



Diagram showing a grid of 10 columns and 10 rows of data points, likely representing a micro-diagnostic chart for the KC750 RDM. The data is organized into a grid structure with various numerical values and labels.

Row	Col 1	Col 2	Col 3	Col 4	Col 5	Col 6	Col 7	Col 8	Col 9	Col 10
1	...	...	...	...	...	...	...	...	...	...
2	...	...	...	...	...	...	...	...	...	...
3	...	...	...	...	...	...	...	...	...	...
4	...	...	...	...	...	...	...	...	...	...
5	...	...	...	...	...	...	...	...	...	...
6	...	...	...	...	...	...	...	...	...	...
7	...	...	...	...	...	...	...	...	...	...
8	...	...	...	...	...	...	...	...	...	...
9	...	...	...	...	...	...	...	...	...	...
10	...	...	...	...	...	...	...	...	...	...



IDENTIFICATION

PRODUCT ID: ZZ-ECKAF-3.1

PRODUCT TITLE: ECKAF301 KC750 MICRO DIAGNOSTIC (L0006)

DECO/DEPO: 3.1

DATE: MAY 1982

MAINTAINED BY: VAX DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1980,1982

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 10754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MEY 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 SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

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

## Table of contents

(1)	24	..RDM Revision History"
(1)	1	..
(1)	3	..DIAGNOSTIC MICROCODE MACROS"
(1)	61	.. RDM Control File Macros"
(1)	72	.. Diagnostic Macros"
(1)	148	..
(1)	149	..MODULE GATE ARRAY MAPS"
(1)	151	.. L0001 MULTIPLIER AND GATE ARRAY LOCATI
(1)	208	.. L0002 GATE ARRAY LOCATION"
(1)	265	.. L0003 GATE ARRAY LOCATION"
(1)	323	.. L0004 GATE ARRAY LOCATION"
(1)	381	.. L0011 & L0016 GATE ARRAY LOCATION"
(1)	439	..
(1)	444	..EQUATED SYMBOLS
(1)	447	..
(1)	3	..TEST 001 DCS ADDRESS REGISTER DATA INTE
(1)	76	..TEST 002 MATCH/CS ADDRESS BUFFER REGIST
(1)	166	..TEST 003 CLOCK CONTROL BITS
(1)	245	..TEST 004 DCS CONTROL FILE DATA PATH
(1)	339	..TEST 005 DCS CONTROL FILE CLEAR
(1)	409	..TEST 006 DCS CHIP SELECT COUNTER
(1)	477	..TEST 007 DCS ADDRESS REGISTER COUNT LOG
(1)	553	..TEST 008 DCS DUAL ADDRESSING
(1)	647	..TEST 009 DCS DATA INTEGRITY
(1)	1043	..TEST 00A HI NEXT BUS DRIVERS
(1)	1136	..TEST 00B DCS ADDRESS DECODE
(1)	1230	..TEST 00C TRAP HALT ENABLE
(1)	1420	..TEST 00D DISABLE HIGH NEXT
(1)	1488	..TEST 00E DCS ADDRESS REGISTER M CLOCK
(1)	1551	..TEST 00F DCS CYCLE TEST
(1)	1877	..TEST 010 TEST TO SCOPE DCS PARITY ERROR
(1)	2194	..TEST 011 DISABLE OF DCS ADDR INCR
(1)	2275	..
(1)	2276	..MAINTENANCE REGISTER FUNCTIONAL TESTS
(1)	2277	..
(1)	2279	..TEST 012 WD AND WH REGISTER DATA INTEGR
(1)	2367	..TEST 013 VBUS DATA INTEGRITY
(1)	2547	..TEST 014 MATCH REGISTER INCREMENT WITH
(1)	2646	..TEST 015 CS ADDRESS MATCH INCR INHIBIT
(1)	2754	..TEST 016 TEST TRPOFF BIT
(1)	2829	..TEST 017 TEST STBINH BIT
(1)	2915	..
(1)	2916	..DCS CONTROL FILE FUNCTIONAL TESTS
(1)	2917	..
(1)	2919	..TEST 018 STOP CLOCK BIT
(1)	3003	..TEST 019 STROBE H REGISTER WITH W BUS
(1)	3094	..TEST 01A W BUS FROM RDM
(1)	3191	..TEST 01B DISABLE OF WH REG FROM WBUS
(1)	3282	..TEST 01C CLEAR DCS ADDRESS BIT
(1)	3364	..TEST 01D ENABLE TRACE REGISTER BIT
(1)	3521	..TEST 01E VBUS STROBE BIT
(1)	3661	..
(1)	3662	..STOP CLOCK FUNCTIONS"
(1)	3663	..
(1)	3665	..TEST 01F CLOCK STOP ON CONTROL STORE PA
(1)	3829	..TEST 020 CS ADDRESS MATCH
(1)	4082	..TEST 021 TEST CLOCK DETECTION CIRCUIT

ZZ-ECKAF-3.1

VAX-11/750 RDM MICRO DIAGNOSTICS

D 1  
6-MAY-1982

Fiche 1 Frame D1

Sequence 3

```
(1) 4150      ..  
(1) 4151      ..TRACE FUNCTIONS  
(1) 4152      ..  
(1) 4154      ..TEST 022   DCS WRITE WITH TRACE ENABLED  
(1) 4273      ..  
(1) 4274      ..CMI FUNCTIONS  
(1) 4275      ..  
(1) 4277      ..TEST 023   TEST RDM TO CMI DATA INTEGRITY  
(1) 4431      ..TEST 024   TEST RDM TO CMI CONTROL AND ST
```

ZZ-  
ECKAF-  
V03

ZZ-ECKAF-3.1  
ECKAF  
V03.01

VAX-11/750 RDM MICRO DIAGNOSTICS  
VAX-11/750 RDM MICRO DIAGNOSTICS

E 1  
6-MAY-1982

Fiche 1 Frame E1  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
9-APR-1982 16:42:54 WRKDS0:[COMET.RDM]TITLE.MAR;301 (1

Sequence 4

Page

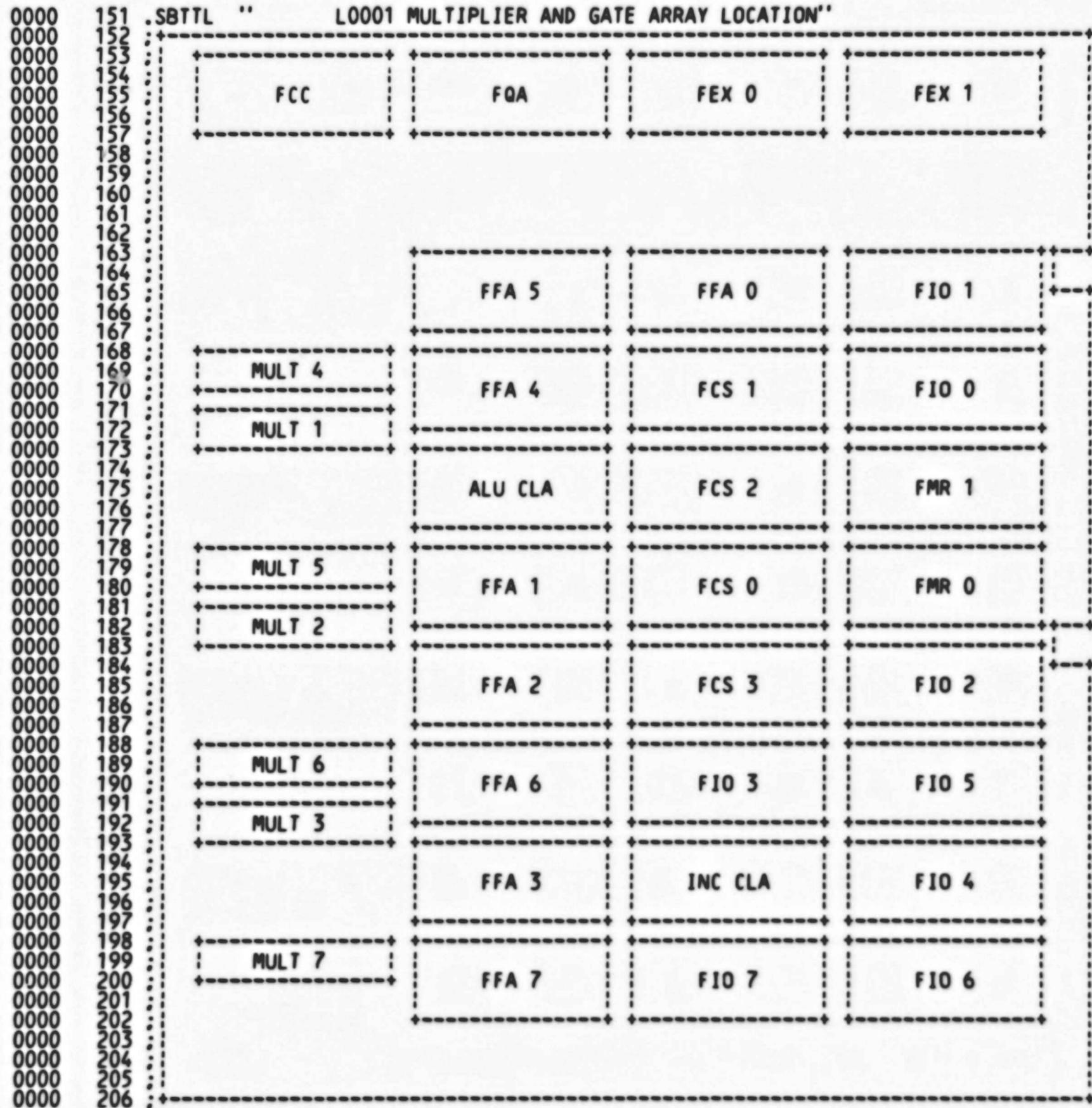
```
0000 1 .TITLE ECKAF VAX-11/750 RDM MICRO DIAGNOSTICS
0000 2 .IDENT /V03.01/
0000 3 :
0000 4 : COPYRIGHT (C) 1980,1981
0000 5 : DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000 6 :
0000 7 : THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE
0000 8 : COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE
0000 9 : ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000 10 : MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000 11 : EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000 12 : TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES
0000 13 : REMAIN IN DEC.
0000 14 :
0000 15 : THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 16 : AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 17 : CORPORATION.
0000 18 :
0000 19 : DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 20 : SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000 21 :
0000 22 :
```

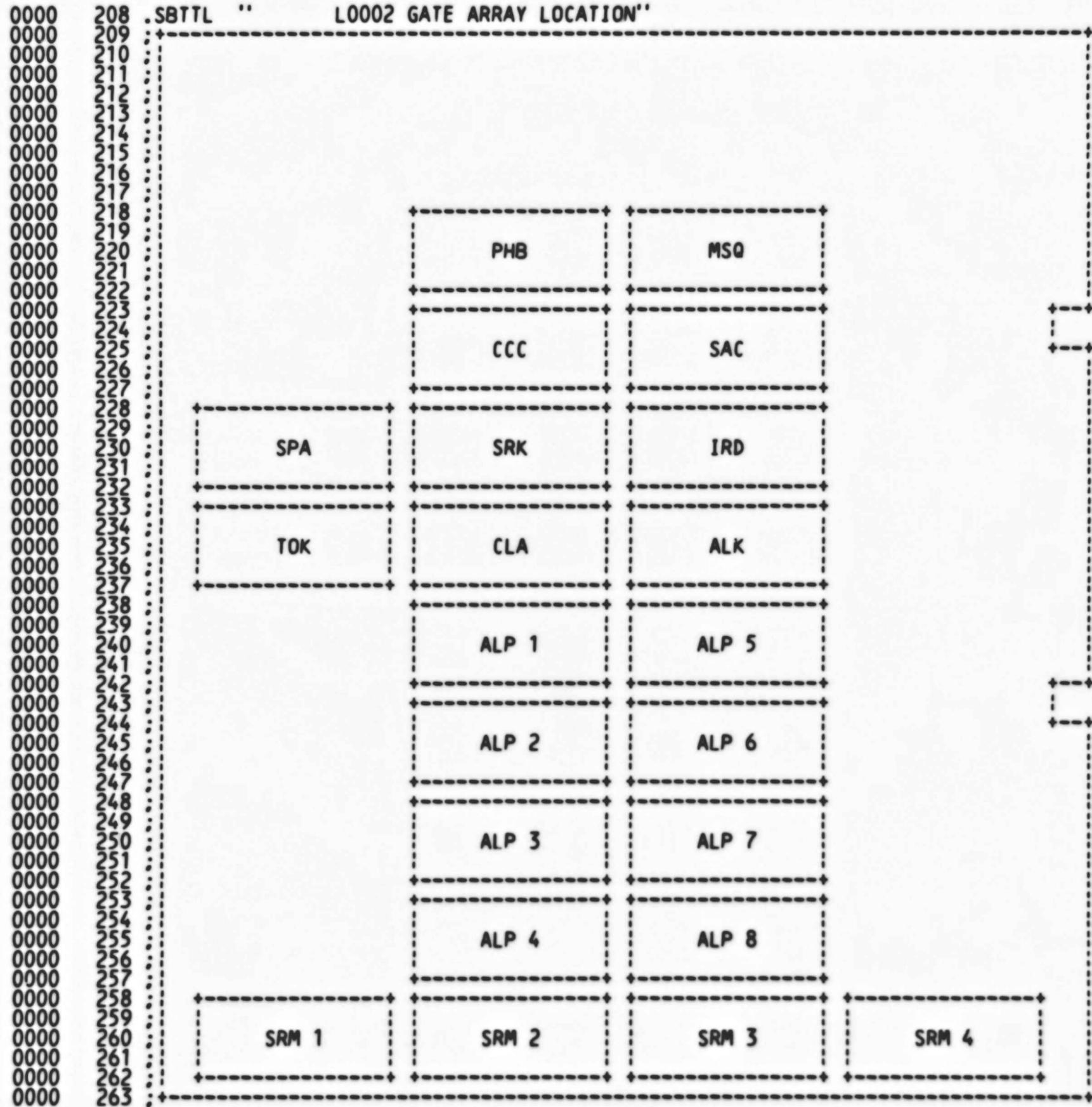
```
0000 24 .SBTTL 'RDM Revision History'
0000 25 :++
0000 26 : RDM MICRO-DIAGNOSTIC PACKAGE
0000 27 :
0000 28 : REVISION HISTORY
0000 29 :
0000 30 : 00-01 Bill Landry 4-FEB-1980
0000 31 : Started file.
0000 32 :
0000 33 : 00-02 Bill Landry 5-FEB-1980
0000 34 : Added MAINT_DCS_ENABL bit to the MAIN32 register.
0000 35 :
0000 36 : 00-03 Bill Landry 27-FEB-1980
0000 37 : Enhanced DCS DATA INTEGRITY test.
0000 38 :
0000 39 : 01-00 Bill Landry/Dave Spain 13-JUN-1980
0000 40 : Added diagnostic macros to tile section and submitted to
0000 41 : RELEASE ENGINEERING.
0000 42 :
0000 43 : 01-01 Bill Landry 25-AUG-1980
0000 44 : Remodeled RDM to CMI tests.
0000 45 :
0000 46 : 02-00 Bill Landry 29-SEP-1980
0000 47 : Second release to SDC
0000 48 :
0000 49 : 02-01 Bill Landry 4-NOV-1980
0000 50 : Added test to check B CLOCK detection circuit
0000 51 :
0000 52 : 03-00 Bill Landry 17-NOV-1980
0000 53 : Third release to SDC
0000 54 : 03-01 Dave Shull 08-Apr-1982
0000 55 : Re-Compiled ECKAF.SRC as version 8.00 of the monitor has BEGINSAs and
0000 56 : ENDSA pseudos removed which changed the offset values for some of the
0000 57 : pseudo opcodes. Re-compiling the source file will assign the correct
0000 58 : pseudo opcode values for compatibility with ECKAA.EXE ver. 800.
0000 59 : Changed the CMC callout to MCO and MC1.
0000 60 :
0000 61 :--
0000 1 .SBTTL ''
```

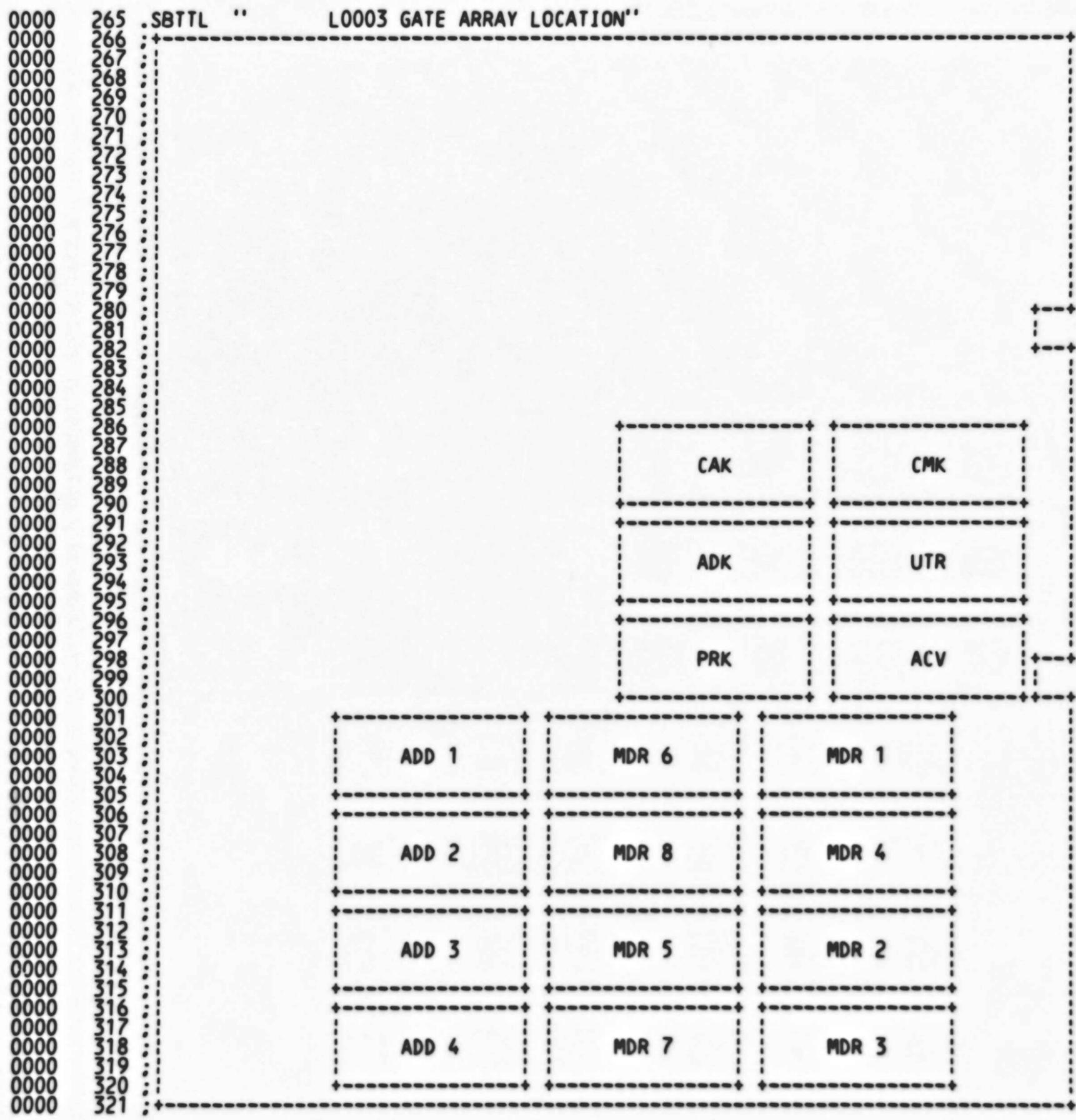
```
0000 3 .SBTTL 'DIAGNOSTIC MICROCODE MACROS''
0000 4 :++
0000 5 : Revision History
0000 6 :
0000 7 : 00-24 Dave Spain 2-Feb-82
0000 8 : Added FPA_R[]+M[] macro.
0000 9 : 00-23 Bill Landry 22-OCT-81
0000 10 : Added PSL_RDM macro
0000 11 : 00-22 Dave Spain 5 Oct 81
0000 12 : Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000 13 : 00-21 Bill Landry 29 Sept 81
0000 14 : Added FPA macro, FPA_M[] FPA_R[].
0000 15 : 00-20 Dave Spain 14 Aug 81
0000 16 : Added the TESTFPA [] macro.
0000 17 : 00-19 Dave Spain 18 July 81
0000 18 : Added the following FPA macros, FPA_RDM and FPA_R[].
0000 19 : 00-18 Dave Spain 24 Sept 80
0000 20 : Added RDM_Q macro. This macro uses the ALU.
0000 21 : 00-17 Bill Landry 23 Sept 80
0000 22 : Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000 23 : 00-16 Dave Spain 27 August 80
0000 24 : Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000 25 : 00-15 Bill Landry 4 June 80
0000 26 : Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000 27 : satisfy validity checks.
0000 28 : 00-14 Dave Spain 23 May 80
0000 29 : Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000 30 : Bill.
0000 31 : 00-13 Dave Spain 14 Apr 80
0000 32 : Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
0000 33 : the COMET micro-code defines to run.
0000 34 : 00-12 Bill Landry 1 Apr 80
0000 35 : Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000 36 : problem in ADD gate array. This is not an APRIL FOOL!
0000 37 : 00-11 Dave Spain 27 Mar 80
0000 38 : Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000 39 : 00-10 Bill Landry 19 Mar 80
0000 40 : Added Q_R[]_RL.P and VA Q.OR.R[]
0000 41 : 00-09 Bill Landry 19 Mar 80
0000 42 : Deleted TB_R[]_RL.P.OR.D
0000 43 : 00-08 Bill Landry 18 Mar 80
0000 44 : Added VA_R[]_RL.P
0000 45 : 00-07 Bill Landry 07 Mar 80
0000 46 : Added Q_M[]_RL.PTE
0000 47 : 00-06 Dave Spain 06 Mar 80
0000 48 : Changed TB_R[]_RL.P macro to TB_R[]_RL.P.OR.D.
0000 49 : 00-05 Dave Spain 03 Mar 80
0000 50 : Added TB_R[]_RL.P macro for Bill Landry.
0000 51 : 00-04 Dave Spain 22 Feb 80
0000 52 : Added CLRTB.VA_R[] macro for Bill Landry.
0000 53 : 00-03 Dave Spain 30 Jan 80
0000 54 : Added Tom Groetzinger's macro definitions.
0000 55 : 00-02 Dave Spain 30 Jan 80
0000 56 : Added Tony Vezza's macro definitions.
0000 57 : 00-01 Dave Spain 29 Jan 80
0000 58 : Macros initiated.
0000 59 :--
```

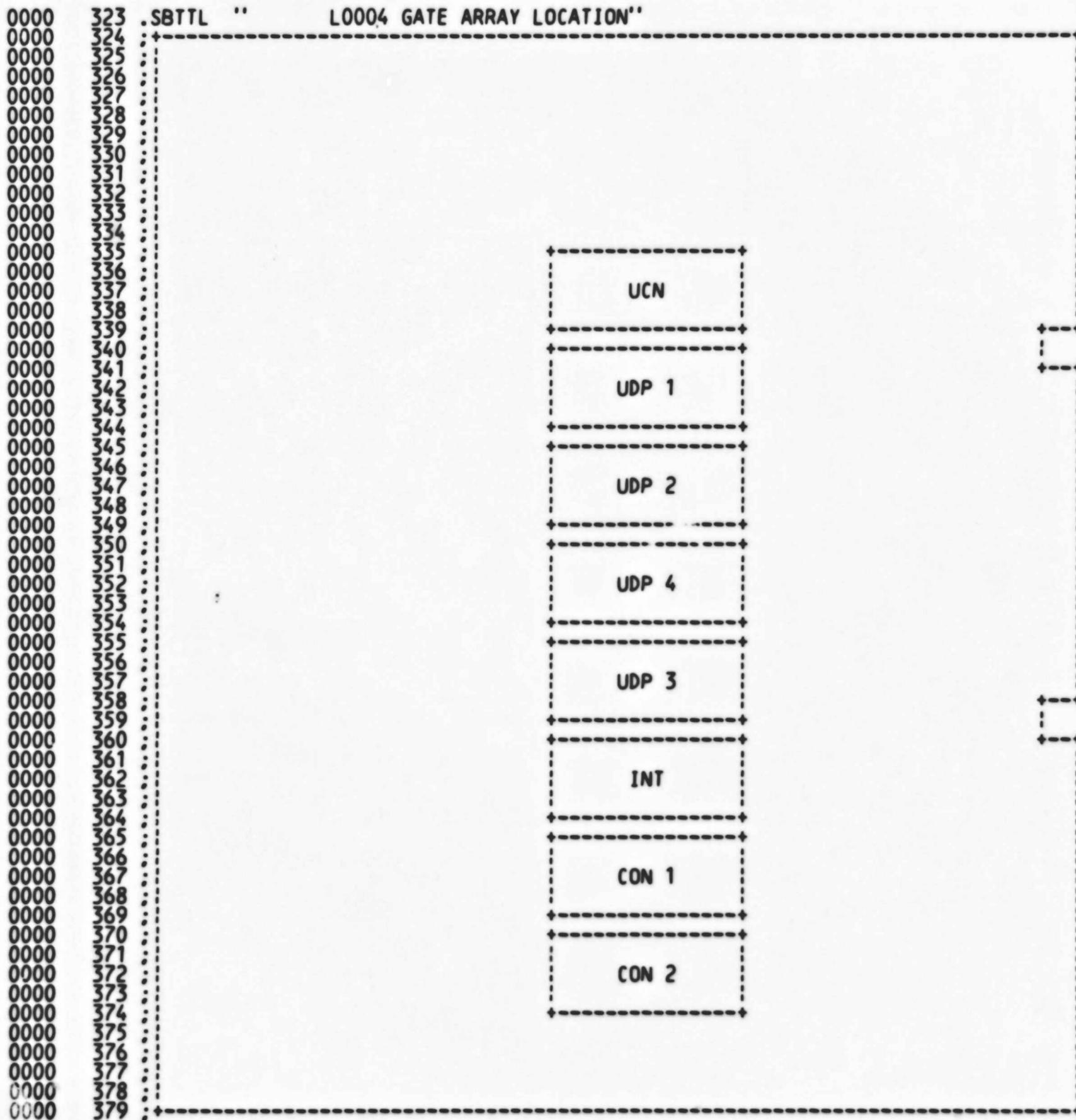
```
0000 60
0000 61          .SBTTL  ''      RDM Control File Macros''
0000 62
0000 63 :STROBE.V.BUS      'RDMCF0/V.STROBE''
0000 64 :DCS.ADDR 0       'RDMCF1/DCS.ADDR.CLR''
0000 65 :LSTOP.CLOCK    'RDMCF2/STOP.CPU.CLOCK''
0000 66 :LATCH.CS.ADDR  'RDMCF3/DIS.STROBE.CS.A''
0000 67 :RDM_WB          'RDMCF4/H.REG.FROM.WB''
0000 68 :WB_RDM          'RDMCF5/WB.FROM.D.REG''
0000 69 :RDM_CMI        'RDMCF6/STROBE.CMI''
0000 70 :INH.CLOCK.SA   'RDMCF7/INH.SA.CLOCK''
0000 71
0000 72          .SBTTL  ''      Diagnostic Macros''
0000 73
0000 74 :CLOCK.EXT      ''CLKX/XTND''
0000 75 :CLRCH.VA_RDM   'WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE''
0000 76 :CLRCH.VA_VA    'RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_WB
0000 77 :CLRTB.VA_D     'RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
0000 78 :                BUS/PRB.RD.PTE''
0000 79 :CLRTB.VA_RDM  'WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE''
0000 80 :CLRTB.VA_R[]  'RSRC/@1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
0000 81 :                BUS/PRB.RD.PTE''
0000 82 :CLRTB.VA_VA   'RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_WB
0000 83 :CLRTB.VA_WB   'WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE''
0000 84 :D_LONLIT      'RSRC/LONLIT,ALPCTL/WX_D.R.Q_D''
0000 85 :D_M[]_LONLIT 'D_LONLIT,MSRC/@1''
0000 86 :D_VA_0        'RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
0000 87 :                WCTRL/VA_WB''
0000 88 :FPA_M[]_FPA_RDM 'MSRC/@1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS''
0000 89 :FPA_M[]_FPA_R[] 'FPA/FPA_MBUS.FPA_WBUS,MSRC/@1,RSRC/@2,MUX/R
0000 90 :                ROT/ZERO,ALU/OR''
0000 91 :FPA_RDM        'WB_RDM,FPA/FPA_DATA.WBUS''
0000 92 :FPA_R[]       'RSRC/@1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000 93 :FPA_R[]+M[]   'RSRC/@1,MSRC/@2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000 94 :MEMSCAR_R[]   'RSRC/@1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB''
0000 95 :M[]_LONLIT    'MSRC/@1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT''
0000 96 :M[]_RDM       'WB_RDM,SPW/MLONG,MSRC/@1''
0000 97 :PC_PC+RDM     'WCTRL/PC_PC+WB_WB_RDM''
0000 98 :PC_PC+4_MB_XB 'MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]''
0000 99 :PC_PC+4_RDM_XB 'RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
0000 100 :                MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]''
0000 101 :PC_RDM       'WB_RDM,WCTRL/PC_WB''
0000 102 :PSE_RDM      'CCPSL/PSE_WB.CCBR_ALUS,WB_RDM''
0000 103 :Q_D_WX_R     'ALPCTL/WX_R.Q_D''
0000 104 :Q_LONLIT     'RSRC/LONLIT,ALPCTL/WX_S.Q_R''
0000 105 :Q_M[]_RL_PTE 'ALPCTL/WX_Q_S,MSRC/@1,ROT/RL.MM.PTE''
0000 106 :Q_R[]_AND_Q  'DQ1/Q_WX,RSRC/@1,MUX/R.Q,ALU/AND''
0000 107 :Q_R[]_OR_Q   'DQ1/Q_WX,RSRC/@1,MUX/R.Q,ALU/OR''
0000 108 :Q_R[]_RL_P  'ALPCTL/WX_Q_S,RSRC/@1,ROT/RL.RR.P''
0000 109 :RDM_ALUF     'ALPCTL/WB_ALUF,RDM_WB''
0000 110 :RDM_FPA      'FPA/WBUS_FPA,RDM_WB''
0000 111 :RDM_LONLIT   'WB_LONLIT,RDM_WB''
0000 112 :RDM_LONLIT.Q_M 'RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB''
0000 113 :RDM_M[]       'MSRC/@1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB''
0000 114 :RDM_M[]_OR_R[] 'WB_M[@1].OR.R[@2],RDM_WB''
0000 115 :RDM_M[]_R[] 'WB_M[@1]-R[@2],RDM_WB''
0000 116 :RDM_PC       'MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB''
```

```
0000 117 :RDM_PCBACK      'RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB''
0000 118 :RDM_Q          'RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB''
0000 119 :RDM_Q_Q_D      'ALPCTL/WX_Q.Q_D,RDM_WB''
0000 120 :RDM_R[]        'RSRC/a1,ROT7ZERO,MUX/R.S,ALU/OR,RDM_WB''
0000 121 :RDM_TB         'WB M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]''
0000 122 :RDM_VA         'MSRC/VA,RSRC7ZERO,MUX/M.R1,ALU/OR,RDM_WB''
0000 123 :RNUM_LONLIT    'ALPCTL/WX_D.R.Q.M,MSRC/RNUM_WBUS,RSRC7LONLIT''
0000 124 :R[]_RDM        'WB_RDM,SPW/RLONG,RSRC/a1''
0000 125 :R[]_DT_Q      'RSRC/a1,D_Q_Q_D,SPW/R.SIZE,V.SIZE/1''
0000 126 :R[]_VA         'RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG''
0000 127 :R[]_WB         'RSRC/a1,SPW/RLONG''
0000 128 :TB_RDM         'WCTRL/TB_WB,WB_RDM,MSRC/VA''
0000 129 :TB_WB         'WCTRL/TB_WB,MSRC/VA''
0000 130 :TESTFPA []    'TESTFPA/a1''
0000 131 :VA_PC+LONLIT+I.PC_PC+I 'RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/OR
0000 132 :              ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
0000 133 :              SIZE[LONG]''
0000 134 :VA_Q.OR.R[]   'WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q''
0000 135 :VA_RDM         'WB_RDM,WCTRL/VA_WB''
0000 136 :VA_R[]_RL.P   'ALPCTL/WX_S,ROT7RL.RR.P,RSRC/a1,WCTRL/VA_WB''
0000 137 :VA_VA-ZLITO[] 'WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLITO,MSRC/VA,
0000 138 :              MUX/M.S,ALU/A-B-CI''
0000 139 :VA_WB         'WCTRL/VA_WB''
0000 140 :WB_LONLIT     'RSRC/LONLIT,ALPCTL/WX_R.Q.M''
0000 141 :WB_LONLIT.Q_M 'RSRC/LONLIT,ALPCTL/WX_R.Q.M''
0000 142 :WB_M[]+R[]    'ALU/A+B+CI,ALUCI/ZERO,MUX7M.R1,MSRC/a1,RSRC/a2''
0000 143 :WB_R[]_Q_D   'RSRC/a1,ALPCTL/WX_R.Q.D''
0000 144 :WDR_R[]      'WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/ZER
0000 145 :WRITE.LONG R[] 'BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
0000 146 :              MUX/R.S,ALU/OR,SIZE[LONG]''
0000 147 :
0000 148 :SBTTL ''
0000 149 :SBTTL 'MODULE GATE ARRAY MAPS''
```

