TEST17: JMP
TEST18: JMP

LOAD THE TEST NUMBER INTO THE DISPLAY
DON'T ITERATE THE REST OF THE SUBTESTS
CLEAR ALL PARITY REGISTERS
SETUP PARITY TRAP RETURN
TEST FOR ANY 8K SYSTEM
SYST > BK
SYST < OR = BK
TEST BANK INDICATOR FOR BANK 1
INITIALIZE BANK INDICATOR TO BANK 2
INITIALIZE MEMORY STARTING ADDRESS
INITIALIZE BANK INDICATOR TO BANK 1
DOES THIS 4K HAVE PARITY?
YES, TEST IT
TEST FOR BANK 1 INDICATOR IF SET GO TO TEST 16
NO, UPDATE MEMORY ADDRESS
UPDATE BIT POINTER
THIS 28K DONE?
NO, BRANCH TO SEE IF NEXT 4K
SHOULD BE TESTED
EXIT
INITIALIZE PATTERN POINTER
SET UPPER LIMIT FOR THIS 4K
INITIALLY CLEAR CORE BLOCK UNDER TEST
INITIALIZE TO SET AE IN ALL REGISTERS
SET ACTION ENABLE IF REGISTER IS PRESENT
GO TO ROUTINE TO EXERCISE THIS 4K WITH THE CURRENT PATTERN

UPDATE PATTERN

LAST PATTERN?

NO LOOP

YES, CLEAR ALL PARITY REGISTERS

UPDATE AND CHECK NEXT 4K

RESTORE TRAP CATCHER

GO TO NEXT TEST

:ROUTINE TO WRITE AND CHECK EACH LOCATION IN 4K (STARTING AT ADDRESS ADRPT) WITH VALUE POINTED TO BY RN

:IN ADRPT (WITH VALUE POINTED TO BY RN)

:MOVE ADRPT, RN

:SETUP RS TO ADDRESS MEMORY LOCATION BEING CHECKED

:WRITE PATTERN INTO MEMORY

:READ TEST LOCATION

:DATA OK?

:YES- BRANCH

:DATA INCORRECT IN LOCATION WHOSE ADDRESS IS IN RS. RN POINTS TO THE DATA WRITTEN.

:UPDATE ADDRESS POINTER

:THIS 4K DONE?

:NO. BRANCH TO TEST NEXT LOCATION

:YES, DID ANY PARITY ERRORS OCCUR WITHOUT TRAPPING?

:YES- BRANCH

:STORE ADDRESS OF REGISTER GETTING ERROR

:PARTY ERROR SET (WITH AE SET) AND NO TRAP OCCURRED. TREG CONTAINS ADDRESS OF PARITY REGISTER WHICH HAS ERROR BIT SET.

:SCAN FOR PARITY ERRORS AND PRINT 18 BIT ADDRESSES OF THOSE FOUND.

:AFTER REPORTING EACH ERROR CLEAR IT

:PARITY TRAP SERVICE (NO TRAPS TO 114 SHOULD OCCUR IN THIS SUBTEST)

:TRAP 14: CMP TREG

:SUBTRACT TREG FROM ADDRESS OF REGISTER WHICH IS IN ERROR

:IF (RESULT IS NOT ZERO) THEN ERROR OCCURRED AT ADDRESS OF REGISTER WHICH HAS ERROR BIT SET

:BLEN (RESULT IS NOT ZERO)

:ADD TREG

:IF ERROR OCCURRED AT ADDRESS OF REGISTER WHICH HAS ERROR BIT SET

:THEN BRANCH TO PARITY ERROR SET
**TEST 15: SCOPE**

**TEST MEMORY WITH SERIES OF PATTERNS ABOVE 28K**

LOAD THE TEST NUMBER INTO THE DISPLAY

**KT11 PRESENT?**

YES - BRANCH

NO, SKIP TEST

CLEAR ALL PARITY REGISTERS

MAP KERNEL 0 TO BANK 0.RW

MAP KERNEL 3 TO THE EXTERNAL BANK

SET KERNEL 1 RW AND TURN ON KT11

MAP KERNEL PAGE 1 TO BEGINNING OF 28-32K

CLEAR FLAG TO INDICATE CHECKING LOWER 64K

SETUP PARITY TRAP RETURN

INITIALIZE BIT POINTERS

DOES THIS 4K HAVE PARITY?

YES, BRANCH TO TEST IT

NO, MAP TO NEXT 4K

UPDATE BIT POINTER

BRANCH IF NOT DONE WITH 64K

DONE WITH 4KB?

YES, BRANCH

NO, SET FLAG INDICATING UPPER 64K

SETUP PARITY MAP WORD

SETUP BIT POINTER FOR UPPER 64K

CONTINUE

INITIALIZE PATTERN POINTER

INITIALIZE VIRTUAL ADDRESS OF MEMORY BEING TESTED

INITIALLY CLEAR CORE BLOCK UNDER TEST

INITIALIZE TO SET ACTION ENABLE IN ALL PARITY REGISTERS
COMPAQ, MEMORY PARITY TEST
COMPAQ AII 13-JAN-78 12:13

1810 006d74 006d71 000001
1811 006d72 012771 000001 000000
1812 006d70 042701 000010 000076
1813 006d6f 006d65 012765 000076
1814 006d64 006d6d 000030

4s: JSR %0, TPCORX

TSF (14)
TST (4)
CMP R1, BNE
BNE 4s
JSR PC CLRPAR

DOMEIS: BR LOPIS
BR LOPIS
BR TEST16
BR TEST16

:PARITY MEMORY TEST ROUTINE USING KT11 AND TESTING MEMORY ABOVE 28K
:WRITES AND CHECKS EACH LOCATION IN 4K USING KERNEL PAGE 1 MAPPED TO CURRENT BANK
:TPCORX: NOP
MOV #200000, R5
MOV .5
MOV (5)
MOV (5)
CMP (4)
BEQ .4s
ERROR .4s

TST (5)
CMP R5, #40000
BLO .4s
CLR #R1

2s: MOV #MPRO, R1
BIT $1, (1)
BNE +10
TST $R1
CMP R1, ADD %10, R1
CMP R1, TREG
BLO 2s
RTS %7

3s: MOV 8R1, TREG
ERRORS

RTS %7

PARITY TRAP SERVICE (NO TRAPS TO 114 SHOULD OCCUR IN THIS SUBTEST)
TRAP15: CLR TREG
MOV #MPRO, R1
BIT $1, (1)
BNE +10

:LOCATE PARITY REGISTER INDICATING ERROR
J03

CCMAF, MEMORY PARITY TEST CCMAF.M11 13-JAN-78 12:13

MACYI 30A(J052) 13-JAN-78 12:27 PAGE 35

SEQ 0035

1866  006400  005771  000000
1867  006409  004007
1868  006441  000010
1869  006442  000076
1870  006443  010383
1871  006444  010002
1872  006445  000005
1873  006446  004767  006036
1874  006447  000003
1875  006448  004767  172316
1876  006449  010404
1877  006450  172316
1878  006451  004767
1879  006452  004767
1880  006453  012826
1881  006454  000207
1882  006455  012826
1883  006456  000016
1884  006457  000016
1885  006458  172410
1886  006459  012777
1887  006460  000004
1888  006461  000004
1889  006462  000004
1890  006463  000004
1891  006464  000004
1892  006465  000004
1893  006466  000004
1894  006467  000004
1895  006468  000004
1896  006469  000004
1897  006470  000004
1898  006471  000004
1899  006472  000004
1900  006473  000004
1901  006474  000004
1902  006475  000004
1903  006476  000004
1904  006477  000004
1905  006478  000004
1906  006479  000004
1907  006480  000004
1908  006481  000004
1909  006482  000004
1910  006483  000004
1911  006484  000004
1912  006485  000004
1913  006486  000004
1914  006487  000004
1915  006488  000004
1916  006489  000004
1917  006490  000004
1918  006491  000004
1919  006492  000004
1920  006493  000004
1921  006494  000004

1866  TST 3(R1)
1867  BXT 2$  ADD #10,R1
1868  CMP R1,TREG
1869  BLX 1$  ERROR
1870  BR 3$  MOV R1,TREG
1871  MOV 172316,ERRORS
1872  JSR %7,PSCAN
1873  RTS %7

1886  006456  000005
1887  006457  000005
1888  006458  000005
1889  006459  000005
1890  006460  000005
1891  006461  000005
1892  006462  000005
1893  006463  000005
1894  006464  000005
1895  006465  000005
1896  006466  000005
1897  006467  000005
1898  006468  000005
1899  006469  000005
1900  006470  000005
1901  006471  000005
1902  006472  000005
1903  006473  000005
1904  006474  000005
1905  006475  000005
1906  006476  000005
1907  006477  000005
1908  006478  000005
1909  006479  000005
1910  006480  000005
1911  006481  000005
1912  006482  000005
1913  006483  000005
1914  006484  000005
1915  006485  000005
1916  006486  000005
1917  006487  000005
1918  006488  000005
1919  006489  000005
1920  006490  000005
1921  006491  000005

