```
1                       .REM ^
2
3
4
5
6
7
8
9
10
11
12
13
14                                  IDENTIFICATION
15                                  --------------
16
17                      PRODUCT CODE:       AC-E992B-MC
18
19                      PRODUCT NAME:       CXKUAB0 KUV11-A MODULE
20
21                      PRODUCT DATE:       SEPTEMBER 1978
22
23                      MAINTAINER:         DEC/X11 SUPPORT GROUP
24
25                      THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
26                      WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT
27                      BY DIGITAL EQUIPMENT CORPORATION.  DIGITAL EQUIPMENT
28                      CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT
29                      MAY APPEAR IN THIS MANUAL.
30
31                      THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE
32                      PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER
33                      SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITALS
34                      COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY
35                      OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.
36
37                      DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR
38                      THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS
39                      NOT SUPPLIED BY DIGITAL.
40
41                  COPYRIGHT (C)  1976,1978 DIGITAL EQUIPMENT CORPORATION
```

1. ABSTRACT

KUA IS AN NBKMOD FOR THE KUV11-AA LSI-11 WRITABLE CONTROL STORE OPTION. IT IS USED IN CONJUCTION WITH THE CPB (FIS EXERCISER) AND FPA (FIS EXERCISER) DECX MODULES TO EXERCISE THE KUV11-AA IN ADDRESS MODE 1 OR 3. THE KUA MODULE RUNS FIRST, RUNS ONLY ONCE, AND WITH THE T-BIT OFF. ITS PURPOSE IS TO RUN A MEMORY TEST ON THE KUV11-AA RAM AND THEN TO LOAD THE RAM WITH THE APPROPRIATE MICRO-CODE FOR THE CPB AND FPA MODULES TO RUN. THUS, CPB AND FPA SHOULD BE CONFIGURED TO RUN ON AN LSI-11, IN ORDER TO FULLY EXERCISE THE KUV11-AA.
ALL ERRORS DETECTED ARE REPORTED ON THE CONSOLE TERMINAL.

2. REQUIREMENTS

HARDWARE: AN LSI-11 (M7264-YC) WITH ONE KUV11-AA, SET UP FOR ADDRESS MODE 1 OR 3 (SEE SECTION 6).

STORAGE:: KUA REQUIRES:
1. DECIMAL WORDS: 1071
2. OCTAL WORDS: 02057
3. OCTAL BYTES: 4136

3. PASS DEFINITION

SINCE THIS IS AN NBKMOD IT RUNS ONLY ONE PASS. THIS CONSISTS OF 4 ITERATIONS OF THE 'MOVI' RAM MEMORY TEST; LOADING OF THE MICRO-CODE INTO THE BOTTOM HALF OF THE KUV11-AA RAM; AND CHECKING THAT THE MICRO-CODE LOADED CORRECTLY.

4. EXECUTION TIME

KUA TAKES APPROXIMATELY 18 SECONDS, RUNNING IN AN MSV11-CD MEMORY.

5. CONFIGURATION REQUIREMENTS

DEFAULT PARAMETERS:
DEVADR: 177540, VECTOR: 1, BR1: 0, DEVCNT: 1

REQUIRED PARAMETERS:
NONE

6. DEVICE/OPTION SETUP

MAKE CERTAIN THAT THE KUV11-AA IS PROPERLY INSTALLED AND THAT THE MODULE (M8018) DIP SWITCHES ARE SET UP FOR EITHER ADDRESS MODE 1 OR 3.
THESE SWITCH SETTINGS ARE:

MODE	SW1	SW2	SW3	SW4	SW5	SW6	SW7	SW8
1	ON	OFF	OFF	ON	OFF	ON	OFF	-
3	OFF	OFF	ON	ON	OFF	ON	OFF	-

7. MODULE OPERATION

A. RUN THE 'MOVI' MEMORY TEST ON THE KUV11-AA RAM MEMORY 4 TIMES; REPORT ERRORS ACCORDING TO SR1 BIT01 SETTING.
B. IF SR1 BIT00 = 1 AND ANY MEMORY ERROR(S) OCCURRED, THEN SKIP TO E.
C. LOAD MICRO-CODE INTO THE KUV11-AA MEMORY.
D. CHECK THAT THE MICRO-CODE WAS LOADED CORRECTLY; REPORT ERROR IF NOT.
E. EXIT THE MODULE.

8. OPERATING OPTIONS

SR1 BIT00 CLEAR(0):
LOAD THE MICRO-CODE FOR CPB AND FPA REGARDLESS OF THE RESULTS OF THE KUV11-AA RAM MEMORY TEST.

SR1 BIT00 SET(1):
IF THE RAM MEMORY TEST FAILS, DO NOT LOAD THE MICRO-CODE FOR CPB AND FPA TO RUN.

SR1 BIT01 CLEAR(0):
IN THE RAM MEMORY TEST, TYPE OUT AN ERROR MESSAGE FOR EACH ERROR ENCOUNTERED.

SR1 BIT01 SET(1):
DO NOT TYPE OUT EACH RAM MEMORY TEST ERROR; JUST TYPE ONE MESSAGE AT THE END INDICATING HOW MANY ERRORS WERE ENCOUNTERED.