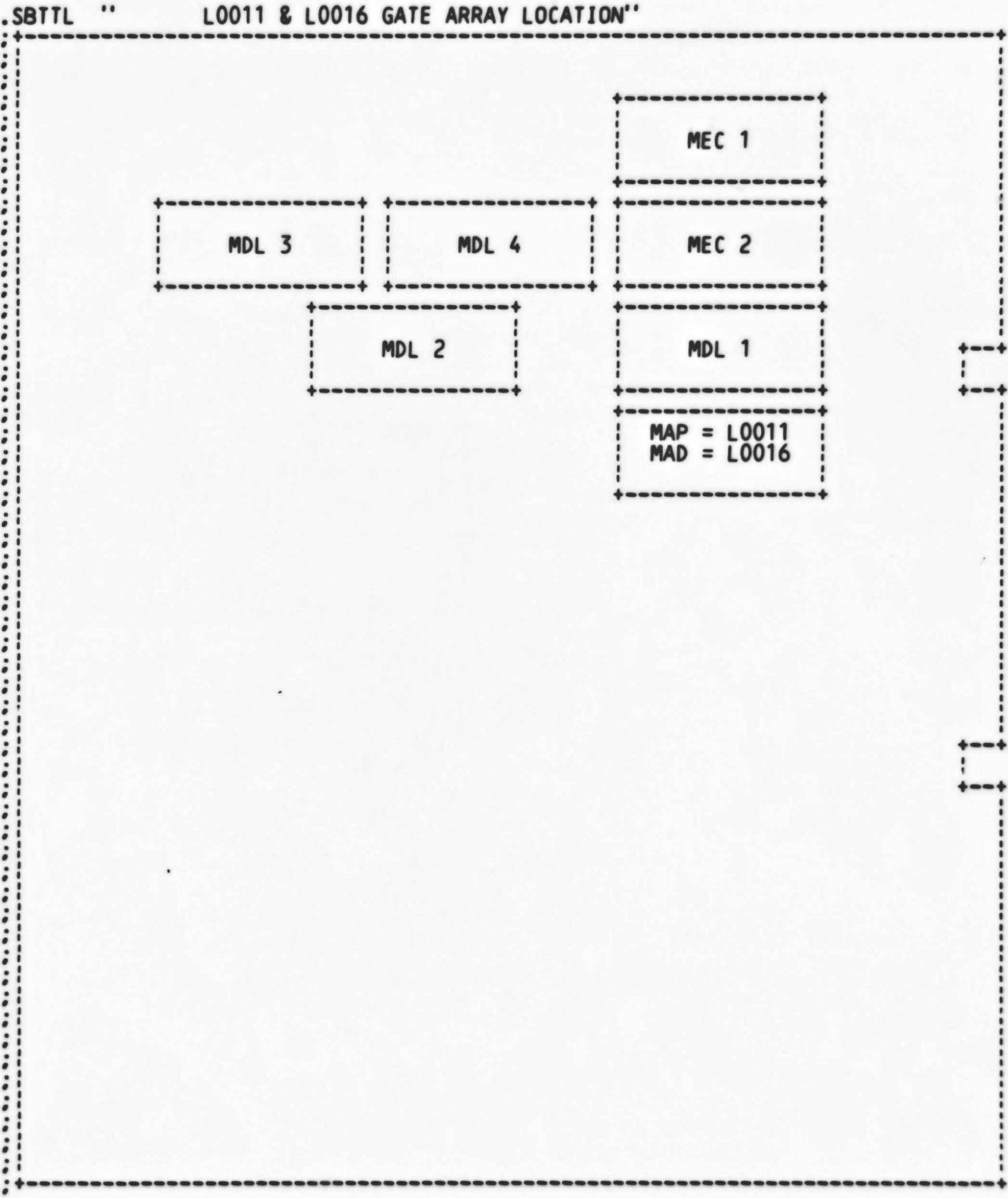








0000 381  
0000 382  
0000 383  
0000 384  
0000 385  
0000 386  
0000 387  
0000 388  
0000 389  
0000 390  
0000 391  
0000 392  
0000 393  
0000 394  
0000 395  
0000 396  
0000 397  
0000 398  
0000 399  
0000 400  
0000 401  
0000 402  
0000 403  
0000 404  
0000 405  
0000 406  
0000 407  
0000 408  
0000 409  
0000 410  
0000 411  
0000 412  
0000 413  
0000 414  
0000 415  
0000 416  
0000 417  
0000 418  
0000 419  
0000 420  
0000 421  
0000 422  
0000 423  
0000 424  
0000 425  
0000 426  
0000 427  
0000 428  
0000 429  
0000 430  
0000 431  
0000 432  
0000 433  
0000 434  
0000 435  
0000 436  
0000 437



```
0000 439      .SBTTL  ""
0000 440      .LIST  MC
0000 441      .LIBRARY /WRKDS0:[COMET]COMETMAC/
0000 442      .LIST  ME
0000 443      .NLIST CND,MC
0000          .SBTTL  'EQUATED SYMBOLS
0000
0000          :
0000          : RDM REGISTER DEFINITIONS:
0000          :
0000          : NOTE: THESE REGISTER DEFINITIONS ARE OFFSET BY THE LOAD/START ADDRESS OF
0000          : THE PROGRAM. FOR THE SIMULATOR VERSION THIS IS 400(X) AND FOR THE
0000          : REAL VERSION IT IS 8400(X).
0000          :
00007400 0000      DCS_DATA_REG      =      ^XF800 + HEAD
00007401 0000      DCS_ADDR_REG_WO =      ^XF801 + HEAD
00007402 0000      DCS_CTRL_REG     =      ^XF802 + HEAD
00007403 0000      DCS_CTRL_FILE    =      ^XF803 + HEAD
00007404 0000      ADDR_MATCH_LO    =      ^XF804 + HEAD
00007405 0000      ADDR_MATCH_HI    =      ^XF805 + HEAD
00007406 0000      STATUS_REG       =      ^XF806 + HEAD
0000          :
00007410 0000      A_REG_BYTE_0     =      ^XF810 + HEAD
00007411 0000      A_REG_BYTE_1     =      ^XF811 + HEAD
00007412 0000      A_REG_BYTE_2     =      ^XF812 + HEAD
00007413 0000      A_REG_BYTE_3     =      ^XF813 + HEAD
00007414 0000      MD_REG_BYTE_0    =      ^XF814 + HEAD
00007415 0000      MD_REG_BYTE_1    =      ^XF815 + HEAD
00007416 0000      MD_REG_BYTE_2    =      ^XF816 + HEAD
00007417 0000      MD_REG_BYTE_3    =      ^XF817 + HEAD
00007418 0000      MH_REG_BYTE_0    =      ^XF818 + HEAD
00007419 0000      MH_REG_BYTE_1    =      ^XF819 + HEAD
0000741A 0000      MH_REG_BYTE_2    =      ^XF81A + HEAD
0000741B 0000      MH_REG_BYTE_3    =      ^XF81B + HEAD
0000741C 0000      CMT_CTRL_REG     =      ^XF81C + HEAD
0000741D 0000      MAINT_REG        =      ^XF81D + HEAD
0000741E 0000      RD_CTRL_REG      =      ^XF81E + HEAD
0000          :
00007420 0000      FRONT_PNL_1      =      ^XF820 + HEAD
00007421 0000      FRONT_PNL_2      =      ^XF821 + HEAD
00007422 0000      CS_ADD_TRAP_LOW  =      ^XF822 + HEAD
00007423 0000      CS_ADD_TRAP_HGH  =      ^XF823 + HEAD
00007424 0000      CS_ADD_BUFF_LOW  =      ^XF824 + HEAD
00007425 0000      CS_ADD_BUFF_HGH  =      ^XF825 + HEAD
00007426 0000      BUFF_ADDR_LOW   =      ^XF826 + HEAD
00007427 0000      DCS_ADDR_REG_RO  =      ^XF827 + HEAD
0000          :
0000742C 0000      WD_REG_BYTE_0     =      ^XF82C + HEAD
0000742D 0000      WD_REG_BYTE_1     =      ^XF82D + HEAD
0000742E 0000      WD_REG_BYTE_2     =      ^XF82E + HEAD
0000742F 0000      WD_REG_BYTE_3     =      ^XF82F + HEAD
00007430 0000      WH_REG_BYTE_0     =      ^XF830 + HEAD
00007431 0000      WH_REG_BYTE_1     =      ^XF831 + HEAD
00007432 0000      WH_REG_BYTE_2     =      ^XF832 + HEAD
00007433 0000      WH_REG_BYTE_3     =      ^XF833 + HEAD
0000          :
0000          : RDM REGISTER NAMES AS DEFINED IN HARDWARE SPEC
0000          :
```

```

00007400 0000 DCSDA=DCS_DATA_REG ;Diagnostic control store data register
00007401 0000 DCSAD=DCS_ADDR_REG_WO ;Diagnostic control store address reg
00007402 0000 DCSCR=DCS_CTRL_REG ;Diagnostic control store control reg
00007403 0000 DCSCF=DCS_CTRL_FILE ;Diagnostic control store control file
00007404 0000 CSAMR=ADDR_MATCH_LO ;Control store address match register
00007406 0000 STATUS=STATUS_REG ;Status register
00007410 0000 MAREG=A_REG_BYTE_0 ;Memory address register
00007414 0000 MDREG=MD_REG_BYTE_0 ;Memory data register
00007418 0000 MHREG=MH_REG_BYTE_0 ;Memory holding register
0000741C 0000 CMICtrl=CMI_CTRL_REG ;CMI control
0000741D 0000 MAIN32=MAINT_REG ;32 Bit path maintenance control
0000741E 0000 RDCTRL=RD_CTRL_REG ;Remote diagnosis control
00007420 0000 FRONT1=FRONT_PNL_1 ;Front panel readback 1
00007421 0000 FRONT2=FRONT_PNL_2 ;Front panel readback 2
00007422 0000 CSTRP=CS_ADD_TRAP_LOW ;Control store address trap register
00007424 0000 CSBUF=CS_ADD_BUFF_LOW ;Control store address buffer
00007426 0000 BADD1=BUFF_ADDR_LOW ;Control store address trace readback
00007427 0000 CLKDCS=DCS_ADDR_REG_RO ;Clock control & DCS address register
0000742C 0000 WDREG=WD_REG_BYTE_0 ;WBUS data register
00007430 0000 WHREG=WH_REG_BYTE_0 ;WBUS holding register

```

... RDM REGISTER BIT DEFINITIONS:

```

... DCS CONTROL REGISTER
00000001 0000 HALT_ON_MATCH = 1
00000002 0000 CLEAR_CTRL_FILE = 2
00000004 0000 TRACE_ENABL = 4
00000008 0000 PAR_CHK_ENABL = 8
00000010 0000 PAR_STOP_ENABL = ^X10
00000020 0000 CLK_CTRL_0 = ^X20 ; ACTIVE LOW
00000040 0000 CLK_CTRL_1 = ^X40 ; ACTIVE LOW
00000080 0000 MICRO_ADDR_INH = ^X80

```

... FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS

CLK_CTRL 1,0	FUNCTION
11	HALT
01	SINGLE MICRO INSTRUCTION
10	SINGLE TICK
00	RUN

... DCS CONTROL FILE ALL OF THESE BITS ARE ACTIVE LOW

```

...
00000001 0000 V_BUS_STROBE = 1
00000002 0000 DCS_ADDR_CLEAR = 2
00000004 0000 STOP_CLOCK = 4
00000008 0000 EN_TRACE_REG = 8
00000010 0000 H_FROM_WBUS = ^X10
00000020 0000 WBUS_FROM_D = ^X20
00000040 0000 STROBE_CMI = ^X40
00000080 0000 SA_CLOCK = ^X80

```

... CS ADDRESS MATCH REGISTER (HIGH)

```

...
00000040 0000 VBUS_SERIAL_IN = ^X40 ; ACTIVE LOW
00000080 0000 VBUS_CLOCK = ^X80

```

```
0000      : STATUS REGISTER
0000      :
00000001 0000      VBUS_SERIAL_OUT =      1
00000002 0000      TRAP_OFF      =      2
00000008 0000      CMI_STATUS_0  =      8
00000010 0000      CMI_STATUS_1  =     ^X10
0000      :
0000      : THE CMI STATUS BITS ARE ENCODED AS FOLLOWS:
0000      :
0000      :     00 = NO RESPONSE
0000      :     01 = UNCORRECTABLE ERROR
0000      :     10 = CORRECTABLE ERROR
0000      :     11 = NO ERRORS
00000020 0000      CMI_COMPLETE =     ^X20
00000080 0000      :
00000080 0000      CLOCK_STOPED =     ^X80
0000      :
0000      : CMI CONTROL REGISTER
0000      :
00000001 0000      CMI_CTRL_CLEAR =      1
00000008 0000      CMI_GO      =      8
00000020 0000      CMI_WRITE   =     ^X20
00000040 0000      CMI_READ    =     ^X40
00000080 0000      SA_START_STOP =     ^X80
0000      :
0000      : 32 BIT MAINTENANCE CONTROL REGISTER
0000      :
00000001 0000      MAINT_TRAP_OFF =      1
00000002 0000      MAINT_STB_INH  =      2
00000004 0000      MAINT_DCS_ENABL =      4
00000008 0000      MAINT_WB_TO_WH =      8
00000010 0000      MAINT_WD_TO_WB =     ^X10
00000020 0000      MAINT_CMI_TO_MH =     ^X20
00000040 0000      MAINT_MD_TO_CMI =     ^X40
00000080 0000      MAINT_A_TO_CMI =     ^X80
0000      :
0000      : RD CONTROL REGISTER
0000      :
00000008 0000      MASTER_HALT_EN =      8
00000010 0000      VBUS_LOAD     =     ^X10
00000020 0000      TRAP_HALT_EN  =     ^X20
00000040 0000      DC_LOW      =     ^X40
00000080 0000      AC_LOW      =     ^X80
0000      :
0000      : FRONT PANEL REGISTER 1
0000      :
00000001 0000      FP_BOOT_0    =      1
00000002 0000      FP_BOOT_1    =      2
00000004 0000      FP_START_0   =      4
00000008 0000      FP_START_1   =      8
00000010 0000      CPD_RUN     =     ^X10
00000020 0000      PARITY_ERROR =     ^X20
0000      :
0000      : FRONT PANEL REGISTER 2
0000      :
00000001 0000      REMOTE      =      1
00000002 0000      FAULT      =      2
```

```

00000004 0000          TEST          =          4
00000000 0000          :
00000000 0000          :
00000000 0000          :
00000040 0000          PB_INIT          =          ^X40      ; ACTIVE LOW
00000080 0000          FRONT_PNL_LOCK =          ^X80
00000000 0000          :
00000000 0000          : DCS ADDRESS REGISTER READ ONLY
00000000 0000          :
00000040 0000          CLK_CTRL_0_RO   =          ^X40      ; ACTIVE HIGH
00000080 0000          CLK_CTRL_1_RO   =          ^X80      ; ACTIVE HIGH
00000000 0000          :
00000000 0000          : MISCELLANEOUS EQUATES:
00000001 0000          BYTE           =          1
00000002 0000          WORD           =          2
00000004 0000          LONG           =          4
0000000A 0000          MICROWORD      =          10
00002C4D 0000          MONITOR_SIZE   =          ^X2C4D
00000400 0000          STACK_SIZE     =          1024
000007B3 0000          M$TEST_SIZE    =          14336 - STACK_SIZE - MONITOR_SIZE
000003F3 0000          M$TEST_SIZE_D  = M$TEST_SIZE - 960
00000000 0000          : ; MAXIMUM SIZE OF A TEST OVERLAY
00000000 0000          : ; MAXIMUM SIZE OF TEST OVERLAY WITH
00000000 0000          : ; 'DEBUG' ENABLED IN
00000001 0000          B             =          BYTE
00000002 0000          W             =          WORD
00000004 0000          L             =          LONG
0000000A 0000          M             =          MICROWORD
00000001 0000          HIGH          =          1
00000000 0000          LOW           =          0
00000000 0000          ONERROR       =          0
00000001 0000          NOERROR      =          1
00000002 0000          ONEQUAL      =          2
00000003 0000          NOTEQUAL     =          3
00000010 0000          MAX_ARGUMENTS =          16      ; BYTE LENGTH OF LARGEST PSEUDO OP
00000036 0000          DCS_SCRATCH_ADR =          54      ; DCS SCRATCH ADDRESS START
000032FC 0000          ERROR_LOG_ADR =          ^XB6FC + HEAD
00000000 0000          : ; START ADDRESS OF MEMORY ERROR LOG
00000000 0000          445          .NLIST    ME
00000000 0000          446          .LIST     MC
00000000 0000          447          .SBTTL   ""
00000000 0000          448          .NLIST    MC ; SHUT-OFF LISTING FOR FIRST 'NEWTST' CALL

```

```
0028 .SBTTL 'TEST 001 DCS ADDRESS REGISTER DATA INTEGRITY
0028 4 :*****
0028 5 :++
0028 6 :
0028 7 : FUNCTIONAL DESCRIPTION:
0028 8 :
0028 9 : This test loads data patterns into the DCS ADDRESS REGISTER
0028 10 : and reads them back thru the Read Only copy of the ADDRESS
0028 11 : REGISTER.
0028 12 :
0028 13 :
0028 14 : TEST ALGORITHM:
0028 15 :
0028 16 : 1. Set up a test loop of 4 iterations.
0028 17 : 2. Generate a test pattern
0028 18 : 3. Load the test pattern into the DCS ADDRESS REGISTER
0028 19 : 4. Test the contents of the read only CLKDCS REGISTER
0028 20 : (masking out the clock control bits)
0028 21 : 5. If no error go to step 2 for next test pattern
0028 22 :
0028 23 :
0028 24 : TEST PATTERNS:
0028 25 :
0028 26 : AA
0028 27 : 55
0028 28 : 33
0028 29 : 0F
0028 30 :
0028 31 :
0028 32 : LOGIC DESCRIPTION:
0028 33 :
0028 34 : This test verifies the ability of the 8085 to load the DCS ADDRESS
0028 35 : REGISTER over the 8085 data path. Test patterns are then read back
0028 36 : to the 8085 via the CLOCK CONTROL & DCS ADDRESS REGISTER where they are
0028 37 : compared for accuracy.
0028 38 :
0028 39 :
0028 40 : ASSUMPTIONS:
0028 41 :
0028 42 : This test assumes the power up self test has completed succesfully.
0028 43 :
0028 44 :
0028 45 : ERROR DESCRIPTION:
0028 46 :
0028 47 : EXPECTED DCS ADDRESS REGISTER CONTENTS
0028 48 : RECEIVED DCS ADDRESS REGISTER CONTENTS
0028 49 : LOOP COUNT
0028 50 :--
0028 51 :*****
0028 52 :
0028 53 :
0028 54 :
0028 55 :
0028 56 : INITIALIZE ; Init the 750 CPU
002A 57 : LOOP I,1,4 ; Setup the loop parameters
0031 58 : SA01PAT I,1$,.. ; Generate the pattern
0039 59 : ERRLOOP ; Set the ERROR loop
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 001 DCS ADDRESS REGISTER DATA INTE  
VAX-11/750 RDM MICRO DIAGNOSTICS

G 2  
6-MAY-1982

6-MAY-1982 19:54:55

Fiche 1 Frame G2

Sequence 19

VAX-11 Macro V02.46

Page 1 (1

'TEST 001 DCS ADDRESS REGISTER DATA INTE 6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1

```
003B 60 LOADREG DCSAD,1$,B ; Write the register
0043 61 CMPREGMSK CLKDCS,1$,,2$,,B, ; Read the data back and check
004F 62 IFERROR i,<<RDM>> ; RDM bad if compare failed
0055 63 ENDLOOP ; Terminate the loop
0058 64 SKIP ; Go to end of test
005F 65
005F 66
005F 67
005F 68
005F 69 BEGIN_TEST_DATA
005F ;-----
005F 70 ;
00000000 005F 71 1$: .LONG 0 ; Test pattern is loaded here
3F 0063 72 2$: .BYTE ^X3F ; Mask for ADDRESS REGISTER
0064 73
0064 74 END_TEST_DATA
0064
0064 ;-----
0064 75 ENDTEST
```

```
0035 .SBTTL 'TEST 002 MATCH/CS ADDRESS BUFFER REGISTERS DATA INTEGRITY
0035 77 :*****
0035 78 :++
0035 79 :
0035 80 : FUNCTIONAL DESCRIPTION:
0035 81 :
0035 82 : This test checks the data integrity of the ADDRESS MATCH register
0035 83 : and the CONTROL STORE BUFFER register.
0035 84 :
0035 85 :
0035 86 : TEST ALGORITHM:
0035 87 :
0035 88 : 1. Create a test loop of 4 iterations
0035 89 : 2. Generate a test pattern
0035 90 : 3. Assert PARITY CHECK and MICRO ADDRESS INHIBIT bits.
0035 91 : 4. Load MATCH register with test pattern
0035 92 : 5. Check CONTROL STORE BUFFER register for correct pattern
0035 93 : 6. Deassert PARITY CHECK and MICRO ADDRESS INHIBIT bits.
0035 94 : 7. Check CONTROL STORE BUFFER register for not test pattern.
0035 95 : 8. If no errors go to step 2 for remaining iterations
0035 96 :
0035 97 :
0035 98 : TEST PATTERNS:
0035 99 :
0035 100 : AAAA
0035 101 : 5555
0035 102 : 3333
0035 103 : 0F0F
0035 104 :
0035 105 :
0035 106 : LOGIC DESCRIPTION:
0035 107 :
0035 108 : This test checks the ability of the 8085 to successfully load test
0035 109 : patterns into the CONTROL STORE ADDRESS MATCH REGISTER and read
0035 110 : them back via the CONTROL STORE ADDRESS BUFFER.
0035 111 :
0035 112 :
0035 113 : ASSUPTIONS:
0035 114 :
0035 115 : This test assumes the power up self test has run successfully.
0035 116 :
0035 117 :
0035 118 : ERROR DESCRIPTION:
0035 119 :
0035 120 : There are two IFERROR statements in this test. Check the PC to
0035 121 : determine which one is failing.
0035 122 :
0035 123 : EXPECTED CONTROL STORE BUFFER ADDRESS
0035 124 : RECEIVED CONTROL STORE BUFFER ADDRESS
0035 125 : LOOP COUNT
0035 126 :
0035 127 : If the second IFERROR statement failed it is possible that the
0035 128 : PARITY CHECK BIT in the DCS CONTROL REGISTER is stuck on.
0035 129 :
0035 130 : --
0035 131 :*****
0035 132 :
```

```

0035 133
0035 134
0035 135 INITIALIZE : Init the 750 CPU
0037 136 LOOP I,1,4 : Init the loop count
003E 137 SA01PAT I,1$,, : Generate the test pattern
0046 138 ERRLOOP : Set the ERROR LOOP
0048 139 LOADREG DCSCR,2$,,B : Set PARITY CHECK &
0050 140 : MICRO ADDRESS INHIBIT
0050 141 LOADREG CSAMR,1$,,W : Load the MATCH register
0058 142 CMPREGMSK CSBUF,1$,,3$,,W, : Check the CS ADDRESS
0064 143 : BUFFER register
0064 144 IFERROR ,,<<RDM>> : Stuck bit in CS ADDRESS PATH
006A 145 LOADREG DCSCR,4$,,B : Clear PARITY CHECK &
0072 146 : MICRO ADDRESS INHIBIT
0072 147 CMPREGMSK CSBUF,1$,,3$,,W,NE : Check that CS ADDRESS
007E 148 : BUFFER register is different
007E 149 : than MATCH register
007E 150 IFERROR ,,<<RDM>>
0084 151 ENDLOOP i : Terminate the test loop
0087 152 SKIP : Go to next test
008E 153
008E 154
008E 155
008E 156 BEGIN_TEST_DATA
008E :-----
008E 157 :
00000000 008E 158 1$: .LONG 0 : Buffer for test patterns
E8 0092 159 2$: .BYTE MICRO ADDR INH!- : Enable Parity Check
0093 160 PAR CRK ENABL!CLK_CTRL_1!CLK_CTRL_0
1F FF 0093 161 3$: .BYTE ^XFF,^XTF : Mask for ADDRESS BUFFER register
60 0095 162 4$: .BYTE CLK_CTRL_1!CLK_CTRL_0 : Deassert Parity Check Enable
0096 163
0096 164 END_TEST_DATA
0096 :-----
0096 165 :
0096 165 ENDTEST

```

```
0017 .SBTTL 'TEST 003 CLOCK CONTROL BITS
0017 167 :*****
0017 168 :++
0017 169
0017 170 : FUNCTIONAL DESCRIPTION:
0017 171
0017 172 : This test ensures that the clock control bits in the DCS CONTROL
0017 173 : register and the CLOCK CONTROL register (read only) function
0017 174 : correctly.
0017 175
0017 176 : TEST ALGORITHM:
0017 177
0017 178 : 1. Clear MASTER HALT ENABLE BIT to allow uninterrupted operation
0017 179 : 2. Create a test loop of 4 iterations
0017 180 : 3. Load test pattern into clock control bits of DCS CONTROL REGISTER
0017 181 : 4. Read CLOCK CONTROL & DCS ADDRESS REGISTER for compliment of
0017 182 : test pattern.
0017 183 : 5. If no error go to step 3 for remaining test patterns.
0017 184
0017 185
0017 186 : TEST PATTERNS:
0017 187
0017 188 : 0
0017 189 : 1
0017 190 : 2
0017 191 : 3
0017 192
0017 193
0017 194 : LOGIC DESCRIPTION:
0017 195
0017 196 : This test checks the ability of the 8085 to load the clock control bits
0017 197 : in the DCS CONTROL REGISTER. The bits are then read back via the
0017 198 : CLOCK CONTROL & DCS ADDRESS REGISTER.
0017 199
0017 200
0017 201 : ASSUMPTIONS:
0017 202
0017 203 : This test assumes the power-up self test has run successfully.
0017 204
0017 205
0017 206 : ERROR DESCRIPTION:
0017 207
0017 208 : EXPECTED CLOCK CONTROL & DCS ADDRESS REGISTER CONTENTS
0017 209 : RECEIVED CLOCK CONTROL & DCS ADDRESS REGISTER CONTENTS
0017 210 : LOOP COUNT
0017 211
0017 212 : --
0017 213 : *****
0017 214
0017 215
0017 216
0017 217
0017 218 : INITIALIZE :Init cpu
0019 219 : LOADREG RDCTRL,4$,,B : Clear Master Halt Enable
0021 220 : LOOP I,1,4 : Init the test loop
0028 221 : ERRLOOP : Set the error loop address
002A 222 : LOADREG DCSCR,1$,I,B : Load the control register
```

```
0032 223      CMPREGMSK      CLKDCS,2$,I,3$,,B,      ; Check contents of CLKDCS REG
003E 224      IFERROR          ;, <<RDM>>          ; Clock control bits stuck
0044 225      ENDLOOP          ; Terminate the test loop
0047 226      SKIP              ; Go to the next test
004E 227
004E 228
004E 229
004E 230      BEGIN_TEST_DATA
004E ;-----
004E 231 ;
80 004E 232 1$: .BYTE MICRO_ADDR_INH
A0 004F 233 .BYTE CLK_CTRL_0!MICRO_ADDR_INH
C0 0050 234 .BYTE CLK_CTRL_1!MICRO_ADDR_INH ; DCS CONTROL register
E0 0051 235 .BYTE CLK_CTRL_1!CLK_CTRL_0!MICRO_ADDR_INH ; Test patterns that get loaded
00 0052 236 2$: .BYTE 0
40 0053 237 .BYTE CLK_CTRL_0_RO
80 0054 238 .BYTE CLK_CTRL_1_RO
C0 0055 239 .BYTE CLK_CTRL_0_RO!CLK_CTRL_1_RO ; Received data from DCS ADDRESS registe
C0 0056 240 3$: .BYTE CLK_CTRL_0_RO!CLK_CTRL_1_RO
00 0057 241 4$: .BYTE 0
0058 242
0058 243      END_TEST_DATA
0058 ;-----
0058 244      ENDTEST
```

```
001F .SBTTL 'TEST 004 DCS CONTROL FILE DATA PATH
001F 246 :*****
001F 247 :++
001F 248 :
001F 249 : FUNCTIONAL DESCRIPTION:
001F 250 :
001F 251 : This test checks the data path to and from the DCS control file.
001F 252 : Implicitly tested is the logic that clears the DCS chip select
001F 253 : counter when the DCS ADDRESS register is loaded.
001F 254 :
001F 255 :
001F 256 : TEST ALGORITHM:
001F 257 :
001F 258 : 1. Create a loop of 64 to select DCS addresses (I)
001F 259 : 2. Create a loop of 4 to select test patterns (J)
001F 260 : 3. Load the DCS ADDRESS register with the selected address
001F 261 : 4. Load the test pattern into the control file
001F 262 : 5. Load the DCS ADDRESS register with the selected address
001F 263 : 6. Load the DCS ADDRESS register with the selected address again
001F 264 : 7. Read the CONTROL FILE register and check for the test pattern
001F 265 : 8. If no error go to step 3 for the remaining test patterns
001F 266 : 9. When loop J exhausted go to step 1 until loop I is exhausted
001F 267 :
001F 268 :
001F 269 : TEST PATTERNS:
001F 270 :
001F 271 : AA
001F 272 : 55
001F 273 : 33
001F 274 : 0F
001F 275 :
001F 276 :
001F 277 : LOGIC DESCRIPTION:
001F 278 :
001F 279 : This test checks the ability of the 8085 to load test patterns
001F 280 : into the DCS CONTROL FILE ram and read them back via the DCS
001F 281 : CONTROL FILE latch. Also tested is the logic that clears the
001F 282 : DCS CHIP SELECT COUNTER and DECODER when the DCS ADDRESS REGISTER
001F 283 : is loaded. You will note that the DCS ADDRESS REGISTER is loaded
001F 284 : twice in a row at one point in the test. This is to allow the
001F 285 : DCS CONTROL FILE LATCH to clock in the CONTROL FILE data that is
001F 286 : output by the DCS CONTROL FILE RAMS as a result of the first DCS
001F 287 : ADDRESS REGISTER load. All 64 CONTROL FILE locations are tested.
001F 288 :
001F 289 :
001F 290 : ASSUMPTIONS:
001F 291 :
001F 292 : This test assumes that the DCS ADDRESS REGISTER DATA INTEGRITY
001F 293 : test has been successfully run.
001F 294 :
001F 295 :
001F 296 : ERROR DESCRIPTION:
001F 297 :
001F 298 : EXPECTED DCS CONTROL FILE REGISTER DATA
001F 299 : RECEIVED DCS CONTROL FILE REGISTER DATA
001F 300 : LOOP COUNT I
001F 301 : LOOP COUNT J
```

```
001F 302 :  
001F 303 :--  
001F 304 :*****  
001F 305  
001F 306  
001F 307  
001F 308 INITIALIZE ; Init the cpu  
0021 309 LOOP I,1,64 ; Address selection loop  
0028 310 LOOP J,1,4 ; Test pattern loop  
002F 311 ERRLOOP ; Error loop address  
0031 312 LOADREG DCSAD,2$,I,B ; Load the DCS ADDRESS register  
0039 313 LOADREG DCSDA,1$,J,B ; Load the control file with  
0041 314 ; the test pattern  
0041 315 LOADREG DCSAD,2$,I,B ; Reload the ADDRESS register  
0049 316 LOADREG DCSAD,2$,I,B ; Reload the ADDRESS register  
0051 317 CMPREG DCSCF,1$,J,B ; Test CONTROL FILE register  
005A 318 IFERROR ,,<<RDM>> ; Failure in control file data  
0060 319 ENDLOOP J ; Terminate the test loop  
0063 320 ENDLOOP I ; End address loop  
0066 321 SKIP ; Go to the next test  
006D 322  
006D 323  
006D 324  
006D 325 BEGIN_TEST_DATA  
006D :-----  
006D 326  
006D 327 1$: .BYTE ^XAA,^X55,^X33,^X0F ; Test patterns that get loaded  
0071 328 2$: .BYTE ^X0,^X1,^X2,^X3,^X4,^X5,^X6,^X7  
0079 329 .BYTE ^X8,^X9,^XA,^XB,^XC,^XD,^XE,^XF  
0081 330 .BYTE ^X10,^X11,^X12,^X13,^X14,^X15,^X16,^X17  
0089 331 .BYTE ^X18,^X19,^X1A,^X1B,^X1C,^X1D,^X1E,^X1F  
0091 332 .BYTE ^X20,^X21,^X22,^X23,^X24,^X25,^X26,^X27  
0099 333 .BYTE ^X28,^X29,^X2A,^X2B,^X2C,^X2D,^X2E,^X2F  
00A1 334 .BYTE ^X30,^X31,^X32,^X33,^X34,^X35,^X36,^X37  
00A9 335 .BYTE ^X38,^X39,^X3A,^X3B,^X3C,^X3D,^X3E,^X3F  
00B1 336  
00B1 337 END_TEST_DATA  
00B1 :-----  
00B1 338 ENDTEST
```

```

001B .SBTTL 'TEST 005 DCS CONTROL FILE CLEAR
001B 340 :*****
001B 341 :++
001B 342 :
001B 343 : FUNCTIONAL DESCRIPTION:
001B 344 :
001B 345 : This test checks that the CLEAR CONTROL FILE bit in the DCS CONTROL
001B 346 : REGISTER, clears the control file.
001B 347 :
001B 348 :
001B 349 : TEST ALGORITHM:
001B 350 :
001B 351 : 1. Load address 0 into DCS ADDRESS REGISTER.
001B 352 : 2. Load the control file with AA.
001B 353 : 3. Load address 0 into DCS ADDRESS REGISTER.
001B 354 : 4. Load address 0 into DCS ADDRESS REGISTER.
001B 355 : 5. Set clear control file bit in DCS CONTROL REGISTER.
001B 356 : 6. Test control file for FF.
001B 357 :
001B 358 :
001B 359 : TEST PATTERNS:
001B 360 :
001B 361 : AA
001B 362 :
001B 363 :
001B 364 : LOGIC DESCRIPTION:
001B 365 :
001B 366 : This test checks the ability of the 8085 to load the DCS CONTROL
001B 367 : REGISTER bit 1. This inturn should cause the control file latch
001B 368 : to clear. The DCS ADDRESS REGISTER is loaded twice at one point
001B 369 : in the test to insure the control file latch is loaded with the
001B 370 : correct ram output.
001B 371 :
001B 372 :
001B 373 : ASSUMPTIONS:
001B 374 :
001B 375 : This test assumes the power up self test has run successfully
001B 376 :
001B 377 :
001B 378 : ERROR DESCRIPTION:
001B 379 :
001B 380 : EXPECTED CONTROL FILE DATA
001B 381 : RECEIVED CONTROL FILE DATA
001B 382 :
001B 383 : --
001B 384 :*****
001B 385 :
001B 386 :
001B 387 :
001B 388 : INITIALIZE
001D 389 : LOADREG DCSAD,1$,,B ; Init the cpu
0025 390 : LOADREG DCSDA,2$,,B ; Select address 0
002D 391 : LOADREG DCSAD,1$,,B ; Load the Control File
0035 392 : LOADREG DCSAD,1$,,B ; Select address 0
003D 393 : LOADREG DCSCR,3$,,B ; Select address 0
0045 394 : LOADREG DCSCR,4$,,B ; Clear the control file
004D 395 : CMPREG DCSCF,5$,,B ;
: ;: Check the control file

```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

"TEST 005 DCS CONTROL FILE CLEAR  
VAX-11/750 RDM MICRO DIAGNOSTICS  
"TEST 005 DCS CONTROL FILE CLEAR

B 3  
6-MAY-1982

Fiche 1 Frame B3  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1  
Sequence 27 Page 2 (1)

```
0056 396 IFERROR      ..<<RDM>>
005C 397 SKIP
0063 398
0063 399 BEGIN_TEST_DATA
0063
0063 :-----
00 0063 400
AA 0063 401 1$: .BYTE 0
62 0064 402 2$: .BYTE ^XAA
60 0065 403 3$: .BYTE CLK_CTRL_1!CLK_CTRL_0!CLEAR_CTRL_FILE
FF 0066 404 4$: .BYTE CLK_CTRL_1!CLK_CTRL_0
0067 405 5$: .BYTE ^XFF
0068 406
0068 407 END_TEST_DATA
0068
0068 :-----
0068 408 ENDTEST
```

ZZ  
EC  
VO

U (