;BRANCH IF PARITY ERROR IS SET
;TRAP TO 114 OCCURRED DURING TEST IS SJTI
;NO PARITY REGISTERS HAVE PARITY ERROR SET
;STORE ADDRESS OF REGISTER GETTING ERROR
;PARITY TRAP TO 114 OCCURRED DUE TO
;PARITY ERROR WHILE EXERCISING MEMORY
;"TREG" CONTAINS ADDRESS OF PARITY REGISTER
;HAVING PARITY ERROR BIT SET
;SCAN MEMORY FOR BAD PARITY AND PRINT 10
;BIT ADDRESSES OF LOCATIONS FOUND
;CLEAR BAD PARITY IN EACH AFTER
;REPORTING IT
;RESTORE STACK POINTER
;RETURN (FROM JSR TO TPCORX) TO
;TEST NEXT PATTERN

;******************************************************************************

FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY FROM 4K TO 28K
WRITE WRONG PARITY AND READ IT BACK WITH ACTION ENABLE SET, MAKING
SURE THAT A TRAP OCCURS, THEN WRITE GOOD PARITY AND MAKE SURE THAT
NO TRAP OCCURS WHEN IT IS READ, MAKE SURE THAT THE ERROR ADDRESS BITS
(PARITY REGISTER BITS 5-11) ARE CORRECT.

******************************************************************************

TEST16: SCOPE

LOAD THE TEST NUMBER INTO THE DISPLAY
SET UP TRAP RETURN
TEST BANK INDICATOR FOR BANK 1
INIT POINTER TO BANK 2

DOES THIS 4K HAVE PARITY?
YES, BRANCH TO TEST IT
TEST FOR BANK 1 INDICATOR IF SET GO OUT
NO, UPDATE MEMORY ADDRESS BY 4K
UPDATE BIT POINTER
THIS 28K DONE?
NO, CHECK NEXT 4K
YES, EXIT
SET UPPER LIMIT THIS 4K
CLEAR ALL PARITY REGISTERS
CLEAR BANK UNDER TEST
NOTE THAT AE AND WHP WERE CLEARED BEFORE TYPING THE ERROR PRINTOUT.

WHEN WRONG PARITY DATA IS READ BACK SHOULDS ENTER HERE VIA TRAP TO 114.

MOV #MPRO,R1
TRAP16A: BIT #IAR1

PARITY TRAP OCCURRED - FIRST CLEAR
WHP AND AE IN ALL REGISTERS

FIND THE REGISTER THAT SENSED THE ERROR

ERROR SET IN MORE THAN ONE PARITY REGISTER
AFTER WRITING WRONG PARITY IN LOCATION
WHOSE ADDRESS IS IN RS

ERROR SET IN MORE THAN ONE PARITY REGISTER
AFTER WRITING WRONG PARITY IN LOCATION
WHOSE ADDRESS IS IN RS

DOES MAP INDICATE THIS PARITY REGISTER CONTROLS THIS MEMORY?
PARITY REGISTER RESPONDED TO MEMORY
NOT INCLUDED IN ITS MAP
PARITY REGISTER'S ADDRESS IS POINTED TO BY RL ADDRESS OF LOCATION CAUSING PARITY ERROR IS IN RS

STORE REGISTER ADDRESS

BRANCH UNTIL ALL THE PARITY
REGISTERS HAVE BEEN CHECKED
SAVE DATA FROM LOCATION UNDER TEST
DID BICB CHANGE DATA?
NO CONTINUE
DATA WAS MODIFIED BY THE BICB WHICH GOT A PARITY ERROR TRAP - SINCE PARITY ERROR TRAP OCCURRED, CONTENTS SHOULD NOT HAVE BEEN MODIFIED, AS CONTAINS ADDRESS OF TEST LOCATION. TREG CONTAINS ADDRESS OF PARITY REGISTER SENSING ERROR
WAS PARITY ERROR SET IN ANY REGISTERS?
YES - BRANCH
PARITY TRAP OCCURRED ON READING WRONG PARITY (WITH AE SET) BUT NO REGISTERS HAD PARITY ERROR BIT SET.
RS CONTAINS THE ADDRESS OF THE TEST LOCATION.
NO TRAP AFTER WRITING AND READING.
WRONG PARITY WITH RE SET- VIRTUAL
ADDRESS OF LOCATION IS IN RS.
(MAPPED THRU KERNEL) PAGE IS 16.
LOW BYTE IF ODDLOC IS POSITIVE.
HIGH BYTE IF IT IS NEGATIVE.
NOTE THAT WWP AND RE WERE CLEARED.
BEFORE ERROR PRINTOUT.

WHERE WRONG PARITY DATA IS READ BACK, SHOULD ENTER HERE VIA TRAP TO 114.
PARITY TRAP OCCURRED- BEFORE CHECKING.
IT, CLEAR WWP AND RE IN ALL REGISTERS.

;FIND REGISTER THAT SENSED THE ERROR
;SETUP PTR TO INDICATOR

;DOES THIS CONTROL EXIST?
;NO BRANCH

;YES- IS ERROR SET?
;NO BRANCH

;YES- WAS IT SET IN ANY OTHER REGISTER ALSO?
;NO- BRANCH

;ERROR SET IN MORE THAN ONE PARITY REGISTER
AFTER READING WRONG PARITY IN LOCATION.
WHOSE VIRTUAL ADDRESS IS IN RS.
DOES MAP INDICATE THAT THIS REGISTER
CONTROLS THIS MEMORY?

;YES- BRANCH
PARITY REGISTER RESPOND TO MEMORY.
NOT INCLUDED IN ITS MAP. RI POINTS TO
THE PARITY REGISTER'S ADDRESS. RS
CONTAINS VIRTUAL ADDRESS OF THE LOCATION
BEING TESTED (MAPPED THRU KERNEL)
PAGE 1.
STORE REGISTER ADDRESS
STORE PTR TO INDICATOR

;INCREMENT PTR TO INDICATOR
LOOP UNTIL ALL THE PARITY REGISTERS
HAVE BEEN CHECKED.
SAVE CONTENTS OF LOCATION UNDER TEST
DID BICB CHANGE DATA?
NO CONTINUE
DATA WAS MODIFIED BY THE BICB WHICH
 GOT A PARITY ERROR TRAP- SINCE PARITY
TRAP OCCURRED, CONTENTS SHOULD NOT
0002 001024 005767 170636 TST TREG
0007 001030 001003 BRN 3%
0012 001018 ERROR
0017 001032 104002

0014 001013 000423 BR 4$
0019 001016 000102 BNE 3$
0024 001010 000001 CMP #1 (R4)
0029 001014 0041 BNE 4$

0034 001014 017701 MOV @TREG, R1
0039 001015 017701 BIC #170037, R1
0044 001016 015052 MOV R5, R2
0049 001017 016377 MOV R5, R2
0054 001018 000300 BIC #163777, R2
0059 001019 006305 SWAB R2
005E 00101A 006305 ASL R2
0063 00101B 006305 ASL R2
0068 00101C 006305 ADD #2KPA1, R2
0073 00101D 001401 CMP R1, R2
0078 00101E 001004 BEQ ERRORS

0083 00101F 002626 4$
0088 001020 005777 CMP (SP)+ (SP)+
0093 001021 005777 CM17: MOV (5) (5)
0098 001022 005777 CLR @TREG
009D 001023 005777 TST $R5
00A2 001024 005777 TST @TREG
00A7 001025 000001 BPL
00A8 001026 004002 ERROR

00B3 001027 005167 170324 COM ODDFLG
00B8 001028 100041 BRN
00BC 001029 005726 TST (5) +
00C4 00102A 005726 CMP R5, #0000
00CC 00102B 000527 TST (5) +
00D1 00102C 005000 BLO 1$
00D6 00102D 000207 RTS $7
00E1 00102E 000167 177360 1$: JMP WWP17R

; PARITY ERROR ADDRESS BITS CORRECT

; PARITY ERROR ADDRESS BITS (PARITY REGISTER BITS 11-5) INCORRECT
; "TREG" CONTAINS ADDRESS OF PARITY REGISTER
; AS CONTAINS THE VIRTUAL ADDRESS OF THE TEST LOCATION
; (MAPPED THRU KERNEL PAGE 1)
; RESTORE TEST LOCATION TO FIX BAD PARITY
; CLEAR ERROR BIT IN THE PARITY REGISTER
; LOG LOCATION TO SEE IF PARITY IS GOOD
; IS PARITY ERROR SET?
; NO BRANCH
; WRITING LOCATION WITH WRITE WRONG
; Parity Error Didn't Clear Bad Parity
; "TREG" CONTAINS THE ADDRESS OF THE PARITY REGISTER
; AS CONTAINS THE VIRTUAL ADDRESS OF THE TEST LOCATION
; (MAPPED THRU KERNEL PAGE 1)
; CHECK BYTE ERROR
; BRANCH IF HIGH BYTE NOT YET TESTED
; UPDATE ADDRESS POINTER
; THIS 4K DONE?
; NO TEST NEXT LOCATION
; YES, RETURN TO CHECK FOR NEXT
; BANK TO BE TESTED
; GO AND TEST NEXT LOCATION
D04

COMFAO, MEMORY PARITY TEST
COMCAF.P11 13-JAN-78 12:13

XFR1: SCOPE

; IF THE FIRST BANK (BANK 0) IS PARITY MEMORY, SETUP TO TEST IT
; COPY THE FIRST 4K TO THE SECOND 4K MOVE THE STACK POINTER TO BANK 1.
; AND THEN JUMP TO THE COPY OF TEST20 IN BANK 1.