9. NON-STANDARD PRINTOUTS

THE KUV11-AA MEMORY TEST ERROR REPORT DOES NOT USE THE STANDARD DEC/X11 ERROR PRINTOUT, SINCE THE MEMORY WORDS ARE 24 BITS LONG. THE ERROR PRINTOUT LOOKS AS FOLLOWS:

```
BAD DATA IN A KUV11-AA RAM WORD.
   RAM          GOOD DATA          BAD DATA
ADDRESS   CSR+4   CSR+2     CSR+4   CSR+2
```

THE FIRST 16 BITS ARE UNDER THE HEADING 'CSR+2'; THE LAST 8 BITS ARE FOUND IN THE LOWER BYTE OF 'CSR+4'. THE UPPER 8 BITS OF 'CSR+4' HAVE NO MEANING AND ARE ALWAYS ZEROES. THE 'RAM ADDRESS' CAN RANGE FROM 0 TO 1777.

```
000000'                         NBKMOD  <KUAB > 177540,1,0,0,1,1,163
000000'                                 MODULE  41000,KUAB ,177540,1,0,0,1,1,163
                                        .TITLE  KUAB DEC/X11 SYSTEM EXERCISER MODULE
                                ;       DDXCOM  VERSION  6       23-MAY-78
                                        .LIST   BIN
                                ;*****************************************************************************
000000'                         BEGIN:
000000' 052513   041101     040 MODNAM: .ASCII  /KUAB / ;MODULE NAME.
000005'     000                 XFLAG:  .BYTE   0PEN            ;USED TO KEEP TRACK OF WBUFF USAGE
000006' 177540                  ADDR:   177540+0                ;1ST DEVICE ADDR.
000010' 000001                  VECTOR: 1+0                     ;1ST DEVICE VECTOR.
000012'     000                 BR1:    .BYTE   PRTY0+0         ;1ST BR LEVEL.
000013'     000                 BR2:    .BYTE   PRTY0+0         ;2ND BR LEVEL.
000014' 000002                  DVID1:  1+1                     ;DEVICE INDICATOR 1.
000016' 000000                  SR1:    OPEN                    ;SWITCH REGISTER 1
000020' 000000                  SR2:    OPEN                    ;SWITCH REGISTER 2
000022' 000000                  SR3:    OPEN                    ;SWITCH REGISTER 3
000024' 000000                  SR4:    OPEN                    ;SWITCH REGISTER 4
                                ;*****************************************************************************
000026' 041000                  STAT:   41000                   ;STATUS WORD.
000030' 000226'                 INIT:   START                   ;MODULE START ADDR.
000032' 000224'                 SPOINT: MODSP                   ;MODULE STACK POINTER.
000034' 000000                  PASCNT: 0                       ;PASS COUNTER.
000036' 000001                  ICONT:  1                       ;# OF ITERATIONS PER PASS=1
000040' 000000                  ICOUNT: 0                       ;LOC TO COUNT ITERATIONS
000042' 000000                  SOFCNT: 0                       ;LOC TO SAVE TOTAL SOFT ERRORS
000044' 000000                  HRDCNT: 0                       ;LOC TO SAVE TOTAL HARD ERRORS
000046' 000000                  SOFPAS: 0                       ;LOC TO SAVE SOFT ERRORS PER PASS
000050' 000000                  HRDPAS: 0                       ;LOC TO SAVE HARD ERRORS PER PASS
000052' 000000                  SYSCNT: 0                       ;# OF SYS ERRORS ACCUMULATED
000054' 000000                  RANNUM: 0                       ;HOLDS RANDOM # WHEN RAND MACRO IS CALLED
000056'                         CONFIG:                         ;RESERVED FOR MONITOR USE
000056' 000000                  RES1:   0                       ;RESERVED FOR MONITOR USE
000060' 000000                  RES2:   0                       ;RESERVED FOR MONITOR USE
000062' 000000                  SVR0:   OPEN                    ;LOC TO SAVE R0.
000064' 000000                  SVR1:   OPEN                    ;LOC TO SAVE R1.
000066' 000000                  SVR2:   OPEN                    ;LOC TO SAVE R2.
000070' 000000                  SVR3:   OPEN                    ;LOC TO SAVE R3.
000072' 000000                  SVR4:   OPEN                    ;LOC TO SAVE R4.
000074' 000000                  SVR5:   OPEN                    ;LOC TO SAVE R5.
000076' 000000                  SVR6:   OPEN                    ;LOC TO SAVE R6.
000100' 000000                  CSRA:   OPEN                    ;ADDR OF CURRENT CSR.
000102'                         SBADR:                          ;ADDR OF GOOD DATA, OR
000102' 000000                  ACSR:   OPEN                    ;CONTENTS OF CSR.
000104'                         WASADR:                         ;ADDR OF BAD DATA, OR
000104' 000000                  ASTAT:  OPEN                    ;STATUS REG CONTENTS.
000106'                         ERRTYP:                         ;TYPE OF ERROR
000106' 000000                  ASB:    OPEN                    ;EXPECTED DATA.
000110' 000000                  AWAS:   OPEN                    ;ACTUAL DATA.
000112' 000226'                 RSTRT:  RESTRT                  ;RESTART ADDRESS AFTER END OF PASS
000114' 000000                  WDTO:   OPEN                    ;WORDS TO MEMORY PER ITERATION
000116' 000000                  WDFR:   OPEN                    ;WORDS FROM MEMORY PER ITERATION
000120' 000000                  INTR:   OPEN                    ;# OF INTERRUPTS PER ITERATION
000122' 000163                  IDNUM:  163                     ;MODULE IDENTIFICATION NUMBER=163