```
001C .SBTTL 'TEST 006 DCS CHIP SELECT COUNTER
001C 410 :*****
001C 411 :++
001C 412 :
001C 413 : FUNCTIONAL DESCRIPTION:
001C 414 :
001C 415 : This test checks that the DCS chip select counter counts thru 11
001C 416 : states and then causes the DCS ADDRESS register to increment.
001C 417 :
001C 418 : TEST ALGORITHM:
001C 419 :
001C 420 : 1. Load the DCS ADDRESS register with 0
001C 421 : 2. Load 10 bytes into the DCS.
001C 422 : 3. Check that DCS ADDRESS register is still 0.
001C 423 : 4. Load 1 more byte into the DCS.
001C 424 : 5. Check that DCS ADDRESS register incremented to 1.
001C 425 :
001C 426 :
001C 427 : TEST PATTERNS:
001C 428 :
001C 429 : none
001C 430 :
001C 431 :
001C 432 : LOGIC DESCRIPTION:
001C 433 :
001C 434 : This test checks the DCS CHIP SELECT COUNTER, DECODER logic to
001C 435 : insure that the dcs address is incremented after 11 bytes are
001C 436 : loaded into dcs.
001C 437 :
001C 438 :
001C 439 : ASSUMPTIONS:
001C 440 :
001C 441 : This test assumes the power up self test has run successfully
001C 442 :
001C 443 :
001C 444 : ERROR DESCRIPTION:
001C 445 :
001C 446 : EXPECTED DCS ADDRESS REGISTER DATA
001C 447 : RECEIVED DCS ADDRESS REGISTER DATA
001C 448 : LOOP COUNT
001C 449 :
001C 450 : --
001C 451 :*****
001C 452 :
001C 453 :
001C 454 :
001C 455 : INITIALIZE ; Init the cpu
001E 456 LOADREG DCSAD,1$,,B ; Load the ADDRESS register
0026 457 LOOP I,1,10 ; Setup a loop to load 10 bytes
002D 458 LOADREG DCSDA,1$,,B ; Load a byte into the DCS
0035 459 ENDLOOP I ; Terminate the loop
0038 460 CMPREGMSK CLKDCS,1$,,3$,,B, ; Test DCS address reg for 0
0044 461 IFERROR ,,<<RDM>> ; ADDRESS register changed
004A 462 LOADREG DCSDA,1$,,B ; Load byte 11 into dcs
0052 463 CMPREGMSK CLKDCS,2$,,3$,,B, ; Test DCS address reg for 1
005E 464 IFERROR ,,<<RDM>> ; ADDRESS reg incorrect
0064 465 SKIP ; Go to next test
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 006 DCS CHIP SELECT COUNTER  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 006 DCS CHIP SELECT COUNTER

D 3  
6-MAY-1982

Fiche 1 Frame D3

Sequence 29

6-MAY-1982 19:54:55 VAX-11 Macro V02.46 Page 2  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1 (1

```
006B 466
006B 467
006B 468
006B 469          BEGIN_TEST_DATA
006B :-----
006B 470
00 006B 471 1$: .BYTE 0          ; Data for ADDRESS register
01 006C 472 2$: .BYTE 1          ; Expected data
3F 006D 473 3$: .BYTE ^X3F      ; Mask for DCS ADDRESS Register
006E 474
006E 475          END_TEST_DATA
006E :-----
006E 476          ENDTEST
```

```
0025 .SBTTL 'TEST 007 DCS ADDRESS REGISTER COUNT LOGIC
0025 478 :*****
0025 479 :++
0025 480 :
0025 481 : FUNCTIONAL DESCRIPTION:
0025 482 :
0025 483 :     This test checks that the DCS ADDRESS register counts from 0
0025 484 :     to 63 and then back to 0.
0025 485 :
0025 486 : TEST ALGORITHM:
0025 487 :
0025 488 :     1. Load the DCS ADDRESS register with 0
0025 489 :     2. Load the DCS with 11 bytes
0025 490 :     3. Check that the DCS ADDRESS register incremented
0025 491 :     4. Do steps 2 and 3 64 times.
0025 492 :
0025 493 :
0025 494 : TEST PATTERNS:
0025 495 :
0025 496 :     NONE
0025 497 :
0025 498 :
0025 499 : LOGIC DESCRIPTION:
0025 500 :
0025 501 :     This test verifies that the DCS addressing logic operates correctly.
0025 502 :     This includes the DCS counter, decoder, and address register.
0025 503 :
0025 504 :
0025 505 : ASSUMPTIONS:
0025 506 :
0025 507 :     This test assumes the power-up self test has run successfully.
0025 508 :
0025 509 :
0025 510 : ERROR DESCRIPTION:
0025 511 :
0025 512 :     EXPECTED DCS ADDRESS REGISTER DATA
0025 513 :     RECEIVED DCS ADDRESS REGISTER DATA
0025 514 :     LOOP COUNT (I = 1 TO 64)
0025 515 :
0025 516 : --
0025 517 : *****
0025 518 :
0025 519 :
0025 520 :
0025 521 :     INITIALIZE                                : Init cpu
0027 522 :     LOADREG DCSAD,4$,.,B                       : 0 to DCS ADDRESS REG
002F 523 :     LOOP I,1,64                                : Setup ADDRESS increment loop
0036 524 :     LOOP J,1,11                                : Load 11 bytes into the DCS
003D 525 :     LOADREG DCSDA,1$,J,B                       : Load the DCS with NOP
0045 526 :     ENDLOOP J                                  : Terminate loop to load dcs
0048 527 :     CMPREGMSK CLKDCS,2$,I,3$,.,B.             : Check that ADDRESS register
0054 528 :                                                    : has the correct count in it
0054 529 :     IFERROR I,<<RDM>>                          : ADDRESS register incorrect
005A 530 :     ENDLOOP I                                  : Terminate the microword loop
005D 531 :     SKIP                                        : Go to the next test
0064 532 :
0064 533 : BEGIN_TEST_DATA
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 007 DCS ADDRESS REGISTER COUNT LOG

VAX-11/750 RDM MICRO DIAGNOSTICS

'TEST 007 DCS ADDRESS REGISTER COUNT LOG

F 3  
6-MAY-1982

Fiche 1 Frame F3

Sequence 31

6-MAY-1982 19:54:55

VAX-11 Macro V02.46

Page 2

6-MAY-1982 19:54:39

WRKDS0:[COMET.RDM]ECKAF.MAR;1

(1

```

0064
0064
U 00, 7700,c800,0364,0300,0470,1801 0064 534 i$:
0064 535 :% 00 NOP ; u-word for DCS
006F 539
006F 540 :+
006F 541 : Following is the expected contents of the ADDRESS register each iteration
006F 542 : of the 'I' loop.
006F 543 :-
0C 0B 0A 09 08 07 06 05 04 03 02 01 006F 544 2$: .BYTE 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
14 13 12 11 10 0F 0E 0D 007B
20 1F 1E 1D 1C 1B 1A 19 18 17 16 15 0083 545 .BYTE 21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40
28 27 26 25 24 23 22 21 008F
34 33 32 31 30 2F 2E 2D 2C 2B 2A 29 0097 546 .BYTE 41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60
3C 3B 3A 39 38 37 36 35 00A3
00 3F 3E 3D 00AB 547 .BYTE 61,62,63,0
00AF 548 3$: .BYTE ^X3F ; Mask for DCS ADDRESS register
00B0 549 4$: .BYTE 0 ; Starting address in DCS
00B1 550
00B1 551 END_TEST_DATA
00B1
00B1
00B1 552 :-----
ENDTEST
```

```
0018 .SBTTL 'TEST 008 DCS DUAL ADDRESSING
0018 554 :*****
0018 555 :++
0018 556 :
0018 557 : FUNCTIONAL DESCRIPTION:
0018 558 :
0018 559 : This test performs a limited dual addressing test of the DCS by
0018 560 : loading a unique data pattern into the control file of each DCS
0018 561 : address then reading them back and checking them.
0018 562 :
0018 563 : TEST ALGORITHM:
0018 564 :
0018 565 : 1. Setup a loop to load all control files
0018 566 : 2. Load the DCS ADDRESS register with an address
0018 567 : 3. Load the control file with its address
0018 568 : 4. Loop to step 2 for 64 iterations
0018 569 : 5. Setup a loop to read all control files
0018 570 : 6. Load the DCS ADDRESS register with an address
0018 571 : 7. Repeat step 6
0018 572 : 8. Test the contents of the control file for its address
0018 573 : 9. If no error go to step 6 for 64 iterations
0018 574 :
0018 575 :
0018 576 : TEST PATTERNS:
0018 577 :
0018 578 : DCS ADDRESS CONTROL FILE
0018 579 : 0 0
0018 580 : 1 1
0018 581 : . .
0018 582 : . .
0018 583 : . .
0018 584 : 3D 3D
0018 585 :
0018 586 :
0018 587 : LOGIC DESCRIPTION:
0018 588 :
0018 589 : This test verifies that the address lines from the DCS ADDRESS register
0018 590 : to the CONTROL FILE ram are in good shape. The address of each
0018 591 : location is written into itself and read back for accuracy. This
0018 592 : test does not check any other rams in the DCS since they cannot be
0018 593 : read back to the 8085. You'll note that the DCS ADDRESS is loaded
0018 594 : twice at one point in the test. This is to enable the control file
0018 595 : latch to load the correct ram output.
0018 596 :
0018 597 :
0018 598 : ASSUMPTIONS:
0018 599 :
0018 600 : This test assumes the power-up self test has run successfully
0018 601 :
0018 602 :
0018 603 : ERROR DESCRIPTION:
0018 604 :
0018 605 : EXPECTED CONTROL FILE DATA
0018 606 : RECEIVED CONTROL FILE DATA
0018 607 : LOOP COUNT
0018 608 :
0018 609 :--
```

```
0018 610 :*****
0018 611
0018 612
0018 613
0018 614 INITIALIZE ; Init the cpu
001A 615 :+
001A 616 : Load the control file of each micro word with a unique pattern
001A 617 :-
001A 618 ERRLOOP ; Error loop point
001C 619 LOOP I,1,64 ; Loop to load the control file
0023 620 LOADREG DCSAD,1$,I,B ; Load the ADDRESS register
002B 621 LOADREG DCSDA,1$,I,B ; Load a data pattern into the
0033 622 ; control file
0033 623 ENDLOOP I ; End loop to load control file
0036 624 :+
0036 625 : Read the control file of each micro word and check for the correct pattern
0036 626 :-
0036 627 LOOP I,1,64 ; Loop to read the control file
003D 628 LOADREG DCSAD,1$,I,B ; Load the ADDRESS register
0045 629 LOADREG DCSAD,1$,I,B ; Load the ADDRESS register
004D 630 CMPREG DCSCF,1$,I,B ; Check the control file
0056 631 IFERROR ;, <<RDM>> ; Incorrect data
005C 632 ENDLOOP I ; END loop to read control file
005F 633 SKIP ; Go to the next test
0066 634
0066 635 BEGIN_TEST_DATA
0066 636
0066 637 -----
0066 638 : Data loaded into the control file
0066 639 :
0066 640 i$: .BYTE 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
0072 641 .BYTE 21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40
007B 642 .BYTE 41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60
0087 643 .BYTE 61,62,63
008F 644
0098 645 END_TEST_DATA
00A3 646 -----
00A6 647
00A6 648
00A6 649
00A6 650
00A6 651
00A6 652
00A6 653
00A6 654
00A6 655
00A6 656
00A6 657
00A6 658
00A6 659
00A6 660
00A6 661
00A6 662
00A6 663
00A6 664
00A6 665
00A6 666
00A6 667
00A6 668
00A6 669
00A6 670
00A6 671
00A6 672
00A6 673
00A6 674
00A6 675
00A6 676
00A6 677
00A6 678
00A6 679
00A6 680
00A6 681
00A6 682
00A6 683
00A6 684
00A6 685
00A6 686
00A6 687
00A6 688
00A6 689
00A6 690
00A6 691
00A6 692
00A6 693
00A6 694
00A6 695
00A6 696
00A6 697
00A6 698
00A6 699
00A6 700
00A6 701
00A6 702
00A6 703
00A6 704
00A6 705
00A6 706
00A6 707
00A6 708
00A6 709
00A6 710
00A6 711
00A6 712
00A6 713
00A6 714
00A6 715
00A6 716
00A6 717
00A6 718
00A6 719
00A6 720
00A6 721
00A6 722
00A6 723
00A6 724
00A6 725
00A6 726
00A6 727
00A6 728
00A6 729
00A6 730
00A6 731
00A6 732
00A6 733
00A6 734
00A6 735
00A6 736
00A6 737
00A6 738
00A6 739
00A6 740
00A6 741
00A6 742
00A6 743
00A6 744
00A6 745
00A6 746
00A6 747
00A6 748
00A6 749
00A6 750
00A6 751
00A6 752
00A6 753
00A6 754
00A6 755
00A6 756
00A6 757
00A6 758
00A6 759
00A6 760
00A6 761
00A6 762
00A6 763
00A6 764
00A6 765
00A6 766
00A6 767
00A6 768
00A6 769
00A6 770
00A6 771
00A6 772
00A6 773
00A6 774
00A6 775
00A6 776
00A6 777
00A6 778
00A6 779
00A6 780
00A6 781
00A6 782
00A6 783
00A6 784
00A6 785
00A6 786
00A6 787
00A6 788
00A6 789
00A6 790
00A6 791
00A6 792
00A6 793
00A6 794
00A6 795
00A6 796
00A6 797
00A6 798
00A6 799
00A6 800
00A6 801
00A6 802
00A6 803
00A6 804
00A6 805
00A6 806
00A6 807
00A6 808
00A6 809
00A6 810
00A6 811
00A6 812
00A6 813
00A6 814
00A6 815
00A6 816
00A6 817
00A6 818
00A6 819
00A6 820
00A6 821
00A6 822
00A6 823
00A6 824
00A6 825
00A6 826
00A6 827
00A6 828
00A6 829
00A6 830
00A6 831
00A6 832
00A6 833
00A6 834
00A6 835
00A6 836
00A6 837
00A6 838
00A6 839
00A6 840
00A6 841
00A6 842
00A6 843
00A6 844
00A6 845
00A6 846
00A6 847
00A6 848
00A6 849
00A6 850
00A6 851
00A6 852
00A6 853
00A6 854
00A6 855
00A6 856
00A6 857
00A6 858
00A6 859
00A6 860
00A6 861
00A6 862
00A6 863
00A6 864
00A6 865
00A6 866
00A6 867
00A6 868
00A6 869
00A6 870
00A6 871
00A6 872
00A6 873
00A6 874
00A6 875
00A6 876
00A6 877
00A6 878
00A6 879
00A6 880
00A6 881
00A6 882
00A6 883
00A6 884
00A6 885
00A6 886
00A6 887
00A6 888
00A6 889
00A6 890
00A6 891
00A6 892
00A6 893
00A6 894
00A6 895
00A6 896
00A6 897
00A6 898
00A6 899
00A6 900
00A6 901
00A6 902
00A6 903
00A6 904
00A6 905
00A6 906
00A6 907
00A6 908
00A6 909
00A6 910
00A6 911
00A6 912
00A6 913
00A6 914
00A6 915
00A6 916
00A6 917
00A6 918
00A6 919
00A6 920
00A6 921
00A6 922
00A6 923
00A6 924
00A6 925
00A6 926
00A6 927
00A6 928
00A6 929
00A6 930
00A6 931
00A6 932
00A6 933
00A6 934
00A6 935
00A6 936
00A6 937
00A6 938
00A6 939
00A6 940
00A6 941
00A6 942
00A6 943
00A6 944
00A6 945
00A6 946
00A6 947
00A6 948
00A6 949
00A6 950
00A6 951
00A6 952
00A6 953
00A6 954
00A6 955
00A6 956
00A6 957
00A6 958
00A6 959
00A6 960
00A6 961
00A6 962
00A6 963
00A6 964
00A6 965
00A6 966
00A6 967
00A6 968
00A6 969
00A6 970
00A6 971
00A6 972
00A6 973
00A6 974
00A6 975
00A6 976
00A6 977
00A6 978
00A6 979
00A6 980
00A6 981
00A6 982
00A6 983
00A6 984
00A6 985
00A6 986
00A6 987
00A6 988
00A6 989
00A6 990
00A6 991
00A6 992
00A6 993
00A6 994
00A6 995
00A6 996
00A6 997
00A6 998
00A6 999
00A6 1000
00A6 1001
00A6 1002
00A6 1003
00A6 1004
00A6 1005
00A6 1006
00A6 1007
00A6 1008
00A6 1009
00A6 1010
00A6 1011
00A6 1012
00A6 1013
00A6 1014
00A6 1015
00A6 1016
00A6 1017
00A6 1018
00A6 1019
00A6 1020
00A6 1021
00A6 1022
00A6 1023
00A6 1024
00A6 1025
00A6 1026
00A6 1027
00A6 1028
00A6 1029
00A6 1030
00A6 1031
00A6 1032
00A6 1033
00A6 1034
00A6 1035
00A6 1036
00A6 1037
00A6 1038
00A6 1039
00A6 1040
00A6 1041
00A6 1042
00A6 1043
00A6 1044
00A6 1045
00A6 1046
00A6 1047
00A6 1048
00A6 1049
00A6 1050
00A6 1051
00A6 1052
00A6 1053
00A6 1054
00A6 1055
00A6 1056
00A6 1057
00A6 1058
00A6 1059
00A6 1060
00A6 1061
00A6 1062
00A6 1063
00A6 1064
00A6 1065
00A6 1066
00A6 1067
00A6 1068
00A6 1069
00A6 1070
00A6 1071
00A6 1072
00A6 1073
00A6 1074
00A6 1075
00A6 1076
00A6 1077
00A6 1078
00A6 1079
00A6 1080
00A6 1081
00A6 1082
00A6 1083
00A6 1084
00A6 1085
00A6 1086
00A6 1087
00A6 1088
00A6 1089
00A6 1090
00A6 1091
00A6 1092
00A6 1093
00A6 1094
00A6 1095
00A6 1096
00A6 1097
00A6 1098
00A6 1099
00A6 1100
00A6 1101
00A6 1102
00A6 1103
00A6 1104
00A6 1105
00A6 1106
00A6 1107
00A6 1108
00A6 1109
00A6 1110
00A6 1111
00A6 1112
00A6 1113
00A6 1114
00A6 1115
00A6 1116
00A6 1117
00A6 1118
00A6 1119
00A6 1120
00A6 1121
00A6 1122
00A6 1123
00A6 1124
00A6 1125
00A6 1126
00A6 1127
00A6 1128
00A6 1129
00A6 1130
00A6 1131
00A6 1132
00A6 1133
00A6 1134
00A6 1135
00A6 1136
00A6 1137
00A6 1138
00A6 1139
00A6 1140
00A6 1141
00A6 1142
00A6 1143
00A6 1144
00A6 1145
00A6 1146
00A6 1147
00A6 1148
00A6 1149
00A6 1150
00A6 1151
00A6 1152
00A6 1153
00A6 1154
00A6 1155
00A6 1156
00A6 1157
00A6 1158
00A6 1159
00A6 1160
00A6 1161
00A6 1162
00A6 1163
00A6 1164
00A6 1165
00A6 1166
00A6 1167
00A6 1168
00A6 1169
00A6 1170
00A6 1171
00A6 1172
00A6 1173
00A6 1174
00A6 1175
00A6 1176
00A6 1177
00A6 1178
00A6 1179
00A6 1180
00A6 1181
00A6 1182
00A6 1183
00A6 1184
00A6 1185
00A6 1186
00A6 1187
00A6 1188
00A6 1189
00A6 1190
00A6 1191
00A6 1192
00A6 1193
00A6 1194
00A6 1195
00A6 1196
00A6 1197
00A6 1198
00A6 1199
00A6 1200
00A6 1201
00A6 1202
00A6 1203
00A6 1204
00A6 1205
00A6 1206
00A6 1207
00A6 1208
00A6 1209
00A6 1210
00A6 1211
00A6 1212
00A6 1213
00A6 1214
00A6 1215
00A6 1216
00A6 1217
00A6 1218
00A6 1219
00A6 1220
00A6 1221
00A6 1222
00A6 1223
00A6 1224
00A6 1225
00A6 1226
00A6 1227
00A6 1228
00A6 1229
00A6 1230
00A6 1231
00A6 1232
00A6 1233
00A6 1234
00A6 1235
00A6 1236
00A6 1237
00A6 1238
00A6 1239
00A6 1240
00A6 1241
00A6 1242
00A6 1243
00A6 1244
00A6 1245
00A6 1246
00A6 1247
00A6 1248
00A6 1249
00A6 1250
00A6 1251
00A6 1252
00A6 1253
00A6 1254
00A6 1255
00A6 1256
00A6 1257
00A6 1258
00A6 1259
00A6 1260
00A6 1261
00A6 1262
00A6 1263
00A6 1264
00A6 1265
00A6 1266
00A6 1267
00A6 1268
00A6 1269
00A6 1270
00A6 1271
00A6 1272
00A6 1273
00A6 1274
00A6 1275
00A6 1276
00A6 1277
00A6 1278
00A6 1279
00A6 1280
00A6 1281
00A6 1282
00A6 1283
00A6 1284
00A6 1285
00A6 1286
00A6 1287
00A6 1288
00A6 1289
00A6 1290
00A6 1291
00A6 1292
00A6 1293
00A6 1294
00A6 1295
00A6 1296
00A6 1297
00A6 1298
00A6 1299
00A6 1300
00A6 1301
00A6 1302
00A6 1303
00A6 1304
00A6 1305
00A6 1306
00A6 1307
00A6 1308
00A6 1309
00A6 1310
00A6 1311
00A6 1312
00A6 1313
00A6 1314
00A6 1315
00A6 1316
00A6 1317
00A6 1318
00A6 1319
00A6 1320
00A6 1321
00A6 1322
00A6 1323
00A6 1324
00A6 1325
00A6 1326
00A6 1327
00A6 1328
00A6 1329
00A6 1330
00A6 1331
00A6 1332
00A6 1333
00A6 1334
00A6 1335
00A6 1336
00A6 1337
00A6 1338
00A6 1339
00A6 1340
00A6 1341
00A6 1342
00A6 1343
00A6 1344
00A6 1345
00A6 1346
00A6 1347
00A6 1348
00A6 1349
00A6 1350
00A6 1351
00A6 1352
00A6 1353
00A6 1354
00A6 1355
00A6 1356
00A6 1357
00A6 1358
00A6 1359
00A6 1360
00A6 1361
00A6 1362
00A6 1363
00A6 1364
00A6 1365
00A6 1366
00A6 1367
00A6 1368
00A6 1369
00A6 1370
00A6 1371
00A6 1372
00A6 1373
00A6 1374
00A6 1375
00A6 1376
00A6 1377
00A6 1378
00A6 1379
00A6 1380
00A6 1381
00A6 1382
00A6 1383
00A6 1384
00A6 1385
00A6 1386
00A6 1387
00A6 1388
00A6 1389
00A6 1390
00A6 1391
00A6 1392
00A6 1393
00A6 1394
00A6 1395
00A6 1396
00A6 1397
00A6 1398
00A6 1399
00A6 1400
00A6 1401
00A6 1402
00A6 1403
00A6 1404
00A6 1405
00A6 1406
00A6 1407
00A6 1408
00A6 1409
00A6 1410
00A6 1411
00A6 1412
00A6 1413
00A6 1414
00A6 1415
00A6 1416
00A6 1417
00A6 1418
00A6 1419
00A6 1420
00A6 1421
00A6 1422
00A6 1423
00A6 1424
00A6 1425
00A6 1426
00A6 1427
00A6 1428
00A6 1429
00A6 1430
00A6 1431
00A6 1432
00A6 1433
00A6 1434
00A6 1435
00A6 1436
00A6 1437
00A6 1438
00A6 1439
00A6 1440
00A6 1441
00A6 1442
00A6 1443
00A6 1444
00A6 1445
00A6 1446
00A6 1447
00A6 1448
00A6 1449
00A6 1450
00A6 1451
00A6 1452
00A6 1453
00A6 1454
00A6 1455
00A6 1456
00A6 1457
00A6 1458
00A6 1459
00A6 1460
00A6 1461
00A6 1462
00A6 1463
00A6 1464
00A6 1465
00A6 1466
00A6 1467
00A6 1468
00A6 1469
00A6 1470
00A6 1471
00A6 1472
00A6 1473
00A6 1474
00A6 1475
00A6 1476
00A6 1477
00A6 1478
00A6 1479
00A6 1480
00A6 1481
00A6 1482
00A6 1483
00A6 1484
00A6 1485
00A6 1486
00A6 1487
00A6 1488
00A6 1489
00A6 1490
00A6 1491
00A6 1492
00A6 1493
00A6 1494
00A6 1495
00A6 1496
00A6 1497
00A6 1498
00A6 1499
00A6 1500
00A6 1501
00A6 1502
00A6 1503
00A6 1504
00A6 1505
00A6 1506
00A6 1507
00A6 1508
00A6 1509
00A6 1510
00A6 1511
00A6 1512
00A6 1513
00A6 1514
00A6 1515
00A6 1516
00A6 1517
00A6 1518
00A6 1519
00A6 1520
00A6 1521
00A6 1522
00A6 1523
00A6 1524
00A6 1525
00A6 1526
00A6 1527
00A6 1528
00A6 1529
00A6 1530
00A6 1531
00A6 1532
00A6 1533
00A6 1534
00A6 1535
00A6 1536
00A6 1537
00A6 1538
00A6 1539
00A6 1540
00A6 1541
00A6 1542
00A6 1543
00A6 1544
00A6 1545
00A6 1546
00A6 1547
00A6 1548
00A6 1549
00A6 1550
00A6 1551
00A6 1552
00A6 1553
00A6 1554
00A6 1555
00A6 1556
00A6 1557
00A6 1558
00A6 1559
00A6 1560
00A6 1561
00A6 1562
00A6 1563
00A6 1564
00A6 1565
00A6 1566
00A6 1567
00A6 1568
00A6 1569
00A6 1570
00A6 1571
00A6 1572
00A6 1573
00A6 1574
00A6 1575
00A6 1576
00A6 1577
00A6 1578
00A6 1579
00A6 1580
00A6 1581
00A6 1582
00A6 1583
00A6 1584
00A6 1585
00A6 1586
00A6 1587
00A6 1588
00A6 1589
00A6 1590
00A6 1591
00A6 1592
00A6 1593
00A6 1594
00A6 1595
00A6 1596
00A6 1597
00A6 1598
00A6 1599
00A6 1600
00A6 1601
00A6 1602
00A6 1603
00A6 1604
00A6 1605
00A6 1606
00A6 1607
00A6 1608
00A6 1609
00A6 1610
00A6 1611
00A6 1612
00A6 1613
00A6 1614
00A6 1615
00A6 1616
00A6 1617
00A6 1618
00A6 1619
00A6 1620
00A6 1621
00A6 1622
00A6 1623
00A6 1624
00A6 1625
00A6 1626
00A6 1627
00A6 1628
00A6 1629
00A6 1630
00A6 1631
00A6 1632
00A6 1633
00A6 1634
00A6 1635
00A6 1636
00A6 1637
00A6 1638
00A6 1639
00A6 1640
00A6 1641
00A6 1642
00A6 1643
00A6 1644
00A6 1645
00A6 1646
00A6 1647
00A6 1648
00A6 1649
00A6 1650
00A6 1651
00A6 1652
00A6 1653
00A6 1654
00A6 1655
00A6 1656
00A6 1657
00A6 1658
00A6 1659
00A6 1660
00A6 1661
00A6 1662
00A6 1663
00A6 1664
00A6 1665
00A6 1666
00A6 1667
00A6 1668
00A6 1669
00A6 1670
00A6 1671
00A6 1672
00A6 1673
00A6 1674
00A6 1675
00A6 1676
00A6 1677
00A6 1678
00A6 1679
00A6 1680
00A6 1681
00A6 1682
00A6 1683
00A6 1684
00A6 1685
00A6 1686
00A6 1687
00A6 1688
00A6 1689
00A6 1690
00A6 1691
00A6 1692
00A6 1693
00A6 1694
00A6 1695
00A6 1696
00A6 1697
00A6 1698
00A6 1699
00A6 1700
00A6 1701
00A6 1702
00A6 1703
00A6 1704
00A6 1705
00A6 1706
00A6 1707
00A6 1708
00A6 1709
00A6 1710
00A6 1711
00A6 1712
00A6 1713
00A6 1714
00A6 1715
00A6 1716
00A6 1717
00A6 1718
00A6 1719
00A6 1720
00A6 1721
00A6 1722
00A6 1723
00A6 1724
00A6 1725
00A6 1726
00A6 1727
00A6 1728
00A6 1729
00A6 1730
00A6 1731
00A6 1732
00A6 1733
00A6 1734
00A6 1735
00A6 1736
00A6 1737
00A6 1738
00A6 1739
00A6 1740
00A6 1741
00A6 1742
00A6 1743
00A6 1744
00A6 1745
00A6 1746
00A6 1747
00A6 1748
00A6 1749
00A6 1750
00A6 1751
00A6 1752
00A6 1753
00A6 1754
00A6 1755
00A6 1756
00A6 1757
00A6 1758
00A6 1759
00A6 1760
00A6 1761
00A6 1762
00A6 1763
00A6 1764
00A6 1765
00A6 1766
00A6 1767
00A6 1768
00A6 1769
00A6 1770
00A6 1771
00A6 1772
00A6 1773
00A6 1774
00A6 1775
00A6 1776
00A6 1777
00A6 1778
00A6 1779
00A6 1780
00A6 1781
00A6 1782
00A6 1783
00A6 1784
00A6 1785
00A6 1786
00A6 1787
00A6 1788
00A6 1789
00A6 1790
00A6 1791
00A6 1792
00A6 1793
00A6 1794
00A6 1795
00A6 1796
00A6 1797
00A6 1798
00A6 1799
00A6 1800
00A6 1801
00A6 1802
00A6 1803
00A
```

```
0017 .SBTTL 'TEST 009 DCS DATA INTEGRITY
0017 648 :*****
0017 649 :++
0017 650 :
0017 651 : FUNCTIONAL DESCRIPTION:
0017 652 :
0017 653 :     This test checks the data integrity of the DCS.
0017 654 :
0017 655 :
0017 656 : TEST ALGORITHM:
0017 657 :
0017 658 :     1. Clear MASTER HALT ENABLE signal
0017 659 :     2. Create a test loop of 64 iterations
0017 660 :     3. Pulse DC LOW to clear any parity errors
0017 661 :     4. Load a test pattern into the selected DCS address
0017 662 :     5. Enable UADDR INHIBIT & PARITY CHECK signals
0017 663 :     6. Get control of CONTROL STORE LINES by loading 1800 into CSAMP REG
0017 664 :     7. Step the COMET clock one micro-instruction
0017 665 :     8. Disable UADDR INHIBIT & PARITY CHECK signals
0017 666 :     9. Test the parity error light on the front panel
0017 667 :    10. If no error go to step 2 for remaining addresses
0017 668 :
0017 669 :
0017 670 : TEST PATTERNS:
0017 671 :
0017 672 :     00000000000000000000
0017 673 :     FFFFFFFFFFFFFFFFFFFF
0017 674 :     AAAAAAAAAAAAAAAAAAAA
0017 675 :     SSSSSSSSSSSSSSSSSSS
0017 676 :     33333333333333333333
0017 677 :     OFOFOFOFOFOFOFOF
0017 678 :     OOFFOOFFOOFFOOFF
0017 679 :     0000FFFF0000FFFF0000
0017 680 :     00000000FFFFFFFF0000
0017 681 :     0000000000000000FFFF
0017 682 :
0017 683 :
0017 684 : LOGIC DESCRIPTION:
0017 685 :
0017 686 :     This test checks the data integrity of the DCS ram lines by
0017 687 :     loading test patterns into each DCS address. Since the rams
0017 688 :     cannot be read back to the 8085 the next best thing to do is
0017 689 :     allow COMET to look at the data for a parity error. This will
0017 690 :     atleast catch any errors that involve an odd number of bits.
0017 691 :
0017 692 :
0017 693 : ASSUMPTIONS:
0017 694 :
0017 695 :     This test assumes that the COMET parity checking logic is
0017 696 :     operational.
0017 697 :
0017 698 :
0017 699 : ERROR DESCRIPTION:
0017 700 :
0017 701 :     EXPECTED CONTENTS OF FRONT1 REGISTER
0017 702 :     RECEIVED CONTENTS OF FRONT1 REGISTER
0017 703 :     LOOP COUNT I
```

```
0017 704 :  
0017 705 :--  
0017 706 :*****  
0017 707 ://///////////////  
0017 708 :+  
0017 709 :SUBTEST #1  
0017 710 :  
0017 711 :      Test all DCS locations with test pattern #1  
0017 712 :--  
0017 713 :      SUBTEST                               : SUBTEST #1  
001A :  
001A ://///////////////  
001A 714 :      INITIALIZE                           : Init the cpu  
001C 715 :      LOADREG          RDCTRL,11$,.B       : Clear Master Halt Enable  
0024 716 :      LOOP              I,1,64             : DCS address loop  
002B 717 :      ERRLOOP          : Error loop point  
002D 718 :      LOADREG          RDCTRL,1$,.B        : Pulse DC LOW  
0035 719 :      LOADREG          RDCTRL,11$,.B       :  
003D 720 :      LOADREG          DCSAD,7$,.I,B       : Address to DCS ADD register  
0045 721 :      LOOP              J,1,11            : Loop to load test pattern  
004C 722 :      LOADREG          DCSDA,20$,.J,B      : Load test pattern  
0054 723 :      ENDLOOP          J                   :  
0057 724 :      LOADREG          DCSAD,7$,.I,B       : Reload DCS ADD register  
005F 725 :      LOADREG          DCSCR,4$,.B         : uADDR INH and Parity Check  
0067 726 :      LOADREG          CSAMR,3$,.W         : 1800 to Match Register  
006F 727 :      LOADREG          DCSCR,5$,.B         : Tick the clock  
0077 728 :      LOADREG          DCSCR,4$,.B         : one micro instruction  
007F 729 :      LOADREG          DCSCR,6$,.B         : uADDR INH and Par. check off  
0087 730 :      CMPREGMSK        FRONT1,30$,.30$,.B. : Check parity error light  
0093 731 :      IFERROR          I,<<RDM>>          : Bit stuck in DCS  
0099 732 :      ENDLOOP          I                   : Terminate the program loop  
009C 733 :  
009C 734 :  
009C 735 :  
009C 736 ://///////////////  
009C 737 :+  
009C 738 :SUBTEST #2  
009C 739 :  
009C 740 :      Test all DCS locations with test pattern #2  
009C 741 :--  
009C 742 :      SUBTEST                               : SUBTEST #2  
009F :  
009F ://///////////////  
009F 743 :      INITIALIZE                           : Init the cpu  
00A1 744 :      LOADREG          RDCTRL,11$,.B       : Clear Master Halt Enable  
00A9 745 :      LOOP              I,1,64             : DCS address loop  
00B0 746 :      ERRLOOP          : Error loop point  
00B2 747 :      LOADREG          RDCTRL,1$,.B        : Pulse DC LOW  
00BA 748 :      LOADREG          RDCTRL,11$,.B       :  
00C2 749 :      LOADREG          DCSAD,7$,.I,B       : Address to DCS ADD register  
00CA 750 :      LOOP              J,1,11            : Loop to load test pattern  
00D1 751 :      LOADREG          DCSDA,21$,.J,B      : Load test pattern  
00D9 752 :      ENDLOOP          J                   :  
00DC 753 :      LOADREG          DCSAD,7$,.I,B       : Reload DCS ADD register  
00E4 754 :      LOADREG          DCSCR,4$,.B         : uADDR INH and Parity Check  
00EC 755 :      LOADREG          CSAMR,3$,.W         : 1800 to Match Register  
00F4 756 :      LOADREG          DCSCR,5$,.B         : Tick the clock
```