MOV $2.BITPT
JMP TEST14

; IF THE PROGRAM IS RUNNING UNDER ACT THEN
; GO TO DONE

JMP DONE

; IF THERE IS A SECOND 4K(BANK 1)?
; BRANCH IF YES, NO- DONE WITH TEST

BIT #1.PMEML
JMP DONE

; CLEAR STATUS REGISTER
OUT: JCR #PS
CLR R1
JMP DONE

; CLEAR ALL PARITY REGISTERS
JSR #7.CLRPAR

; RO IS COUNTER TO MOVE 4K
MOV #17770,RO

; R1 POINTS TO LOCATION IN BANK O
CLR R1

; COPY FROM BANK O TO BANK 1
MOV $R1,20000(R1)

; MOVE POINTER
MOV R0

; DONE WITH 4K
JMP DONE

; YES, MOVE STACK POINTER TO POINT TO BANK 1
ADD $20000.SP

; UPDATE addresses used in ERROR TypeOUT
JMP TEST20+20010

; GO TO TEST 20 IN BANK 1

************************************************************************************

; IF FIRST 4K IS PARITY MEMORY, CHECK IT WITH A SERIES OF PATTERNS
; THIS SUBTEST IS RUN IN BANK 1 (20000 ABOVE THE ADDRESSES IN THE LISTING)

MOV SPR🏀$(-SP)

; THESE 2 LINES DO THE SAME AS A SCOPE WITHOUT
; USING AN EMT

MOV #00105PLAY
JSR #2.SCOREC

; LOAD THE TEST NUMBER INTO THE DISPLAY

MOV #20,00105PLAY
JSR #7.CLRPAR

; CLEAR ALL PARITY REGISTERS
JSR #7.CLRPAR

; INITIALIZE PATTERN POINTER
MOV $PARPAT+20000,R4

; INITIALLY CLEAR BANK 0
CLR RO
CLR $R5
CLR $R6
CLR $R7

; SETUP TRAP RETURN
MOV $TRAP20+20000,AR

; SETUP TO SET ACTION ENABLE IN ALL
; PARITY REGISTERS PRESENT
MOV #MPRO+20000,R1

************************************************************************************
2314 010530 001003
2315 010532 012771 000001 000000
2316 010540 062791 000010 000010
2317 010544 020127 020266
2318 010550 103765
2319 010552 004767 000000 000000
2320 010555 000767 000022
2321 010556 005724 000000
2322 010560 005714
2323 010562 001373
2324 010564 004767 003222
2325 010570 012737 000116 000116
2326 010576 000526

BNE .+10
MOV .R
c(R1)
ADD #10,R1
CMP R1,PARVEC+2
BLO 2$

; SET ACTION ENABLE IF REGISTER IS PRESENT
; BRANCH UNTIL ALL REGISTER ADDRESSES
; HAVE BEEN CHECKED
; EXERCISE THIS 4K
; UPDATE PATTERN
; LAST PATTERN?
; NO LOOP
; CLEAR ALL PARITY REGISTERS
; RESTORE TRAP CATCHER
; GO TO NEXT TEST

3$: JSR \%1, CKBKO
TST (4)$
TST (4)$
BNE 3$
DONE20: JSR PC, CLRPAR
MOV #PARVEC+2, #PARVEC
BR TEST21

F04

; PARITY MEMORY TEST ROUTINE
; WRITES AND CHECKS EACH LOCATION IN BANK D (EXCEPT 114 AND 116)
; WITH VALUE POINTED TO BY R4

XBDK: CLR R4
1$: MOV (4), (5)
    CMP (4), (5)
    BEQ 6%
    MOV #0, - (SP)
    JSR PC, EER
    JSR PC, EER
    JMP DONE20

4$: TST (5)+
    CMP R5, #114
    AND R4, R5
    CMP R5, #20000
    BLO 1%
    CLR TREG
    MOV @MPRO+20000, R1
    BNE +10
    TST @R1
    ADD @R1
    CMP RI, @TREG+20000
    BLO 2%
    RTS 2%
    MOV @SPS, -(SP)
    JSR PC, EER
3$: JSR PC, EER

7$: CMP (SP)+, (SP)+
    JMP DONE20

; PARITY TRAP SERVICE (NO TRAPS TO 114 SHOULD OCCUR IN THIS SUBTEST)

TRAP20: CLR TREG
1$: MOV @MPRO+20000, R1
    BNE +10
    TST @R1
    ADD @R1
    CMP RI, @TREG+20000
    BLO 1%
    MOV @SPS, -(SP)
    JSR PC, EER
2$: JSR PC, EER

5$: BR 5%
    SAVE CONTENTS OF PARITY REGISTER
    CLEAR PARITY REGISTER
    READ LOCATION
    PARITY ERROR SET?
3$: CLR R1
    TST @R3
    BSET 3$
    4$: BSET 4$

Done20: MOV @MPRO+20000, R1
        JMP DONE20


G04

FORCE PARITY ERROR IN EACH LOCATION IN BANK 0
NOTE THAT THIS SUBTEST IS EXECUTED IN BANK 0 (200000 ABOVE ADDRESSES)
IN THE LISTING MAKE SURE THAT WROAN PARITY IN EACH BYTE CAN BE DETECTED
AND THAT WHEN GOOD PARITY IS WRITTEN AND READ NO PARITY ERROR IS DETECTED
CHECK ERROR ADDRESS BITS (PARITY REGISTER BITS 5-11)

TEST 21: MOV #SPS 00000 MOV @RSP 00000

SAME AS SCOPE WITHOUT DOING ENT

LOAD THE TEST NUMBER INTO THE DISPLAY
CLEAR ALL PARITY REGISTERS
CLEAR PARITY ERRORS IN BANK 0

WRITE WRONG PARITY TEST ROUTINE
USING SAME DATA VALUE WRITES AND CHECKS PARITY IN WRONG STATE
AND THEN IN CORRECT STATE TO PROVE THAT PARITY BITS TOGGLE

WHEN 21: CLR HR CLP 00000 MOV @SPS 00000
CLEAR FLAG TO INDICATE TESTING LOW BYTE
STORE DATA FOR USE BY ERROR TYPEOUT ROUTINE
INITIALIZE LOCATION
SETUP TO SET WRITE WRONG PARITY
IN ALL REGISTERS