                                        .REPT   SPSIZ           ;MODULE STACK STARTS HERE.
000040
                                        .NLIST
```



```
                                        .WORD   0
                                        .LIST
                                        .ENDR
000224'                         MODSP:
                                ;*****************************************************************************
```

```
                                ;USER EQUATES AND CONSTANTS

000001                          BIT0    = 000001
000002                          BIT1    = 000002
000004                          BIT2    = 000004
000010                          BIT3    = 000010
000020                          BIT4    = 000020
000040                          BIT5    = 000040
000100                          BIT6    = 000100
000200                          BIT7    = 000200
000400                          BIT8    = 000400
001000                          BIT9    = 001000
002000                          BIT10   = 002000
004000                          BIT11   = 004000
010000                          BIT12   = 010000
020000                          BIT13   = 020000
040000                          BIT14   = 040000
100000                          BIT15   = 100000
000000                          R0      = %0
000001                          R1      = %1
000002                          R2      = %2
000003                          R3      = %3
000004                          R4      = %4
000005                          R5      = %5
000006                          R6      = %6
000007                          R7      = %7
000006                          SP      = %6
000007                          PC      = %7

000224' 000000                  TIMES:  0               ;WILL KEEP TRACK OF RAM MEMORY TEST ITERATIONS
```



```
000226'                         START:
000226' 016701  177554          RESTRT: MOV     ADDR,R1         ;GET THE BASE ADDRESS (CSR)
000232' 005011                          CLR     (R1)            ;MAKE SURE THAT ENABLE (CSR BIT12) IS CLEAR
000234' 005067  177604                  CLR     HRDCNT          ;SET RAM MEMORY TEST ERROR COUNT TO ZERO
000240' 012767  000004  177756          MOV     #4,TIMES        ;SET ITERATION COUNT TO 10 OCTAL

                                ;CLEAR THE KUV11-AA RAM MEMORY

000246' 005011                  CLRRAM: CLR     (R1)            ;SET UP CSR FOR Q-BUS ACCESS, RAM ADDRESS 0
000250' 012700  002000                  MOV     #2000,R0        ;SET UP LOOP COUNTER
000254' 005061  000002          1$:     CLR     2(R1)           ;WRITE ZEROES...
000260' 005061  000004                  CLR     4(R1)           ;...TO RAM WORD
000264' 005211                          INC     (R1)            ;INCREMENT RAM ADDRESS
000266' 005300                          DEC     R0              ;DONE CLEARING THE WCS RAM?
000270' 001371                          BNE     1$              ;BRANCH BACK AND CONTINUE IF NOT

                                ;*************************************************************************
                                ;* TEST RAM MEMORY USING 'MOVI' MEMORY TEST
                                ;*
                                ;*AFTER THE KUV11-AA RAM HAS BEEN CLEARED, THE ALGORITHM PROCEEDS AS FOLLOWS:
                                ;*
                                ;*(1) READ RAM LOCATION 0 AND VERIFY THAT IT CONTAINS ZEROES,
                                ;*     WRITE ALL ONES TO RAM LOCATION 0,
                                ;*     READ LOCATION 0 AND VERIFY THAT IT CONTAINS ALL ONES;
                                ;*(2) REPEAT THE ABOVE FOR LOCATIONS 1,2,...,1777 (I.E., ALL OF THE RAM).
                                ;*     AT THIS POINT THE RAM WILL BE FULL OF ONES.
                                ;*(3) REPEAT THE ABOVE, WRITING ZEROES THIS TIME.  AFTER THIS THE RAM WILL
                                ;*     AGAIN BE FULL OF ZEROES.
                                ;*(4) REPEAT ALL OF THE ABOVE, EXCEPT START AT THE TOP OF THE RAM AND WORK
                                ;*     DOWNWARDS (RAM LOCATIONS 1777-0).
                                ;*     AT THIS POINT THE TEST HAS SEQUENCED THROUGH THE RAM FOUR TIMES.
                                ;*(5) REPEAT THE ABOVE, EXCEPT USE BIT 1 AS THE LEAST SIGNIFICANT BIT WHEN
                                ;*     FORMING THE RAM ADDRESSES.  THE RAM ADDRESS WILL OVERFLOW AFTER CHECKING
                                ;*     ALL EVEN RAM ADDRESSES, AND THE OVERFLOW BIT IS SIMPLY WRAPPED AROUND
                                ;*     TO BIT 0 OF THE RAM ADDRESS.  TESTING CONTINUES UNTIL ALL RAM ADDRESSES
                                ;*     HAVE BEEN TESTED.
                                ;*     REPEAT AGAIN, THIS TIME WITH BIT 2 AS THE LSB IN FORMING THE RAM
                                ;*     ADDRESSES; AND AGAIN WITH BIT 3 AS THE LSB; AND SO ON UNTIL ALL BITS
                                ;*     OF THE RAM ADDRESS HAVE SERVED AS THE LSB.  SINCE THERE ARE 10 BITS