```
00FC 757 LOADREG DCSCR,4$,B ; one micro instruction
0104 758 LOADREG DCSCR,6$,B ; uADDR INH and Par. check off
010C 759 CMPREGMSK FRONT1,31$,30$,B. ; Check parity error light
0118 760 IFERROR ; Bit stuck in DCS
011E 761 ENDLOOP ; Terminate the program loop
0121 762
0121 763
0121 764
0121 765 :////////////////////
0121 766 :+
0121 767 :SUBTEST #3
0121 768 :
0121 769 : Test all DCS locations with test pattern #3
0121 770 :-
0121 771 :SUBTEST ; SUBTEST #3
0124
0124 :////////////////////
0124 772 INITIALIZE ; Init the cpu
0126 773 LOADREG RDCTRL,11$,B ; Clear Master Halt Enable
012E 774 LOOP I,1,64 ; DCS address loop
0135 775 ERRLOOP ; Error loop point
0137 776 LOADREG RDCTRL,1$,B ; Pulse DC LOW
013F 777 LOADREG RDCTRL,11$,B ;
0147 778 LOADREG DCSAD,7$,I,B ; Address to DCS ADD register
014F 779 LOOP J,1,11 ; Loop to load test pattern
0156 780 LOADREG DCSDA,22$,J,B ; Load test pattern
015E 781 ENDLOOP ;
0161 782 LOADREG DCSAD,7$,I,B ; Reload DCS ADD register
0169 783 LOADREG DCSCR,4$,B ; uADDR INH and Parity Check
0171 784 LOADREG CSAMP,3$,W ; 1800 to Match Register
0179 785 LOADREG DCSCR,5$,B ; Tick the clock
0181 786 LOADREG DCSCR,4$,B ; one micro instruction
0189 787 LOADREG DCSCR,6$,B ; uADDR INH and Par. check off
0191 788 CMPREGMSK FRONT1,32$,30$,B. ; Check parity error light
019D 789 IFERROR ; Bit stuck in DCS
01A3 790 ENDLOOP ; Terminate the program loop
01A6 791
01A6 792
01A6 793
01A6 794 :////////////////////
01A6 795 :+
01A6 796 :SUBTEST #4
01A6 797 :
01A6 798 : Test all DCS locations with test pattern #4
01A6 799 :-
01A6 800 :SUBTEST ; SUBTEST #4
01A9
01A9 :////////////////////
01A9 801 INITIALIZE ; Init the cpu
01AB 802 LOADREG RDCTRL,11$,B ; Clear Master Halt Enable
01B3 803 LOOP I,1,64 ; DCS address loop
01BA 804 ERRLOOP ; Error loop point
01BC 805 LOADREG RDCTRL,1$,B ; Pulse DC LOW
01C4 806 LOADREG RDCTRL,11$,B ;
01CC 807 LOADREG DCSAD,7$,I,B ; Address to DCS ADD register
01D4 808 LOOP J,1,11 ; Loop to load test pattern
01DB 809 LOADREG DCSDA,23$,J,B ; Load test pattern
```

```
01E3 810      ENDLOOP      J      ;
01E6 811      LOADREG     DCSDA,7$,I,B  ; Reload DCS ADD register
01EE 812      LOADREG     DCSCR,4$,B  ; uADDR INH and Parity Check
01F6 813      LOADREG     CSAMR,3$,W  ; 1800 to Match Register
01FE 814      LOADREG     DCSCR,5$,B  ; Tick the clock
0206 815      LOADREG     DCSCR,4$,B  ; one micro instruction
020E 816      LOADREG     DCSCR,6$,B  ; uADDR INH and Par. check off
0216 817      CMPREGMSK   FRONT1,33$,,30$,B, ; Check parity error light
0222 818      IFERROR     ,,<<RDM>> ; Bit stuck in DCS
0228 819      ENDLOOP     I      ; Terminate the program loop
0228 820
0228 821
0228 822
0228 823 :////////////////////
0228 824 :+
0228 825 :SUBTEST #5
0228 826 :
0228 827 :      Test all DCS locations with test pattern #5
0228 828 :-
0228 829 :      SUBTEST      : SUBTEST #5
022E
022E :////////////////////
022E 830      INITIALIZE ; Init the cpu
0230 831      LOADREG     RDCTRL,11$,B  ; Clear Master Halt Enable
0238 832      LOOP      I,1,64 ; DCS address loop
023F 833      ERRLOOP ; Error loop point
0241 834      LOADREG     RDCTRL,1$,B  ; Pulse DC LOW
0249 835      LOADREG     RDCTRL,11$,B ;
0251 836      LOADREG     DCSDA,7$,I,B  ; Address to DCS ADD register
0259 837      LOOP      J,1,11 ; Loop to load test pattern
0260 838      LOADREG     DCSDA,24$,J,B ; Load test pattern
0268 839      ENDLOOP     J      ;
0268 840      LOADREG     DCSDA,7$,I,B  ; Reload DCS ADD register
0273 841      LOADREG     DCSCR,4$,B  ; uADDR INH and Parity Check
0278 842      LOADREG     CSAMR,3$,W  ; 1800 to Match Register
0283 843      LOADREG     DCSCR,5$,B  ; Tick the clock
0288 844      LOADREG     DCSCR,4$,B  ; one micro instruction
0293 845      LOADREG     DCSCR,6$,B  ; uADDR INH and Par. check off
0298 846      CMPREGMSK   FRONT1,34$,,30$,B, ; Check parity error light
02A7 847      IFERROR     ,,<<RDM>> ; Bit stuck in DCS
02AD 848      ENDLOOP     I      ; Terminate the program loop
02B0 849
02B0 850
02B0 851
02B0 852 :////////////////////
02B0 853 :+
02B0 854 :SUBTEST #6
02B0 855 :
02B0 856 :      Test all DCS locations with test patterns #6
02B0 857 :-
02B0 858 :      SUBTEST      : SUBTEST #6
02B3
02B3 :////////////////////
02B3 859      INITIALIZE ; Init the cpu
02B5 860      LOADREG     RDCTRL,11$,B  ; Clear Master Halt Enable
02BD 861      LOOP      I,1,64 ; DCS address loop
02C4 862      ERRLOOP ; Error loop point
```

```
02C6 863 LOADREG RDCTRL,1$,,B ; Pulse DC LOW
02CE 864 LOADREG RDCTRL,11$,,B ;
02D6 865 LOADREG DCSAD,7$,,I,B ; Address to DCS ADD register
02DE 866 LOOP J,1,11 ; Loop to load test pattern
02E5 867 LOADREG DCSDA,25$,,J,B ; Load test pattern
02ED 868 ENDLOOP J ;
02F0 869 LOADREG DCSAD,7$,,I,B ; Reload DCS ADD register
02F8 870 LOADREG DCSCR,4$,,B ; uADDR INH and Parity Check
0300 871 LOADREG CSAMR,3$,,W ; 1800 to Match Register
0308 872 LOADREG DCSCR,5$,,B ; Tick the clock
0310 873 LOADREG DCSCR,4$,,B ; one micro instruction
0318 874 LOADREG DCSCR,6$,,B ; uADDR INH and Par. check off
0320 875 CMPREGMSK FRONT1,35$,,30$,,B. ; Check parity error light
032C 876 IFERROR ;, <<RDM>> ; Bit stuck in DCS
0332 877 ENDLOOP I ; Terminate the program loop
0335 878
0335 879
0335 880
0335 881 :////////////////////
0335 882 :+
0335 883 :SUBTEST #7
0335 884 :
0335 885 : Test all DCS locations with test pattern #7
0335 886 :-
0335 887 : SUBTEST ; SUBTEST #7
0338
0338 :////////////////////
0338 888 INITIALIZE ; Init the cpu
033A 889 LOADREG RDCTRL,11$,,B ; Clear Master Halt Enable
0342 890 LOOP I,1,64 ; DCS address loop
0349 891 ERRLOOP ; Error loop point
034B 892 LOADREG RDCTRL,1$,,B ; Pulse DC LOW
0353 893 LOADREG RDCTRL,11$,,B ;
035B 894 LOADREG DCSAD,7$,,I,B ; Address to DCS ADD register
0363 895 LOOP J,1,11 ; Loop to load test pattern
036A 896 LOADREG DCSDA,26$,,J,B ; Load test pattern
0372 897 ENDLOOP J ;
0375 898 LOADREG DCSAD,7$,,I,B ; Reload DCS ADD register
037D 899 LOADREG DCSCR,4$,,B ; uADDR INH and Parity Check
0385 900 LOADREG CSAMR,3$,,W ; 1800 to Match Register
038D 901 LOADREG DCSCR,5$,,B ; Tick the clock
0395 902 LOADREG DCSCR,4$,,B ; one micro instruction
039D 903 LOADREG DCSCR,6$,,B ; uADDR INH and Par. check off
03A5 904 CMPREGMSK FRONT1,36$,,30$,,B. ; Check parity error light
03B1 905 IFERROR ;, <<RDM>> ; Bit stuck in DCS
03B7 906 ENDLOOP I ; Terminate the program loop
03BA 907
03BA 908
03BA 909
03BA 910 :////////////////////
03BA 911 :+
03BA 912 :SUBTEST #8
03BA 913 :
03BA 914 : Test all DCS locations with test pattern #8
03BA 915 :-
03BA 916 : SUBTEST ; SUBTEST #8
03BD
```

```
03BD :////////////////////
03BD 917 INITIALIZE : Init the cpu
03BF 918 LOADREG RDCTRL,11$,,B : Clear Master Halt Enable
03C7 919 LOOP I,1,64 : DCS address loop
03CE 920 ERRLOOP : Error loop point
03D0 921 LOADREG RDCTRL,1$,,B : Pulse DC LOW
03D8 922 LOADREG RDCTRL,11$,,B :
03E0 923 LOADREG DCSAD,7$,,I,B : ...
03E8 924 LOOP J,1,11 : Address to DCS ADD register
03EF 925 LOADREG DCSDA,27$,,J,B : Loop to load test pattern
03F7 926 ENDLOOP J : Load test pattern
03FA 927 LOADREG DCSAD,7$,,I,B : ...
0402 928 LOADREG DCSCR,4$,,B : Reload DCS ADD register
040A 929 LOADREG CSAMR,3$,,W : uADDR INH and Parity Check
0412 930 LOADREG DCSCR,5$,,B : 1800 to Match Register
041A 931 LOADREG DCSCR,4$,,B : Tick the clock
0422 932 LOADREG DCSCR,6$,,B : one micro instruction
042A 933 CMPREGMSK FRONT1,37$,,30$,,B. : uADDR INH and Par. check off
0436 934 IFERROR ,,<<RDM>> : Check parity error light
043C 935 ENDLOOP I : Bit stuck in DCS
043F 936 : Terminate the program loop
043F 937
043F 938
043F 939 :////////////////////
043F 940 :+
043F 941 :SUBTEST #9
043F 942 :
043F 943 : Test all DCS locations with test pattern #9
043F 944 :-
043F 945 :SUBTEST : SUBTEST #9
0442 :////////////////////
0442 946 INITIALIZE : Init the cpu
0444 947 LOADREG RDCTRL,11$,,B : Clear Master Halt Enable
044C 948 LOOP I,1,64 : DCS address loop
0453 949 ERRLOOP : Error loop point
0455 950 LOADREG RDCTRL,1$,,B : Pulse DC LOW
045D 951 LOADREG RDCTRL,11$,,B :
0465 952 LOADREG DCSAD,7$,,I,B : ...
046D 953 LOOP J,1,11 : Address to DCS ADD register
0474 954 LOADREG DCSDA,28$,,J,B : Loop to load test pattern
047C 955 ENDLOOP J : Load test pattern
047F 956 LOADREG DCSAD,7$,,I,B : ...
0487 957 LOADREG DCSCR,4$,,B : Reload DCS ADD register
048F 958 LOADREG CSAMR,3$,,W : uADDR INH and Parity Check
0497 959 LOADREG DCSCR,5$,,B : 1800 to Match Register
049F 960 LOADREG DCSCR,4$,,B : Tick the clock
04A7 961 LOADREG DCSCR,6$,,B : one micro instruction
04AF 962 CMPREGMSK FRONT1,38$,,30$,,B. : uADDR INH and Par. check off
04BB 963 IFERROR ,,<<RDM>> : Check parity error light
04C1 964 ENDLOOP I : Bit stuck in DCS
04C4 965 : Terminate the program loop
04C4 966
04C4 967
04C4 968 :////////////////////
04C4 969 :+
04C4 970 :SUBTEST #10
```

```
04C4 971 :  
04C4 972 : Test all DCS locations with test pattern #10  
04C4 973 :-  
04C4 974 SUBTEST ; SUBTEST #10  
04C7  
04C7 :////////////////////  
04C7 975 INITIALIZE ; Init the cpu  
04C9 976 LOADREG RDCTRL,11$,,B ; Clear Master Halt Enable  
04D1 977 LOOP I,1,64 ; DCS address loop  
04D8 978 ERRLOOP ; Error loop point  
04DA 979 LOADREG RDCTRL,1$,,B ; Pulse DC LOW  
04E2 980 LOADREG RDCTRL,11$,,B ;  
04EA 981 LOADREG DCSAD,7$,I,B ; Address to DCS ADD register  
04F2 982 LOOP J,1,11 ; Loop to load test pattern  
04F9 983 LOADREG DCSDA,29$,J,B ; Load test pattern  
0501 984 ENDLOOP J ;  
0504 985 LOADREG DCSAD,7$,I,B ; Reload DCS ADD register  
050C 986 LOADREG DCSCR,4$,,B ; uADDR INH and Parity Check  
0514 987 LOADREG CSAMR,3$,,W ; 1800 to Match Register  
051C 988 LOADREG DCSCR,5$,,B ; Tick the clock  
0524 989 LOADREG DCSCR,4$,,B ; one micro instruction  
052C 990 LOADREG DCSCR,6$,,B ; uADDR INH and Par. check off  
0534 991 CMPREGMSK FRONT1,39$,,30$,,B, ; Check parity error light  
0540 992 IFERROR I,<<RDM>> ; Bit stuck in DCS  
0546 993 ENDLOOP ; Terminate the program loop  
0549 994 SKIP ; Go to next test  
0550 995  
0550 996  
0550 997  
0550 998
```

BEGIN\_TEST\_DATA

```
0550 999 :-----  
40 0550 1000 1$: .BYTE DC LOW  
1800 0551 1001 3$: .WORD DCS START ; Start Address of DCS Space  
E8 0553 1002 4$: .BYTE MICRO_ADDR_INH!PAR_CHK_ENABL!CLK_CTRL_0!CLK_CTRL_1  
A8 0554 1003 5$: .BYTE MICRO_ADDR_INH!PAR_CHK_ENABL!CLK_CTRL_0  
60 0555 1004 6$: .BYTE CLK_CTRL_0!CLK_CTRL_1  
07 06 05 04 03 02 01 00 0556 1005 7$: .BYTE 0,1,2,3,4,5,6,7  
0F 0E 0D 0C 0B 0A 09 08 055E 1006 .BYTE 8,9,10,11,12,13,14,15  
17 16 15 14 13 12 11 10 0566 1007 .BYTE 16,17,18,19,20,21,22,23  
1F 1E 1D 1C 1B 1A 19 18 056E 1008 .BYTE 24,25,26,27,28,29,30,31  
27 26 25 24 23 22 21 20 0576 1009 .BYTE 32,33,34,35,36,37,38,39  
2F 2E 2D 2C 2B 2A 29 28 057E 1010 .BYTE 40,41,42,43,44,45,46,47  
37 36 35 34 33 32 31 30 0586 1011 .BYTE 48,49,50,51,52,53,54,55  
3F 3E 3D 3C 3B 3A 39 38 058E 1012 .BYTE 56,57,58,59,60,61,62,63  
61 0596 1013 8$: .BYTE CLK_CTRL_1!CLK_CTRL_0!HALT_ON_MATCH  
0597 1014 10$:  
U 00, 7700,C800,0364,0300,0470,1801 0597 1015 :% 00 NOP ; Good data for dcs  
00 00 00 00 00 00 00 00 00 00 00 05A2 1019 11$: .BYTE 0  
FF 05A3 1020 20$: .BYTE ^XFF,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00  
AA FF 05AE 1021 21$: .BYTE ^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF  
55 55 55 55 55 55 55 55 55 55 FF 05B9 1022 22$: .BYTE ^XFF,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA  
33 33 33 33 33 33 33 33 33 33 FF 05C4 1023 23$: .BYTE ^XFF,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55  
0F FF 05CF 1024 24$: .BYTE ^XFF,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33  
FF 00 FF 00 FF 00 FF 00 FF 00 FF 05DA 1025 25$: .BYTE ^XFF,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F  
05E5 1026 26$: .BYTE ^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 009 DCS DATA INTEGRITY  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 009 DCS DATA INTEGRITY

C 4  
6-MAY-1982

Fiche 1 Frame C4

Sequence 41

6-MAY-1982 19:54:55 VAX-11 Macro V02.46 Page 3  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1 (1

```
00 00 FF FF 00 00 FF FF 00 00 FF 05F0 1027 27$: .BYTE ^XFF,^X00,^X00,^XFF,^XFF,^X00,^X00,^XFF,^XFF,^X00,^X00
00 00 FF FF FF FF 00 00 00 00 FF 05FB 1028 28$: .BYTE ^XFF,^X00,^X00,^X00,^X00,^XFF,^XFF,^XFF,^XFF,^X00,^X00
FF FF 00 00 00 00 00 00 00 00 FF 0606 1029 29$: .BYTE ^XFF,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^XFF,^XFF
20 0611 1030 30$: .BYTE PARITY_ERROR ; Results table
20 0612 1031 31$: .BYTE PARITY_ERROR
20 0613 1032 32$: .BYTE PARITY_ERROR
20 0614 1033 33$: .BYTE PARITY_ERROR
20 0615 1034 34$: .BYTE PARITY_ERROR
20 0616 1035 35$: .BYTE PARITY_ERROR
20 0617 1036 36$: .BYTE PARITY_ERROR
20 0618 1037 37$: .BYTE PARITY_ERROR
20 0619 1038 38$: .BYTE PARITY_ERROR
20 061A 1039 39$: .BYTE PARITY_ERROR
```

061B 1040  
061B 1041 END\_TEST\_DATA

061B  
061B  
061B 1042 ;-----  
ENDTEST

%MACRO-W-GENWRN, Generated WARNING: TEST MAY BE TOO LARGE WITH DEBUG MONITOR;

ZZ-  
ECK  
V03

```

0018      .SBTTL 'TEST 00A HI NEXT BUS DRIVERS
0018 1044 :*****
0018 1045 :++
0018 1046 :
0018 1047 : FUNCTIONAL DESCRIPTION:
0018 1048 :
0018 1049 :     This test checks the CS ADDRESS Bus Drivers for the HIGH Next field.
0018 1050 :
0018 1051 :
0018 1052 : TEST ALGORITHM:
0018 1053 :
0018 1054 :     1. Create a test loop of 4 iterations
0018 1055 :     2. Block traps from the RDM by clearing MASTER HALT ENABLE
0018 1056 :     3. Load micro word into DCS address 0
0018 1057 :     4. Put micro word on cs lines by performing a FETCH pseudo op
0018 1058 :     5. Check CSBUF register for correct HIGH NEXT FIELD
0018 1059 :     6. If no error go to step 2 for remaining test patterns
0018 1060 :
0018 1061 :
0018 1062 : TEST PATTERNS:
0018 1063 :
0018 1064 :     ITERATION      TEST PATTERN
0018 1065 :           1          2AAA
0018 1066 :           2          1555
0018 1067 :           3          3333
0018 1068 :           4          0F0F
0018 1069 :
0018 1070 :
0018 1071 : LOGIC DESCRIPTION:
0018 1072 :
0018 1073 :     This test verifies the correct operation of the HIGH NEXT FIELD
0018 1074 :     bus drivers (cs address bits <13:6>) by loading test patterns into
0018 1075 :     the NEXT field of micro instructions and reading them back through
0018 1076 :     the CS ADDRESS BUFFER register. Note that all other bits are masked
0018 1077 :     out by the CMPREGMSK pseudo op.
0018 1078 :
0018 1079 :
0018 1080 : ASSUMPTIONS:
0018 1081 :
0018 1082 :     This test assumes the power up self test has run successfully.
0018 1083 :
0018 1084 :
0018 1085 : ERROR DESCRIPTION:
0018 1086 :
0018 1087 :     EXPECTED CSBUF REGISTER CONTENTS
0018 1088 :     RECEIVED CSBUF REGISTER CONTENTS
0018 1089 :     LOOP COUNT
0018 1090 :
0018 1091 : --
0018 1092 :*****
0018 1093 :
0018 1094 :
0018 1095 :
0018 1096 :     INITIALIZE          : Init cpu
001A 1097 :     LOOP                : Loop for 4 patterns
0021 1098 :     LOADREG             : Clear Master Halt Enable
0029 1099 :     ERRLOOP             : Error loop point

```

```
002B 1100 LOADDCS 1$,I,0,1 ; Load a micro word
0032 1101 FETCH 0 ; Fetch the micro word
0037 1102 CMPREGMSK CSBUF,2$,I,3$,,W, ; Check the CS Address Bus
0043 1103 IFERROR ;, <<RDM>> ; Check if there is an error
0049 1104 ENDLOOP I ; End test loop
004C 1105 SKIP ; Go to next test
0053 1106
0053 1107
```

BEGIN\_TEST\_DATA

```
0053 1108 :-----
0053 1109 : The following micro words contain the test patterns for the HIGH NEXT field
0053 1110 :
```

```
U 00, 7700,C800,0364,0300,0470,2AAA
U 00, 7700,C800,0364,0300,0470,1555
U 00, 7700,4800,0364,0300,0470,3333
U 00, 7700,4800,0364,0300,0470,0F0F
```

```
0053 1111 1$:
0053 1112 :% 00 NEXT/2AAA
005E 1116 :% 00 NEXT/1555
0069 1120 :% 00 NEXT/3333
0074 1124 :% 00 NEXT/0F0F
007F 1128
007F 1129 2$: .WORD ^X2AAA,^X1555,^X3333,^X0F0F ; Expected CS ADDRESS
0087 1130
0087 1131 3$: .WORD ^X1FC0 ; CS ADDRESS mask
0089 1132 4$: .BYTE 0
```

0F0F 3333 1555 2AAA

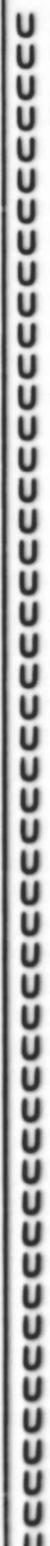
1FC0  
00

END\_TEST\_DATA

```
008A 1133
008A 1134
008A 1135 :-----
008A ENDTEST
```

```
0017 .SBTTL 'TEST 00B DCS ADDRESS DECODE
0017 1137 :*****
0017 1138 :++
0017 1139 :
0017 1140 : FUNCTIONAL DESCRIPTION:
0017 1141 :
0017 1142 : This test ensures that the DCS ADDRESS decoder functions correctly.
0017 1143 : The decode of DCS ADDRESS space has been checked. This test checks
0017 1144 : that not DCS ADDRESS space does not enable the DCS.
0017 1145 :
0017 1146 :
0017 1147 : TEST ALGORITHM:
0017 1148 :
0017 1149 : 1. Load a micro word into DCS address 0 with a 0 next field
0017 1150 : 2. Block traps from the RDM by clearing MASTER HALT ENABLE
0017 1151 : 3. Create a test loop of 6 iterations
0017 1152 : 4. Set uADDR INHIBIT & PARITY CHECK
0017 1153 : 5. Put test pattern into CSAMR register
0017 1154 : 6. Step the clock one micro instruction
0017 1155 : 7. Clear uADDR INHIBIT & PARITY CHECK
0017 1156 : 8. Test the CSBUF register for not a 0 next field
0017 1157 : 9. If no error go to step 4 for remaining test patterns
0017 1158 :
0017 1159 :
0017 1160 : TEST PATTERNS:
0017 1161 :
0017 1162 : ITERATION TEST PATTERN
0017 1163 : 1 0800
0017 1164 : 2 1000
0017 1165 : 3 2000
0017 1166 : 4 2800
0017 1167 : 5 3000
0017 1168 : 6 3800
0017 1169 :
0017 1170 :
0017 1171 : LOGIC DESCRIPTION:
0017 1172 :
0017 1173 : This test insures that the DCS ADDRESS SPACE logic does not enable
0017 1174 : the RDM when address other than 18xx are on the CS ADDRESS BUS.
0017 1175 : If the RDM is enabled there should be a bus conflict with two people
0017 1176 : driving the bus at once. The theory here is that the 0 state of the
0017 1177 : DCS NEXT FIELD will win out and the error condition can be detected.
0017 1178 :
0017 1179 :
0017 1180 : ASSUMPTIONS:
0017 1181 :
0017 1182 : This test assumes the power up self test has run successfully
0017 1183 :
0017 1184 :
0017 1185 : ERROR DESCRIPTION:
0017 1186 :
0017 1187 : EXPECTED CSBUF REGISTER DATA (NOT 0)
0017 1188 : RECEIVED CSBUF REGISTER DATA
0017 1189 : LOOP COUNT
0017 1190 :
0017 1191 : --
0017 1192 :*****
```

```
0017 1193  
0017 1194  
0017 1195  
0017 1196 INITIALIZE : Init the cpu  
0019 1197 LOADDCS 1$,0,1 : Load uword with 0 next field  
0020 1198 LOADREG RDCTRL,8$,B : Clear Master Halt Enable  
0028 1199 LOOP 1,1,6 : Test loop of 6 iterations  
002F 1200 ERRLOOP : Error loop point  
0031 1201 LOADREG DCSAD,8$,B : 0 to DCS ADDRESS register  
0039 1202 LOADREG DCSCR,3$,B : uADDR INH and Parity Check  
0041 1203 LOADREG CSAMR,2$,I,W : Load test pattern  
0049 1204 LOADREG DCSCR,4$,B : Tick the clock  
0051 1205 LOADREG DCSCR,3$,B : one micro instruction  
0059 1206 LOADREG DCSCR,5$,B : uADDR INH and Par. Check to 0  
0061 1207 CMPREGMSK CSBUF,7$,9$,W,NE : Check for not 0 pattern  
006D 1208 IFERROR : Check if error  
0073 1209 ENDLOOP I : End test loop  
0076 1210 SKIP : Go to next test  
007D 1211  
007D 1212 BEGIN_TEST_DATA  
007D :-----  
007D 1213 :  
007D 1214 1$:  
U 00, 7700,4800,0364,0300,0470,0000 007D 1215 ;% 00 NEXT/0000  
0088 1219  
2800 3000 2800 2000 1000 0800 0088 1220 2$: .WORD ^X800,^X1000,^X2000,^X2800,^X3000,^X2800  
E8 0094 1221 3$: .BYTE MICRO_ADDR_INH!PAR_CHK_ENABL!CLK_CTRL_1!CLK_CTRL_0  
A8 0095 1222 4$: .BYTE MICRO_ADDR_INH!PAR_CHK_ENABL!CLK_CTRL_0  
60 0096 1223 5$: .BYTE CLK_CTRL_1!CLK_CTRL_0  
0000 0097 1224 7$: .WORD ^X0000  
00 0099 1225 8$: .BYTE 0  
3FFF 009A 1226 9$: .WORD ^X3FFF  
009C 1227  
009C 1228 END_TEST_DATA  
009C :-----  
009C 1229 :  
ENDTEST
```







```
0093 1340 :SUBTEST #3
0093 1341 :
0093 1342 : Check that clock does not stop if TRAP HALT ENABLE not set
0093 1343 :-
0093 1344 : SUBTEST : SUBTEST #3
0096 :
0096 : ////////////////////////////////////////////////////////////////////
0096 1345 LOADREG DCSCR,1$.B : Set parity check
009E 1346 LOADREG CSAMR,2$.W : 0 on CS BUS
00A6 1347 ERRLOOP : Error loop point
00AB 1348 LOADREG RDCTRL,2$.B : Clear MASTER HALT ENABLE
00B0 1349 LOADREG RDCTRL,10$.B : Set MASTER HALT ENABLE
00BB 1350 LOADREG DCSCR,6$.B : Start the clock
00C0 1351 CMPREGMSK CLKDCS,2$.4$.B. : Check clock still running
00CC 1352 LOADREG DCSCR,5$.B : Stop the clock
00D4 1353 IFERROR ..<<RDM>> : Clock had stopped
00DA 1354
00DA 1355
00DA 1356 :
00DA 1357 : ////////////////////////////////////////////////////////////////////
00DA 1358 :+
00DA 1359 :SUBTEST #4
00DA 1360 :
00DA 1361 : Check that the clock does not stop if DCS is selected
00DA 1362 :-
00DA 1363 : SUBTEST : SUBTEST #4
00DD :
00DD : ////////////////////////////////////////////////////////////////////
00DD 1364 ERRLOOP : Error loop point
00DF 1365 LOADREG DCSCR,1$.B : Set parity check
00E7 1366 LOADREG CSAMR,20$.W : 1800 on CS BUS
00EF 1367 LOADREG RDCTRL,2$.B : Clear Master Halt En
00F7 1368 LOADREG DCSCR,21$.B : Tick the clock
00FF 1369 LOADREG DCSCR,5$.B : one u-inst
0107 1370 LOADREG RDCTRL,3$.B : MASTER HALT & TRAP EN
010F 1371 LOADREG DCSCR,6$.B : Start the clock
0117 1372 CMPREGMSK CLKDCS,2$.4$.B. : Check clock still running
0123 1373 LOADREG DCSCR,5$.B : Stop the clock
012B 1374 IFERROR ..<<RDM>> : Clock had stopped
0131 1375
0131 1376
0131 1377 :
0131 1378 : ////////////////////////////////////////////////////////////////////
0131 1379 :+
0131 1380 :SUBTEST #5
0131 1381 :
0131 1382 : Check that the clock does not stop if MASTER HALT ENABLE is clear
0131 1383 :-
0131 1384 : SUBTEST : SUBTEST #5
0134 :
0134 : ////////////////////////////////////////////////////////////////////
0134 1385 LOADREG DCSCR,1$.B : Set parity check
013C 1386 LOADREG CSAMR,2$.W : 0 on CS BUS
0144 1387 ERRLOOP : Error loop point
0146 1388 LOADREG RDCTRL,2$.B : Clear Master Halt En
014E 1389 LOADREG RDCTRL,30$.B : Set Trap Halt Enable
0156 1390 LOADREG DCSCR,6$.B : Start the clock
```

015E 1391 CMPREGMSK CLKDCS,2\$,4\$,B, ; Check clock still running  
016A 1392 LOADREG DCSCR,5\$,B ; Stop the clock  
0172 1393 IFERROR ; Clock had stopped  
0178 1394 SKIP ; Go to next test  
017F 1395  
017F 1396

BEGIN\_TEST\_DATA

017F ;-----  
017F 1397  
E8 017F 1398 1\$: .BYTE CLK\_CTRL\_1!CLK\_CTRL\_0!PAR\_CHK\_ENABL!MICRO\_ADDR\_INH  
0000 0180 1399 2\$: .WORD 0  
28 0182 1400 3\$: .BYTE MASTER\_HALT\_EN!TRAP\_HALT\_EN  
C0 0183 1401 4\$: .BYTE CLK\_CTRL\_0\_RO!CLK\_CTRL\_1\_RO  
60 0184 1402 5\$: .BYTE CLK\_CTRL\_1!CLK\_CTRL\_0  
88 0185 1403 6\$: .BYTE PAR\_CHK\_ENABL!MICRO\_ADDR\_INH  
0186 1404  
08 0186 1405 10\$: .BYTE MASTER\_HALT\_EN  
0187 1406  
1800 0187 1407 20\$: .WORD DCS\_START  
A8 0189 1408 21\$: .BYTE PAR\_CHK\_ENABL!MICRO\_ADDR\_INH!CLK\_CTRL\_0  
018A 1409  
20 018A 1410 30\$: .BYTE TRAP\_HALT\_EN  
018B 1411  
0B 0A 09 08 07 06 05 04 03 02 01 00 018B 1412 40\$: .BYTE 0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22  
16 15 14 13 12 11 10 0F 0E 0D 0C 0197  
22 21 20 1F 1E 1D 1C 1B 1A 19 18 17 01A2 1413 .BYTE 23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42  
36 35 34 33 32 2A 29 28 27 26 25 24 23 01AE  
31 30 2F 2E 2D 2C 2B 01B6 1414 .BYTE 43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61  
3D 3C 3B 3A 39 38 37 01C2  
3F 3E 01C9 1415 .BYTE 62,63  
FF 01CB 1416 41\$: .BYTE ^XFF  
01CC 1417  
01CC 1418 END\_TEST\_DATA  
01CC  
01CC ;-----  
01CC 1419 ENDTEST

```
0016 .SBTTL 'TEST OOD DISABLE HIGH NEXT
0016 1421 :*****
0016 1422 :++
0016 1423 :
0016 1424 : FUNCTIONAL DESCRIPTION:
0016 1425 :
0016 1426 :     Check that DISABLE HIGH NEXT disables the HIGH NEXT drivers.
0016 1427 :
0016 1428 :
0016 1429 : TEST ALGORITHM:
0016 1430 :
0016 1431 :     1. Load the HIGH NEXT latch with a known test pattern
0016 1432 :     2. Assert uADDR INHIBIT.
0016 1433 :     3. Check that CS ADDRESS is not the contents of the CS ADDR Buffer
0016 1434 :
0016 1435 :
0016 1436 : TEST PATTERNS:
0016 1437 :
0016 1438 :     2AAA
0016 1439 :
0016 1440 :
0016 1441 : LOGIC DESCRIPTION:
0016 1442 :
0016 1443 :     This test detects whether or not the DISABLE HIGH NEXT signal coming
0016 1444 :     from COMET is able to shut of the HIGH NEXT FIELD bus drivers.
0016 1445 :
0016 1446 :
0016 1447 : ASSUMPTIONS:
0016 1448 :
0016 1449 :     This test assumes that COMET is sending the DISABLE HIGH NEXT signal
0016 1450 :     when it receives uADDR INHIBIT from the RDM.
0016 1451 :
0016 1452 :
0016 1453 : ERROR DESCRIPTION:
0016 1454 :
0016 1455 :     NOT EXPECTED CSBUF DATA
0016 1456 :     RECEIVED CSBUF DATA
0016 1457 :
0016 1458 : --
0016 1459 :*****
0016 1460 :
0016 1461 :
0016 1462 :
0016 1463 :     INITIALIZE
0018 1464 :     LOADDCS          1$..0.1
001F 1465 :     ERRLOOP
0021 1466 :     FETCH           0
0026 1467 :     LOADREG         DCSCR,2$..B
002E 1468 :     CMPREGMSK       CSBUF,5$..3$..W,NE
003A 1469 :     LOADREG         DCSCR,4$..B
0042 1470 :     IFERROR         ..<<RDM>>
0048 1471 :     SKIP
004F 1472 :
004F 1473 :     BEGIN_TEST_DATA
004F 1474 :-----
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST OOD DISABLE HIGH NEXT  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST OOD DISABLE HIGH NEXT

M 4  
6-MAY-1982

Fiche 1 Frame M4

Sequence 51

6-MAY-1982 19:54:55 VAX-11 Macro V02.46 Page 4  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1 (1

```
U 00, 7700, C800, 0364, 0300, 0470, 2AAA 004F 1475 1$:  
004F 1476 :% 00 NEXT/2AAA  
005A 1480  
EO 005A 1481 2$: .BYTE MICRO ADDR_INH!CLK_CTRL_1!CLK_CTRL_0  
1FC0 005B 1482 3$: .WORD ^X1FC0 ; Mask for HIGH NEXT  
60 005D 1483 4$: .BYTE CLK_CTRL_1!CLK_CTRL_0  
2AAA 005E 1484 5$: .WORD ^X2AAA  
0060 1485  
0060 1486 END_TEST_DATA  
0060  
0060  
0060 1487 :-----  
ENDTEST
```