WHEN 21A: MOV @SPS 00000, R1
SET TEST ADDRESS POINTER

WHEN 31: BIT #1 (1)
;SET WRITE WRONG PARITY

WHEN 32: MOV @WEP 3R1
;YES, BRANCH

WHEN 33: MOV #253, R5
;NO, WRITE WRONG PARITY IN LOW BYTE

WHEN 34: MOV #253, R5
;TESTING HIGH BYTE
; DETECT WRONG PARITY WITH DATA
; WRITE WRONG PARITY IN HIGH BYTE
; CLEAR WWP IN ALL PARITY REGISTERS
; SET UP TO CLEAR WWP IN ALL PARITY REGISTERS
; POINTER TO INDICATOR
; CLEAR WWP IN ALL PARITY REGISTERS
; CHECK TO SEE IF ANY REGISTER DETECTED
; THE PARITY ERROR
; CHECK PARITY REGISTER
; BRANCH IF PARITY ERROR IS NOT SET
; WAS PARITY ERROR SET IN ANY OTHER REGISTER?
; SET UP TO DO ERROR CALL VIA A JSR
; ERROR- PARITY ERROR SET IN MORE THAN
; ONE REGISTER AFTER WRITING WRONG PARITY
; IN LOCATION WHOSE ADDRESS IS IN RS
; DOES MAP INDICATE THAT THIS REGISTER
; CONTROLS THIS MEMORY?
; YES, BRANCH
; NO- SETUP TO DO ERROR CALL VIA A JSR
; ERROR- PARITY REGISTER RESPONDED
; TO MEMORY WHICH IS NOT IN ITS MAP.
; RS POINTS TO PARITY REGISTER'S ADDRESS.
; STORE REGISTER ADDRESS
; BRANCH UNTIL ALL REGISTERS HAVE BEEN
; CHECKED
; WAS PARITY ERROR SET IN ANY REGISTER?
; YES- BRANCH
; NO- SETUP TO DO ERROR CALL VIA A JSR
; ERROR- NO REGISTER HAS PARITY ERROR
; SET AFTER READING WRONG PARITY IN LOCATION
; WHOSE ADDRESS IS IN RS
; IS THIS A CORE PAR REG?
; NO BRANCH(MOS PAR REG DOES NOT
; STORE ADDR BITS OF PAR ERROR)
; GET PARITY REGISTER CONTENTS
; MASK ALL BUT ERROR ADDRESS BITS
; GET ADDRESS OF LOCATION UNDER TEST
; POSITION BITS IN R2
PARITY ERROR ADDRESS BITS CORRECT?
BRANCH IF YES
NO- SETUP TO DO ERROR CALL VIA A JSR
ERROR- ADDRESS BITS (PARITY REGISTER)
BITS 5-11 INCORRECT- ADDRESS OF PARITY
REGISTER IS CONTAINED IN LOCATION "TREG"
ADDRESS OF TEST LOCATION IS IN RS
RESTORE TEST LOCATION TO FIX BAD PARITY
CLEAR ERROR BIT IN PARITY REGISTER
READ LOCATION TO SET IF PARITY IS GOOD
CHECK PARITY ERROR BIT
BRANCH IF NOT SET
SETUP TO DO ERROR CALL VIA A JSR
ERROR- WRITING LOCATION WITH WRITE
WRONG PARITY CLEAR DIDN'T CLEAR BAD PARITY
ADDRESS OF LOCATION IS IN RS
"TREG" +20000 CONTAINS THE ADDRESS
OF THE PARITY REGISTER
TOGGLE BYTE INDICATOR
BRANCH IF HIGH BYTE NOT YET TESTED
UPDATE ADDRESS POINTER
THIS 4K DONE?
LOOP TILL ALL 4K HAS BEEN TESTED
CLEAR ALL PARITY REGISTERS
SETUP TO CALL SCOPE VIA JSR
SCOPE

COPY SECOND 4K BANK BACK TO FIRST 4K AND RETURN TO FIRST 4K BANK
XFR: MOV R17770,RO
RO IS USED AS A COUNTER
CLAR 005001 POINTS TO THE CURRENT LOCATION
COPY BANK 1 TO BANK 0

RESTORE STACK POINTER

DOES THE CONSOLE HAVE A SWITCH REG.?
IF YES THEN GO TO 25
OTHERWISE REINITIALIZE THE ADDRESS OF SOFTWARE
SWITCH REGISTER
RETURN TO BANK 0

AT THIS POINT EXECUTION RETURNS TO BANK 0

REINITIALIZE STACK POINTER
;TYPE BELP, "END PASS:" AND PASS COUNT
;ALLOW TIME FOR END PASS MESSAGE TO PRINT
;BRANCH IF NO
;GO TO MONITOR

;SWITCH B SET?
;HALT AT END OF PASS SET
;IF NO TERMINAL SKIP
;WAIT FOR IFF TO FINISH SO THAT RESET
;DON'T Clobber THE BELL
;OUTPUT A NULL

;PASS COUNT

;CREATE MAP INDICATING WHERE 4K BLOCKS OF MEMORY ARE PRESENT
;MAP MEMORY USING KT11 - MAX OF 124K POSSIBLE

;MAP MEMORY USING KT11 - MAX OF 124K POSSIBLE

;INDICATE KT11 PRESENT
;SET UP CORE MAPS
;SET UP 4K POINTER
;SET UP FOR TIME CUTS
;UPDATE TEST ADDRESS
;UPDATE BANK POINTER
;BRANCH IF NOT DONE WITH 64K SECTION
;NO, INIT T'BANK TO SELECT KERNEL PAGE 1
;SET UP FOR ACCESS TO 28K BANK
;INITIALLY MAP ALL PAGES MR. BANK 0
;MAP KERNAL 0 TO BANK 0, RW
;MAP KERNAL 7 TO THE EXTERNAL BANK, RW
;SET KERNAL 1 RW AND TURN ON K'TII
;INDICATE K'TII NOW IN USE
;TURN ON K'TII
;SHIFT BANK INDICATOR
;BRANCH IF FIRST 64K NOT DONE
;IF FIRST 64K DONE, SETUP FOR
;SECOND 64K
;INDICATE NOW TESTING HIGH 64K
;UP TO EXTERNAL BANK YET?
;NO, CONTINUE
;YES, TURN OFF K'TII AND EXIT
;IS THIS 4K PRESENT?
;NO- BRANCH
;YES, MAP PAGE 1 TO THIS BANK
;GO TEST FOR PARITY MEMORY

;******************************************************************************
;ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
;AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
;******************************************************************************

;MAP:
;CLEAR ALL PARITY REGISTORS PRESENT
;THE PARITY REGISTORS CONTROL MEMORY
;AS FOLLOWS:
;SET UP POINTER
;DONE WITH MAP?
;YES, BRANCH
;INITIALIZE TRANSITION FLAG (USED TO
;FIGURE MEMORY LIMITS)
;INITIALIZE TO INDICATE NOTHING TYPED FOR
;THIS REGISTER YET
;GET LOW 64K MEMORY MAP WORD
;INITIALIZE 4K POINTER
;DOES THIS CONTROL EXIST?
;NO, GET ADDRES OF NEXT ONE
;YES, PRINT ITS ADDRESS

;******************************************************************************
;******************************************************************************
KEEP TRACK OF N OF K OF CORE
DOES THIS PARITY REGISTER CONTROL THIS 4K?
NO- BRANCH
YES- DOES IT CONTROL PREVIOUS 4K?
YES- DON'T TYPE IT
NO- SET FLAG INDICATING TRANSITION
CONVERT K CORE TO ASCII

TYPE "CONTROLS", AND ADDRESS OF CORE

INDICATE TYPED

DOES THIS PARITY REGISTER CONTROL PREVIOUS 4K?
NO- SKIP PRINTING
YES- TRANSITION OCCURRED- CLEAR FLAG
CONVERT K CORE TO ASCII

TYPE RIGHT AND RETURN

INDICATE TYPED

UPDATE BIT POINTER TO NEXT 4K
TEST NEXT 4K IF NOT DONE WITH 1ST 64K
64-128K DONE?
NO- BRANCH
YES- WAS ANY PARITY MEMORY FOUND FOR THIS REGISTER?
NO PARITY MEMORY WAS FOUND FOR THIS
REGISTER- EITHER WRITE WRONG PARITY
FAILED, PARITY ERROR GENERATE OR
DETECT FAILED, OR THE PARITY ERROR
BIT FAILED TO SET

UPDATE TO MAP WORD FOR THE UPPER 64K
RESET BIT POINTER
INDICATE LOW 64K DONE
LOOP
RETURN WHEN DONE

ROUTINE TO HANDLE BK SYSTEMS

SET UP POINTER FOR BANK 1
D05

MEMORY PARITY TEST

13-JAN-78 12:27 PAGE 55

;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT

CLRPAR: MOV R1,-(SP)
MOV R2,-(SP)
MOV R3,PC,R1
ADD #PRO-.R1
MOV R4,PC,R2
ADD #REG-.R2
1$: BIT @1,R1
IS THIS REGISTER PRESENT?
CLR @1,R1
CLEAR ALL PARITY REGISTERS

;ROUTINE TO MAP KERNAL 0 TO BANK 0 READ/WRITE

;KERNEL 1 READ/WRITE BUT BANK MAPPED BY CALLING ROUTINE.

;KERNAL 7 TO EXTERNAL BANK, READ/WRITE

MAP1: MOV #7/MOB6, #KPDAR
MOV #7/MOB6, #KPDAR
MOV #7/MOB6, #KPDAR
MOV #7/MOB6, #KPDAR
CLR #KPDAR
MOV R1, #KPDRO
RTS %7

;ROUTINE TO LOCATE THE FIRST PARITY MEMORY ADDRESS (ABOVE BANK 0)

;CORRESPONDING TO A GIVEN PARITY REGISTER-REQUIRES THAT THE ROUTINES

;MAPPED AND MAPREC HAVE ALREADY BEEN RUN

;TO USE, PUT THE ADDRESS OF THE REGISTER IN R0 (I.E. POINT TO

;TO MAP TABLE)-THE DESIRED ADDRESS IS RETURNED IN R1, USING KERNEL PAGE 1 IF

;K11 IS PRESENT

LOCATM: MOV R2,-(SP)
MOV #00,R2
SKIP USE OF BANK 0
1$: MOV #01,R1

;K11 PRESENT?