                                ;*     IN THE RAM ADDRESS (TO COVER THE RANGE 0-1777), THIS MEANS THAT STEP (5)
                                ;*     WILL BE DONE 10 TIMES IN ALL.
                                ;*************************************************************************

000272' 012702  177777          RAMTST: MOV     #177777,R2      ;R2 WILL CONTAIN TEST DATA
000276' 005003                          CLR     R3              ;R3 WILL CONTAIN TEST DATA
000300' 005004                          CLR     R4              ;R4 WILL CONTAIN THE RAM ADDRESS
000302' 012705  000001                  MOV     #1,R5           ;R5 WILL HAVE THE LEAST SIGNIFICANT BIT FOR
                                                                ;GENERATING THE RAM ADDRESS
000306' 012700  002000                  MOV     #2000,R0        ;SET UP LOOP COUNTER
                                ;BEGINNING OF TEST LOOP
000312' 010411                  1$:     MOV     R4,(R1)         ;PUT ADDRESS INTO THE CSR
000314' 020361  000002                  CMP     R3,2(R1)        ;CORRECT DATA?
000320' 001003                          BNE     2$              ;BRANCH TO ERROR IF NOT
000322' 120361  000004                  CMPB    R3,4(R1)        ;CORRECT DATA?
```

```
                        BEQ     3$              ;BRANCH OVER ERROR IF OK
000326' 001404
000330' 010367 177552   2$:     MOV     R3,ASB          ;GO TO ERROR REPORTING SUBROUTINE
000334' 004767 000536           JSR     PC,MEMERR       ;WRITE TEST DATA TO RAM BITS 0-15
000340' 010261 000002   3$:     MOV     R2,2(R1)        ;WRITE TEST DATA TO RAM BITS 16-23
000344' 010261 000004           MOV     R2,4(R1)        ;CORRECT DATA?
000350' 020261 000002           CMP     R2,2(R1)        ;BRANCH TO ERROR IF NOT
000354' 001003                  BNE     4$
000356' 120261 000004           CMPB    R2,4(R1)        ;CORRECT DATA?
000362' 001404                  BEQ     5$              ;BRANCH OVER ERROR IF YES
000364' 010267 177516   4$:     MOV     R2,ASR          ;GO TO ERROR REPORTING SUBROUTINE
000370' 004767 000502           JSR     PC,MEMERR       ;GENERATE A RAM ADDRESS
000374' 060504          5$:     ADD     R5,R4           ;ADDRESS OVERFLOW?
000376' 032704 002000           BIT     #BIT10,R4       ;BRANCH IF NOT
000402' 001403                  BEQ     6$
000404' 042704 002000           BIC     #BIT10,R4       ;ADD ADDRESS OVERFLOW...
000410' 005204                  INC     R4              ;...TO BOTTOM OF ADDRESS
000412' 005300          6$:     DEC     R0              ;LOOP DONE?
000414' 001335                  BNE     1$              ;CONTINUE TEST LOOP IF NOT
000416' 005102                  COM     R2              ;R2 GETS ONES COMPLEMENT OF ITSELF
000420' 005103                  COM     R3              ;R3 GETS ONES COMPLEMENT OF ITSELF
000422' 012700 002000           MOV     #2000,R0        ;SET UP LOOP COUNTER
000426' 005004                  CLR     R4              ;INITIALIZE THE RAM ADDRESS
000430' 005702                  TST     R2              ;HAVE WE GONE THROUGH TWICE?
000432' 001727                  BEQ     1$              ;NO, GO BACK AND DO IT AGAIN WITH COMPLEMENTED DATA
                        ;DO THE SAME AS ABOVE EXCEPT TEST RAM MEMORY FROM TOP TO BOTTOM (1777-0)
000434' 012704 001777           MOV     #1777,R4        ;SET UP RAM ADDRESS
000440' 010411          7$:     MOV     R4,(R1)         ;PUT ADDRESS INTO THE CSR
000442' 020361 000002           CMP     R3,2(R1)        ;CORRECT DATA?
000446' 001003                  BNE     8$              ;BRANCH TO ERROR IF NOT
000450' 120361 000004           CMPB    R3,4(R1)        ;CORRECT DATA?
000454' 001404                  BEQ     9$              ;BRANCH OVER ERROR IF OK
000456' 010367 177424   8$:     MOV     R3,ASB          ;GO TO MEMORY ERROR REPORTING SUBROUTINE
000462' 004767 000410           JSR     PC,MEMERR       ;WRITE TEST DATA TO RAM BITS 0-15
000466' 010261 000002   9$:     MOV     R2,2(R1)        ;WRITE TEST DATA TO RAM BITS 16-23
000472' 010261 000004           MOV     R2,4(R1)        ;CORRECT DATA?
000476' 020261 000002           CMP     R2,2(R1)        ;BRANCH TO ERROR IF NOT
000502' 001003                  BNE     10$
000504' 120261 000004           CMPB    R2,4(R1)        ;CORRECT DATA?
000510' 001404                  BEQ     11$             ;BRANCH OVER ERROR IF YES
000512' 010267 177370   10$:    MOV     R2,ASB          ;GO TO MEMORY ERROR REPORTING SUBROUTINE
000516' 004767 000354           JSR     PC,MEMERR       ;GENERATE A RAM ADDRESS
000522' 160504          11$:    SUB     R5,R4           ;BRANCH IF NO ADDRESS UNDERFLOW OCCURED
000524' 100002                  BPL     12$             ;CORRECT THE ADDRESS UNDERFLOW
000526' 062704 001777           ADD     #1777,R4        ;LOOP DONE?
000532' 005300          12$:    DEC     R0              ;CONTINUE TEST LOOP IF NOT
000534' 001341                  BNE     7$              ;R2 GETS ONES COMPLEMENT OF ITSELF
000536' 005102                  COM     R2              ;R3 GETS ONES COMPLEMENT OF ITSELF
000540' 005103                  COM     R3              ;SET UP LOOP COUNTER
000542' 012700 002000           MOV     #2000,R0        ;INITIALIZE THE RAM ADDRESS