```
0021 .SBTTL 'TEST 00E DCS ADDRESS REGISTER M CLOCK
0021 1489 :*****
0021 1490 :++
0021 1491 :
0021 1492 : FUNCTIONAL DESCRIPTION:
0021 1493 :
0021 1494 :     This test checks that the DCS ADDRESS register increments with
0021 1495 :     M CLOCK.
0021 1496 :
0021 1497 :
0021 1498 : TEST ALGORITHM:
0021 1499 :
0021 1500 :     1. Load a NOP into DCS address 0
0021 1501 :     2. Load the NOP into the CS LATCHES via the FETCH pseudo op
0021 1502 :     3. Test the CLKDCS register to see if it incremented to 1
0021 1503 :
0021 1504 :
0021 1505 : TEST PATTERNS:
0021 1506 :
0021 1507 :     none
0021 1508 :
0021 1509 :
0021 1510 : LOGIC DESCRIPTION:
0021 1511 :
0021 1512 :     This test checks that the DCS ADDRESS register is clocked by M clock.
0021 1513 :     The FETCH pseudo op should tick the clock once in order to load the
0021 1514 :     first DCS micro instruction.
0021 1515 :
0021 1516 :
0021 1517 : ASSUMPTIONS:
0021 1518 :
0021 1519 :     This test assumes the power up self test has been run successfully.
0021 1520 :
0021 1521 :
0021 1522 : ERROR DESCRIPTION:
0021 1523 :
0021 1524 :     EXPECTED CLKDCS REGISTER CONTENTS
0021 1525 :     RECEIVED CLKDCS REGISTER CONTENTS
0021 1526 :
0021 1527 : --
0021 1528 :*****
0021 1529 :
0021 1530 :
0021 1531 :
0021 1532 :     INITIALIZE                                ; Init the cpu
0023 1533 :     LOADDCS 1$,,0,1                          ; Load a NOP micro word
002A 1534 :     FETCH 0                                   ; Start execution
002F 1535 :     CMPREGMSK CLKDCS,2$,,3$,,B.             ; Test the ADDRESS REG
003B 1536 :     IFERROR ..<<RDM>>                        ; Check if error
0041 1537 :     SKIP                                       ; Go to the next test
0048 1538 :
0048 1539 :     BEGIN_TEST_DATA
0048 :
0048 :-----
0048 1540 :
0048 1541 1$:
0048 1542 ;% 00 NOP                                ; NOP Micro Instruction
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 00E DCS ADDRESS REGISTER M CLOCK

VAX-11/750 RDM MICRO DIAGNOSTICS

'TEST 00E DCS ADDRESS REGISTER M CLOCK

B 5  
6-MAY-1982

6-MAY-1982 19:54:55  
6-MAY-1982 19:54:39

Fiche 1 Frame B5

VAX-11 Macro V02.46  
WRKDS0:[COMET.RDM]ECKAF.MAR;1

Sequence 53

Page 5  
(1

```
01 0053 1546 2$: .BYTE 1
3F 0054 1547 3$: .BYTE ^X3F
0055 1548
0055 1549 END_TEST_DATA
0055
0055 ;-----
0055 1550 ENDTEST
```

```

0013 .SBTTL 'TEST OOF DCS CYCLE TEST
0013 1552 :*****
0013 1553 :++
0013 1554 :
0013 1555 : FUNCTIONAL DESCRIPTION:
0013 1556 :
0013 1557 :     This test checks that NOP micro instructions in the DCS will cycle
0013 1558 :     at full speed and won't cause errors that stop the clock.
0013 1559 :
0013 1560 :
0013 1561 : TEST ALGORITHM:
0013 1562 :
0013 1563 :     1. Load all 64 DCS locations with NOP micro instructions
0013 1564 :     2. FETCH the first micro instruction
0013 1565 :     3. Let the clock free run
0013 1566 :     4. Test that the clock is still running
0013 1567 :     5. Stop the clock
0013 1568 :
0013 1569 :
0013 1570 : TEST PATTERNS:
0013 1571 :
0013 1572 :     none
0013 1573 :
0013 1574 :
0013 1575 : LOGIC DESCRIPTION:
0013 1576 :
0013 1577 :     This test takes advantage of the fact that the DCS ADDRESS register
0013 1578 :     returns to 0 after counting up to 64. This allows the COMET to
0013 1579 :     continuously cycle on the NOP micro instructions in DCS. This test
0013 1580 :     is probably the best way to detect DCS parity errors. See the next
0013 1581 :     test if you suspect parity errors in DCS.
0013 1582 :
0013 1583 :
0013 1584 : ASSUMPTIONS:
0013 1585 :
0013 1586 :     This test assumes that the parity checking logic in COMET is
0013 1587 :     operational.
0013 1588 :
0013 1589 :
0013 1590 : ERROR DESCRIPTION:
0013 1591 :
0013 1592 :     EXPECTED CLKDCS DATA
0013 1593 :     RECEIVED CLKDCS DATA
0013 1594 :
0013 1595 : --
0013 1596 :*****
0013 1597 :
0013 1598 :
0013 1599 :
0015 1600 : ERRLOOP : Error loop point
0017 1601 : INITIALIZE : INIT THE CPU
001E 1602 : LOADDCS : Load the DCS
0023 1603 : FETCH : Fetch location 0
002B 1604 : LOADREG : Start the clock
0037 1605 : CMPREGMSK : Test clock stop bits
003F 1606 : LOADREG : Stop the clock
003F 1607 : IFERROR : Unexpected clock stop
          1$.0.64
          0
          DCSCR,2$.B
          CLKDCS,4$.3$.B.
          DCSCR,5$.B
          .,<<RDM>>

```

0045 1608 SKIP  
004C 1609  
004C 1610 BEGIN\_TEST\_DATA

; Go to next text

004C 1611

004C 1612 1\$:

U 00. 7700.c800.0364.0300.0470.1801 004C 1613 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0057 1617 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0062 1621 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 006D 1625 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0078 1629 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0083 1633 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 008E 1637 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0099 1641 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00A4 1645 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00AF 1649 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00BA 1653 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00C5 1657 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00D0 1661 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00DB 1665 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00E6 1669 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00F1 1673 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 00FC 1677 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0107 1681 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0112 1685 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 011D 1689 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0128 1693 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0133 1697 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 013E 1701 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0149 1705 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0154 1709 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 015F 1713 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 016A 1717 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0175 1721 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0180 1725 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 018B 1729 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0196 1733 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01A1 1737 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01AC 1741 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01B7 1745 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01C2 1749 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01CD 1753 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01D8 1757 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01E3 1761 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01EE 1765 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 01F9 1769 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0204 1773 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 020F 1777 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 021A 1781 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0225 1785 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0230 1789 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 023B 1793 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0246 1797 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0251 1801 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 025C 1805 :% 00 NOP  
U 00. 7700.c800.0364.0300.0470.1801 0267 1809 :% 00 NOP

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST OOF DCS CYCLE TEST  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST OOF DCS CYCLE TEST

E 5  
6-MAY-1982

Fiche 1 Frame E5

Sequence 56

6-MAY-1982 19:54:55 VAX-11 Macro V02.46 Page 5  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1 (1

```
U 00, 7700, C800, 0364, 0300, 0470, 1801 0272 1813 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 027D 1817 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 0288 1821 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 0293 1825 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 029E 1829 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02A9 1833 :% C0 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02B4 1837 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02BF 1841 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02CA 1845 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02D5 1849 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02E0 1853 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02EB 1857 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 02F6 1861 :% 00 NOP
U 00, 7700, C800, 0364, 0300, 0470, 1801 0301 1865 :% 00 NOP
      030C 1869
      10 030C 1870 2$: .BYTE PAR_STOP_ENABL
      CO 030D 1871 3$: .BYTE CLK_CTRL_1_RO!CLK_CTRL_0_RO
      00 030E 1872 4$: .BYTE 0
      60 030F 1873 5$: .BYTE CLK_CTRL_1!CLK_CTRL_0
      0310 1874
      0310 1875 END_TEST_DATA
      0310
      0310
      0310 1876 :-----
      ENDTEST
```

```
0023 .SBTTL 'TEST 010 TEST TO SCOPE DCS PARITY ERROR
0023 1878 :*****
0023 1879 :++
0023 1880 :
0023 1881 : FUNCTIONAL DESCRIPTION:
0023 1882 :
0023 1883 :     Test to cycle through DCS memory and scope out bad bits.
0023 1884 :
0023 1885 :
0023 1886 : TEST ALGORITHM:
0023 1887 :
0023 1888 :     1. Execute NOPS in DCS memory
0023 1889 :     2. Test CLKDCS register for address FF
0023 1890 :
0023 1891 :
0023 1892 : TEST PATTERNS:
0023 1893 :
0023 1894 :     none
0023 1895 :
0023 1896 :
0023 1897 : LOGIC DESCRIPTION:
0023 1898 :
0023 1899 :     This test is identical to the previous test except that a BRSTCLK
0023 1900 :     pseudo op is used to run the clock. This allows you to use the
0023 1901 :     SE ST CY command and single step through DCS. By stopping at the
0023 1902 :     address before which you suspect of having a parity error you can scope
0023 1903 :     the output of each DCS ram for correct output.
0023 1904 :
0023 1905 :
0023 1906 : ASSUMPTIONS:
0023 1907 :
0023 1908 :     This test assumes that the parity checking logic in COMET is
0023 1909 :     operational.
0023 1910 :
0023 1911 :
0023 1912 : ERROR DESCRIPTION:
0023 1913 :
0023 1914 :     EXPECTED CLKDCS REGISTER DATA
0023 1915 :     RECEIVED CLKDCS REGISTER DATA
0023 1916 :
0023 1917 : --
0023 1918 :*****
0023 1919 :
0023 1920 :
0023 1921 :
0023 1922 :
0023 1923 : ERRLOOP                               : Error loop point
0025 1924 : INITIALIZE                            : Init the cpu
0027 1925 : EXPUTRAP                               : Possible u-trap
0029 1926 : LOADDCS                                : Load nops into DCS
0030 1927 : FETCH                                  : Start DCS program
0035 1928 : BRSTCLK                                : End DCS program
0039 1929 : CMPREG                                  : Test stop address
0042 1930 : IFERROR                                 : Unexpected stop
0048 1931 : SKIP                                    : Go to next test
004F 1932 :
004F 1933 :
```

```
004F 1934  
004F 1935          BEGIN_TEST_DATA  
004F  
004F  
004F 1936  
FF 004F 1937 1$: .BYTE ^XFF  
0050 1938 2$:  
U 00. 7700.C800.0364.0300.0470.1801 0050 1939 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 005B 1943 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0066 1947 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0071 1951 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 007C 1955 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0087 1959 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0092 1963 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 009D 1967 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00A8 1971 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00B3 1975 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00BE 1979 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00C9 1983 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00D4 1987 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00DF 1991 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00EA 1995 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 00F5 1999 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0100 2003 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 010B 2007 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0116 2011 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0121 2015 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 012C 2019 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0137 2023 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0142 2027 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 014D 2031 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0158 2035 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0163 2039 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 016E 2043 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0179 2047 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0184 2051 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 018F 2055 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 019A 2059 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01A5 2063 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01B0 2067 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01BB 2071 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01C6 2075 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01D1 2079 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01DC 2083 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01E7 2087 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01F2 2091 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 01FD 2095 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0208 2099 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0213 2103 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 021E 2107 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0229 2111 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0234 2115 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 023F 2119 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 024A 2123 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0255 2127 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 0260 2131 :% 00 NOP  
U 00. 7700.C800.0364.0300.0470.1801 026B 2135 :% 00 NOP
```



```

001D .SBTTL 'TEST 011 DISABLE OF DCS ADDR INCR
001D 2195 :*****
001D 2196 :++
001D 2197 :
001D 2198 : FUNCTIONAL DESCRIPTION:
001D 2199 :
001D 2200 : This test checks that the DCS ADDRESS register does not increment
001D 2201 : if the DCS is not selected.
001D 2202 :
001D 2203 :
001D 2204 : TEST ALGORITHM:
001D 2205 :
001D 2206 : 1. Clear MASTER HALT ENABLE
001D 2207 : 2. Load micro code into DCS
001D 2208 : 3. Fetch DCS ADDRESS 0 and step the clock twice
001D 2209 : 4. Test CLKDCS register for DCS ADDRESS 01
001D 2210 : 5. If no error go to next test
001D 2211 :
001D 2212 :
001D 2213 : TEST PATTERNS:
001D 2214 :
001D 2215 : none
001D 2216 :
001D 2217 :
001D 2218 : LOGIC DESCRIPTION:
001D 2219 :
001D 2220 : This test makes sure that the DCS ADDRESS register does not
001D 2221 : increment on M clocks if the CS address is not in DCS space.
001D 2222 :
001D 2223 :
001D 2224 : ASSUMPTIONS:
001D 2225 :
001D 2226 : This test assumes that the power-up self test has run successfully
001D 2227 :
001D 2228 :
001D 2229 : ERROR DESCRIPTION:
001D 2230 :
001D 2231 : EXPECTED CLKDCS REGISTER CONTENTS
001D 2232 : RECEIVED CLKDCS REGISTER CONTENTS
001D 2233 :
001D 2234 : --
001D 2235 :*****
001D 2236 :
001D 2237 :
001D 2238 :
001D 2239 : INITIALIZE
001F 2240 : LOADREG RDCTRL,6$,,B : Init the cpu
0027 2241 : LOADDCS 1$,,0,3 : Clear MASTER HALT ENABLE
002E 2242 : FETCH 0 : Load NOP micro instructions
0033 2243 : LOOP 1,1,2 : Start execution
003A 2244 : LOADREG DCSCR,2$,,B : Loop to tick the clock
0042 2245 : LOADREG DCSCR,3$,,B : Tick the clock
004A 2246 : ENDLOOP 1 : ...
004D 2247 : CMPREGMSK CLKDCS,4$,,5$,,B, : End loop
0059 2248 : IFERROR ..<<RDM>> : Test DCS ADDRESS
005F 2249 : SKIP : Incorrect address
0066 2250 : : Go to next test

```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 011 DISABLE OF DCS ADDR INCR  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 011 DISABLE OF DCS ADDR INCR

J 5  
6-MAY-1982

Fiche 1 Frame J5  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1  
Sequence 61 Page 5 (1

```
0066 2251          BEGIN_TEST_DATA
0066
0066 :-----
0066 2252
0066 2253 1$:
0066 2254 :% C0  NOP,NEXT/0          ; Micro code
U 00. 7700,4800,0364,0300,0470,0000 0071 2258 :% 01  NOP
U 01. 7700,C800,0364,0300,0470,1802 007C 2262 :% 02  NOP
U 02. 7700,4800,0364,0300,0470,1803 0087 2266
20 0087 2267 2$: .BYTE CLK_CTRL_0
60 0088 2268 3$: .BYTE CLK_CTRL_1!CLK_CTRL_0
01 0089 2269 4$: .BYTE 1
3F 008A 2270 5$: .BYTE ^X3F
00 008B 2271 6$: .BYTE 0
008C 2272
008C 2273          END_TEST_DATA
008C :-----
008C 2274
008C 2275 .SBTTL ""
008C 2276 .SBTTL 'MAINTENANCE REGISTER FUNCTIONAL TESTS
008C 2277 .SBTTL ""
008C 2278          ENDTEST
```

```
0026 .SBTTL 'TEST 012 WD AND WH REGISTER DATA INTEGRITY
0026 2280 :*****
0026 2281 :++
0026 2282 :
0026 2283 : FUNCTIONAL DESCRIPTION:
0026 2284 :
0026 2285 : This test loads test patterns into the WD register and reads them
0026 2286 : back through the WH register.
0026 2287 :
0026 2288 :
0026 2289 : TEST ALGORITHM:
0026 2290 :
0026 2291 : 1. Create a test loop of 6 iterations
0026 2292 : 2. Generate a test pattern
0026 2293 : 3. Load WD register with the test pattern.
0026 2294 : 4. Enable WD register onto the WBUS
0026 2295 : 5. Strobe the WH register to read the WBUS
0026 2296 : 6. Test the WH register for the test pattern
0026 2297 : 7. Disable the WD register from the WBUS
0026 2298 : 8. Test that the WH register did not change
0026 2299 : 9. If no errors go to step 3 for remaining test patterns
0026 2300 :
0026 2301 :
0026 2302 : TEST PATTERNS:
0026 2303 :
0026 2304 : AAAAAAA
0026 2305 : 5555555
0026 2306 : 3333333
0026 2307 : 0F0F0F0F
0026 2308 : 00FF00FF
0026 2309 : 0000FFFF
0026 2310 :
0026 2311 :
0026 2312 : LOGIC DESCRIPTION:
0026 2313 :
0026 2314 : This test checks the ability of the RDM to load test patterns
0026 2315 : into the WD register and drive them onto the WBUS. The patterns
0026 2316 : are then read back through the WH register and checked for accuracy.
0026 2317 : Any errors in this data path should be detected by this test.
0026 2318 :
0026 2319 :
0026 2320 : ASSUMPTIONS:
0026 2321 :
0026 2322 : This test assumes that the COMET WBUS is operational.
0026 2323 :
0026 2324 :
0026 2325 : ERROR DESCRIPTION:
0026 2326 :
0026 2327 : Be sure to check the failing PC as there are two IFERROR pseudo ops
0026 2328 : used in this test.
0026 2329 :
0026 2330 : EXPECTED WH REGISTER CONTENTS
0026 2331 : RECEIVED WH REGISTER CONTENTS
0026 2332 : LOOP COUNT
0026 2333 :
0026 2334 : --
0026 2335 :*****
```

```
0026 2336
0026 2337
0026 2338
0026 2339      INITIALIZE      : Init the cpu
0028 2340      LOOP          I,1,6    : Set the test loop parameters
002F 2341      SA01PAT       I,1$,..  : Generate the test pattern
0037 2342      ERRLOOP      : Error loop point
0039 2343      LOADREG      WDREG,1$,..L : Test pattern to WD register
0041 2344      LOADREG      MAIN32,2$,..B : Enable the WD reg to WBUS
0049 2345      LOADREG      MAIN32,3$,..B : Strobe the WH Register
0051 2346      LOADREG      MAIN32,2$,..B :
0059 2347      CMPREG      WHREG,1$,..L : Test the WH register
0062 2348      IFERROR      ,,<<RDM>> : Error between WD and WH reg
0068 2349
0068 2350      LOADREG      MAIN32,4$,..B : Disable the WD register
0070 2351      CMPREG      WHREG,1$,..L : Test the WH register
0079 2352      IFERROR      ,,<<RDM>> : WH register changed
007F 2353      ENDLOOP      I        : Terminate the program loop
0082 2354      SKIP
0089 2355
0089 2356      BEGIN_TEST_DATA
0089
0089 ;-----
0089 2357
0000008D 0089 2358 1$:      .BLKB      4          : Storage for test pattern
15 008D 2359 2$:      .BYTE      MAINT_WD TO WB!MAINT_TRAP_OFF!-
008E 2360      MAINT_DCS ENABL          :ENABLE WD REGISTER ONTO WBUS
1D 008E 2361 3$:      .BYTE      MAINT_WD TO WB!MAINT_WB TO WH!-
008F 2362      MAINT_TRAP_OFF!MAINT_DCS ENABL
05 008F 2363 4$:      .BYTE      MAINT_TRAP_OFF!MAINT_DCS ENABL : Disable the WD register
0090 2364
0090 2365      END_TEST_DATA
0090
0090 ;-----
0090 2366      ENDTEST
```

```
0018 .SBTTL 'TEST 013 VBUS DATA INTEGRITY
0018 2368 :*****
0018 2369 :++
0018 2370 :
0018 2371 : FUNCTIONAL DESCRIPTION:
0018 2372 :
0018 2373 : This test checks that the VBUS can be loaded (via shifting) with all
0018 2374 : zero's and all one's and that the LOAD Control Line causes data to
0018 2375 : be loaded into the VBUS.
0018 2376 :
0018 2377 :
0018 2378 : TEST ALGORITHM:
0018 2379 :
0018 2380 : SUBTEST #1
0018 2381 : 1. Create a test loop of 64 iterations
0018 2382 : 2. Clock 0's into the VBUS
0018 2383 : 3. Go to step 2 for 64 iterations
0018 2384 : 4. Create a test loop of 64 iterations
0018 2385 : 5. Test output of VBUS for 0's
0018 2386 : 6. If no error continue
0018 2387 : 7. Shift VBUS one bit
0018 2388 : 8. Go to step 5 for 64 iterations
0018 2389 :
0018 2390 : SUBTEST #2
0018 2391 : 1. Create a test loop of 64 iterations
0018 2392 : 2. Clock 1's into the VBUS
0018 2393 : 3. Go to step 2 for 64 iterations
0018 2394 : 4. Create a test loop of 64 iterations
0018 2395 : 5. Test output of VBUS for 1's
0018 2396 : 6. If no error continue
0018 2397 : 7. Shift VBUS one bit
0018 2398 : 8. Go to step 5 for 64 iterations
0018 2399 :
0018 2400 : SUBTEST #3
0018 2401 : 1. Fetch first DCS micro instruction and leave uADDR INHIBIT asserted
0018 2402 : 2. Load 1800 into CSAMR and put on CS ADDRESS bus
0018 2403 : 3. Step the COMET clock one micro instruction
0018 2404 : 4. Load the VBUS
0018 2405 : 5. Test the VBUS parity error bit
0018 2406 :
0018 2407 :
0018 2408 : TEST PATTERNS:
0018 2409 :
0018 2410 : none
0018 2411 :
0018 2412 :
0018 2413 : LOGIC DESCRIPTION:
0018 2414 :
0018 2415 : This test checks that all 0's and all 1's can be shifted into and read
0018 2416 : from the VBUS. In addition a parity error is loaded into DCS address 1
0018 2417 : and allowed onto the COMET CS BUS. The VBUS is loaded at this time to
0018 2418 : see if this function is operational.
0018 2419 :
0018 2420 :
0018 2421 : ASSUMPTIONS:
0018 2422 :
0018 2423 : This test assumes that the COMET parity checking logic and VBUS
```

```
0018 2424 : are operational.
0018 2425 :
0018 2426 :
0018 2427 : ERROR DESCRIPTION:
0018 2428 :
0018 2429 : SUBTEST #1 & 2
0018 2430 : EXPECTED STATUS REGISTER CONTENTS
0018 2431 : RECEIVED STATUS REGISTER CONTENTS
0018 2432 : LOOP COUNT
0018 2433 :
0018 2434 : SUBTEST #3
0018 2435 : EXPECTED VBUS BIT 1D
0018 2436 : RECEIVED VBUS BIT 1D
0018 2437 :
0018 2438 : Bit<8> contains the value
0018 2439 : Bits <7:0> contain the bit position
0018 2440 : --
0018 2441 : *****
0018 2442 : ////////////////////////////////////////////////////////////////////
0018 2443 : +
0018 2444 : SUBTEST #1
0018 2445 :
0018 2446 : Fill the VBUS with all 0's and check the output
0018 2447 : -
0018 2448 : SUBTEST ; SUBTEST #1
0018 : ////////////////////////////////////////////////////////////////////
0018 2449 : INITIALIZE ; Init the cpu
001D 2450 : ERRLOOP ; Error loop point
001F 2451 : LOOP I,1,64 ; Create a test loop of 64
0026 2452 : LOADREG CSAMR,4$,W ; Clock the VBUS
002E 2453 : LOADREG CSAMR,5$,W
0036 2454 : ENDLOOP I ;
0039 2455 : LOOP I,1,64 ; End test loop
0040 2456 : CMPREGMSK STATUS,2$,3$,B ; Create a test loop of 64
004C 2457 : IFERROR <<RDM>> ; Check for zero's coming out
0052 2458 : LOADREG CSAMR,4$,W ; Incorrect data
005A 2459 : LOADREG CSAMR,5$,W ; Clock the VBUS
0062 2460 : ENDLOOP I ;
0065 2461 : End test loop
0065 2462 :
0065 2463 : ////////////////////////////////////////////////////////////////////
0065 2464 : +
0065 2465 : SUBTEST #2
0065 2466 :
0065 2467 : Fill the VBUS with 1's and check the output
0065 2468 : -
0065 2469 : SUBTEST ; SUBTEST #2
0068 : ////////////////////////////////////////////////////////////////////
0068 2470 : ERRLOOP ; Error loop point
006A 2471 : LOOP I,1,64 ; Shift 64 bits into the VBUS
0071 2472 : LOADREG CSAMR,1$,W ; Clock the VBUS
0079 2473 : LOADREG CSAMR,2$,W
0081 2474 : ENDLOOP I ;
0084 2475 : LOOP I,1,64 ; End test loop
0088 2476 : CMPREGMSK STATUS,3$,3$,B ; Check 64 bits
: Check serial out for one
```

```
0097 2477          IFERROR          ..<<RDM>>          : Incorrect output
009D 2478          LOADREG          CSAMR,1$,W          : Clock the VBUS
00A5 2479          LOADREG          CSAMR,2$,W          :
00AD 2480          ENDLOOP          I          : End test loop
00B0 2481
00B0 2482
00B0 2483 :////////////////////
00B0 2484 :+
00B0 2485 :SUBTEST #3
00B0 2486 :
00B0 2487 :          Test load of VBUS
00B0 2488 :-
00B0 2489 :          SUBTEST          : SUBTEST #3
00B3
00B3 :////////////////////
00B3 2490          INITIALIZE          : Init the cpu
00B5 2491          LDFIELD          25$,B,31$,78,78,....NP. : Load parity error
00C6 2492          LOADDCS          30$,0,3          : Load test u-code
00CD 2493          ERRLOOP          : Error loop point
00CF 2494          FETCH          0,INH          : Latch DCS address 0
00D4 2495          LOADREG          CSAMR,21$,W          : Load 1800 into CSAMR
00DC 2496          LOADREG          DCSCR,22$,B          : Put 1800 onto CS ADDR BUS
00E4 2497          LOADREG          DCSCR,23$,B          : Tick the COMET clock
00EC 2498          LOADREG          DCSCR,22$,B          :
00F4 2499          LOADREG          RDCTRL,24$,B          : Load the VBUS
00FC 2500          LOADREG          RDCTRL,25$,B          :
0104 2501          CMPVBUS          26$          : Test the VBUS for cs parity
0109 2502          IFERROR          ..<<RDM>>          : VBUS didn't load
010F 2503          SKIP          : Go to next test
0116 2504
0116 2505
0116 2506
0116 2507          BEGIN_TEST_DATA
0116
0116 :-----
0116 2508
0116 2509 :+
0116 2510 : Subtest 1 and 2 data
0116 2511 :-
8000 0116 2512 1$: .WORD ^X8000          : VBUS_CLOCK
0000 0118 2513 2$: .WORD 0
01 011A 2514 3$: .BYTE VBUS_SERIAL_OUT
C000 011B 2515 4$: .WORD ^XC000          : VBUS_SERIAL_IN!VBUS_CLOCK
4000 011D 2516 5$: .WORD ^X4000          : VBUS_SERIAL_IN
011F 2517
011F 2518 :+
011F 2519 : Subtest 3 data
011F 2520 :-
1800 011F 2521 21$: .WORD DCS_START
E8 0121 2522 22$: .BYTE PAR_CHK_ENABL!MICRO_ADDR_INH!CLK_CTRL_0!CLK_CTRL_1
A8 0122 2523 23$: .BYTE PAR_CHK_ENABL!MICRO_ADDR_INH!CLK_CTRL_0
10 0123 2524 24$: .BYTE VBUS_LOAD
00 0124 2525 25$: .BYTE ^X00          : Clear VBUS_LOAD
01 0125 2526 26$: .BYTE 1          : VBUS data
0126 2527          VBUSDATA          <^X1D>,HIGH          : CS parity error
0127 2528
0127 2529 30$:          : Test u-code
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 013 VBUS DATA INTEGRITY  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 013 VBUS DATA INTEGRITY

C 6  
6-MAY-1982

Fiche 1 Frame C6  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1  
Sequence 67  
Page 6 (1)

```
U 00, 7700,C800,0364,0300,0470,1801 0127 2530 :% 00 NOP
0132 2534 31$:
U 01, 7700,C800,0364,0300,0470,1802 0132 2535 :% 01 NOP ; Force parity error
U 02, 7700,4800,0364,0300,0470,1803 013D 2539 :% 02 NOP
0148 2543
0148 2544
0148 2545 END_TEST_DATA
0148
0148 ;-----
0148 2546 ENDTEST
```

ZZ-  
ECK  
V03

```
002A .SBTTL 'TEST 014 MATCH REGISTER INCREMENT WITH M CLOCK
002A 2548 :*****
002A 2549 :++
002A 2550 :
002A 2551 : FUNCTIONAL DESCRIPTION:
002A 2552 :
002A 2553 : This test checks that the Match Register will increment with M Clock
002A 2554 : when PARITY CHECK is ascerted. The following patterns are used to test
002A 2555 : the carry logic:
002A 2556 :
002A 2557 :
002A 2558 : TEST ALGORITHM:
002A 2559 :
002A 2560 : 1. Create a test loop of 14 iterations
002A 2561 : 2. Initialize the cpu
002A 2562 : 3. Clear MASTER HALT ENABLE
002A 2563 : 4. Set PARITY CHECK and uADDR INHIBIT
002A 2564 : 5. Load the test pattern into CSAMR register
002A 2565 : 6. Step the COMET clock one micro instruction
002A 2566 : 7. Test the CSBUF register to see that the test pattern incremented
002A 2567 : 8. If no error continue
002A 2568 : 9. Clear PARITY CHECK and uADDR INHIBIT
002A 2569 : 10. Loop to step 2 for remaining test patterns
002A 2570 :
002A 2571 :
002A 2572 : TEST PATTERNS
002A 2573 :
002A 2574 : 0001
002A 2575 : 0003
002A 2576 : 0007
002A 2577 : 000F
002A 2578 : 001F
002A 2579 : 003F
002A 2580 : 007F
002A 2581 : 00FF
002A 2582 : 01FF
002A 2583 : 03FF
002A 2584 : 07FF
002A 2585 : 0FFF
002A 2586 : 1FFF
002A 2587 : 3FFF
002A 2588 : 7FFF
002A 2589 : FFFF
002A 2590 :
002A 2591 :
002A 2592 : LOGIC DESCRIPTION:
002A 2593 :
002A 2594 : This test checks the abiltiy of the CSAMR register to count
002A 2595 : when PARITY CHECK is enabled. Test patterns are loaded into
002A 2596 : CSAMR, the clock is ticked, and the resulting pattern read
002A 2597 : back to the 8085 via the CSBUF register. Any failures in this
002A 2598 : test should be in the gating of M clock to the CSAMR register.
002A 2599 :
002A 2600 :
002A 2601 : ASSUMPTIONS:
002A 2602 :
002A 2603 : This test assumes that the COMET CS ADDRESS BUS and M clock
```

```
002A 2604 : are operational.
002A 2605 :
002A 2606 :
002A 2607 : ERROR DESCRIPTION:
002A 2608 :
002A 2609 : EXPECTED CSBUF CONTENTS
002A 2610 : RECEIVED CSBUF CONTENTS
002A 2611 : LOOP COUNT
002A 2612 :
002A 2613 : --
002A 2614 : *****
002A 2615 :
002A 2616 :
002A 2617 :
002A 2618 : LOOP I,1,14 : Setup the pattern select loop
0031 2619 : ERRLOOP : Error loop point
0033 2620 : INITIALIZE : Clearout any Parity Errors
0035 2621 : LOADREG RDCTRL,7$,,B : Clear MASTER HALT ENABLE
003D 2622 : LOADREG DCSCR,1$,,B : Set PARITY CHECK & uADDR INH
0045 2623 : LOADREG CSAMR,2$,I,W : Load the Test Pattern
004D 2624 : LOADREG DCSCR,3$,,B : Tick the Clock
0055 2625 : LOADREG DCSCR,1$,,B :
005D 2626 : CMPREGMSK CSBUF,4$,I,5$,,W. : Test CSBUF for correct result
0069 2627 : IFERROR <<RDM>> : Did not increment
006F 2628 : LOADREG DCSCR,6$,,B : Clear PAR CHECK & uADDR INH
0077 2629 : ENDLOOP : End test loop
007A 2630 : SKIP : Go to next test
0081 2631 :
0081 2632 : BEGIN_TEST_DATA
0081 :
0081 2633 : -----
0081 2634 1$: .BYTE CLK_CTRL_1!CLK_CTRL_0!PAR_CHK_ENABL!MICRO_ADDR_INH
007F 003F 001F 000F 0007 0003 0001 0082 2635 2$: .WORD 1,3,7,^XF,^X1F,^X3F,^X7F,^XFF,^X1FF,^X3FF,^X7FF,^XFFF
00FF 07FF 03FF 01FF 00FF 0090
009A 2636 : .WORD ^X1FFF,^X3FFF
0080 0040 0020 0010 0008 0004 0002 009E 2637 3$: .BYTE CLK_CTRL_0!PAR_CHK_ENABL!MICRO_ADDR_INH
0800 0400 0200 0100 00AD 2638 4$: .WORD 2,4,8,^XT0,^X20,^X40,^X80,^X100,^X200,^X400,^X800
0000 2000 1000 00B5 2639 : .WORD ^X1000,^X2000,0
1FFF 00BB 2640 5$: .WORD ^X1FFF
60 00BD 2641 6$: .BYTE CLK_CTRL_1!CLK_CTRL_0
00 00BE 2642 7$: .BYTE 0
00BF 2643 :
00BF 2644 : END_TEST_DATA
00BF :
00BF 2645 : -----
00BF : ENDTEST
```