;YES, BRANCH

;NO, CHECK ONLY FOR MEMORY IN FIRST 28K

;ERROR RETURN

;IF NO MEMORY FOUND IN 1ST 28K, GIVE

;YES

;NO

;ERROR

;RETURN

;FAIL

;ERROR

;RETURN

;FAIL

;ERROR

;RETURN

;FAIL
E5

SEQ 0056

BNE LOCAT1
ADD $200 , R2

BCE $1

MOV R1 , R1

BR ( RQ )

BCE $1

MOV R1 , R1

MOV ( SP )+ , R2

RTS $7

LOCAT1 : TST NOKT

BR $2

MOV R2 , R1

BR ( RQ )

BR ( SP )+ , R2

ADD LSAV , R1

MOV (SP)+ , R2

LAV : 0

: SAVE LOADER IN TOP OF FIRST 4K

: SAVE CONTENTS OF TIMEOUT VECTOR

: SETUP TO RTI ON TIMEOUT

: SAVE REGISTERS

: IF TIMEOUT, C BIT WILL STILL BE SET

: IF NO TIMEOUT, C BIT WILL BE CLEAR

: TIMEOUT OCCURRED, CHECK FOR NEXT LOWER BANK

: ONLY 4K OF 1ST 28K PRESENT- EXIT

: THIS BANK IS HIGHEST OF 1ST 28K- COPY

: LOADER AREA TO BANK 0

: INDICATE LOADER HAS BEEN MOVED TO

: THE TOP OF THE FIRST 4K

: YES, BRANCH

: NO, CHECK TO SEE IF NEXT 4K CORRESPONDS

: CHECK HIGH 64K

: NO PARITY MEMORY CORRESPONDS TO

: THIS REGISTER- RETURN WITH ERROR

: NO K11 PRESENT?

: YES- SETUP R1 TO REFERENCE 4K

: BANK USING KERNEL PAGE

: SET UP R1 TO REFERENCE 4K BANK

: WITHOUT K11

: RESTORE R2

: AND RETURN
;ROUTINE TO RESTORE THE LOADER FROM BANK 0 (WHERE IT WAS SAVED) TO THE
;HIGHEST BANK IN THE FIRST 28K OF MEMORY

LDRSVD: TST
BEQ RSTL0
MOV #2, R1
MOV #20000, R1
#152234, R1
1S: SEC
BCC 2S
SUB #20000, R1
CMP R1, #20000
#152234, R1
RSTL0
2S: MOV #32234, R2
BR RSTL0
3S: MOV (R2) + (R1) +
CMP R2, #40000
BLO 3S
RSTMG: RTS PC
;TYPE MESSAGE "LOADER RESTORED"
;TO HIGHEST BANK IN FIRST 28K

;SCAN ALL MEMORY FOR BAD PARITY, TYPE 18 BIT ADDRESSES OF
;LOCATIONS FOUND TO BE BAD, WRITE INTO LOCATIONS WITH GOOD PARITY

PSCAN: MOV #1, R1
MOV R2, (SP)
MOV R3, (SP)
MOV R4, (SP)
MOV R5, (SP)
MOV R6, (SP)
MOV R7, (SP)
MOV #6, R4
CLR #3
MOV R1, (SP)
CLR #16, #114
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLR #16
CLEAR REGISTERS AND LOCATIONS TO BE
ALTERNED

;SETUP TIMEOUT TRAPCATCHER

;SETUP PARITY TRAP TRAPCATCHER

;KILL PRESENT?

;YES, BRANCH

;R2 CONTAINS TEST ADDRESS

;R3 USED AS A BIT POINTER

;CLEAR ALL PARITY REGISTERS

;READ LOCATION TO CHECK FOR BAD PARITY

;SETUP TO SCAN REGISTERS FOR PARITY

;ERROR SET
G05

COMPAQ, MEMORY PARITY TEST
COMPAQ P11 13-JAN-78 12:13

014674 001003 000000 000000 000000
BNE .+10 :PARITY ERROR SET?

014676 005771 000000
TST #1(R1) :YES- BRANCH

014677 038424 000000
BRA 6S :NO- CHECK NFAT REGISTER

014678 002701 000000 000000
ADD #10,R1

014679 000812 000000
CMP R1,R1

01467A 003265 000000
BLO 5S :LOOP UNTIL ALL REGISTERS HAVE BEEN CHECKED

01467B 004644 000000 000000
ADD #2,R2

01467C 00279F 001777
BIT #17777,R2

01467D 00017F 000000
CMP R1,R3

01467E 001353 000000
BNE 1S :MOVE ADDRESS POINTER

01467F 000000
ADD #4K

014680 001393 000000
BIT R3,MEML

014681 023B27 000000
BHS PSSCAN

014682 003036 000000
BIT R3,MEML

014683 016470 000000
BNE 2S :IS THIS MEMORY PRESENT?

014684 068702 000000
ADD #20000,R2

014685 007660 000000
BR 5S :YES, GO TEST IT

014686 010267 000000 000010
6S: MOV R2,PSADRS

014687 004567 000000 000002
JSR R5,OEACHV

014688 015020 000000
PSADRS

014689 017940 000000
MPERI

01468A 000000
TYPE

01468B 014718 000000
MPER

01468C 014720 000000
BIT #2000,85WR

01468D 014722 002777 000000
BIT #164144

01468E 014724 014000
HALT

01468F 014726 000000
CLR #1(R1)

014690 014728 001000
TST #0(R1)

014691 01472A 000000
BPL .+4

014692 01472C 000000
ERROR 4S

014693 01472E 014000
PSSCAN: JSR PC, CLRPAR

014694 014730 004276 17704
PSCANX: MOV (SP)+,#8116

014695 014732 012637 00011
MOV (SP)+,#8114

014696 014734 012637 000006
MOV (SP)+,#816

014697 014736 012637 000004
MOV (SP)+,#814

014698 014738 012637 00002
MOV (SP)+,#813

014699 01473A 012637 000004
MOV (SP)+,#812

01469A 01473C 012637 000001
MOV (SP)+,#811

01469B 01473E 000000
MOV (SP)+,#87

01469C 014740 000000
RIS .+7

01469D 014742 000000
PSADRS: 0

01469E 014744 000000
PSAD: 0

01469F 014746 000000
STD: 0

0146A0 014748 000000
STC: 0

0146A1 01474A 000035
RETURN

0146A2 01474C 000027
:SCAN ALL MEMORY FOR BAD PARITY USING KTI1

0146A3 015024 011774 164222
:TYPE 18 BIT ADDRESSES OF LOCATIONS FOUND BAD, AND WRITE GOOD PARITY BACK IN

0146A4 015030 003239 000001
PSCAN: MOV @KPARI,-(SP)

0146A5 015036 000000 177572
:SAVE CONTENTS OF KERNEL PAR!

0146A6 015040 004767 176646
:SKIP IF KTI1 IS ALREADY ON

0146A7 015044 000000
BNE 1S

0146A8 015046 000100
JSR PC, KPAR

0146A9 015050 000000
:MAP KERNEL 0 TO BANK 0,RW
H05

MAP KERNEL 7 TO THE EXTERNAL BANK, RW
MAP KERNEL 1 RW, AND TURN ON KT11
CLEAR FLAG TO INDICATE CHECKING FIRST 64K
INITIALIZE TO BANK 0
R3 IS USED AS A BIT POINTER
TESTING TOP 64K?
YES, BRANCH
NO, IS PARITY MEMORY PRESENT IN THIS 4K?
YES, GO TEST IT
NO, CHECK FOR NEXT 4K
IS PARITY MEMORY PRESENT IN THIS 4K?
YES, GO TEST IT
NO, MAP TO NEXT 4K

BRANCH IF NOT END OF 64K
END OF TOP 64K?
YES, GET READY TO EXIT
NO, SET FLAG INDICATING DONE WITH LOWER 64K

R2 USED AS ADDRESS POINTER
CLEAR ALL PARITY REGISTERS
READ LOCATION
SETUP TO SCAN REGISTERS FOR PARITY ERROR SET

PARITY ERROR SET?
YES, BRANCH
NO, CHECK NEXT

LOOP UNTIL ALL REGISTERS HAVE BEEN CHECKED
UPDATE TEST ADDRESS POINTER
DONE WITH BANK?
NO, LOOP
YES, GO CHECK FOR ANOTHER BANK
PARITY ERROR OCCURRED- GET 18 BIT
OCTAL ADDRESS OF BAD LOCATION

CONVERT LOW 16 OCTAL BITS TO ASCII
CHANGE TO ASCII FOR 18 BITS

TYPE ADDRESS OF LOCATION WITH BAD PARITY

SWITCH IO SET?