000546' 012704 001777           MOV     #1777,R4        ;HAVE WE GONE THROUGH TWICE?
000552' 005702                  TST     R2              ;GO BACK AND DO IT AGAIN WITH COMPLEMENTED DATA IF NOT
000554' 001731                  BEQ     7$

                        ;GENERATE A NEW LEAST SIGNIFICANT BIT FOR RAM ADDRESS GENERATION AND,
                        ;IF NOT DONE WITH THE TEST, GO BACK AND RUN SOME MORE
000556' 006305                  ASL     R5              ;SHIFT THE LSB FOR SUBSEQUENT RAM ADDRESS GENERATION
```



```
000560' 005004                  CLR     R4              ;INITIALIZE THE RAM ADDRESS
000562' 032705 002000           BIT     #BIT10,R5       ;SEE IF WE ARE DONE WITH THE RAM MEMORY TEST
000566' 001002                  BNE     13$             ;BRANCH IF YES
000570' 000167 177516           JMP     1$              ;CONTINUE TESTING IF NOT
000574' 005367 177424   13$:    DEC     TIMES           ;ALL ITERATIONS DONE?
000600' 001402                  BEQ     14$             ;BRANCH IF YES
000602' 000167 177440           JMP     CLRRAM          ;NO, RUN RAM MEMORY TEST AGAIN

000606' 005767 177232   14$:    TST     HRDCNT          ;ANY RAM MEMORY ERRORS?
000612' 001410                  BEQ     15$             ;BRANCH IF NOT
                                ;*****************************************************************
                                                        ;CONVERT HRDCNT TO ASCII AND
000614' 104420  000000' 000044'         OTOA$,BEGIN,HRDCNT,COUNT        ;STORE AT COUNT
000622' 001614'
                                ;*****************************************************************
000624' 104403  000000' 001242'         MSGN$,BEGIN,ERR2        ;ASCII MESSAGE CALL WITH COMMON HEADER
000632' 000403                  BR      16$             ;SKIP PRINTOUT
000634'                 15$:
000634' 104403  000000' 001246'         MSGN$,BEGIN,RAMOK       ;ASCII MESSAGE CALL WITH COMMON HEADER

                        ;*******************************************************************************
                        ;*
                        ;* THE NEXT ROUTINE LOADS MICRO-CODE FROM THE TABLES AT 'UCODE' & 'UCODE1'  INTO
                        ;* THE KUV11-AA RAM AND CHECKS THAT IT LOADED CORRECTLY.
                        ;*
                        ;*******************************************************************************

                        ;LOAD THE MICRO-CODE
000642' 036727 177150  000001 16$:      BIT     SR1,#BIT0       ;LOAD MICRO-CODE EVEN IF RAM MEMORY TEST HAD ERRORS?
000650' 001403                  BEQ     17$             ;BRANCH IF YES
000652' 005767 177166           TST     HRDCNT          ;SEE IF ANY RAM MEMORY TEST ERRORS OCCURED
000656' 001103                  BNE     ABORT           ;YES, BRANCH TO LOADER ABORT
000660' 005011          17$:    CLR     (R1)            ;SET RAM ADDRESS TO 0
000662' 012702 002070'          MOV     #UCODE,R2       ;GET FIRST MICRO-CODE TABLE ADDRESS
000666' 012261 000002   18$:    MOV     (R2)+,2(R1)     ;LOAD A KUV11-AA RAM WORD, BITS 15-0
000672' 021127 000777           CMP     (R1),#777       ;DONE?
000676' 001402                  BEQ     19$             ;BRANCH IF YES
000700' 005211                  INC     (R1)            ;INCREMENT THE RAM ADDRESS
000702' 000771                  BR      18$             ;BRANCH BACK AND CONTINUE LOADING MICRO-CODE
000704' 012702 004070'  19$:    MOV     #UCODE1,R2      ;GET SECOND MICRO-CODE TABLE ADDRESS
000710' 012200          20$:    MOV     (R2)+,R0        ;GET A TABLE ENTRY
000712' 001414                  BEQ     22$             ;BRANCH OUT OF MICRO-CODE LOADER IF A ZERO TABLE
                                                        ;ENTRY IS ENCOUNTERED
000714' 012703 000001           MOV     #BIT0,R3        ;PRE-LOAD R3 WITH RAM DATA
000720' 005700                  TST     R0              ;SEE IF BIT15 IS SET
000722' 100004                  BPL     21$             ;BRANCH IF NOT
000724' 012703 000002           MOV     #BIT1,R3        ;PRE-LOAD R3 WITH RAM DATA
000730' 042700 100000           BIC     #BIT15,R0       ;STRIP OFF BIT 15
000734' 010011          21$:    MOV     R0,(R1)         ;LOAD THE RAM ADDRESS
000736' 010361 000004           MOV     R3,4(R1)        ;LOAD THE RAM WORD, BITS 23-16
000742' 000762                  BR      20$

                        ;CHECK FOR CORRECT MICRO-CODE LOAD
000744' 005011          22$:    CLR     (R1)            ;SET RAM ADDRESS TO 0
000746' 012702 002070'          MOV     #UCODE,R2       ;GET FIRST MICRO-CODE TABLE ADDRESS
000752' 022261 000002   23$:    CMP     (R2)+,2(R1)     ;CORRECT DATA IN RAM, BITS 15-0?
```

```
000756' 001026                  BNE     27$             ;BRANCH TO ERROR IF NOT
000760' 021127  000777          CMP     (R1),#777       ;DONE?
000764' 001402                  BEQ     24$             ;BRANCH IF YES
000766' 005211                  INC     (R1)            ;INCREMENT RAM ADDRESS
000770' 000770                  BR      23$             ;BRANCH BACK AND CONTINUE CHECKING MICRO-CODE
000772' 012702  004070'  24$:   MOV     #UCODE1,R2      ;GET SECOND MICRO-CODE TABLE ADDRESS