```

0022 .SBTTL 'TEST 015 CS ADDRESS MATCH INCR INHIBIT
0022 2647 :*****
0022 2648 :++
0022 2649 :
0022 2650 : FUNCTIONAL DESCRIPTION
0022 2651 :
0022 2652 : This test checks to make sure that when the Control Store Address Match
0022 2653 : Registers are placed into counter mode and enabled onto the CS Bus they
0022 2654 : will stop incrementing whenever a control store parity error occurs.
0022 2655 :
0022 2656 :
0022 2657 : TEST ALGORITHM:
0022 2658 :
0022 2659 : 1. Fetch the first micro instruction
0022 2660 : 2. Set 1800 into the CSAMR register
0022 2661 : 3. Set PARITY CHECK and uADDR INHIBIT
0022 2662 : 4. Clear MASTER HALT ENABLE
0022 2663 : 5. Let the clock free run
0022 2664 : 6. Stop the clock
0022 2665 : 7. Test the CSBUF register to see that CSAMR halted on parity error
0022 2666 : 8. If no error continue
0022 2667 : 9. Clear PARITY CHECK and uADDR INHIBIT
0022 2668 :
0022 2669 :
0022 2670 : TEST PATTERNS:
0022 2671 :
0022 2672 : none
0022 2673 :
0022 2674 :
0022 2675 : LOGIC DESCRIPTION:
0022 2676 :
0022 2677 : This test checks that the CSAMR clock signal is disabled whenever
0022 2678 : a parity error is encountered. Location 1801 of DCS contains a
0022 2679 : parity error which should stop the CSAMR at 1802.
0022 2680 :
0022 2681 :
0022 2682 : ASSUMPTIONS:
0022 2683 :
0022 2684 : This test assumes the parity checking logic in COMET is operational
0022 2685 :
0022 2686 :
0022 2687 : ERROR DESCRIPTION:
0022 2688 :
0022 2689 : EXPECTED CSBUF CONTENTS
0022 2690 : RECEIVED CSBUF CONTENTS
0022 2691 :
0022 2692 : --
0022 2693 :*****
0022 2694 :
0022 2695 :
0022 2696 :
0022 2697 : INITIALIZE
0024 2698 : LDFIELD 8$ ,B,11$,78,78,....NP, : Init the cpu
0035 2699 : LOADDCS 10$ ,0,6 : Force parity error
003C 2700 : FETCH 0,INH : Load test u-code
0041 2701 : LOADREG CSAMR,7$ ,W : Start DCS program
0049 2702 : LOADREG DCSCR,1$ ,B : Load 1800 into CSAMR
: Set PARITY CHECK & uADDR INH

```

0051 2703 LOADREG RDCTRL,6\$,,B ; Clear MASTER HALT ENABLE  
0059 2704 LOADREG DCSCR,2\$,,B ; Start the clock  
0061 2705 LOADREG DCSCR,1\$,,B ; Stop the clock  
0069 2706 CMPREGMSK CSBUF,3\$,,4\$,,W, ; Test contents of CSBUF  
0075 2707 IFERROR <<RDM>> ; Incorrect value  
007B 2708 LOADREG DCSCR,5\$,,B ; Clr PARITY CHECK & uADDR INH  
0083 2709 SKIP  
008A 2710  
008A 2711  
008A 2712

BEGIN\_TEST\_DATA

008A 2713 :-----  
E8 008A 2714 1\$: .BYTE CLK\_CTRL\_0!CLK\_CTRL\_1!- ; Enable CSAMR onto CS  
008B 2715 PAR\_CHK\_ENABL!MICRO\_ADDR\_INH ; BUS with clock halted  
88 008B 2716 2\$: .BYTE PAR\_CHK\_ENABL!MICRO\_ADDR\_INH ; Let clock run  
1802 008C 2717 3\$: .WORD ^X1802 ; Test results  
3FFF 008E 2718 4\$: .WORD ^X3FFF ; Mask for CMPREGMSK pseudo-op  
60 0090 2719 5\$: .BYTE CLK\_CTRL\_0!CLK\_CTRL\_1 ; Stop the clock  
00 0091 2720 6\$: .BYTE ^X00 ; Clear MASTER HALT ENABLE  
1800 0092 2721 7\$: .WORD ^X1800 ; Address for CSAMR  
00 0094 2722 8\$: .BYTE 0  
0095 2723 10\$:  
U 00, 7700,c800,0364,0300,0470,1801 0095 2724 :% 00 NOP  
00A0 2728 11\$:  
U 01, 7700,c800,0364,0300,0470,1802 00A0 2729 :% 01 NOP  
U 02, 7700,4800,0364,0300,0470,1803 00AB 2733 :% 02 NOP  
U 03, 7700,c800,0364,0300,0470,1804 00B6 2737 :% 03 NOP  
U 04, 7700,4800,0364,0300,0470,1805 00C1 2741 :% 04 NOP  
U 05, 7700,4800,0364,0300,0470,1806 00CC 2745 :% 05 NOP  
00D7 2749  
00D7 2750  
00D7 2751  
00D7 2752

END\_TEST\_DATA

00D7 2753 :-----  
ENDTEST

```

0014 .SBTTL 'TEST 016 TEST TRPOFF BIT
0014 2755 :*****
0014 2756 :++
0014 2757 :
0014 2758 : FUNCTIONAL DESCRIPTION:
0014 2759 :
0014 2760 : This test verifies that the TRPOFF bit in the MAIN32 register is
0014 2761 : operational.
0014 2762 :
0014 2763 : TEST ALGORITHM:
0014 2764 :
0014 2765 : 1. Create a test loop of 10
0014 2766 : 2. Load the MAIN32 register with not the TRPOFF bit
0014 2767 : 3. Perform step 2 ten times
0014 2768 : 4. Test STATUS register to see that TRPOFF bit is clear
0014 2769 : 5. Create a test loop of 10
0014 2770 : 6. Set the TRPOFF bit in the MAIN32 register
0014 2771 : 7. Perform step 6 ten times
0014 2772 : 8. Test STATUS register to see that TRPOFF bit is set
0014 2773 : 9. If no error go to next test
0014 2774 :
0014 2775 :
0014 2776 : TEST PATTERNS:
0014 2777 :
0014 2778 : none
0014 2779 :
0014 2780 :
0014 2781 : LOGIC DESCRIPTION:
0014 2782 :
0014 2783 : This test clears and sets the TRPOFF bit in the MAIN32 register 10
0014 2784 : times. This is done to allow you to scope out the 8085 trap logic
0014 2785 : should you suspect errors. Otherwise the only indication of failure
0014 2786 : you will get with this test is if the STATUS register is unable to read
0014 2787 : the TRPOFF bit back.
0014 2788 :
0014 2789 :
0014 2790 : ASSUMPTIONS:
0014 2791 :
0014 2792 : This test assumes the power up self test has run successfully
0014 2793 :
0014 2794 :
0014 2795 : ERROR DESCRIPTION:
0014 2796 :
0014 2797 : EXPECTED STATUS REGISTER CONTENTS
0014 2798 : RECEIVED STATUS REGISTER CONTENTS
0014 2800 :
0014 2801 :-
0014 2802 :*****
0014 2803 :
0014 2804 :
0014 2805 : INITIALIZE : Init the cpu
0016 2806 : ERRLOOP : Error loop point
0018 2807 : LOOP I,1,10 : Loop of 10 iterations
001F 2808 : LOADREG MAIN32,3$,,B : Clear TRPOFF bit
0027 2809 : ENDLOOP I : End loop
002A 2810 : CMPREGMSK STATUS,4$,,2$,,B. : Test for TRPOFF clear

```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 016 TEST TRPOFF BIT  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 016 TEST TRPOFF BIT

I 6  
6-MAY-1982

Fiche 1 Frame 16 Sequence 73  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46 Page 7  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1 (1

```
0036 2811 IFERROR      ; <<RDM>>      ; Bit not clear
003C 2812 LOOP          ; i,1,10      ; Loop of 10 iterations
0043 2813 LOADREG       ; MAIN32,1$,B ; Set TRPOFF bit
004B 2814 ENDLOOP      ; I           ; End loop
004E 2815 CMPREGMSK    ; STATUS,2$,2$,B ; Test for TRPOFF set
005A 2816 IFERROR      ; <<RDM>>      ; Bit not set
0060 2817 SKIP         ;             ; Go to next test
0067 2818
0067 2819
0067 2820 BEGIN_TEST_DATA
0067 2821 ;-----
05 0067 2822 1$: .BYTE MAINT_TRAP_OFF!MAINT_DCS_ENABL
02 0068 2823 2$: .BYTE TRAP_OFF
04 0069 2824 3$: .BYTE MAINT_DCS_ENABL
00 006A 2825 4$: .BYTE ^X0
006B 2826
006B 2827 END_TEST_DATA
006B 2828 ;-----
006B 2828 ENDTEST
```

```

0014 .SBTTL 'TEST 017 TEST STBINH BIT
0014 2830 :*****
0014 2831 :++
0014 2832 :
0014 2833 : FUNCTIONAL DESCRIPTION:
0014 2834 :
0014 2835 :     This test checks the ability of the STBINH bit to block loading of
0014 2836 :     the DCS CONTROL FILE.
0014 2837 :
0014 2838 :
0014 2839 : TEST ALGORITHM:
0014 2840 :
0014 2841 :     1. Create a loop of 2 iterations
0014 2842 :     2. Load CONTROL FILE location 0 with 1's and location 1 with 0's
0014 2843 :     3. Load the DCS ADDRESS register with address 0
0014 2844 :     4. Set the STBINH bit in the MAIN32 register
0014 2845 :     5. Load the DCS ADDRESS register with address 1
0014 2846 :     6. Test the control file for all 1's
0014 2847 :     7. If no error continue
0014 2848 :     8. Clear the STBINH bit in the MAIN32 register
0014 2849 :     9. Load the DCS ADDRESS register with address 1
0014 2850 :    10. Test the control file for all 0's
0014 2851 :    11. If no error go to next test
0014 2852 :
0014 2853 :
0014 2854 : TEST PATTERNS:
0014 2855 :
0014 2856 :     DCS ADDRESS      CONTROL FILE
0014 2857 :         0              FF
0014 2858 :         1              00
0014 2859 :
0014 2860 :
0014 2861 : LOGIC DESCRIPTION:
0014 2862 :
0014 2863 :     This test ensures that the STBINH bit of MAIN32 blocks loading of
0014 2864 :     the DCS CONTROL FILE. DCS addresses are loaded into the address
0014 2865 :     register twice to allow the control file latch to gate the proper
0014 2866 :     data.
0014 2867 :
0014 2868 :
0014 2869 : ASSUMPTIONS:
0014 2870 :
0014 2871 :     This test assumes the power up self test has run successfully
0014 2872 :
0014 2873 :
0014 2874 : ERROR DESCRIPTION:
0014 2875 :
0014 2876 :     EXPECTED DCS CONTROL FILE DATA
0014 2877 :     RECEIVED DCS CONTROL FILE DATA
0014 2878 :
0014 2879 :
0014 2880 :*****
0014 2881 :
0014 2882 :
0014 2883 :     INITIALIZE          : Init the cpu
0016 2884 :     ERRLOOP             : Error loop
0018 2885 :     LOOP                1,1,2 : Test loop of 2

```

```
001F 2886      LOADREG      DCSAD,1$,I,B      ; Load control file address
0027 2887      LOADREG      DCSDA,2$,I,B      ; Load control file data
002F 2888      ENDLOOP      I                ; End test loop
0032 2889      LOADREG      DCSAD,1$,,B      ; Load address 0
003A 2890      LOADREG      DCSAD,1$,,B      ;
0042 2891      LOADREG      MAIN32,3$,,B     ; Set STBINH bit
004A 2892      LOADREG      DCSAD,4$,,B      ; Load address 1
0052 2893      LOADREG      DCSAD,4$,,B      ;
005A 2894      CMPREG      DCSCF,2$,,B      ; Test control file for 1's
0063 2895      IFERROR      ,,<<RDM>>      ; Incorrect data
0069 2896      LOADREG      MAIN32,5$,,B     ; Clear STBINH bit
0071 2897      LOADREG      DCSAD,4$,,B      ; Load address 1
0079 2898      CMPREG      DCSCF,1$,,B      ; Test control file for 0's
0082 2899      IFERROR      ,,<<RDM>>      ; Incorrect data
0088 2900      SKIP          ; Go to next test
008F 2901
008F 2902
008F 2903      BEGIN_TEST_DATA
008F
008F
008F 2904      ;-----
00 008F 2905 1$:      .BYTE      ^X0
01 0090 2906      .BYTE      ^X1
FF 0091 2907 2$:      .BYTE      ^XFF
00 0092 2908      .BYTE      ^X00
07 0093 2909 3$:      .BYTE      MAINT_STB_INH!MAINT_TRAP_OFF!MAINT_DCS_ENABL
01 0094 2910 4$:      .BYTE      ^X1
05 0095 2911 5$:      .BYTE      MAINT_TRAP_OFF!MAINT_DCS_ENABL
0096 2912
0096 2913      END_TEST_DATA
0096
0096
0096 2914      ;-----
0096 2915      .SBTTL      ""
0096 2916      .SBTTL      "DCS CONTROL FILE FUNCTIONAL TESTS"
0096 2917      .SBTTL      ""
0096 2918      ENDTEST
```

```

0013 .SBTTL 'TEST 018 STOP CLOCK BIT
0013 2920 :*****
0013 2921 :++
0013 2922 :
0013 2923 : FUNCTIONAL DESCRIPTION:
0013 2924 :
0013 2925 : This test checks the ability of the clock stop bit in the DCS
0013 2926 : CONTROL FILE to stop the COMET clock.
0013 2927 :
0013 2928 :
0013 2929 : TEST ALGORITHM:
0013 2930 :
0013 2931 : 1. Load the test micro code into DCS
0013 2932 : 2. Strobe MASTER HALT ENABLE to clear the clock control logic
0013 2933 : 3. Fetch the first micro instruction
0013 2934 : 4. Let the clock free run
0013 2935 : 5. Test the CLKDCS register for clock stopped at DCS address 03
0013 2936 : 6. Turn clock off
0013 2937 : 7. If no error go to next test
0013 2938 :
0013 2939 :
0013 2940 : TEST PATTERNS:
0013 2941 :
0013 2942 : none
0013 2943 :
0013 2944 :
0013 2945 : LOGIC DESCRIPTION:
0013 2946 :
0013 2947 : This test checks the ability of the CLOCK STOP bit in the DCS
0013 2948 : CONTROL FILE to stop the COMET clock. The DCS ADDRESS register
0013 2949 : should halt with address 03 in it.
0013 2950 :
0013 2951 :
0013 2952 : ASSUMPTIONS:
0013 2953 :
0013 2954 : This test assumes that COMET can run NOP instructions
0013 2955 :
0013 2956 :
0013 2957 : ERROR DESCRIPTION:
0013 2958 :
0013 2959 : EXPECTED CLKDCS REGISTER CONTENTS
0013 2960 : RECEIVED CLKDCS REGISTER CONTENTS
0013 2961 :
0013 2962 : --
0013 2963 :*****
0013 2964 :
0013 2965 :
0013 2966 :
0013 2967 : INITIALIZE : Init the cpu
0015 2968 : LOADDCS 1$,,0,3 : Load the micro words
001C 2969 : ERRLOOP : Error loop point
001E 2970 : LOADREG RDCTRL,2$,,B : Clear MASTER HALT ENABLE
0026 2971 : LOADREG RDCTRL,5$,,B : Set MASTER HALT ENABLE
002E 2972 : FETCH 0 : Fetch the first instruction
0033 2973 : LOADREG DCSCR,2$,,B : Start the clock
003B 2974 : CMPREG CLKDCS,3$,,B : Check for clock stopped and
0044 2975 : : correct address

```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 018 STOP CLOCK BIT  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 018 STOP CLOCK BIT

M 6  
6-MAY-1982

Fiche 1 Frame M6

Sequence 77

6-MAY-1982 19:54:55 VAX-11 Macro V02.46 Page 7  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1 (1

```
0044 2976 LOADREG DCSCR,4$,,B ; Stop the clock
004C 2977 IFERROR ,,<<RDM>> ; Check if error
0052 2978 SKIP ; Go to next test
0059 2979
0059 2980 BEGIN_TEST_DATA
0059
0059 ;-----
0059 2981
0059 2982 1$:
U 00, 7700,C800,0364,0300,0470,1801 0059 2983 :% 00 NOP ; Micro code
U 01, 7700,C800,0364,0300,0470,1802 0064 2987 :% 01 NOP
U 02, 7300,4800,0364,0300,0470,1803 006F 2991 :% 02 NOP,STOP.CLOCK
007A 2995
00 007A 2996 2$: .BYTE 0
C3 007B 2997 3$: .BYTE 3 ! CLK_CTRL_1_RO!CLK_CTRL_0_RO
60 007C 2998 4$: .BYTE CLK_CTRL_1!CLK_CTRL_0
08 007D 2999 5$: .BYTE MASTER_HALT_EN
007E 3000
007E 3001 END_TEST_DATA
007E
007E ;-----
007E 3002 ENDTEST
```

```
0021 .SBTTL 'TEST 019 STROBE H REGISTER WITH W BUS
0021 3004 :*****
0021 3005 :++
0021 3006 :
0021 3007 : FUNCTIONAL DESCRIPTION:
0021 3008 :
0021 3009 : This test checks the ability of the WH register to latch data
0021 3010 : coming from the COMET LONGLIT register via the WBUS.
0021 3011 :
0021 3012 :
0021 3013 : TEST ALGORITHM:
0021 3014 :
0021 3015 : 1. Create a test loop of 6 iterations
0021 3016 : 2. Generate a test pattern and put in LONGLIT field of test micro code
0021 3017 : 3. Load micro code into DCS
0021 3018 : 4. Fetch location 0 of DCS
0021 3019 : 5. Let the clock free run
0021 3020 : 6. Stop the clock
0021 3021 : 7. Test the contents of the WH register
0021 3022 : 8. If no error go to step 2 for remaining test patterns
0021 3023 :
0021 3024 :
0021 3025 : TEST PATTERNS:
0021 3026 :
0021 3027 : AAAAAAAA
0021 3028 : 55555555
0021 3029 : 33333333
0021 3030 : 0F0F0F0F
0021 3031 : 00FF00FF
0021 3032 : 0000FFFF
0021 3033 :
0021 3034 :
0021 3035 : LOGIC DESCRIPTION:
0021 3036 :
0021 3037 : This test checks the ability of the WH register to latch data from
0021 3038 : WBUS by setting the HRWBUS bit in the CONTROL FILE. This should
0021 3039 : cause the WH register to load at DCS ADDRESS 01 with the data contained
0021 3040 : in the LONGLIT register.
0021 3041 :
0021 3042 :
0021 3043 : ASSUMPTIONS:
0021 3044 :
0021 3045 : This test assumes that the DPM module in COMET is in fairly good shape.
0021 3046 :
0021 3047 :
0021 3048 : ERROR DESCRIPTION:
0021 3049 :
0021 3050 : EXPECTED WH REGISTER CONTENTS
0021 3051 : RECEIVED WH REGISTER CONTENTS
0021 3052 : LOOP COUNT
0021 3053 :
0021 3054 : --
0021 3055 : *****
0021 3056 :
0021 3057 :
0021 3058 :
0021 3059 : INITIALIZE ; Init the cpu
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 019 STROBE H REGISTER WITH W BUS  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 019 STROBE H REGISTER WITH W BUS

B 7  
6-MAY-1982

Fiche 1 Frame B7

Sequence 79

6-MAY-1982 19:54:55  
6-MAY-1982 19:54:39

VAX-11 Macro V02.46  
WRKDS0:[COMET.RDM]ECKAF.MAR;1

Page 7  
(1

ZZ-1  
ECKAF  
V03

```
0023 3060 LOOP I,1,6 ; Check 6 patterns
002A 3061 SA01PAT I,1$,LONLIT,2$ ; Test pattern to LONLIT
0032 3062 LOADDCS 1$,,0,3 ; Load micro instructions
0039 3063 ERRLOOP ; Error loop point
003B 3064 FETCH 0 ; Fetch location 0
0040 3065 LOADREG DCSCR,3$,,B ; Start the clock
0048 3066 LOADREG DCSCR,5$,,B ; Stop the clock
0050 3067 CMPREG WHREG,2$,,L ; Check received data
0059 3068 IFERROR ; See if error
005F 3069 ENDL00P ; End test loop
0062 3070 SKIP ; Go to next test
0069 3071
0069 3072 BEGIN_TEST_DATA
0069 ;-----
0069 3073 ;
0069 3074 1$: ; Micro code
0069 3075 :% 00 LONLIT [0]
U 00, 7700,7800,7FFF,FFFF,8470,1801 0074 3079 :% 01 RDM_LONLIT.Q_M
U 01, 6700,8800,0047,03D4,0470,1802 007F 3083 :% 02 STOP.CLOCK
U 02, 7300,4800,0364,0300,0470,1803 008A 3087
00000000 008A 3088 2$: .LONG 0 ; Test pattern storage
00 008E 3089 3$: .BYTE 0
60 008F 3090 5$: .BYTE CLK_CTRL_0!CLK_CTRL_1
0090 3091
0090 3092 END_TEST_DATA
0090 ;-----
0090 3093 ENDTEST
```

```

0013 .SBTTL 'TEST 01A W BUS FROM RDM
0013 3095 :*****
0013 3096 :++
0013 3097 :
0013 3098 : FUNCTIONAL DESCRIPTION:
0013 3099 :
0013 3100 : This test writes 6 data patterns from the D register into R[0]
0013 3101 : and reads R[0] back into the H register.
0013 3102 :
0013 3103 :
0013 3104 : TEST ALGORITHM:
0013 3105 :
0013 3106 : 1. Load the micro code into DCS
0013 3107 : 2. Create a test loop of 6 iterations
0013 3108 : 3. Generate a test pattern
0013 3109 : 4. Strobe MASTER HALT ENABLE to clear the clock control logic
0013 3110 : 5. Load test pattern into WD register
0013 3111 : 6. Run the micro instructions in DCS
0013 3112 : 7. Test the WH register for the test pattern
0013 3113 : 8. If no error go to step 3 for remaining test patterns
0013 3114 :
0013 3115 :
0013 3116 : TEST PATTERNS:
0013 3117 :
0013 3118 : AAAAAAAA
0013 3119 : 55555555
0013 3120 : 33333333
0013 3121 : 0F0F0F0F
0013 3122 : 00FF00FF
0013 3123 : 0000FFFF
0013 3124 :
0013 3125 :
0013 3126 : LOGIC DESCRIPTION:
0013 3127 :
0013 3128 : This test checks the ability of the WD register to load data onto
0013 3129 : the WBUS when the WBUSDR CONTROL FILE bit is set. The results
0013 3130 : are then read back via the WH register.
0013 3131 :
0013 3132 :
0013 3133 : ASSUMPTIONS:
0013 3134 :
0013 3135 : This test assumes that the COMET DPM module is operational.
0013 3136 :
0013 3137 :
0013 3138 : ERROR DESCRIPTION:
0013 3139 :
0013 3140 : EXPECTED WH REGISTER CONTENTS
0013 3141 : RECEIVED WH REGISTER CONTENTS
0013 3142 : LOOP COUNT
0013 3143 : --
0013 3144 : *****
0013 3145 :
0013 3146 :
0013 3147 :
0013 3148 : INITIALIZE ; Init the cpu
0015 3149 : LOADDCS ; Load the micro instructions
001C 3150 : LOOP I$,0,4 ; Create a test loop of 6
I,1,6

```

```
0023 3151 SA01PAT I,2$ ; Generate the test pattern
002B 3152 ERRLOOP ; Error loop point
002D 3153 LOADREG RDCTRL,3$,,B ; Clear MASTER HALT ENABLE
0035 3154 LOADREG RDCTRL,4$,,B ; Set MASTER HALT ENABLE
003D 3155 LOADREG WDREG,2$,,L ; Test pattern to WD register
0045 3156 FETCH 0 ; Start execution
004A 3157 LOADREG DCSCR,3$,,B ; Start the clock
0052 3158 LOADREG DCSCR,5$,,B ; Stop the clock
005A 3159 CMPREG WHREG,2$,,L ; Check the received data
0063 3160 IFERROR ,,<<RDM>> ; See if failure
0069 3161 ENDLOOP I ; End test loop
006C 3162 SKIP ; Go to next test
0073 3163
0073 3164
```

BEGIN\_TEST\_DATA

```
0073 ;-----
0073 3165 ;
0073 3166 1$: ; Micro code
0073 3167 :% 00 NOP
007E 3171 :% 01 R[0] RDM
0089 3175 :% 02 RDM WB,WB R[0]
0094 3179 :% 03 STOP.CLOCK
009F 3183
009F 3184 2$: .LONG 0
00A3 3185 3$: .BYTE 0
00A4 3186 4$: .BYTE MASTER HALT EN
00A5 3187 5$: .BYTE CLK_CTRL_1!CLK_CTRL_0
00A6 3188
00A6 3189 END_TEST_DATA
00A6 ;-----
00A6 3190 ENDTEST
```

U 00. 7700,c800,0364,0300,0470,1801  
U 01. 5700,8840,0364,0300,0470,1802  
U 02. 6700,4800,5BE4,0300,0470,1803  
U 03. 7300,c800,0364,0300,0470,1804

00000000  
00  
08  
60

ZZ-  
ECKA  
V03  
  
U 0  
U 0  
  
U 0  
U 0  
U 0  
U 0

```

0020 .SBTTL 'TEST 01B DISABLE OF WH REG FROM WBUS
0020 3192 :*****
0020 3193 :++
0020 3194 :
0020 3195 : FUNCTIONAL DESCRIPTION:
0020 3196 :
0020 3197 :     This test checks that the WH register is not loaded by the bit in
0020 3198 :     the control file if the DCS is not selected.
0020 3199 :
0020 3200 :
0020 3201 : TEST ALGORITHM:
0020 3202 :
0020 3203 :     1. Load micro code into DCS
0020 3204 :     2. Load test pattern into WD register
0020 3205 :     3. Enable the WD register onto the WBUS
0020 3206 :     4. Latch WH register with data on WBUS
0020 3207 :     5. Fetch DCS ADDRESS 0
0020 3208 :     6. Tick the clock twice
0020 3209 :     7. Test contents of WH register to see if it didn't change
0020 3210 :     8. If no error go to next test
0020 3211 :
0020 3212 :
0020 3213 : TEST PATTERNS:
0020 3214 :
0020 3215 :     12345678
0020 3216 :
0020 3217 :
0020 3218 : LOGIC DESCRIPTION:
0020 3219 :
0020 3220 :     This test loads an initial test pattern into the WH register. Micro
0020 3221 :     code is then executed that forces the CS ADDRESS out of DCS space.
0020 3222 :     The WH register is then tested to ensure it did not strobe when the
0020 3223 :     CS ADDRESS was out of range.
0020 3224 :
0020 3225 :
0020 3226 : ASSUMPTIONS:
0020 3227 :
0020 3228 :     This test assumes that the COMET DPM module is operational
0020 3229 :
0020 3230 :
0020 3231 : ERROR DESCRIPTION:
0020 3232 :
0020 3233 :     EXPECTED WH REGISTER CONTENTS
0020 3234 :     RECEIVED WH REGISTER CONTENTS
0020 3235 :
0020 3236 : --
0020 3237 : *****
0020 3238 :
0020 3239 :
0020 3240 :
0020 3241 : INITIALIZE
0022 3242 : LOADDCS 1$,0,3 : Init the cpu
0029 3243 : LOADREG WDREG,2$,,L : Load the Micro words
0031 3244 : LOADREG MAIN32,6$,,B : Load initial test pattern
0039 3245 : LOADREG MAIN32,7$,,B : Enable WD reg onto the WBUS
0041 3246 : LOADREG MAIN32,6$,,B : Strobe the WH reg
0049 3247 : LOADREG MAIN32,8$,,B : Disable WD reg from the WBUS

```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 01B DISABLE OF WH REG FROM WBUS  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 01B DISABLE OF WH REG FROM WBUS

F 7  
6-MAY-1982

Fiche 1 Frame F7

Sequence 83

6-MAY-1982 19:54:55

VAX-11 Macro V02.46

Page 8

6-MAY-1982 19:54:39

WRKDS0:[COMET.RDM]ECKAF.MAR;1

(1

```
0051 3248      FETCH      0      ; Start execution
0056 3249      LOOP      I,1,2    ; Loop to tick the clock
005D 3250      LOADREG   DCSCR,3$,,B ; Tick the clock
0065 3251      LOADREG   DCSCR,4$,,B ;
006D 3252      ENDLOOP   I      ; End loop
0070 3253      CMPREG    WHREG,2$,,L ; Test the WH register contents
0079 3254      IFERROR   ,,<<RDM>> ; Incorrect data
007F 3255      SKIP      ; Go to next test
0086 3256
0086 3257      BEGIN_TEST_DATA
0086
0086 -----
0086 3258
0086 3259 1$:
U 00, 7700,7800,7FFF,FFFF,8470,1801 0086 3260 :% 00 LONLIT [0] ; Micro code
U 01, 6700,0800,0047,03D4,0470,0000 0091 3264 :% 01 RDM_LONLIT.Q_M,NEXT/0 ; Init the LONLIT register
U 02, 7700,4800,0364,0300,0470,1803 009C 3268 :% 02 NOP ; Try to read the LONLIT reg
00A7 3272
12345678 00A7 3273 2$: .LONG ^X12345678
20 00AB 3274 3$: .BYTE CLK_CTRL_0
40 00AC 3275 4$: .BYTE CLK_CTRL_1!CLK_CTRL_1
15 00AD 3276 6$: .BYTE MAINT_WD_TO_WB!MAINT_TRAP_OFF!MAINT_DCS_ENABL
1D 00AE 3277 7$: .BYTE MAINT_WD_TO_WB!MAINT_WB_TO_WH!MAINT_TRAP_OFF!MAINT_DCS_ENABL
05 00AF 3278 8$: .BYTE MAINT_TRAP_OFF!MAINT_DCS_ENABL
00B0 3279
00B0 3280      END_TEST_DATA
00B0
00B0 -----
00B0 3281      ENDTEST
```

ZZ-  
ECK  
V03

```

001A .SBTTL 'TEST 01C CLEAR DCS ADDRESS BIT
001A 3283 :*****
001A 3284 :++
001A 3285 :
001A 3286 : FUNCTIONAL DESCRIPTION:
001A 3287 :
001A 3288 :     This test checks that the CLEAR DCS ADDR bit in the control file
001A 3289 :     functions correctly.
001A 3290 :
001A 3291 :
001A 3292 : TEST ALGORITHM:
001A 3293 :
001A 3294 :     1. Load micro code into DCS
001A 3295 :     2. Fetch location 1 of DCS
001A 3296 :     3. Tick the clock one micro instruction
001A 3297 :     4. Test the CLKDCS register for an address of 0
001A 3298 :     5. If no error go to next test
001A 3299 :
001A 3300 :
001A 3301 : TEST PATTERNS:
001A 3302 :
001A 3303 :     none
001A 3304 :
001A 3305 :
001A 3306 : LOGIC DESCRIPTION:
001A 3307 :
001A 3308 :     This test ensures that the DCS ADDRESS register is forced back
001A 3309 :     to address 0 when the DCSACL bit is encountered in the DCS CONTROL
001A 3310 :     FILE.
001A 3311 :
001A 3312 :
001A 3313 : ASSUMPTIONS:
001A 3314 :
001A 3315 :     This test assumes the power-up self test has run successfully
001A 3316 :
001A 3317 :
001A 3318 : ERROR DESCRIPTION:
001A 3319 :
001A 3320 :     EXPECTED CLKDCS REGISTER CONTENTS
001A 3321 :     RECEIVED CLKDCS REGISTER CONTENTS
001A 3322 :
001A 3323 : --
001A 3324 : *****
001A 3325 :
001A 3326 :
001A 3327 :
001A 3328 :     INITIALIZE
001C 3329 :     LOADDCS          1$,0.4
0023 3330 :     ERRLOOP
0025 3331 :     FETCH           1
002A 3332 :     LOADREG        DCSCR,3$,B
0032 3333 :     LOADREG        DCSCR,4$,B
003A 3334 :     CMPREG         CLKDCS,2$,B
0043 3335 :     IFERROR        ..,<<RDM>>
0049 3336 :     SKIP
0050 3337 :
0050 3338 : BEGIN_TEST_DATA
: Init the cpu
: Load the DCS
: Error loop point
: Start execution
: Tick the clock
:
: Test DCS ADDRESS register
: Incorrect data
: Go to next test

```

```
0050  
0050 ;-----  
0050 3339  
0050 3340 1$: ; Micro code  
U 00, 7700,c800,0364,0300,0470,1801 0050 3341 :% 00 NOP  
U 01, 7700,c800,0364,0300,0470,1802 005B 3345 :% C1 NOP  
U 02, 7500,4800,0364,0300,0470,1803 0066 3349 :% 02 DCS.ADDR_0  
U 03, 7700,c800,0364,0300,0470,1804 0071 3353 :% 03 NOP  
007C 3357  
C0 007C 3358 2$: .BYTE CLK_CTRL_1_RO!CLK_CTRL_0_RO  
20 007D 3359 3$: .BYTE CLK_CTRL_0  
60 007E 3360 4$: .BYTE CLK_CTRL_1!CLK_CTRL_0  
007F 3361  
007F 3362 END_TEST_DATA  
007F  
007F 3363 ;-----  
ENDTEST
```

```
001E .SBTTL 'TEST 01D ENABLE TRACE REGISTER BIT
001E 3365 :*****
001E 3366 :++
001E 3367 :
001E 3368 : FUNCTIONAL DESCRIPTION:
001E 3369 :
001E 3370 : This test checks the operation of the ENTRCE bit in the DCS CONTROL
001E 3371 : FILE.
001E 3372 :
001E 3373 :
001E 3374 : TEST ALGORITHM:
001E 3375 :
001E 3376 : SUBTEST #1
001E 3377 : 1. Load the micro code to perform the test
001E 3378 : 2. Fetch location 0 of DCS
001E 3379 : 3. Step the clock one micro instruction
001E 3380 : 4. Test the CSTRP register for the correct address
001E 3381 : 5. If no error go to step 1 for the remaining test patterns
001E 3382 :
001E 3383 : SUBTEST #2
001E 3384 : 1. Load the micro code to perform the test
001E 3385 : 2. Fetch location 0 of DCS
001E 3386 : 3. Run the micro code
001E 3387 : 4. Test the CSTRP register for the correct address
001E 3388 : 5. If no error go to next test
001E 3389 :
001E 3390 :
001E 3391 : TEST PATTERNS:
001E 3392 :
001E 3393 : SUBTEST #1
001E 3394 : 1AAA
001E 3395 : 1D55
001E 3396 : 1CCC
001E 3397 : 1BC3
001E 3398 : 183F
001E 3399 :
001E 3400 : SUBTEST #2
001E 3401 : 18AA
001E 3402 :
001E 3403 :
001E 3404 : LOGIC DESCRIPTION:
001E 3405 :
001E 3406 : This test first checks that the CSTRP register is constantly being
001E 3407 : updated with the CS ADDRESS. Subtest 2 checks that the register
001E 3408 : remains unchanged when the ENTRCE bit in the DCS CONTROL FILE is set.
001E 3409 :
001E 3410 :
001E 3411 : ASSUMPTIONS:
001E 3412 :
001E 3413 : This test assumes the CS ADDRESS BUS is operational
001E 3414 :
001E 3415 :
001E 3416 : ERROR DESCRIPTION:
001E 3417 :
001E 3418 : EXPECTED CSTRP REGISTER CONTENTS
001E 3419 : RECEIVED CSTRP REGISTER CONTENTS
001E 3420 : LOOP COUNT (subtest #1)
```

ZZ-  
ECK  
V03  
  
U 0  
U 0  
  
18  
  
1F  
  
U 0



ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 01D ENABLE TRACE REGISTER BIT  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 01D ENABLE TRACE REGISTER BIT