NO - BRANCH
HALT ON BAD PARITY SET
REWRITE LOCATION CONTAINING BAD PARITY
CLEAR PARITY ERROR BIT
READ LOCATION TO SEE IF PARITY IS NOW GOOD
CHECK PARITY ERROR BIT

REWRITING LOCATION DID NOT CLEAR BAD PARITY
GO TEST NEXT LOCATION

-PIC ROUTINE TO OUTPUT A SERIES OF ASCII MESSAGES (CALLED VIA JSR RS)

PIC: MOV)*(R5)+, TYPBX
GET ADDRESS OF MESSAGE
CMP 0-1 TYPBX
TERMINATOR?
BNE TYPAX
NO, BRANCH
RTS ½S
YES, RETURN

TYPAX: MOV $(TYPAX), -(SP)
SETUP TO CALL TYPE ROUTINE VIA JSR
TYPBX
PC, $TYPE
TYPE ASCII MESSAGE

SUBROUTINE FOR OCTAL TO ASCII CONVERSION

OACNV: MOV *(RS)+, OACNVX
GET OCTAL VALUE
MOV *(RS)+, OACCDST
GET DESTINATION ADDRESS
MOV *(RS)+, OACNT
GET CONVERT COUNT
MOV OACNVX, -(SP)
DEVELOP ADDRESS TO STORE 1ST CHAR.
OACNYVA MOV OACNVX, -(SP)
ISOLATE LEAST SIGNIFICANT DIGIT
ADC #0, OACDESt
CONVERT DIGIT TO ASCII
MOV *(RS)+, OACDST
STORE ASCII CHARACTER
DEC OACNT
DONE ALL DIGITS?
ROR OACNVX
BRANCH IF NOT DONE
ROR OACNVX
DONE, EXIT

SUBROUTINE FOR BINARY TO DECIMAL ASCII CONVERSION

BDCNV: SAVO
SAVE REGS
MOV #DEVAL, %0
SET UP ADDR TO STORE DECIMAL ASCII
3336 015732 012737 015776 000024 PWRDN: MOV #PWRUP, @24
3337 015740 012701 000566 MOV #MPRO, R1
3338 015744 012711 000001 JS: BIT R1, @3R1
3339 015750 001002 BNE +6
3340 015752 005071 000000 CLR @R1)
3341 015756 0627C1 000010 ADD @10, @1
3342 015762 020127 000766 CMP R1, @TREG

; SET UP FOR POWER UP
; CLEAR PARITY REGISTERS IN CASE
; MAP IS SET
```
BLO 1%
MOV SP,SPSRV ;POWER DOWN WALT
MOV SPSPSRV ;SET UP FOR POWER DOWN
PWRUP: MOV #PWRDN,#24 ;STALL SO OUTPUT WON'T BE GARBLED
MPWRF ;TYPE RECOVERY MESSAGE
RSTART ;RESTART

EMT HANDLER

EMTINT: MOV (SP),-(SP) ;GET SAVED PC
SUB #2,(SP) ;DECUMENT PC BY 2
MOV #3,(SP) ;CALL
CPB (SP),EMTLIM ;CHECK IF CALL WITHIN LIMITS
BLO EMTA ;CALL IS NOT WITHIN LIMITS
HALT ;EMT ARG X 2
EMTA: ROL (SP) ;REMOVE 7 MSB
MOV @SP,(SP) ;FORM EMT RTN ADDRESS
ADD #EMTMTAB,(SP) ;GO TO EMT RETURN
JMP @SP

:EMT DEFINITIONS AND ASSIGNMENTS
:EMTMTAB:

TYPE=EMT+EMTX
$TYPE
SPECT=EMT+EMTX
SCPEC
ERROR=EMT+EMTX
ERR
ERRRP=EMT+EMTX
ERRP
ERRRS=EMT+EMTX
ERRS
SAV04=EMT+EMTX
SV04
RST04=EMT+EMTX
RST4
RST05=EMT+EMTX
RST5
SAV05=EMT+EMTX
SV05
EMTLIM: EMTX-1

:SUBROUTINE TO SAVE REGS 0-4
SV04: MOV (SP)+,-12,(SP) ;MOVE PC+PS UP STACK
3397 016126 012866 17724
MOV (SP)+,-12,(SP)
3398 016132 012767 17724
MOV *(RTI,SV05)
```
016140 000411

:SUBROUTINE TO SAVE REGS 0-5 + PLACE EM'T PC IN R5

$0053: MOV  MOVS
016142 0177:7 000240 000030 $0064: MOV  MOVS
016150 000400 177762

:SUBROUTINE TO SAVE REGS 0-5

$0064: MOV  MOVS
016152 012666 177762
016156 012666 177762

$0075: MOV  MOVS
016160 010346 177762
016164 010346 177762

$0086: MOV  MOVS
016166 010346 177762
016169 010346 177762

$0097: MOV  MOVS
016170 010146 177762
016173 010046 177762

$00A8: MOV  MOVS
016179 010046 177762
01617B 024646 177762

$00B9: MOV  MOVS
016186 000002 177762
016192 000002 177762

$00CB: MOV  MOVS
0161A8 000002 177762
0161B0 000002 177762

:SUBROUTINE TO RESTORE REGS 0-4

$00E8: MOV  MOVS
016210 022626 177762
016212 022626 177762

$00F9: MOV  MOVS
016216 022626 177762
016218 022626 177762

$010A: MOV  MOVS
016220 022626 177762
016222 022626 177762

$011B: MOV  MOVS
016224 022626 177762
016226 022626 177762

$012C: MOV  MOVS
016228 022626 177762
01622A 022626 177762

$013B: MOV  MOVS
01622C 022626 177762
01622E 022626 177762

$014D: MOV  MOVS
016230 022626 177762
016232 022626 177762

:SUBROUTINE TO RESTORE REGS 0-5

$015E: MOV  MOVS
016236 010566 000020
016239 010566 000020

:ROUTINE TO LOOP THRU A SINGLE INSTRUCTION TEST

:LOAD THE STARTING ADDRESS OF THE TEST

:YOU WISH TO RUN (THE ADDRESS OF THE TEST)

:TAG) AT THE 1ST HALT, SET SWITCH REGISTER
OPTIONS AT THE 2ND HALT
NOTE THAT SWI WILL MUST BE DOWN AFTER THE 2ND HALT

; CLRT: CLR $00

HALT: MOV $0000,0

; WAIT FOR STARTING ADDRESS
LOAD STARTING ADDRESS IN RETURN
ADD 2 TO POINT TO INSTRUCTION AFTER
SET SR OPTIONS
SET FLAG
CHECK SWI2
BRANCH IF NOT SET
CLEAR TRACE BIT
SKIP NEXT INSTRUCTION
SET TRACE BIT
JUMP TO TEST

; SCOPE AND/OR ITERATION LOOP FOR EACH TEST 64 TIMES
A SETUP ROUTINE SHOULD INITIALIZE RETURN AND IMAX

; SETUP ROUTINE SHOULD INITIALIZE RETURN AND IMAX

; TEST SR FOR SCOPE
; YES, SCOPE
; NO-TEST FOR ITERATION
; INHIBIT ITERATION
; First PASS?
; YES, INHIBIT ITERATIONS
; USING SINGLE SUBTEST STARTUP?
; YES LOOP
; COMPARE CURRENT COUNT TO MAX NUMBER
; EXIT-DONE
; INCRTMENT COUNT
; REPOSTITION STACK
; RESTORE PREVIOUS PROCESSOR STATUS
; REPEAT TEST
; IF USING TESTX STARTUP, RETURN TO NORMAL FLOW
; CLEAR COUNT
; SAVE SCOPE RETURN POINTER
; ITERATION COUNT
; COUNT LOCATION FOR ITERATION LOOP
; ADDRESS OF LAST TEST

; ASCII MESSAGES

MED: .ASCII "MED:"

MNOM: .ASCII "MNOM:

MPC: .ASCII "MPC:

MICNT: .ASCII "MICNT:

MSTR: .ASCII "MSTR:

MTREG: .ASCII "MTREG:

MPR DATA: }

; ASCII MESSAGES

MED: .ASCII "MED:"

MNOM: .ASCII "MNOM:

MPC: .ASCII "MPC:

MICNT: .ASCII "MICNT:

MSTR: .ASCII "MSTR:

MTREG: .ASCII "MTREG:

MPR DATA: 
CCMFAF0, MEMORY PARITY TEST
MCC11 30A(1052)  13-JAN-78  12:27  PAGE 67

MT:  .ASCIZ 'NO PARITY MEMORY FOUND'<15><12>

MTR:  .ASCIZ 'NO PARITY REGISTER FOUND'<15><12>

MPRAF:  .ASCIZ '<15><12>

MTIT:  .ASCIZ '<15><15><12><15><12>'CCMFAF0 MS11, MA11-P, MF11-LP PARITY MEMORY TESTS'