000776' 012200          25$:    MOV     (R2)+,R0        ;GET A TABLE ENTRY
001000' 001423                  BEQ     28$             ;BRANCH OUT OF MICRO-CODE CHECKER IF A ZERO TABLE
                                                        ;ENTRY IS ENCOUNTERED
001002' 012703  000001          MOV     #BIT0,R3        ;PRE-LOAD R3 WITH EXPECTED RAM DATA
001006' 005700                  TST     R0              ;SEE IF BIT15 IS SET
001010' 100004                  BPL     26$             ;BRANCH IF NOT
001012' 012703  000002          MOV     #BIT1,R3        ;PRE-LOAD R3 WITH EXPECTED RAM DATA
001016' 042700  100000          BIC     #BIT15,R0       ;STRIP OFF BIT 15
001022' 010011          26$:    MOV     R0,(R1) ;LOAD THE RAM ADDRESS
001024' 020361  000004          CMP     R3,4(R1)        ;CHECK FOR CORRECT DATA IN RAM BITS 23-16
001030' 001001                  BNE     27$             ;BRANCH TO ERROR IF BAD
001032' 000761                  BR      25$             ;CONTINUE CHECKING

001034'                 27$:
001034' 104403  000000' 001256' MSGN$,BEGIN,LDBAD       ;ASCII MESSAGE CALL WITH COMMON HEADER
001042' 005267  176776          INC     HRDCNT          ;ADD THIS LOAD ERROR TO THE ERROR COUNT
001046' 000405                  BR      END

001050'                 28$:
001050' 104403  000000' 001252' MSGN$,BEGIN,LOADOK      ;ASCII MESSAGE CALL WITH COMMON HEADER
001056' 052711  010000          BIS     #BIT12,(R1)     ;SET KUV11-AA ENABLE BIT, TO ALLOW MIB ACCESS TO THE RAM
001062'                 END:
001062' 104413  000000'         ENDIT$,BEGIN            ;SIGNAL END OF ITERATION.
                                                        ;MONITOR SHALL TEST END OF PASS

001066'                 ABORT:
001066' 104403  000000' 001262' MSGN$,BEGIN,LDABRT      ;ASCII MESSAGE CALL WITH COMMON HEADER
001074' 000772                  BR      END


                        ;*****************************************************************************
                        ;*
                        ;*SUBROUTINE    MEMERR
                        ;*
                        ;*THIS SUBROUTINE REPORTS KUV11-AA RAM MEMORY ERRORS.  EXPECTED DATA IS PASSED
                        ;*VIA 'ASB'; THE BAD DATA AND FAILING RAM ADDRESS ARE OBTAINED THROUGH THE KUV11-AA
                        ;*DEVICE REGISTERS.  THE BASE REGISTER ADDRESS OF THE KUV11-AA IS PASSED VIA
                        ;*'ADDR'.  THE STANDARD MONITOR CALLS 'MSGN' AND 'OACNV' ARE MADE IN THE SUBTOUTINE,
                        ;*AND USE GENERAL REGISTER 5.  NO OTHER GENERAL PURPOSE REGISTERS ARE USED.
                        ;*THIS SUBROUTINE IS CALLED AS FOLLOWS:          JSR     PC,MEMERR
                        ;*
                        ;*****************************************************************************
001076' 005267  176742  MEMERR: INC     HRDCNT          ;COUNT THE NUMBER OF RAM MEMORY ERRORS
001102' 036727  176710  000002  BIT     SR1,#BIT1       ;REPORT EACH ERROR?
001110' 001050          BNE     1$                      ;BRANCH OVER ERROR REPORTING IF NO
                                ;*************************************************************
                                                        ;CONVERT ASB TO ASCII AND
                                                        ;STORE AT GOODLO
001112' 104420  000000' 000106' OTOA$,BEGIN,ASB,GOODLO
001120' 001462'
```



```
                                ;*************************************************************
001122' 042767  177400  176756  BIC     #177400,ASB     ;STRIP OFF HIGH BYTE
                                ;*************************************************************
                                                        ;CONVERT ASB TO ASCII AND
                                                        ;STORE AT GOODHI
001130' 104420  000000' 000106' OTOA$,BEGIN,ASB,GOODHI
001136' 001453'
                                ;*************************************************************
001140' 011167  000070          MOV     (R1),GUNCH      ;CONVERT FAILING RAM ADDRESS TO ASCII
                                ;*************************************************************
                                                        ;CONVERT GUNCH TO ASCII AND
                                                        ;STORE AT ADDRES
001144' 104420  000000' 001234' OTOA$,BEGIN,GUNCH,ADDRES
001152' 001442'
                                ;*************************************************************
001154' 112767  000040  000260  MOVB    #40,ADDRES      ;PUT TWO ASCII SPACES INTO ADDRES
001162' 112767  000040  000253  MOVB    #40,ADDRES+1    ;
001170' 016167  000002  000036  MOV     2(R1),GUNCH     ;CONVERT BAD DATA TO ASCII
                                ;*************************************************************
                                                        ;CONVERT GUNCH TO ASCII AND
                                                        ;STORE AT BADLO
001176' 104420  000000' 001234' OTOA$,BEGIN,GUNCH,BADLO
001204' 001502'