K 7  
6-MAY-1982

6-MAY-1982 19:54:55  
6-MAY-1982 19:54:39

Fiche 1 Frame K7

Sequence 88

VAX-11 Macro V02.46 Page 8  
WRKDS0:[COMET.RDM]ECKAF.MAR;1 (1

```

        60 00D7 3490 4$: .BYTE CLK_CTRL_1!CLK_CTRL_0 ; Halt
        00D8 3491
1AAA 00D8 3492 5$: .WORD ^X1AAA ; Expected data
1D55 00DA 3493 .WORD ^X1D55
1CCC 00DC 3494 .WORD ^X1CCC
1BC3 00DE 3495 .WORD ^X1BC3
183F 00E0 3496 .WORD ^X183F
        00E2 3497
3FFF 00E2 3498 6$: .WORD ^X3FFF ; CS ADDRESS Mask
        00E4 3499
        00E4 3500 ;+
        00E4 3501 ; Subtest 2 data
        00E4 3502 ;-
        00E4 3503 10$:
U 00, 7700,4800,0364,0300,0470,18AA 00E4 3504 ;% 00 NEXT/18AA ; Data for CSTRP register
U 01, 7F00,4800,0364,0300,0470,1855 00EF 3508 ;% 01 NEXT/1855,LATCH.CS.ADDR ; Inhibit the TRAP reg strobe
U 02, 7300,4800,0364,0300,0470,1803 00FA 3512 ;% 02 STOP.CLOCK ; Stop the clock
        0105 3516
18AA 0105 3517 11$: .WORD ^X18AA ; Expected data
        0107 3518
        0107 3519 END_TEST_DATA
        0107
        0107
        0107 3520 ;-----
        ENDTEST
```

```
0014 .SBTTL 'TEST 01E VBUS STROBE BIT
0014 3522 :*****
0014 3523 :++
0014 3524 :
0014 3525 : FUNCTIONAL DESCRIPTION:
0014 3526 :
0014 3527 : This test checks that the VSTB bit in the DCS CONTROL FILE operates
0014 3528 : correctly.
0014 3529 :
0014 3530 :
0014 3531 : TEST ALGORITHM:
0014 3532 :
0014 3533 : SUBTEST #1
0014 3534 : 1. Load good parity into location 0 of DCS so cpu will init ok
0014 3535 : 2. Load bad parity into location 0 of DCS for test
0014 3536 : 3. Fill the VBUS with 0's
0014 3537 : 4. Fetch location 0 of DCS and step the clock one micro instruction
0014 3538 : 5. Test the VBUS for the parity error bit
0014 3539 : 6. If no error go to subtest 2
0014 3540 :
0014 3541 : SUBTEST #2
0014 3542 : 1. Load good parity into location 0 of DCS so cpu will init ok
0014 3543 : 2. Load bad parity into location 0 of DCS for test
0014 3544 : 3. Fill the VBUS with 0's
0014 3545 : 4. Fetch location 0 of DCS and step the clock one micro instruction
0014 3546 : 5. Test the VBUS for parity bit not set
0014 3547 : 6. If no error go to next test
0014 3548 :
0014 3549 :
0014 3550 : TEST PATTERNS:
0014 3551 :
0014 3552 : none
0014 3553 :
0014 3554 :
0014 3555 : LOGIC DESCRIPTION:
0014 3556 :
0014 3557 : This test ensures that the VSTB bit in the DCS CONTROL FILE loads the
0014 3558 : VBUS at the proper time. Subtest 1 forces a parity error with the
0014 3559 : bit set; thus causing the VBUS to load. Subtest 2 is the same as 1
0014 3560 : except the VSTB bit is not set in the test micro instruction.
0014 3561 : Therefore the VBUS should not load the parity error bit.
0014 3562 :
0014 3563 :
0014 3564 : ASSUMPTIONS:
0014 3565 :
0014 3566 : This test assumes that the VBUS and parity checking logic within
0014 3567 : COMET are operational.
0014 3568 :
0014 3569 :
0014 3570 : ERROR DESCRIPTION:
0014 3571 :
0014 3572 : EXPECTED VBUS RESULTS
0014 3573 : RECEIVED VBUS RESULTS
0014 3574 :
0014 3575 : bit <8> contains the value
0014 3576 : bits <7:0> contain the bit position
0014 3577 :
```

```
0014 3578 :--
0014 3579 :*****
0014 3580 :////////////////////////////////////
0014 3581 :+
0014 3582 :SUBTEST #1
0014 3583 :
0014 3584 :     Test VBUS load with VSTB bit set
0014 3585 :--
0014 3586 :     SUBTEST                               : SUBTEST #1
0017 :
0017 :////////////////////////////////////
0017 3587 :     ERRLOOP                               : Error loop point
0019 3588 :     LOADDCS 12$,0,1                       : Put good parity in location 0
0020 3589 :     INITIALIZE                             : Init the cpu
0022 3590 :     LDFIELD 13$,B,1$,78,78,....NP,       : Force parity error
0033 3591 :     LOADDCS 1$,0,1                         : Put bad parity in location 0
003A 3592 :     LOOP I,1,64                            : Create loop of 64
0041 3593 :     LOADREG CSAMR,2$,W                     : Fill the VBUS Bus with zero's
0049 3594 :     LOADREG CSAMR,3$,W
0051 3595 :     ENDLOOP I                               : End of loop
0054 3596 :     FETCH 0                                : Fetch the micro word
0059 3597 :     LOADREG DCSCR,4$,B                     : Tick the clock one
0061 3598 :     LOADREG DCSCR,5$,B                     : micro instruction
0069 3599 :     CMPVBUS 6$                             : Check for Parity Error
006E 3600 :     IFERROR ..<<RDM>>                     : VBUS incorrect
0074 3601 :
0074 3602 :
0074 3603 :
0074 3604 :////////////////////////////////////
0074 3605 :+
0074 3606 :SUBTEST #2
0074 3607 :
0074 3608 :     Test VBUS does not load with VSTB bit not set
0074 3609 :--
0074 3610 :     SUBTEST                               : SUBTEST #2
0077 :
0077 :////////////////////////////////////
0077 3611 :     ERRLOOP                               : Error loop point
0079 3612 :     LOADDCS 12$,0,1                       : Put good parity in location 0
0080 3613 :     INITIALIZE                             : Init the cpu
0082 3614 :     LDFIELD 13$,B,10$,78,78,....NP,      : Force parity error
0093 3615 :     LOADDCS 10$,0,1                       : Put bad parity in location 0
009A 3616 :     LOOP I,1,64                            : Create loop of 64
00A1 3617 :     LOADREG CSAMR,2$,W                     : Fill VBUS with 0's
00A9 3618 :     LOADREG CSAMR,3$,W
00B1 3619 :     ENDLOOP I                               : End loop
00B4 3620 :     FETCH 0                                : Start execution
00B9 3621 :     LOADREG DCSCR,4$,B                     : Tick the clock one
00C1 3622 :     LOADREG DCSCR,5$,B                     : micro instruction
00C9 3623 :     CMPVBUS 11$                            : Check for no parity error
00CE 3624 :     IFERROR ..<<RDM>>                     : VBUS incorrect
00D4 3625 :     SKIP                                     : Go to next test
00DB 3626 :
00DB 3627 :     BEGIN_TEST_DATA
00DB :
00DB :-----
00DB 3628 :
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 01E VBUS STROBE BIT  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 01E VBUS STROBE BIT

N 7  
6-MAY-1982

Fiche 1 Frame N7  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1  
Sequence 91  
Page 8 (1)

```
U 00, 7600,C800,0364,0300,0470,1801 00DB 3629 1$:  
00DB 3630 :% 00 NOP,STROBE.V.BUS ; Reverse a parity field  
00E6 3634 12$:  
U 01, 7700,C800,0364,0300,0470,1802 00E6 3635 :% 01 NOP  
00F1 3639  
C000 00F1 3640 2$: .WORD ^XC000 ; VBUS_SERIAL_IN!VBUS_CLOCK  
4000 00F3 3641 3$: .WORD ^X4000 ; VBUS_SERAIL_IN  
20 00F5 3642 4$: .BYTE CLK_CTRL_0  
60 00F6 3643 5$: .BYTE CLK_CTRL_1!CLK_CTRL_0  
01 00F7 3644 6$: .BYTE 1  
00F8 3645 VBUSDATA <^X1D>,HIGH ; CS Parity Error H  
00F9 3646  
00F9 3647 :+  
00F9 3648 : Subtest 2 data  
00F9 3649 :-  
U 00, 7700,C800,0364,0300,0470,1801 00F9 3650 10$:  
01 0104 3651 :% 00 NOP  
0105 3656 11$: .BYTE 1  
00 0106 3657 13$: VBUSDATA <^X1D>,LOW ; CS Parity Error H  
0107 3658 .BYTE 0  
0107 3659 END_TEST_DATA  
0107  
0107 :-----  
0107 3660  
0107 3661 .SBTTL ""  
0107 3662 .SBTTL "STOP CLOCK FUNCTIONS"  
0107 3663 .SBTTL ""  
0107 3664 ENDTEST
```

ZZ-  
ECI  
VO:

```
002D .SBTTL 'TEST 01F CLOCK STOP ON CONTROL STORE PARITY ERROR
002D 3666 :*****
002D 3667 :++
002D 3668 :
002D 3669 : FUNCTIONAL DESCRIPTION:
002D 3670 :
002D 3671 : This test checks that the clock control logic responds correctly to
002D 3672 : parity errors.
002D 3673 :
002D 3674 :
002D 3675 : TEST ALGORITHM:
002D 3676 :
002D 3677 : SUBTEST #1
002D 3678 : 1. Load good parity into DCS address 2 so cpu will init ok
002D 3679 : 2. Load bad parity into DCS address 2 for test
002D 3680 : 3. Fetch DCS address 0
002D 3681 : 4. Put 1800 onto CS ADDRESS BUS and block COMET CS ADDRESS lines
002D 3682 : 5. Set PARSTP and MSTP bits so RDM can detect parity errors
002D 3683 : 6. Let the clock free run
002D 3684 : 7. Test that clock stopped at DCS address 03
002D 3685 : 8. If no error go to subtest 2
002D 3686 :
002D 3687 : SUBTEST #2
002D 3688 : 1. Load good parity into DCS address 2 so cpu will init ok
002D 3689 : 2. Load bad parity into DCS address 2 for test
002D 3690 : 3. Fetch DCS address 0
002D 3691 : 4. Put 1800 onto CS ADDRESS BUS and block COMET CS ADDRESS lines
002D 3692 : 5. Clear PARSTP bit so the RDM cannot detect parity errors
002D 3693 : 6. Let the clock free run
002D 3694 : 7. Test that clock stopped at DCS address 05
002D 3695 : 8. If no error go to next test
002D 3696 :
002D 3697 :
002D 3698 : TEST PATTERNS:
002D 3699 :
002D 3700 : none
002D 3701 :
002D 3702 :
002D 3703 : LOGIC DESCRIPTION:
002D 3704 :
002D 3705 : This test checks that the clock control logic responds properly to
002D 3706 : parity errors depending on the state of the PARSTP bit in the
002D 3707 : DCSCR register. In subtest 1 the PARSTP bit is set and the RDM
002D 3708 : should detect the parity error and halt the clock. In subtest 2
002D 3709 : the PARSTP bit is not set; thus the RDM should ignore the parity
002D 3710 : error and continue to execute DCS micro instructions.
002D 3711 :
002D 3712 :
002D 3713 : ASSUMPTIONS:
002D 3714 :
002D 3715 : This test assumes the power up self test has run successfully
002D 3716 : and the COMET parity detection logic is operational.
002D 3717 :
002D 3718 :
002D 3719 : ERROR DESCRIPTION:
002D 3720 :
002D 3721 : EXPECTED CONTENTS OF CLKDCS REGISTER
```

```
002D 3722 : RECEIVED CONTENTS OF CLKDCS REGISTER
002D 3723 :
002D 3724 :--
002D 3725 :*****
002D 3726 :////////////////////////////////////
002D 3727 :+
002D 3728 :SUBTEST #1
002D 3729 :
002D 3730 : Test clock control logic with PARSTP set
002D 3731 :--
002D 3732 : SUBTEST ; SUBTEST #1
0030 :
0030 :////////////////////////////////////
0030 3733 : LOADDCS 1$,,2,1 ; Good parity for DCS address 2
0037 3734 : INITIALIZE ; Init the cpu
0039 3735 : LDFIELD 11$,,B,2$,,78,78,,,,NP, ; Force parity error
004A 3736 : LOADDCS 2$,,2,1 ; Bad parity for DCS address 2
0051 3737 : ERRLOOP ; Error loop point
0053 3738 : FETCH 0,INH ; Set DCS add 0 into latches
0058 3739 : LOADREG CSAMR,3$,,W ; Load 1800 into CSAMR
0060 3740 : LOADREG DCSCR,4$,,B ; Set PARSTP bit
0068 3741 : LOADREG RDCTRL,5$,,B ; Set MSTP bit
0070 3742 : LOADREG DCSCR,6$,,B ; Start the clock
0078 3743 : CMPREG CLKDCS,7$,,B, ; Test for clock stop at DCS 03
0081 3744 : LOADREG DCSCR,4$,,B ; Stop the clock
0089 3745 : IFERROR ..<<RDM>> ; Incorrect clock stop
008F 3746 :
008F 3747 :
008F 3748 :
008F 3749 :////////////////////////////////////
008F 3750 :+
008F 3751 :SUBTEST #2
008F 3752 :
008F 3753 : Test clock control logic with PARSTP not set
008F 3754 :--
008F 3755 : SUBTEST ; SUBTEST #2
0092 :
0092 :////////////////////////////////////
0092 3756 : LOADDCS 1$,,2,1 ; Good parity for DCS address 2
0099 3757 : INITIALIZE ; Init the cpu
0098 3758 : LOADDCS 2$,,2,1 ; Bad parity for DCS address 2
00A2 3759 : ERRLOOP ; Error loop point
00A4 3760 : FETCH 0,INH ; Set DCS add 0 into latches
00A9 3761 : LOADREG CSAMR,3$,,W ; Load 1800 into CSAMR
00B1 3762 : LOADREG DCSCR,8$,,B ; Clear PARSTP bit
00B9 3763 : LOADREG RDCTRL,5$,,B ; Set MSTP bit
00C1 3764 : LOADREG DCSCR,9$,,B ; Start the clock
00C9 3765 : CMPREG CLKDCS,10$,,B, ; Test clock stop at DCS 05
00D2 3766 : LOADREG DCSCR,8$,,B ; Stop the clock
00DA 3767 : IFERROR ..<<RDM>> ; Incorrect clock stop
00E0 3768 : SKIP ; Go to next test
00E7 3769 :
00E7 3770 :
00E7 3771 :
00E7 3772 : BEGIN_TEST_DATA
00E7 :-----
00E7
```

```
U 02, 7700,4800,0364,0300,0470,1803 00E7 3773
00E7 3774 1$:
00E7 3775 :% 02 NOP ; Good parity for DCS add 2
00F2 3779 2$:
U 02, 7700,4800,0364,0300,0470,1803 00F2 3780 :% 02 NOP ; Bad parity for DCS add 2
1800 00FD 3784 3$: ; 1800 to CSAMR
F8 00FF 3785 4$: .WORD DCS_START
0100 3786 .BYTE PAR_CHK_ENABL!PAR_STOP_ENABL!-
0100 3787 CLK_CTL_0!CLK_CTL_1!-
08 0100 3788 5$: .BYTE MICRO_ADDR_INH
98 0101 3789 6$: .BYTE MASTER_HALT_EN
0102 3790 .BYTE PAR_CHK_ENABL!PAR_STOP_ENABL!-
C4 0102 3791 7$: .BYTE MICRO_ADDR_INH
E8 0103 3792 8$: .BYTE ^XC4 ; Clock stopped at DCS add 03
0104 3793 .BYTE PAR_CHK_ENABL!MICRO_ADDR_INH!-
88 0104 3794 9$: .BYTE CLK_CTL_0!CLK_CTL_1
C5 0105 3795 10$: .BYTE PAR_CHK_ENABL!MICRO_ADDR_INH
00 0106 3796 11$: .BYTE ^XC5 ; Clock stopped at DCS add 05
0107 3797 .BYTE 0
0107 3798
0107 3799 BEGIN_CPU_DATA
0002 3800
0002 3801 DCS_DATA
0001 3802
U 00, 7700,c800,0364,0300,0470,1801 0001 3803 :% 00 NOP
U 01, 7700,c800,0364,0300,0470,1802 000C 3807 :% 01 NOP
U 02, 7700,4800,0364,0300,0470,1803 0017 3811 :% 02 NOP
U 03, 7700,c800,0364,0300,0470,1804 0022 3815 :% 03 NOP
U 04, 7300,4800,0364,0300,0470,1805 002D 3819 :% 04 NOP,STOP.CLOCK
0038 3823
0038 3824 END_CPU_DATA
0107 3825
0107 3826
0107 3827 END_TEST_DATA
0107
0107
0107 3828 -----
ENDTEST
```

```
0015 .SBTTL 'TEST 020 CS ADDRESS MATCH
0015 3830 :*****
0015 3831 :++
0015 3832 :
0015 3833 : FUNCTIONAL DESCRIPTION:
0015 3834 :
0015 3835 : This test checks that the clock will stop on a CS address match. It
0015 3836 : also checks the data integrity of the address comparator.
0015 3837 :
0015 3838 :
0015 3839 : TEST ALGORITHM:
0015 3840 :
0015 3841 : SUBTEST #1
0015 3842 : 1. Load test micro code into DCS
0015 3843 : 2. Fetch location 0 of DCS
0015 3844 : 3. Load match address into CSAMR
0015 3845 : 4. Start the clock with the STPMCH bit set
0015 3846 : 5. Test the CLKDCS register for clock stop at the correct address
0015 3847 : 6. If no error go to subtest 2
0015 3848 :
0015 3849 : SUBTEST #2
0015 3850 : 1. Create a loop of 24 iterations
0015 3851 : 2. Setup the next field of a micro word with the test pattern
0015 3852 : 3. Load the micro word into DCS
0015 3853 : 4. Fetch location 0 of DCS
0015 3854 : 5. Load the match address into CSAMR
0015 3855 : 6. Start the clock with the STPMCH bit set
0015 3856 : 7. Test the CLKDCS register for clock stop at the correct address
0015 3857 : 8. If no error go to step 2 for the remaining test patterns
0015 3858 :
0015 3859 : SUBTEST #3
0015 3860 : 1. Load the next field of a micro word with 1800
0015 3861 : 2. Load the micro word into DCS
0015 3862 : 3. Create a loop of 11 iterations
0015 3863 : 4. Fetch location 0 of DCS
0015 3864 : 5. Load the match address into CSAMR
0015 3865 : 6. Start the clock with the STPMCH bit set
0015 3866 : 7. Test the CLKDCS register to see that a match did not take place
0015 3867 : 8. If no error go to step 4 for the remaining test patterns
0015 3868 :
0015 3869 : SUBTEST #4
0015 3870 : 1. Create a test loop of 11 iterations
0015 3871 : 2. Setup the next field of a micro word with the test pattern
0015 3872 : 3. Load the micro word into DCS
0015 3873 : 4. Fetch location 0 of DCS
0015 3874 : 5. Load 1800 into CSAMR
0015 3875 : 6. Start the clock with the STPMCH bit set
0015 3876 : 7. Test the CLKDCS register to see that a match did not take place
0015 3877 : 8. If no error go to step 2 for the remaining test patterns
0015 3878 :
0015 3879 :
0015 3880 : TEST PATTERNS:
0015 3881 :
0015 3882 : SUBTEST #1:
0015 3883 : CSAMR NEXT FIELD
0015 3884 : 1820 1820
0015 3885 :
```

		SUBTEST #2		
		ITERATION	CSAMR	NEXT FIELD
0015	3886			
0015	3887			
0015	3888	1	1800	
0015	3889	2	1801	
0015	3890	3	1802	
0015	3891	4	1804	
0015	3892	5	1808	
0015	3893	6	1810	
0015	3894	7	1820	
0015	3895	8	1840	
0015	3896	9	1880	
0015	3897	10	1900	
0015	3898	11	1A00	
0015	3899	12	1C00	
0015	3900	13	1FFF	
0015	3901	14	1FFE	
0015	3902	15	1FFD	
0015	3903	16	1FFB	
0015	3904	17	1FF7	
0015	3905	18	1FEF	
0015	3906	19	1FDF	
0015	3907	20	1FBF	
0015	3908	21	1F7F	
0015	3909	22	1EFF	
0015	3910	23	1DFF	
0015	3911	24	1BFF	

		SUBTEST #3		
		ITERATION	CSAMR	NEXT FIELD
0015	3912			
0015	3913			
0015	3914	1	1802	1800
0015	3915	2	1804	1800
0015	3916	3	1808	1800
0015	3917	4	1810	1800
0015	3918	5	1820	1800
0015	3919	6	1840	1800
0015	3920	7	1880	1800
0015	3921	8	1900	1800
0015	3922	9	1A00	1800
0015	3923	10	1C00	1800
0015	3924	11	1FFF	1800

		SUBTEST #4		
		ITERATION	CSAMR	NEXT FIELD
0015	3925			
0015	3926			
0015	3927			
0015	3928	1	1800	1802
0015	3929	2	1800	1804
0015	3930	3	1800	1808
0015	3931	4	1800	1810
0015	3932	5	1800	1820
0015	3933	6	1800	1840
0015	3934	7	1800	1880
0015	3935	8	1800	1900
0015	3936	9	1800	1A00
0015	3937	10	1800	1C00
0015	3938	11	1800	1FFF

0015 3939 :  
0015 3940 :  
0015 3941 :  
0015 3942 : LOGIC DESCRIPTION:

0015 3943 :  
0015 3944 : This test checks the ability of MATCH register to detect CS ADDRESS  
0015 3945 : matches and to stop the clock via the clock control logic. Subtest 1  
0015 3946 : and 2 test for a match condition. Subtest 3 and 4 test for a not match  
0015 3947 : condition.  
0015 3948 :  
0015 3949 :  
0015 3950 : ASSUMPTIONS:

0015 3951 :  
0015 3952 : This test assumes the CS ADDRESS BUS is operational  
0015 3953 :  
0015 3954 : ERROR DESCRIPTION:

0015 3955 :  
0015 3956 :  
0015 3957 : EXPECTED CLKDCS REGISTER CONTENTS  
0015 3958 : RECEIVED CLKDCS REGISTER CONTENTS  
0015 3959 : LOOP COUNT (subtest 2,3,4)  
0015 3960 :  
0015 3961 : --

0015 3962 : \*\*\*\*\*  
0015 3963 : //  
0015 3964 : +

0015 3965 : SUBTEST #1  
0015 3966 :  
0015 3967 : Test for match on address 1820  
0015 3968 : -

0015 3969 : SUBTEST : SUBTEST #1  
0018 :  
0018 : //  
0018 3970 : INITIALIZE : Init the cpu  
001A 3971 : LOADDCS 1\$,0,4 : Load some micro words  
0021 3972 : ERRLOOP : Error loop point  
0023 3973 : FETCH 0 : Start execution  
0028 3974 : LOADREG CSAMR,2\$,W : Load address to match on  
0030 3975 : LOADREG DCSCR,3\$,B : Start the clock and set Halt  
0038 3976 : : on Address Match  
0038 3977 : CMPREG CLKDCS,4\$,B : Test CLKDCS for correct stop  
0041 3978 : LOADREG DCSCR,5\$,B : Stop the clock  
0049 3979 : IFERROR ..<RDM> : Incorrect stop  
004F 3980 :  
004F 3981 :  
004F 3982 : //  
004F 3983 : +

004F 3984 : SUBTEST #2  
004F 3985 :  
004F 3986 : Test for a match condition using various patterns  
004F 3987 : -

004F 3988 : SUBTEST : SUBTEST #2  
0052 :  
0052 : //  
0052 3989 : LOOP I,1,24 : Set loop for 24 test patterns  
0059 3990 : LDFIELD 10\$,I,W,11\$,0,13, : Set next field of a microword  
006A 3991 : LOADDCS 11\$,1,1 : Load the microword  
0071 3992 : ERRLOOP : Error loop point  
0073 3993 : FETCH 0 : Start execution  
0078 3994 : LOADREG CSAMR,10\$,I,W : Load address to match on  
0080 3995 : LOADREG DCSCR,3\$,B : Start clock with STPMCH

```

0088 3996      CMPREG      CLKDCS,4$,,B      ; Test clock stop location
0091 3997      LOADREG     DCSCR,5$,,B      ; Stop the clock
0099 3998      IFERROR     ,,<<RDM>>      ; Incorrect clock stop
009F 3999      ENDLOOP     I              ; End test loop
00A2 4000
00A2 4001
00A2 4002 :////////////////////////////////////////////////////
00A2 4003 :+
00A2 4004 :SUBTEST #3
00A2 4005 :
00A2 4006 :      Check for not match with various patterns
00A2 4007 :-
00A2 4008 :      SUBTEST                          : SUBTEST #3
00A5 :////////////////////////////////////////////////////
00A5 4009      LDFIELD     20$,,W,11$,0,13,  ; Set next field to 1800
00B6 4010      LOADDCS     11$,,1,1        ; Load the microword
00BD 4011      LOOP        I,1,11         ; Set loop of 11 test patterns
00C4 4012      ERRLOOP     0              ; Error loop point
00C6 4013      FETCH      0              ; Start execution
00CB 4014      LOADREG     CSAMR,10$+2,I,W  ; Load address to match on
00D3 4015      LOADREG     DCSCR,3$,,B     ; Start clock with STPMCH
00DB 4016      CMPREG     CLKDCS,21$,,B   ; Check clock stop location
00E4 4017      LOADREG     DCSCR,5$,,B     ; Stop the clock
00EC 4018      IFERROR     ,,<<RDM>>      ; Incorrect clock stop
00F2 4019      ENDLOOP     I              ; End test loop
00F5 4020
00F5 4021
00F5 4022 :////////////////////////////////////////////////////
00F5 4023 :+
00F5 4024 :SUBTEST #4
00F5 4025 :
00F5 4026 :      Test more patterns for a not match condition
00F5 4027 :-
00F5 4028 :      SUBTEST                          : SUBTEST #4
00F8 :////////////////////////////////////////////////////
00F8 4029      LOOP        I,1,11         ; Set loop for 11 test patterns
00FF 4030      LDFIELD     10$+2,I,W,11$,0,13, ; Set next field of a microword
0110 4031      LOADDCS     11$,,1,1        ; Load the microword
0117 4032      ERRLOOP     0              ; Error loop point
0119 4033      FETCH      0              ; Start execution
011E 4034      LOADREG     CSAMR,20$,,W    ; Load address to match on
0126 4035      LOADREG     DCSCR,3$,,B     ; Start clock with STPMCH
012E 4036      CMPREG     CLKDCS,21$,,B   ; Test clock stop location
0137 4037      LOADREG     DCSCR,5$,,B     ; Stop the clock
013F 4038      IFERROR     ,,<<RDM>>      ; Incorrect clock stop
0145 4039      ENDLOOP     I              ; End test loop
0148 4040      SKIP
014F 4041
014F 4042      BEGIN_TEST_DATA
014F :-----
014F 4043
014F 4044 1$:
014F 4045 :% 00  NEXT/18AA
015A 4049 :% 01  NEXT/1820

```





```
0066 4139  
0066 4140 BEGIN_TEST_DATA  
0066  
0066 ;-----  
60 0066 4141  
80 0066 4142 1$: .BYTE CLK_CTRL_0!CLK_CTRL_1 ;HALT THE CLOCK  
80 0067 4143 2$: .BYTE ^X80  
80 0068 4144 3$: .BYTE ^X80 ;MASK FOR CMPREGMSK PSEUDO OP  
40 0069 4145 4$: .BYTE DC_LOW ;FORCE DC LOW  
00 006A 4146 5$: .BYTE ^X00 ;COMPARE DATA  
0068 4147  
0068 4148 END_TEST_DATA  
0068  
0068 ;-----  
0068 4149  
0068 4150 .SBTTL ""  
0068 4151 .SBTTL "TRACE FUNCTIONS"  
0068 4152 .SBTTL ""  
0068 4153 ENDTST
```

ZZ-  
ECK  
Sym  
VBU  
VBU  
VBU  
VBU  
VB  
V\_B  
W-  
WBU  
WDR  
WD-  
WD-  
WD-  
WD-  
WHR  
WH-  
WH-  
WH-  
WH-  
WOR

```
0021 .SBTTL 'TEST 022 DCS WRITE WITH TRACE ENABLED
0021 4155 :*****
0021 4156 :++
0021 4157 :
0021 4158 : FUNCTIONAL DESCRIPTION:
0021 4159 :
0021 4160 :     This test checks that the first two bytes of the DCS are written
0021 4161 :     with the TRAP register when trace is enabled.
0021 4162 :
0021 4163 : TEST ALGORITHM:
0021 4164 :
0021 4165 :     1. Deselect DCS by loading 0 into the CSAMR
0021 4166 :     2. Write 1's into the first two bytes of DCS address 0
0021 4167 :     3. Load 0's into the CSTRP register from the CSAMR
0021 4168 :     4. Load 1's into the CSAMR
0021 4169 :     5. Select DCS address 0 and with the DTRACE bit set tick the clock
0021 4170 :     6. Test that the CLKDCS register contains address 1
0021 4171 :     7. If no error test the DCSCF ram address 0 for 0's
0021 4172 :     8. If no error test the BADD1 ram address 0 for 0's
0021 4173 :     9. If no error go to the next test
0021 4174 :
0021 4175 :
0021 4176 : TEST PATTERNS:
0021 4177 :
0021 4178 :     none
0021 4179 :
0021 4180 :
0021 4181 : LOGIC DESCRIPTION:
0021 4182 :
0021 4183 :     This test checks the trace function by loading 1's into the first
0021 4184 :     two bytes of DCS address 0. Then with the DTRACE bit set and 0's
0021 4185 :     in the CSTRP register the clock is ticked. This should cause the
0021 4186 :     contents of CSTRP to be written into the first two bytes of DCS
0021 4187 :     address 0.
0021 4188 :
0021 4189 :
0021 4190 : ASSUMPTIONS:
0021 4191 :
0021 4192 :     This test assumes that the CS ADDRESS BUS is operational
0021 4193 :
0021 4194 :
0021 4195 : ERROR DESCRIPTION:
0021 4196 :
0021 4197 :     IFERROR #1:
0021 4198 :         EXPECTED CONTENTS OF CLKDCS REGISTER
0021 4199 :         RECEIVED CONTENTS OF CLKDCS REGISTER
0021 4200 :
0021 4201 :     IFERROR #2:
0021 4202 :         EXPECTED CONTENTS OF DCSCF RAM LOCATION 0
0021 4203 :         RECEIVED CONTENTS OF DCSCF RAM LOCATION 0
0021 4204 :
0021 4205 :     IFERROR #3:
0021 4206 :         EXPECTED CONTENTS OF BADD1 RAM LOCATION 0
0021 4207 :         RECEIVED CONTENTS OF BADD1 RAM LOCATION 0
0021 4208 :
0021 4209 :     --
0021 4210 :*****
```



ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 022 DCS WRITE WITH TRACE ENABLED  
VAX-11/750 RDM MICRO DIAGNOSTICS  
'TEST 022 DCS WRITE WITH TRACE ENABLED

N 8  
6-MAY-1982

Fiche 1 Frame N8  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1  
Sequence 104 Page 10 (1)

```
EC 00F5 4266 7$: .BYTE CLK_CTRL_1!CLK_CTRL_0!PAR_CHK_ENABL!MICRO_ADDR_INH!TRACE_ENABL
AC 00F6 4267 8$: .BYTE CLK_CTRL_0!PAR_CHK_ENABL!MICRO_ADDR_INH!TRACE_ENABL
3F 00F7 4268 9$: .BYTE ^X3F
01 00F8 4269 10$: .BYTE 1
   00F9 4270
   00F9 4271          END_TEST_DATA
   00F9
   00F9
   00F9 4272 ;-----
   00F9 4273          .SBTTL ""
   00F9 4274          .SBTTL ""CMI FUNCTIONS
   00F9 4275          .SBTTL ""
   00F9 4276          ENDTEST
```

ZZ-  
ECI  
VA)

Mac

WRK  
SYS  
TOI

188

The  
10

WRK

```
0023 .SBTTL 'TEST 023 TEST RDM TO CMI DATA INTEGRITY
0023 4278 :*****
0023 4279 :++
0023 4280 :
0023 4281 : FUNCTIONAL DESCRIPTION:
0023 4282 :
0023 4283 : THIS IS THE FIRST TEST OF THE CMI BUS. IT WILL CHECK TO SEE IF
0023 4284 : ANY DATA BITS ARE STUCK.
0023 4285 :
0023 4286 :
0023 4287 : TEST ALGORITHM:
0023 4288 :
0023 4289 : SUBTEST #1
0023 4290 : 1. CREATE A TEST LOOP OF 6
0023 4291 : 2. GENERATE A TEST PATTERN
0023 4292 : 3. LOAD THE TEST PATTERN INTO THE RDM MA REGISTER
0023 4293 : 4. ENABLE MA REGISTER ONTO CMI BUS
0023 4294 : 5. READ CMI BUS INTO RDM MH REGISTER
0023 4295 : 6. CLEAR THE RDM MAIN32 REGISTER
0023 4296 : 7. TEST THE MH REGISTER FOR THE CORRECT PATTERN
0023 4297 : 8. IF AN ERROR GO TO SUBTEST 3 ELSE GO TO STEP 2 FOR REMAINING PATT.
0023 4298 :
0023 4299 : SUBTEST #2
0023 4300 : 1. CREATE A TEST LOOP OF 6
0023 4301 : 2. GENERATE A TEST PATTERN
0023 4302 : 3. LOAD THE TEST PATTERN INTO THE RDM MD REGISTER
0023 4303 : 4. ENABLE MD REGISTER ONTO CMI BUS
0023 4304 : 5. READ CMI BUS INTO RDM MH REGISTER
0023 4305 : 6. CLEAR THE RDM MAIN32 REGISTER
0023 4306 : 7. TEST THE MH REGISTER FOR THE CORRECT PATTERN
0023 4307 : 8. IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0023 4308 :
0023 4309 : SUBTEST #3
0023 4310 : 1. CREATE A TEST LOOP OF 6
0023 4311 : 2. GENERATE A TEST PATTERN
0023 4312 : 3. LOAD THE TEST PATTERN INTO THE RDM MD REGISTER
0023 4313 : 4. ENABLE MD REGISTER ONTO CMI BUS
0023 4314 : 5. READ CMI BUS INTO RDM MH REGISTER
0023 4315 : 6. CLEAR THE RDM MAIN32 REGISTER
0023 4316 : 7. TEST THE MH REGISTER FOR THE CORRECT PATTERN
0023 4317 : 8. IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0023 4318 : 9. IF LOOP EXHUSTED FORCE AN ERROR TO CALL OUT RDM MA REGISTER
0023 4319 :
0023 4320 :
0023 4321 : TEST PATTERNS:
0023 4322 :
0023 4323 : LOOP COUNT TEST PATTERN
0023 4324 : 1 AAAAAAA
0023 4325 : 2 5555555
0023 4326 : 3 3333333
0023 4327 : 4 0F0F0F0F
0023 4328 : 5 00FF00FF
0023 4329 : 6 0000FFFF
0023 4330 :
0023 4331 :
0023 4332 : LOGIC DESCRIPTION:
0023 4333 :
```

0023 4334 : THIS TEST SIMPLY TRIES TO DRIVE BITS ONTO THE CMI DATA BUS FROM THE  
0023 4335 : RDM MA AND MD REGISTERS. THEY ARE THEN READ BACK ONTO THE RDM MH  
0023 4336 : REGISTER. IF THE TEST FAILS, IT IS EITHER THE RDM, OR BITS ARE BEING  
0023 4337 : HELD BY SOMEOTHER MODULE ON THE CMI. WHAT MODULES ARE ON THE CMI WILL  
0023 4338 : DEPEND ON THE SYSTEM CONFIGURATION. DON'T FORGET THINGS LIKE WCS, MBA,  
0023 4339 : etc.

0023 4341 :  
0023 4342 : ASSUMPTIONS:

0023 4343 :  
0023 4344 : THIS TEST ASSUMES THE RDM POWER-UP SELF TEST HAS RUN SUCCESSFULLY  
0023 4345 :  
0023 4346 :

0023 4347 : ERROR DESCRIPTION:

0023 4348 :  
0023 4349 : EXPECTED TEST PATTERN ON CMI  
0023 4350 : RECEIVED TEST PATTERN FROM CMI  
0023 4351 : LOOP COUNT  
0023 4352 :

0023 4353 :--  
0023 4354 :\*\*\*\*\*  
0023 4355 :////////////////////////////////////

0023 4356 :+  
0023 4357 :SUBTEST #1

0023 4358 :  
0023 4359 : TEST MAREG TO CMI

0023 4360 :-  
0023 4361 : SUBTEST 1 ;SUBTEST #1

0026 :  
0026 :////////////////////////////////////  
0026 4362 : INITIALIZE ;INIT THE CPU  
0028 4363 : LOOP I,1,6 ;CREATE A TEST LOOP OF 6  
002F 4364 : SA01PAT I,1\$,.. ;GENERATE A TEST PATTERN  
0037 4365 : LOADREG MAREG,1\$,..L ;PUT TEST PATTERN IN MAREG  
003F 4366 : LOADREG MAIN32,2\$,..B ;ENABLE MAREG TO CMI  
0047 4367 : LOADREG MAIN32,3\$,..B ;STROBE MHREG  
004F 4368 : LOADREG MAIN32,6\$,..B ;CLEAR THE MAIN32 REGISTER  
0057 4369 : CMPREG MHREG,1\$,..L ;TEST FOR CORRECT TEST PATTERN  
0060 4370 : SKIP 100\$,ONERROR ;TRY COMET TO MAIN MEMORY  
0067 4371 : ENDOOP I ;END TEST LOOP

006A 4372 :  
006A 4373 :  
006A 4374 :  
006A 4375 :////////////////////////////////////

006A 4376 :+  
006A 4377 :SUBTEST #2

006A 4378 :  
006A 4379 : TEST MDREG TO CMI

006A 4380 :-  
006A 4381 : SUBTEST 2 ;SUBTEST #2

006D :  
006D :////////////////////////////////////  
006D 4382 : INITIALIZE ;INIT THE CPU  
006F 4383 : LOOP I,1,6 ;CREATE A TEST LOOP OF 6  
0076 4384 : SA01PAT I,1\$,.. ;GENERATE A TEST PATTERN  
007E 4385 : ERRLOOP ;ERROR LOOP POINT  
0080 4386 : LOADREG MDREG,1\$,..L ;PUT TEST PATTERN INTO MDREG