MPWAF:  .ASCIZ '<15><12>'POWER FAILED'

MPGEND:  .BYTE 007

MPGEND:  .ASCIZ '<15><12>'END PASS = '

MPCINT:  .ASCIZ '<15><15><12>'BAD PAR FOUND IN LOC'

MPSBR:  .ASCIZ '<15><15><12>'BAD PAR FOUND IN LOC'

MPSERI:  .ASCIZ '<15><15><12>'REGISTER AT'

MX1:  .ASCIZ '<15><15><12>'MONITOR WILL NOT BE RESTORED FOR AN BK SYSTEM'

NOMON:  .ASCIZ '<15><15><15><12>'BAD PARITY SCAN COMPLETE'

PSMSG:  .ASCIZ '<15><12>'BAD PARITY SCAN COMPLETE'

PSMSG:  .ASCIZ '<15><12>'BAD PARITY SCAN COMPLETE'
MLDRSV: .ASCIIZ 'LOADERS SAVED IN BANK 1'

MPCOR: .ASCIIZ 'CORE PARITY REGISTER'

MPAROS: .ASCIIZ 'MOS PARITY REGISTER'

.EVEN
.END
<table>
<thead>
<tr>
<th>DISPRE</th>
<th>1661*</th>
<th>1777*</th>
<th>1897*</th>
<th>2082*</th>
<th>2083*</th>
<th>2303*</th>
<th>2415*</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOME</td>
<td>01124*</td>
<td>406*</td>
<td>732*</td>
<td>2270*</td>
<td>2273*</td>
<td>2540*</td>
<td>2547*</td>
</tr>
<tr>
<td>DOME10</td>
<td>00432*</td>
<td>1323*</td>
<td>1387*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME12</td>
<td>00054*</td>
<td>1419*</td>
<td>456*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME13</td>
<td>00530*</td>
<td>1584*</td>
<td>1649*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME14</td>
<td>00561*</td>
<td>1895*</td>
<td>1970*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME16</td>
<td>00065*</td>
<td>1796*</td>
<td>1823*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME17</td>
<td>00725*</td>
<td>1915*</td>
<td>1925*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME20</td>
<td>01054*</td>
<td>2104*</td>
<td>2116*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME24</td>
<td>00314*</td>
<td>1013*</td>
<td>1079*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME36</td>
<td>00326*</td>
<td>1103*</td>
<td>1131*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOME37</td>
<td>00377*</td>
<td>1280*</td>
<td>1288*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DSMA</td>
<td>177570</td>
<td>366*</td>
<td>515*</td>
<td>362*</td>
<td>362*</td>
<td>415*</td>
<td>415*</td>
</tr>
<tr>
<td>EMAT</td>
<td>01608*</td>
<td>3362*</td>
<td>362*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMINT</td>
<td>016030</td>
<td>397*</td>
<td>3358*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMTIL</td>
<td>016120</td>
<td>3397*</td>
<td>3397*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>EMTAB</td>
<td>016076</td>
<td>3367*</td>
<td>3342*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ENX</td>
<td>000011</td>
<td>3373*</td>
<td>3374*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERR</td>
<td>013562</td>
<td>2336*</td>
<td>2357*</td>
<td>2375*</td>
<td>2396*</td>
<td>2461*</td>
<td>2482*</td>
</tr>
<tr>
<td>ERR1</td>
<td>013532</td>
<td>2356*</td>
<td>2357*</td>
<td>2375*</td>
<td>2396*</td>
<td>2461*</td>
<td>2482*</td>
</tr>
<tr>
<td>ERR2</td>
<td>013534</td>
<td>2205*</td>
<td>2232*</td>
<td>2290*</td>
<td>2307*</td>
<td>2307*</td>
<td>2307*</td>
</tr>
<tr>
<td>ERR3</td>
<td>013646</td>
<td>2204*</td>
<td>2232*</td>
<td>2290*</td>
<td>2307*</td>
<td>2307*</td>
<td>2307*</td>
</tr>
<tr>
<td>ERR4</td>
<td>013666</td>
<td>2203*</td>
<td>2232*</td>
<td>2290*</td>
<td>2307*</td>
<td>2307*</td>
<td>2307*</td>
</tr>
<tr>
<td>ERR5</td>
<td>013700</td>
<td>2202*</td>
<td>2232*</td>
<td>2290*</td>
<td>2307*</td>
<td>2307*</td>
<td>2307*</td>
</tr>
<tr>
<td>ERR2</td>
<td>013710</td>
<td>2203*</td>
<td>2232*</td>
<td>2290*</td>
<td>2294*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRORP</td>
<td>104403</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERRAS</td>
<td>013364</td>
<td>2204*</td>
<td>2232*</td>
<td>2290*</td>
<td>2307*</td>
<td>2307*</td>
<td>2307*</td>
</tr>
<tr>
<td>F1TLE</td>
<td>005640</td>
<td>1677*</td>
<td>1679*</td>
<td>743*</td>
<td>745*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMPAR</td>
<td>001644</td>
<td>766*</td>
<td>798*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMPPA</td>
<td>002012</td>
<td>754*</td>
<td>793*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMPCR</td>
<td>002020</td>
<td>792*</td>
<td>795*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>GMPRO</td>
<td>002054</td>
<td>802*</td>
<td>805*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H1TAR</td>
<td>006300</td>
<td>430*</td>
<td>468*</td>
<td>1688*</td>
<td>1691*</td>
<td>1722*</td>
<td>1916*</td>
</tr>
<tr>
<td>HIORD</td>
<td>006553</td>
<td>440*</td>
<td>453*</td>
<td>2589*</td>
<td>2705*</td>
<td>2732*</td>
<td>2757*</td>
</tr>
<tr>
<td>ICMT</td>
<td>016460</td>
<td>2910*</td>
<td>3486*</td>
<td>3486*</td>
<td>3486*</td>
<td>3493*</td>
<td>3493*</td>
</tr>
<tr>
<td>IMAX</td>
<td>016456</td>
<td>280*</td>
<td>885*</td>
<td>996*</td>
<td>1662*</td>
<td>3486*</td>
<td>3492*</td>
</tr>
<tr>
<td>IND10</td>
<td>001014</td>
<td>269*</td>
<td>763*</td>
<td>834*</td>
<td>88*</td>
<td>907*</td>
<td>1007*</td>
</tr>
<tr>
<td>IND11</td>
<td>001016</td>
<td>269*</td>
<td>763*</td>
<td>834*</td>
<td>88*</td>
<td>907*</td>
<td>1007*</td>
</tr>
</tbody>
</table>
GO6

ENCAP MEMORY PARITY TEST
MACY11 30A(1052) 13-JAN-78 12:27 PAGE 72
CROSS REFERENCE TABLE -- USER SYMBOLS

INDC12 001090 568#
INDC13 001092 569#
INDC15 001096 570#
INDC15 000774 571#
INDC3 000776 572#
INDC2 001000 573#
INDC5 001002 574#
INDC7 001006 575#
INDC8 001010 576#
INDC9 001012 577#