                                ;*************************************************************
001206' 016167  000004  000020  MOV     4(R1),GUNCH     ;CONVERT BAD DATA TO ASCII
                                ;*************************************************************
                                                        ;CONVERT GUNCH TO ASCII AND
                                                        ;STORE AT BADHI
001214' 104420  000000' 001234' OTOA$,BEGIN,GUNCH,BADHI
001222' 001473'
                                ;*************************************************************
                                                        ;TYPE OUT THE ERROR MESSAGE AND DATA
001224' 104403  000000' 001236' MSGN$,BEGIN,ERR1        ;ASCII MESSAGE CALL WITH COMMON HEADER
001232' 000207          1$:     RTS     PC              ;RETURN FROM THE SUBROUTINE
```

```
                                                ;TEMP LOC. FOR OCTAL TO ASCII CONV.
001234' 000000          GUNCH:  .WORD   0

001236' 001266'         ERR1:   ERROR1
001240' 177777                  177777

001242' 001511'         ERR2:   ERROR2
001244' 177777                  177777

001246' 001625'         RAMOK:  MEMOK
001250' 177777                  177777

001252' 001667'         LOADOK: LOK
001254' 177777                  177777

001256' 001731'         LDBAD:  LBAD
001260' 177777                  177777

001262' 001775'         LDABRT: LABORT
001264' 177777                  177777

001266' 041045  042101  042040  ERROR1: .ASCII  '%BAD DATA IN A KUV11-AA RAM WORD.'
001274' 052101  020101  047111
001302' 040440  045440  053125
001310' 030461  040455  020101
001316' 040522  020115  047527
001324' 042122  000056

001327'     045 020040  040522          .ASCII  '%  RAM      GOOD DATA       BAD DATA'
001334' 020115  020040  020040
001342' 043440  047517  020104
001350' 040504  040524  020040
001356' 020040  020040  041040
001364' 042101  042040  052101
001372'     101

001373'     045 042101  051104          .ASCII  '%ADDRESS  CSR+4  CSR+2    CSR+4  CSR+2%'
001400' 051505  020123  041440
001406' 051123  032053  020040
001414' 051503  025522  020062
001422' 020040  041440  051123
001430' 032053  020040  051503
001436' 025522  022462

001442' 000006          ADDRES: .BLKB   6               ;BAD RAM ADDRESS
001450'     040     040     040 GOODHI: .BYTE   40,40,40        ;ASCII SPACES
001453' 000006          GOODLO: .BLKB   6               ;EXPECTED DATA, BITS 23-16
001461'     040                 .BYTE   40              ;ASCII SPACE
001462' 000006          BADHI:  .BLKB   6               ;EXPECTED DATA, BITS 15-0
001470'     040     040     040 .BYTE   40,40,40        ;ASCII SPACES
001473' 000006          BADLO:  .BLKB   6               ;ACTUAL DATA, BITS 23-16
001501'     040                 .BYTE   40              ;ASCII SPACE
001502' 000006                  .BLKB   6               ;ACTUAL DATA, BITS 15-0
001510'     000                 .BYTE   0               ;ASCII MESSAGE TERMINATOR
```



```
001511'     045 040522  020115  ERROR2: .ASCII  '%RAM MEMORY TEST UNSUCCESSFUL; TOTAL NUMBER OF ERRORS (IN OCTAL) = '
001516' 042515  047515  054522
001524' 052040  051505  020124
001532' 047125  052523  041503
001540' 051505  043123  046125
001546' 020073  047524  040524
001554' 020114  052516  041115
001562' 051105  047440  020106
001570' 051105  047522  051522
001576' 024040  047111  047440
001604' 052103  046101  020051
001612' 020075

001614' 000006          COUNT:  .BLKB   6
001622'     015                 .BYTE   15
001623'     012                 .BYTE   12
001624'     000                 .BYTE   0               ;ASCII MESSAGE TERMINATOR

001625'     045 040522  020115  MEMOK:  .ASCIZ  '%RAM MEMORY TEST WAS SUCCESSFUL.%'
001632' 042515  047515  054522
001640' 052040  051505  020124
001646' 040527  020123  052523
001654' 041503  051505  043123
001662' 046125  022456      000

001667'     045 044515  051103  LOK:    .ASCIZ  '%MICRO-CODE LOAD WAS SUCCESSFUL.%'
001674' 026517  047503  042504
001702' 046040  040517  020104
001710' 040527  020123  052523
001716' 041503  051505  043123
001724' 046125  022456      000

001731'     045 051105  047522  LBAD:   .ASCIZ  '%ERROR OCCURED IN MICRO-CODE LOAD.%'
001736' 020122  041517  052503
001744' 042522  020104  047111
001752' 046440  041511  047522
001760' 041455  042117  020105
001766' 047514  042101  022456
001774'     000

001775'     045 044515  051103  LABORT: .ASCIZ  '%MICRO-CODE LOAD IS ABORTED SINCE RAM MEMORY TEST FAILED.%'
002002' 026517  047503  042504
002010' 046040  040517  020104
002016' 051511  040440  047502
002024' 052122  042105  051440
002032' 047111  042503  051040
002040' 046501  046440  046505
002046' 051117  020131  042524
002054' 052123  043040  044501
002062' 042514  027104  000045

                        .EVEN
```

```
                                UCODE:  072411
                                        ;*
                                        ;*
                                UCODE1: 000123
                                        ;*
                                        ;*
                                        000000

                                .END
```