```
0088 4387          LOADREG          MAIN32,4$,,B          :ENABLE MDREG TO CMI
0090 4388          LOADREG          MAIN32,5$,,B          :STROBE MHREG
0098 4389          LOADREG          MAIN32,6$,,B          :CLEAR THE MAIN32 REGISTER
00A0 4390          CMPREG           MHREG,1$,,L           :TEST FOR CORRECT PATTERN
00A9 4391          IFERROR          I,<<RDM>>             :MDREG ON RDM IS BAD
00AF 4392          ENDLOOP          I                     :END TEST LOOP
00B2 4393          SKIP
00B9 4394
00B9 4395
00B9 4396 :////////////////////
00B9 4397 :+
00B9 4398 :SUBTEST #3
00B9 4399 :
00B9 4400 :          TEST MDREG TO CMI
00B9 4401 :-
00B9 4402 100$: SUBTEST          3          :SUBTEST #3
00BC
00BC :////////////////////
00BC 4403          INITIALIZE          :INIT THE CPU
00BE 4404          LOOP              I,1,6              :CREATE A TEST LOOP OF 6
00C5 4405          SA01PAT          I,1$,..            :GENERATE A TEST PATTERN
00CD 4406          ERRLOOP
00CF 4407          LOADREG          MDREG,1$,,L          :PUT TEST PATTERN INTO MDREG
00D7 4408          LOADREG          MAIN32,4$,,B          :ENABLE MDREG TO CMI
00DF 4409          LOADREG          MAIN32,5$,,B          :STROBE MHREG
00E7 4410          LOADREG          MAIN32,6$,,B          :CLEAR MAIN32 REGISTER
00EF 4411          CMPREG           MHREG,1$,,L           :TEST FOR CORRECT PATTERN
00F8 4412          IFERROR          I,<<CMI>>             :SOMETHING IS TIED TO CMI
00FE 4413          ENDLOOP          I                     :END TEST LOOP
0101 4414          CMPREG           MHREG,1$,,L,NE        :FORCE AN ERROR
010A 4415          IFERROR
010F 4416          SKIP
0116 4417
0116 4418
0116 4419
0116 4420 BEGIN_TEST_DATA
0116
0116 :-----
0116 4421
00000000 0116 4422 1$: .LONG ^X00000000          :TEST PATTERN STORAGE
81 011A 4423 2$: .BYTE ^X81          :ENABLE MAREG TO CMI
A1 011B 4424 3$: .BYTE ^XA1          :STROBE MHREG
41 011C 4425 4$: .BYTE ^X41          :ENABLE MDREG TO CMI
61 011D 4426 5$: .BYTE ^X61          :STROBE MHREG
01 011E 4427 6$: .BYTE ^X01          :CLEAR MAIN32 REGISTER
011F 4428
011F 4429 END_TEST_DATA
011F
011F :-----
011F 4430 ENDTEST
```

```
0027 .SBTTL 'TEST 024 TEST RDM TO CMI CONTROL AND STATUS
0027 4432 :*****
0027 4433 :++
0027 4434 :
0027 4435 : FUNCTIONAL DESCRIPTION:
0027 4436 :
0027 4437 : THIS TEST WILL PERFORM A CMI BUS CYCLE FROM THE RDM TO DETERMINE IF
0027 4438 : THE CONTROL AND STATUS LINES ARE OPERATIONAL.
0027 4439 :
0027 4440 : TEST ALGORITHM:
0027 4441 :
0027 4442 : SUBTEST #1
0027 4443 : 1. CREATE A TEST LOOP OF 6
0027 4444 : 2. GENERATE A TEST PATTERN
0027 4445 : 3. LOAD THE FIRST DCS MICROINSTRUCTION
0027 4446 : 4. ALLOW THE CPU CLOCK TO FREE RUN
0027 4447 : 5. CLEAR THE CMICTL REGISTER
0027 4448 : 6. LOAD THE MAREG WITH A WRITE COMMAND
0027 4449 : 7. LOAD THE TEST PATTERN INTO THE MDREG
0027 4450 : 8. ALLOW THE RDM TO PERFORM THE BUS CYCLE
0027 4451 : 9. STOP THE CPU CLOCK
0027 4452 : 10. TEST THE MHREG FOR THE TEST PATTERN
0027 4453 : 11. IF NO ERROR TEST THE STATUS REGISTER FOR GOOD STATUS
0027 4454 : 12. IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
0027 4455 :
0027 4456 : SUBTEST #2
0027 4457 : 1. SAME AS SUBTEST 1 EXCEPT CMI CYCLE IS TO A NONEXISTANT ADDRESS
0027 4458 : 2. TEST STATUS REGISTER FOR NO RESPONSE
0027 4459 :
0027 4460 :
0027 4461 : TEST PATTERNS:
0027 4462 :
0027 4463 : ITERATION TEST PATTERN
0027 4464 : 1 AAAAAAAA
0027 4465 : 2 55555555
0027 4466 : 3 33333333
0027 4467 : 4 0F0F0F0F
0027 4468 : 5 00FF00FF
0027 4469 : 6 0000FFFF
0027 4470 :
0027 4471 :
0027 4472 : LOGIC DESCRIPTION:
0027 4473 :
0027 4474 : THIS TEST ATTEMPTS TO CHECK THE CONTROL AND STATUS LINES ON THE CMI BY
0027 4475 : PERFORMING A WRITE OPERATION TO MAIN MEMORY. BY SIMULTANEOUSLY SETTING
0027 4476 : THE WRITE AND READ BITS IN THE RDM CMICTL REGISTER YOU CAN CAUSE THE
0027 4477 : RDM TO PERFORM A CMI CYCLE. THE DATA LOADED INTO THE MHREG WILL BE
0027 4478 : WHAT IS SENT FROM THE MDREG.
0027 4479 :
0027 4480 :
0027 4481 : ASSUMPTIONS:
0027 4482 :
0027 4483 : THIS TEST ASSUMES THE CPU CLOCK IS OPERATIONAL.
0027 4484 :
0027 4485 :
0027 4486 : ERROR DESCRIPTION:
0027 4487 :
```

```
0027 4488 : SUBTEST #1:
0027 4489 : FIRST IFERROR
0027 4490 : EXPECTED DATA FROM CMI
0027 4491 : RECEIVED DATA FROM CMI
0027 4492 : LOOP COUNT I
0027 4493 :
0027 4494 : SECOND IFERROR
0027 4495 : EXPECTED STATUS FROM CMI
0027 4496 : RECEIVED STATUS FROM CMI
0027 4497 : LOOP COUNT I
0027 4498 :
0027 4499 : SUBTEST #2:
0027 4500 : EXPECTED STATUS FROM CMI
0027 4501 : RECEIVED STATUS FROM CMI
0027 4502 :
0027 4503 : --
0027 4504 : *****
0027 4505 : +
0027 4506 : SUBTEST #1
0027 4507 :
0027 4508 : TEST CMI CONTROL AND STATUS LINES
0027 4509 : -
0027 4510 : SUBTEST :SUBTEST #1
002A :
002A : ////////////////////////////////////////////////////////////////////
002A 4511 : INITIALIZE :INIT THE CPU
002C 4512 : LOOP I,1,6 :CREATE A TEST LOOP OF 6
0033 4513 : SA01PAT I,5$. :GENERATE A TEST PATTERN
003B 4514 : ERRLOOP :ERROR LOOP POINT
003D 4515 : FETCH 0 :START DCS PROGRAM
0042 4516 : LOADREG DCSCR,2$,,B :START THE CPU CLOCK
004A 4517 : LOADREG CMICTL,3$,,B :CLEAR THE CMICTL REGISTER
0052 4518 : LOADREG MAREG,4$,,L :LOAD MAREG WITH WRITE COMMAND
005A 4519 : LOADREG MDREG,5$,,L :LOAD MDREG WITH WRITE DATA
0062 4520 : LOADREG CMICTL,6$,,B :PERFORM CMI WRITE/READ
006A 4521 : LOADREG DCSCR,7$,,B :STOP THE CPU CLOCK
0072 4522 : CMPREG MHREG,5$,,L :TEST FOR CORRECT TEST PATTERN
007B 4523 : IFERROR <<CMI>> :CMI OR RDM FAILURE
0081 4524 : CMPREGMSK STATUS,8$,,8$,,B :TEST FOR GOOD STATUS
008D 4525 : IFERROR <<MC0><MC1><CMI>> :CMI, CMC OR RDM FAILURE
0095 4526 : ENDL00P I :END TEST LOOP
0098 4527 :
0098 4528 :
0098 4529 :
0098 4530 : ////////////////////////////////////////////////////////////////////
0098 4531 : +
0098 4532 : SUBTEST #2
0098 4533 :
0098 4534 : TEST STATUS LINES FOR NONEXISTANT MEMORY
0098 4535 : -
0098 4536 : SUBTEST :SUBTEST #2
0098 :
0098 : ////////////////////////////////////////////////////////////////////
0098 4537 : INITIALIZE :INIT THE CPU
009D 4538 : ERRLOOP :ERROR LOOP POINT
009F 4539 : FETCH 0 :START DCS PROGRAM
00A4 4540 : LOADREG DCSCR,2$,,B :START CPU CLOCK
```

```
00AC 4541      LOADREG      CMICTL,3$,,B      ;CLEAR THE CMICTL REGISTER
00B4 4542      LOADREG      MAREG,9$,,L      ;LOAD NONEXISTANT ADDRESS
00BC 4543      LOADREG      CMICTL,6$,,L      ;PERFORM CMI WRITE/READ
00C4 4544      LOADREG      DCSCR,7$,,B      ;STOP THE CPU CLOCK
00CC 4545      CMPREGMSK    STATUS,10$,,11$,,B, ;TEST FOR NO RESPONSE
00D8 4546      IFERROR      ,,<<CMI>>      ;INCORRECT STATUS
00DE 4547      INITIALIZE      ;INIT THE CPU
00E0 4548      SKIP          ;GO TO NEXT TEST
00E7 4549
00E7 4550
00E7 4551
00E7 4552 BEGIN_TEST_DATA
00E7 4553 ;-----
00E7 4554 2$:      .BYTE      ^X00      ;CLEAR REGISTERS
00E8 4555 3$:      .BYTE      ^X01      ;CLEAR CMICTL REGISTER
F8000000 00E9 4556 4$:      .LONG      ^XF8000000 ;WRITE COMMAND TO ADDRESS ZERO
00000000 00ED 4557 5$:      .LONG      ^X00000000 ;TEST PATTERN STORAGE
68 00F1 4558 6$:      .BYTE      ^X68      ;PERFORM CMI WRITE/READ
60 00F2 4559 7$:      .BYTE      ^X60      ;STOP CPU CLOCK
38 00F3 4560 8$:      .BYTE      ^X38      ;GOOD STATUS
F8F40000 00F4 4561 9$:      .LONG      ^XF8F40000 ;WRITE TO NONEXISTANT MEMORY
20 00F8 4562 10$:     .BYTE      ^X20      ;NO RESPONSE STATUS
38 00F9 4563 11$:     .BYTE      ^X38      ;MASK FOR CMPREGMSK PSEUDO OP
00FA 4564
00FA 4565
00FA 4566 BEGIN_CPU_DATA
0002 4567
0002 4568 DCS_DATA
0001 4569
U 00, 7700,c800,0364,0300,0470,1801 0001 4570 :% 00  NOP
U 01, 7500,c800,0364,0300,0470,1802 000C 4574 :% 01  NOP,DCS.ADDR_0      ;RUN B CLOCK
0017 4578
0017 4579 END_CPU_DATA
00FA 4580
00FA 4581
00FA 4582
00FA 4583 END_TEST_DATA
00FA 4584 ;-----
00FA 4584 ENDTEST
```

ZZ-ECKAF-3.1  
ECKAF  
V03.01

'TEST 024 TEST RDM TO CMI CONTROL AND ST

VAX-11/750 RDM MICRO DIAGNOSTICS

'TEST 024 TEST RDM TO CMI CONTROL AND ST

H 9  
6-MAY-1982

6-MAY-1982

6-MAY-1982

Fiche 1 Frame H9

19:54:55

19:54:39

VAX-11 Macro V02.46

WRKDS0:[COMET.RDM]ECKAF.MAR;1

Sequence 111

Page 10

(1

0002 4587 .END

\$CONTROL_C_CODE=	0000000A	CSAMR	=	00007404	END_TEST_29	00000107	R	3C			
\$FILE_NOT_FOUND=	00000009	CSBUF	=	00007424	END_TEST_3	00000058	R	08			
\$STN	=	00000000	CSTRP	=	00007422	END_TEST_30	00000107	R	3E		
\$TEST_NUMBER	=	00000025	CS_ADD_BUFF_HGH=	00007425	END_TEST_31	00000107	R	40			
ACV	=	0000000F	CS_ADD_BUFF_LOW=	00007424	END_TEST_32	000001BE	R	46			
AC_LOW	=	00000080	CS_ADD_TRAP_HGH=	00007423	END_TEST_33	00000068	R	48			
ADD	=	00000019	CS_ADD_TRAP_LOW=	00007422	END_TEST_34	000000F9	R	4A			
ADDR_MATCH_HI	=	00007405	CYCLE_FLAG	=	00000008	END_TEST_35	0000011F	R	4C		
ADDR_MATCH_LO	=	00007404	CYCLE_KEY	=	00000020	END_TEST_36	000000FA	R	4E		
ADK	=	0000000B	D0	=	00000004	END_TEST_4	000000B1	R	0A		
ALK	=	00000000	D1	=	00000002	END_TEST_5	00000068	R	0C		
ALP	=	00000016	D2	=	00000000	END_TEST_6	0000006E	R	0E		
ARG_CNT	=	000000D9	R	4E	DATA_FLAG	=	00000000	END_TEST_7	000000B1	R	10
A_REG_BYTE_0	=	00007410	DATA_LENGTH	=	00000001	END_TEST_8	000000A6	R	12		
A_REG_BYTE_1	=	00007411	DCSAD	=	00007401	END_TEST_9	0000061B	R	14		
A_REG_BYTE_2	=	00007412	DCSCF	=	00007403	EN_TRACE_REG	=	00000008			
A_REG_BYTE_3	=	00007413	DCSCR	=	00007402	ERRLOG_FLAG	=	00000000			
B	=	00000001	DCSDA	=	00007400	ERROR_FLAG	=	00000010			
BADD1	=	00007426	DCS_ADDR_CLEAR=	00000002	ERROR_LOG_ADR	=	000032FC				
BELL_FLAG	=	00000008	DCS_ADDR_REG_RO=	00007427	ERR_LOG_INIT	=	00000001				
BELL_KEY	=	0000000E	DCS_ADDR_REG_WO=	00007401	EXP_STALL_FLAG	=	00000010				
BUFF_ADDR_LOW	=	00007426	DCS_CTRL_FILE	=	00007403	EXP_UTRAP_FLAG	=	00000040			
BURST_STOP_FLAG=	00000080	DCS_CTRL_REG	=	00007402	FAC	=	00000020				
BYTE	=	00000001	DCS_DATA_REG	=	00007400	FAULT	=	00000002			
CACHE_FLAG	=	00000000	DCS_FLAG	=	00000000	FCC	=	00000015			
CACHE_FLAG1	=	00000000	DCS_FLAG1	=	00000000	FCS	=	00000021			
CACHE_LENGTH	=	00000001	DCS_HEAD	=	00000000	R	4F	FETCH_FLAG	=	00000008	
CAK	=	0000000A	DCS_LENGTH	=	00000001	FEX	=	0000001E			
CCC	=	00000004	DCS_SCRATCH_ADR=	00000036	FFH	=	00000024				
CCS	=	00000005	DCS_START	=	00001800	FFL	=	00000023			
CF_KEY	=	00000024	DC [OW	=	00000040	FIC	=	0000001F			
CL	=	00000003	DIAGNOSE_FLAG	=	00000004	FIO	=	0000001D			
CLEAR_CTRL_FILE=	00000002	DPM	=	00000000	FLAG_KEY	=	00000004				
CLKDCS	=	00007427	DUM1	=	FFFFFFFF	FMR	=	00000022			
CLK_CTRL_0	=	00000020	ENDTEST_LOC	=	000000FF	R	4E	FPA	=	00000002	
CLK_CTRL_0_RO	=	00000040	END_SLICE	=	00000025	FP_BOOT_0	=	00000001			
CLK_CTRL_1	=	00000040	END_TEST_1	00000064	R	04	FP_BOOT_1	=	00000002		
CLK_CTRL_1_RO	=	00000080	END_TEST_10	0000008A	R	16	FP_START_0	=	00000004		
CLOCK_STOPED	=	00000080	END_TEST_11	0000009C	R	18	FP_START_1	=	00000008		
CMI	=	00000007	END_TEST_12	000001CC	R	1A	FQA	=	00000014		
CMICTL	=	0000741C	END_TEST_13	00000060	R	1C	FRONT1	=	00007420		
CMI_COMPLETE	=	00000020	END_TEST_14	00000055	R	1E	FRONT2	=	00007421		
CMI_CTRL_CLEAR	=	00000001	END_TEST_15	00000310	R	20	FRONT_PNL_1	=	00007420		
CMI_CTRL_REG	=	0000741C	END_TEST_16	00000305	R	22	FRONT_PNL_2	=	00007421		
CMI_GO	=	00000008	END_TEST_17	0000008C	R	24	FRONT_PNL_LOCK	=	00000080		
CMI_READ	=	00000040	END_TEST_18	00000090	R	26	HALT_FLAG	=	00000001		
CMI_STATUS_0	=	00000008	END_TEST_19	00000148	R	28	HALT_KEY	=	00000008		
CMI_STATUS_1	=	00000010	END_TEST_2	00000096	R	06	HALT_ON_MATCH	=	00000001		
CMI_WRITE	=	00000020	END_TEST_20	000000BF	R	2A	HEAD	=	FFFF7C00		
CMK	=	0000000E	END_TEST_21	000000D7	R	2C	HIGH	=	00000001		
COMPOSIT_LENGTH=	00000003	END_TEST_22	0000006B	R	2E	H FROM WBUS	=	00000010			
CON	=	00000011	END_TEST_23	00000096	R	30	IB_FLAG	=	00000020		
CONTINUE_FLAG	=	00000002	END_TEST_24	0000007E	R	32	IB_KEY	=	00000012		
CONTROL_C_FLAG	=	00000001	END_TEST_25	00000090	R	34	INSTR_FLAG	=	00000004		
CONT_FLAG	=	00000020	END_TEST_26	000000A6	R	36	INSTR_KEY	=	00000018		
CONT_KEY	=	0000001C	END_TEST_27	000000B0	R	38	INT	=	00000010		
CPU_RUN	=	00000010	END_TEST_28	0000007F	R	3A	IRD	=	00000005		

I\_INDEX\_FLAG = 00000000  
I\_INIT = 00000000  
J\_INDEX\_FLAG = 00000000  
J\_INIT = 00000000  
K\_INDEX\_FLAG = 00000000  
K\_INIT = 00000000  
L = 00000004  
LIST\_END = 000000DE R 4E  
LIST\_HEAD = 000000DA R 4E  
LIST\_LENGTH = 00000004  
LITERAL = 00000000  
LONG = 00000004  
LOOP\_ERROR\_FLAG = 00000020  
LOOP\_FLAG = 00000002  
LOOP\_KEY = 0000000A  
LOST\_FLAG = 00000010  
LOST\_KEY = 0000001A  
LOW = 00000000  
M = 0000000A  
MSTEST\_SIZE = 000007B3  
MSTEST\_SIZE\_D = 000003F3  
MA0 = 00000008  
MA1 = 0000000A  
MAD = 00000013  
MAIN32 = 0000741D  
MAINT\_A\_TO\_CMI = 00000080  
MAINT\_CMI\_TO\_MH = 00000020  
MAINT\_DCS\_ENABL = 00000004  
MAINT\_MD\_TO\_CMI = 00000040  
MAINT\_REG = 0000741D  
MAINT\_STB\_INH = 00000002  
MAINT\_TRAP\_OFF = 00000001  
MAINT\_WB\_TO\_WH = 00000008  
MAINT\_WD\_TO\_WB = 00000010  
MAP = 00000012  
MAREG = 00007410  
MASTER\_HALT\_EN = 00000008  
MAX\_ARGUMENTS = 00000010  
MA\_KEY = 00000038  
MCD = 00000003  
MCI = 00000009  
MDL = 00000018  
MDR = 00000018  
MDREG = 00007414  
MD\_KEY = 00000034  
MD\_REG\_BYTE\_0 = 00007414  
MD\_REG\_BYTE\_1 = 00007415  
MD\_REG\_BYTE\_2 = 00007416  
MD\_REG\_BYTE\_3 = 00007417  
MEI = 0000001A  
MHREG = 00007418  
MH\_REG\_BYTE\_0 = 00007418  
MH\_REG\_BYTE\_1 = 00007419  
MH\_REG\_BYTE\_2 = 0000741A  
MH\_REG\_BYTE\_3 = 0000741B  
MIC = 00000001  
MICROWORD = 0000000A

MICRO\_ADDR\_INH = 00000080  
MONITOR\_SIZE = 00002C4D  
MSQ = 00000008  
MT\_KEY = 0000002E  
N = 00000001  
NER\_FLAG = 00000004  
NER\_KEY = 0000000C  
NOERROR = 00000001  
NOTEQUAL = 00000003  
ONEQUAL = 00000002  
ONERROR = 00000000  
OP\_BEGIN\_LOOP = 00000008  
OP\_BURST\_CLOCK = 00000002  
OP\_COMPARE\_REG = 00000004  
OP\_COMPARE\_REGM = 00000006  
OP\_COMPARE\_VB = 00000020  
OP\_DUMPLOG = 00000030  
OP\_ENABL\_STALL = 00000038  
OP\_ENABL\_TRAP = 0000002E  
OP\_END\_FILE = 00000024  
OP\_END\_LOOP = 0000000A  
OP\_END\_TEST = 0000000C  
OP\_ERRLOG = 00000032  
OP\_ERROR\_LOOP = 0000000E  
OP\_FETCH = 00000022  
OP\_IF\_ERROR = 00000012  
OP\_INC\_INDEX = 0000002A  
OP\_INIT = 00000014  
OP\_LOAD\_DCS = 00000000  
OP\_LOAD\_FIELD = 00000026  
OP\_LOAD\_INDEX = 00000028  
OP\_LOAD\_REG = 00000016  
OP\_MASK = 00000018  
OP\_NEW\_TEST = 0000001A  
OP\_PATTERN\_GEN = 00000010  
OP\_PRINT\_N = 00000036  
OP\_PRINT\_S = 00000034  
OP\_SAVE\_INDEX = 0000002C  
OP\_SKIP = 0000001C  
OP\_SUB\_TEST = 0000001E  
OVERLAY\_HEAD = 00000000 R 53  
PARITY\_ERROR = 00000020  
PAR\_CHR\_ENABL = 00000008  
PAR\_STOP\_ENABL = 00000010  
PASS\_KEY = 00000002  
PB\_INIT = 00000040  
PB\_KEY = 00000036  
PC\_KEY = 00000030  
PHB = 00000007  
PRK = 0000000C  
PS\_KEY = 0000003E  
QA\_FLAG = 00000040  
QA\_KEY = 00000014  
QV\_FLAG = 00000002  
QV\_KEY = 00000028  
RDCTRL = 0000741E  
RDM = 00000004

RDM\_FLAG = 00000004  
RDM\_KEY = 0000002A  
RDM\_LAST = 00000008  
RD\_CTRL\_REG = 0000741E  
REMOTE = 00000001  
RTEMP\_FLAG = 00000000  
RTEMP\_FLAG1 = 00000000  
RTEMP\_LENGTH = 00000001  
RT\_KEY = 0000002C  
SAC = 00000009  
SA\_CLOCK = 00000080  
SA\_FLAG = 00000010  
SA\_KEY = 00000010  
SA\_START\_STOP = 00000080  
SBE\_FLAG = 00000020  
SBE\_KEY = 00000042  
SECTION\_FLAG = 00000080  
SF\_KEY = 00000040  
SOMM\_FLAG = 00000001  
SOMM\_KEY = 00000006  
SPA = 00000002  
SRK = 00000001  
SRM = 00000017  
SR\_KEY = 0000003C  
STACK\_SIZE = 00000400  
STATUS = 00007406  
STATUS\_REG = 00007406  
STEP\_KEY = 0000001E  
STOP\_CLOCK = 00000004  
STROBE\_CMI = 00000040  
ST\_KEY = 0000001E  
TEMP = 00000052  
TEMP1 = 0000002E  
TEMP2 = 00000019  
TEMP3 = 00000002  
TEMP4 = 00000002 R 4D  
TEST = 00000004  
TEST\_FLAG = 00000000  
TEST\_HEAD = 00000024 R 4E  
TEST\_KEY = 00000000  
TEST\_LENGTH = 00000002  
TEST\_SECT\_HEAD = 00000000 R 4E  
THIS\_TEST = 00000024  
TICK\_FLAG = 00000002  
TICK\_KEY = 00000022  
TOK = 00000006  
TRACE\_ENABL = 00000004  
TRAP\_HALT\_EN = 00000020  
TRAP\_OFF = 00000002  
TR\_FLAG = 00000080  
TR\_KEY = 00000016  
TSTSPAN\_FLAG = 00000040  
UBI = 00000006  
UDP = 0000001C  
UTR = 0000000D  
VA\_KEY = 00000032  
VBDS\_CLOCK = 00000080

ZZ-ECKAF-3.1 Symbol table  
ECKAF  
Symbol table

VAX-11/750 RDM MICRO DIAGNOSTICS

K 9  
6-MAY-1982

Fiche 1 Frame K9  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1  
Sequence 114  
Page 11  
(1)

VBUS\_COMPARE = 00000080  
VBUS\_LOAD = 00000010  
VBUS\_SERIAL\_IN = 00000040  
VBUS\_SERIAL\_OUT = 00000001  
VB\_KEY = 00000026  
V\_BUS\_STROBE = 00000001  
W = 00000002  
WBUS\_FROM\_D = 00000020  
WDREG = 0000742C  
WD\_KEY = 0000003A  
WD\_REG\_BYTE\_0 = 0000742C  
WD\_REG\_BYTE\_1 = 0000742D  
WD\_REG\_BYTE\_2 = 0000742E  
WD\_REG\_BYTE\_3 = 0000742F  
WHREG = 00007430  
WH\_REG\_BYTE\_0 = 00007430  
WH\_REG\_BYTE\_1 = 00007431  
WH\_REG\_BYTE\_2 = 00007432  
WH\_REG\_BYTE\_3 = 00007433  
WORD = 00000002

PSECT name	Allocation	! Psect synop PSECT No.		Attributes										
. ABS .	00000000 ( 0.)	00	( 0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE
. BLANK .	00000000 ( 0.)	01	( 1.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
DIRECTORY	0000007E ( 126.)	02	( 2.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0001_D	00000002 ( 2.)	03	( 3.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0001_T	0000007E ( 126.)	04	( 4.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0002_D	00000002 ( 2.)	05	( 5.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0002_T	0000007E ( 126.)	06	( 6.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0003_D	00000002 ( 2.)	07	( 7.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0003_T	0000007E ( 126.)	08	( 8.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0004_D	00000002 ( 2.)	09	( 9.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0004_T	0000007E ( 126.)	0A	( 10.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0005_D	00000002 ( 2.)	0B	( 11.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0005_T	0000007E ( 126.)	0C	( 12.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0006_D	00000002 ( 2.)	0D	( 13.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0006_T	0000007E ( 126.)	0E	( 14.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0007_D	00000002 ( 2.)	0F	( 15.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0007_T	0000007E ( 126.)	10	( 16.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0008_D	00000002 ( 2.)	11	( 17.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0008_T	0000007E ( 126.)	12	( 18.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0009_D	00000002 ( 2.)	13	( 19.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0009_T	0000007E ( 126.)	14	( 20.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0010_D	00000002 ( 2.)	15	( 21.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0010_T	0000007E ( 126.)	16	( 22.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0011_D	00000002 ( 2.)	17	( 23.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0011_T	0000007E ( 126.)	18	( 24.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0012_D	00000002 ( 2.)	19	( 25.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0012_T	000001FE ( 510.)	1A	( 26.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0013_D	00000002 ( 2.)	1B	( 27.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0013_T	0000007E ( 126.)	1C	( 28.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0014_D	00000002 ( 2.)	1D	( 29.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0014_T	0000007E ( 126.)	1E	( 30.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0015_D	00000002 ( 2.)	1F	( 31.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0015_T	0000037E ( 894.)	20	( 32.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0016_D	00000002 ( 2.)	21	( 33.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0016_T	0000037E ( 894.)	22	( 34.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0017_D	00000002 ( 2.)	23	( 35.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0017_T	0000007E ( 126.)	24	( 36.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0018_D	00000002 ( 2.)	25	( 37.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0018_T	0000007E ( 126.)	26	( 38.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0019_D	00000002 ( 2.)	27	( 39.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0019_T	0000017E ( 382.)	28	( 40.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0020_D	00000002 ( 2.)	29	( 41.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0020_T	0000007E ( 126.)	2A	( 42.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0021_D	00000002 ( 2.)	2B	( 43.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0021_T	0000007E ( 126.)	2C	( 44.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0022_D	00000002 ( 2.)	2D	( 45.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0022_T	0000007E ( 126.)	2E	( 46.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0023_D	00000002 ( 2.)	2F	( 47.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0023_T	0000007E ( 126.)	30	( 48.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0024_D	00000002 ( 2.)	31	( 49.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE

X_0024_T	000000FE	( 254.)	32 ( 50.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0025_D	00000002	( 2.)	33 ( 51.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0025_T	000000FE	( 254.)	34 ( 52.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0026_D	00000002	( 2.)	35 ( 53.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0026_T	000000FE	( 254.)	36 ( 54.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0027_D	00000002	( 2.)	37 ( 55.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0027_T	000000FE	( 254.)	38 ( 56.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0028_D	00000002	( 2.)	39 ( 57.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0028_T	000000FE	( 254.)	3A ( 58.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0029_D	00000002	( 2.)	3B ( 59.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0029_T	0000017E	( 382.)	3C ( 60.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0030_D	00000002	( 2.)	3D ( 61.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0030_T	0000017E	( 382.)	3E ( 62.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0031_D	00000002	( 2.)	3F ( 63.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0031_T	00000180	( 384.)	40 ( 64.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0031_GDCS	00000038	( 56.)	41 ( 65.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0031_ERTEMP	00000001	( 1.)	42 ( 66.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0031_FCACHE	00000001	( 1.)	43 ( 67.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0031_HFILL	00000044	( 68.)	44 ( 68.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0032_D	00000002	( 2.)	45 ( 69.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0032_T	000001FE	( 510.)	46 ( 70.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0033_D	00000002	( 2.)	47 ( 71.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0033_T	0000007E	( 126.)	48 ( 72.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0034_D	00000002	( 2.)	49 ( 73.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0034_T	0000017E	( 382.)	4A ( 74.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0035_D	00000002	( 2.)	4B ( 75.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0035_T	0000017E	( 382.)	4C ( 76.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0036_D	00000002	( 2.)	4D ( 77.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0036_T	00000101	( 257.)	4E ( 78.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0036_GDCS	00000017	( 23.)	4F ( 79.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0036_ERTEMP	00000001	( 1.)	50 ( 80.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0036_FCACHE	00000001	( 1.)	51 ( 81.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0036_HFILL	00000065	( 101.)	52 ( 82.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
X_0037_D	00000000	( 0.)	53 ( 83.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE
A_DIRECTORY	00000002	( 2.)	54 ( 84.)	NOPIC	USR	CON	REL	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE

Phase	Page faults	CPU Time	Elapsed Time
Initialization	25	00:00:00.04	00:00:01.02
Command processing	30	00:00:00.21	00:00:02.17
Pass 1	4309	00:01:48.35	00:04:04.07
Symbol table sort	3	00:00:00.70	00:00:01.05
Pass 2	4206	00:00:29.18	00:00:39.51
Symbol table output	34	00:00:00.21	00:00:00.46
Psect synopsis output	26	00:00:00.30	00:00:00.51
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	8637	00:02:19.00	00:04:48.80

The working set limit was 1000 pages.  
 1445173 bytes (2823 pages) of virtual memory were used to buffer the intermediate code.  
 There were 30 pages of symbol table space allocated to hold 362 non-local and 234 local symbols.  
 5097 source lines were read in Pass 1, producing 194 object records in Pass 2.  
 99 pages of virtual memory were used to define 45 macros.

ZZ-ECKAF-3.1 Psect synopsis  
ECKAF  
VAX-11 Macro Run Statistics

VAX-11/750 RDM MICRO DIAGNOSTICS

N 9  
6-MAY-1982

Fiche 1 Frame N9  
6-MAY-1982 19:54:55 VAX-11 Macro V02.46  
6-MAY-1982 19:54:39 WRKDS0:[COMET.RDM]ECKAF.MAR;1  
Sequence 117  
Page 11 (1)

Macro library name

-----+  
! Macro lib  
Macros defined  
-----

WRKDS0:[COMET]COMETMAC.MLB;801  
SYSSYSROOT:[SYSLIB]STARLET.MLB;1  
TOTALS (all libraries)

45  
0  
45

1887 GETS were required to define 45 macros.

There were 0 errors, 1 warning and 0 information messages, on lines:  
1042 (1)

WRKDS0:[COMET.RDM]TITLE+\_\_DRBO:[COMET]COMETASM+WRKDS0:[COMET.RDM]ECKAF/LIS=ECKAF/OBJ=ECKAF