INDX17= 010060 2097# 2108# 2181#
INST1 001414 1339# 134# 136#
INST2 001422 1345# 1367#
KPAR0 001250 568# 2979#
KPAR1 001252 669# 1785# 1792# 2094# 2100# 2229 2600# 2740# 3015# 3164 3171# 3180# 3188#
KPAR2 001254 670#
KPAR3 001256 671# 2978#
KPAR4 001258 664# 2975#
KPAR5 001260 665# 2975#
KPAR6 001262 666# 2977#
KSTAR 001264 659# 2650# 2718 2726#
LDMS0 017560 3082 3616#
LDMS2 014416 3052# 5059# 3065
LOCAT2 014432 1014 1104# 1158 1324 1420 1407 1586 2991#
LOCAT1 014224 3000 3005# 3013#
LOGCA 011672 611 2560#
LOGP10 004066 1319# 1331# 1386
LOGP11 004068 1411 1415# 1427 1454
LOGP12 004070 1477 1481# 1494 1555
LOGP13 005124 1581# 1599# 1647
LOGP14 005436 1672 1674# 1683
LOGP15 005706 1794 1794# 1800
LOGP21 001240 2424# 2477#
LOGP29 002760 1008# 1012
LOGP3 003216 1098# 1102
LOGP5 003372 1151 1155
LOGP7 003734 1279# 1287# 1297
LOGP15 005116 1792 1822
LOGP5 003270 1010# 1031 1055
LOGP5 003270 1100# 1115# 1130
LOGP7 003394 1153 1169# 1261
LOGP8 003396 1241# 1296# 1797# 2095# 2103 2105#
LOGP8 003396 1241# 1296# 1797# 2095# 2103 2105#
LOGP10 004060 1317# 1322
LOGP11 004062 1410 1416
LOGP12 004616 1476 1494
LOGP13 005116 1578# 1583
LOGP14 005446 1676# 1706
LOGP16 006264 1900 1904#
LOGP9 003726 1277# 1281
MAPRA 001770 5899# 2621#
MAPRB 012162 5899# 2621#
<table>
<thead>
<tr>
<th>Memory Location</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAPMEM</td>
<td>011756</td>
</tr>
<tr>
<td>MAPBR</td>
<td>012446</td>
</tr>
<tr>
<td>MAPR</td>
<td>012548</td>
</tr>
<tr>
<td>MAPRD</td>
<td>012549</td>
</tr>
<tr>
<td>MAPREG</td>
<td>012550</td>
</tr>
<tr>
<td>MAP1</td>
<td>010002</td>
</tr>
<tr>
<td>MCR</td>
<td>016666</td>
</tr>
<tr>
<td>MAR</td>
<td>017154</td>
</tr>
<tr>
<td>MARH</td>
<td>017151</td>
</tr>
<tr>
<td>MARQ</td>
<td>016550</td>
</tr>
<tr>
<td>MARE</td>
<td>001045</td>
</tr>
<tr>
<td>MAFUT</td>
<td>000546</td>
</tr>
<tr>
<td>MEQ</td>
<td>016464</td>
</tr>
<tr>
<td>MICTNT</td>
<td>016510</td>
</tr>
<tr>
<td>MSLRSV</td>
<td>017157</td>
</tr>
<tr>
<td>MADDR</td>
<td>016754</td>
</tr>
<tr>
<td>MADEV</td>
<td>016750</td>
</tr>
<tr>
<td>MPRAT</td>
<td>016775</td>
</tr>
<tr>
<td>MPRAS</td>
<td>017026</td>
</tr>
<tr>
<td>MONCNT</td>
<td>011754</td>
</tr>
<tr>
<td>MRC</td>
<td>016742</td>
</tr>
<tr>
<td>MFCNT</td>
<td>017401</td>
</tr>
<tr>
<td>MPEND</td>
<td>017361</td>
</tr>
<tr>
<td>MPK</td>
<td>016743</td>
</tr>
<tr>
<td>MPRDO</td>
<td>017274</td>
</tr>
<tr>
<td>MPROR</td>
<td>017570</td>
</tr>
<tr>
<td>MPRROS</td>
<td>017731</td>
</tr>
<tr>
<td>MPK8</td>
<td>000842</td>
</tr>
<tr>
<td>MPK8</td>
<td>000856</td>
</tr>
<tr>
<td>MPR1</td>
<td>000576</td>
</tr>
<tr>
<td>MPR10</td>
<td>000706</td>
</tr>
<tr>
<td>MPR11</td>
<td>000718</td>
</tr>
<tr>
<td>MPR12</td>
<td>000726</td>
</tr>
<tr>
<td>MPR13</td>
<td>000736</td>
</tr>
<tr>
<td>MPR14</td>
<td>000746</td>
</tr>
<tr>
<td>MPR15</td>
<td>000756</td>
</tr>
<tr>
<td>MPR2</td>
<td>000606</td>
</tr>
<tr>
<td>MPR3</td>
<td>000616</td>
</tr>
<tr>
<td>MPR4</td>
<td>000626</td>
</tr>
<tr>
<td>MPR5</td>
<td>000636</td>
</tr>
<tr>
<td>MPR8</td>
<td>000646</td>
</tr>
<tr>
<td>MPR9</td>
<td>000656</td>
</tr>
<tr>
<td>MPSR</td>
<td>017410</td>
</tr>
<tr>
<td>MPSER</td>
<td>017440</td>
</tr>
<tr>
<td>MPSER1</td>
<td>017441</td>
</tr>
<tr>
<td>MSTR</td>
<td>016459</td>
</tr>
<tr>
<td>MSTR</td>
<td>016517</td>
</tr>
<tr>
<td>MSTRX</td>
<td>016561</td>
</tr>
<tr>
<td>MSTRX1</td>
<td>016605</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table above represents memory parity test data for MACY 110A (30A) system, with values indicating parity checks for various memory locations.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSTRX3</td>
<td>016621</td>
</tr>
<tr>
<td>MSTRX5</td>
<td>016635</td>
</tr>
<tr>
<td>MT</td>
<td>016649</td>
</tr>
<tr>
<td>MTIT</td>
<td>016655</td>
</tr>
<tr>
<td>MMTOP</td>
<td>026400</td>
</tr>
<tr>
<td>MNUM</td>
<td>026409</td>
</tr>
<tr>
<td>MTR</td>
<td>026412</td>
</tr>
<tr>
<td>MTREG</td>
<td>026416</td>
</tr>
<tr>
<td>MTCOR</td>
<td>026414</td>
</tr>
<tr>
<td>MTYFG</td>
<td>000062</td>
</tr>
<tr>
<td>MX1</td>
<td>017447</td>
</tr>
<tr>
<td>MX2</td>
<td>017546</td>
</tr>
<tr>
<td>MXK</td>
<td>017466</td>
</tr>
<tr>
<td>NOMON</td>
<td>017466</td>
</tr>
<tr>
<td>NOP</td>
<td>000240</td>
</tr>
<tr>
<td>NOREC</td>
<td>000240</td>
</tr>
<tr>
<td>NRALL</td>
<td>013171</td>
</tr>
<tr>
<td>ORCDST</td>
<td>015520</td>
</tr>
<tr>
<td>OACMT</td>
<td>015522</td>
</tr>
<tr>
<td>OACNV</td>
<td>015416</td>
</tr>
<tr>
<td>OACWNY</td>
<td>015440</td>
</tr>
<tr>
<td>OACNY</td>
<td>015416</td>
</tr>
<tr>
<td>ODPLG</td>
<td>000056</td>
</tr>
<tr>
<td>OMEM</td>
<td>000000</td>
</tr>
<tr>
<td>OMP</td>
<td>001330</td>
</tr>
<tr>
<td>OPRAP</td>
<td>001036</td>
</tr>
<tr>
<td>PARVEC</td>
<td>000114</td>
</tr>
<tr>
<td>PASCNT</td>
<td>01752</td>
</tr>
<tr>
<td>PDEFNO</td>
<td>013136</td>
</tr>
<tr>
<td>PORTAB</td>
<td>001328</td>
</tr>
<tr>
<td>PERF</td>
<td>100000</td>
</tr>
<tr>
<td>PHMNH</td>
<td>001072</td>
</tr>
<tr>
<td>PHML</td>
<td>001056</td>
</tr>
<tr>
<td>PHMNX</td>
<td>001074</td>
</tr>
<tr>
<td>PS</td>
<td>177776</td>
</tr>
<tr>
<td>PSAORS</td>
<td>015200</td>
</tr>
<tr>
<td>PSCAN</td>
<td>014514</td>
</tr>
<tr>
<td>PSCAMH</td>
<td>015025</td>
</tr>
<tr>
<td>PSCANX</td>
<td>014725</td>
</tr>
<tr>
<td>PSCAN1</td>
<td>015024</td>
</tr>
<tr>
<td>PSCK1</td>
<td>015136</td>
</tr>
<tr>
<td>PSLDP</td>
<td>015064</td>
</tr>
<tr>
<td>PSMNSG</td>
<td>017603</td>
</tr>
<tr>
<td>PSXNK</td>
<td>015110</td>
</tr>
<tr>
<td>PSXTST</td>
<td>015144</td>
</tr>
<tr>
<td>PHRDN</td>
<td>015732</td>
</tr>
<tr>
<td>PHRP</td>
<td>015776</td>
</tr>
<tr>
<td>RELUC</td>
<td>000564</td>
</tr>
<tr>
<td>REBARV</td>
<td>001030</td>
</tr>
<tr>
<td>RISVC</td>
<td>001032</td>
</tr>
</tbody>
</table>

**CROSS REFERENCE TABLE -- USER SYMBOLS**

**SEQ 0073**
<table>
<thead>
<tr>
<th>EMTDEF</th>
<th>388H</th>
<th>3373</th>
<th>3376</th>
<th>3378</th>
<th>3380</th>
<th>3382</th>
<th>3384</th>
<th>3386</th>
<th>3388</th>
<th>3390</th>
<th>3392</th>
<th>3394</th>
<th>3396</th>
<th>3398</th>
<th>3400</th>
</tr>
</thead>
<tbody>
<tr>
<td>TESTX</td>
<td>2083</td>
<td>2303</td>
<td>2415</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ABS: 017772 000

ERRORS DETECTED: 0

CCMFAF.CCMFAF.LST/CRF/SOL=CCMFAF.P11
RUN-TIME: 4.91 SECONDS
RUN-TIME RATIO: 242/15=16.3
CORE USED: 9K (17 PAGES)