CROSS REFERENCE TABLE -- USER SYMBOLS

```
ABORT   001066R          406     461#
ACSR    000102R          210#
ADDR    000006R          176#    265
ADDRES  001442R          498     501*    502*    567#
ADDR22= 001000           228#
ASB     000106R          214#    319*    327*    349*    357*    484     487*    491
ASTAT   000104R          212#
AWAS    000110R          215#
BADHI   001473R          514     573#
BADLO   001502R          507     575#
BEGIN   000000R          173#    387     390     393     450     455     458     462     484     491     498     507     514
                         518
BIT0  = 000001           228#    232#    403     418     439
BIT1  = 000002           228#    233#    421     442     479
BIT10 = 002000           228#    242#    330     332     375
BIT11 = 004000           228#    243#
BIT12 = 010000           228#    244#    456
BIT13 = 020000           228#    245#
BIT14 = 040000           228#    246#
BIT15 = 100000           228#    247#    422     443
BIT2  = 000004           228#    234#
BIT3  = 000010           228#    235#
BIT4  = 000020           228#    236#
BIT5  = 000040           228#    237#
BIT6  = 000100           228#    238#
BIT7  = 000200           228#    239#
BIT8  = 000400           228#    240#
BIT9  = 001000           228#    241#
BREAK$= 104407           228#
BR1     000012R          178#
BR2     000013R          179#
BTOD$ = 104421           228#
CDATA$= 104412           228#
CLRRAM  000246R          272#    380
CONFIG  000056R          198#
COUNT   001614R          387     593#
CSRA    000100R          208#
DATCK$= 104411           228#
DATER$= 104404           228#
DVID1   000014R          180#
END     001062R          452     457#    463
ENDIT$= 104413           228#    458
END$  = 104410           228#
ERROR1  001266R          527     546#
ERROR2  001511R          530     561#
ERRTYP  000106R          213#
ERR1    001236R          518     527#
ERR2    001242R          390     530#
EXIT$ = 104400           228#
GETPA$= 104415           228#
GOODHI  001453R          491     569#
GOODLO  001462R          484     571#
GUNCH   001234R          494*    498     503*    507     510*    514     525#
GWBUF$= 104414           228#
HRDCNT  000044R          193#    267*    382     387     405     451*    478*
HRDER$= 104405           228#
```

```
HRDPAS  000050R          195#
ICONT   000036R          190#
ICOUNT  000040R          191#
IDNUM   000122R          220#
INIT    000030R          187#
INTR    000120R          219#
LABORT  001775R          542     620#
LBAD    001731R          539     613#
LDABRT  001262R          462     542#
LDBAD   001256R          450     539#
LOADOK  001252R          455     536#
LOK     001667R          536     606#
MAP22$= 104416           228#
MEMERR  001076R          320     328     350     358     478#
MEMOK   001625R          533     598#
MODNAM  000000R          174#
MODSP   000224R          188     226#
MSGN$ = 104403           228#    390     393     450     455     462     518
MSGS$ = 104402           228#
MSG$  = 104401           228#
NULL  = 000000           228#
OPEN  = 000000           175     181     182     183     184     201     202     203     204     205     206     207     208
                         210     212     214     215     217     218     219     228#
OTOA$ = 104420           228#    387     484     491     498     507     514
PASCNT  000034R          189#
PIRQ$ = 000004           228#
POPSP = 005726           228#
POPSP2= 022626           228#
PRTY  = 000000           228#
PRTY0 = 000000           178     179     228#
PRTY1 = 000040           228#
PRTY2 = 000100           228#
PRTY3 = 000140           228#
PRTY4 = 000200           228#
PRTY5 = 000240           228#
PRTY6 = 000300           228#
PRTY7 = 000340           228#
PS    = 177776           228#
PSW   = 177776           228#
PUSH  = 005746           228#
PUSH2 = 024646           228#
RAMOK   001246R          393     533#
RAMTST  000272R          307#
RAND$ = 104417           228#
RANNUM  000054R          197#
RESTRT  000226R          216     265#
RES1    000056R          199#
RES2    000060R          200#
RSTRT   000112R          216#
SBADR   000102R          209#
SOFCNT  000042R          192#
SOFERS= 104406           228#
SOFPAS  000046R          194#
SPDINT  000032R          188#
SPSIZ = 000040             1#    221
SR1     000016R          181#    403     479
```



```
SR2     000020R          182#
SR3     000022R          183#
SR4     000024R          184#
START   000226R          187     264#
STAT    000026R          186#
SVR0    000062R          201#
SVR1    000064R          202#
SVR2    000066R          203#
SVR3    000070R          204#
SVR4    000072R          205#
SVR5    000074R          206#
SVR6    000076R          207#
SYSCNT  000052R          196#
TIMES   000224R          260#    268*    378*
TRPDFD= 000022           228#
UCODE   002070R          408     428     635#
UCODE1  004070R          414     435     640#
VECTOR  000010R          177#
WASADR  000104R          211#
WDFR    000116R          218#
WDTO    000114R          217#
XFLAG   000005R          175#
.     = 004136R          567#    569#    571#    573#    575#    593#

. ABS.  000000     000
        004136     001

 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 XKUAB0,XKUAB0/SOL/CRF:SYM=DDXCOM,XKUAB0
 RUN-TIME: 1 2 .2 SECONDS
 RUN-TIME RATIO: 36/4=7.8
 CORE USED:  7K  (13 PAGES)
```