ECKAF·3.1 KC750 RDM MICRO DIAG

EP-ECKAF-DL-3.1
FICHE 1 OF 1

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

Table of contents

```
(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
```

```
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 ;
```

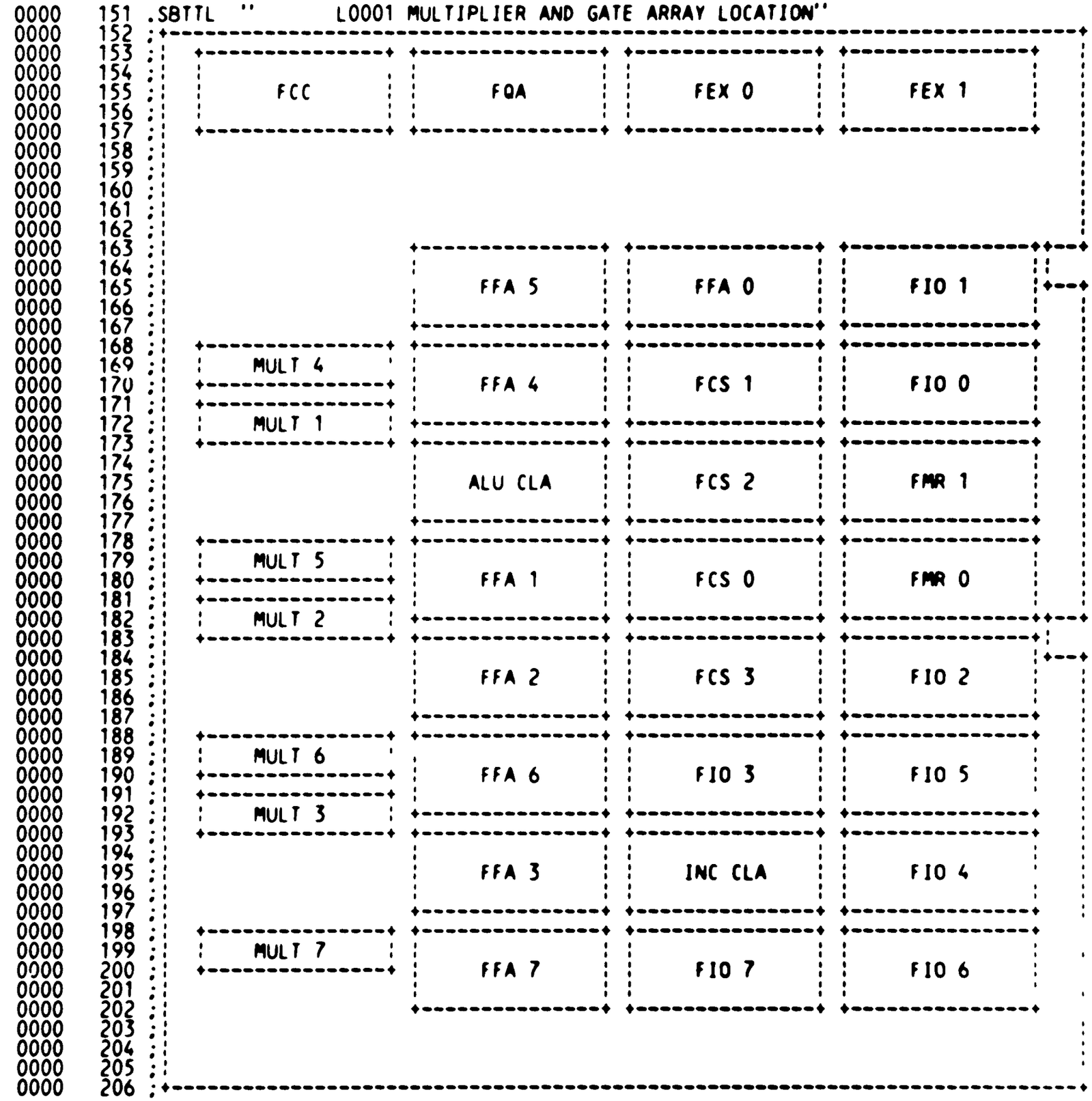
```
                        24          .SBTTL  'RDM Revision History'
                        25 ;++
                        26 ; RDM MICRO-DIAGNOSTIC PACKAGE
                        27 ;
                        28 ; REVISION HISTORY
                        29 ;
                        30 ; 00-01 Bill Landry     4-FEB-1980
                        31 ;       Started file.
                        32 ;
                        33 ; 00-02 Bill Landry     5-FEB-1980
                        34 ;       Added MAINT_DCS_ENABL bit to the MAIN32 register.
                        35 ;
                        36 ; 00-03 Bill Landry     27-FEB-1980
                        37 ;       Enhanced DCS DATA INTEGRITY test.
                        38 ;
                        39 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
                        40 ;       Added diagnostic macros to tile section and submitted to
                        41 ;       RELEASE ENGINERRING.
                        42 ;
                        43 ; 01-01 Bill Landry     25-AUG-1980
                        44 ;       Remodeled RDM to CMI tests.
                        45 ;
                        46 ; 02-00 Bill Landry     29-SEP-1980
                        47 ;       Second release to SDC
                        48 ;
                        49 ; 02-01 Bill Landry     4-NOV-1980
                        50 ;       Added test to check B CLOCK detection circuit
                        51 ;
                        52 ; 03-00 Bill Landry     17-NOV-1980
                        53 ;       Third release to SDC
                        54 ; 03-01 Dave Shull      08-Apr-1982
                        55 ;       Re-Compiled ECKAF.SRC as version 8.00 of the monitor has BEGINSA and
                        56 ;       ENDSA pseudos removed which changed the offset values for some of the
                        57 ;       pseudo opcodes.  Re-compiling the source file will assign the correct
                        58 ;       pseudo opcode values for compatibility with ECKAA.EXE ver. 800.
                        59 ;       Changed the CMC callout to MC0 and MC1.
                        60 ;
                        61 ;--
                         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-ZLI'0[] 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 ;       Deleated 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/a1,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/a1''
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/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS''
0000     89 ;FPA_M[] FPA_R[]                         'FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX/R
                                                    ROT/ZERO,ALU/OR''
0000     90 ;                                       'WB_RDM,FPA/FPA_DATA.WBUS''
0000     91 ;FPA_RDM                                'RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_DAT
0000     92 ;FPA_R[]                        'RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FPA/
0000     93 ;FPA_R[]+M[]                    'RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB''
0000     94 ;MEMSCAR_R[]                    'MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT''
0000     95 ;M[]_LONLIT                             'WB_RDM,SPW/MLONG,MSRC/a1''
0000     96 ;M[]_RDM                        'WCTRL/PC_PC+WB,WB_RDM''
0000     97 ;PC_PC+RDM                      'MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]''
0000     98 ;PC_PC+4 MB_XB                  'RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
0000     99 ;PC_PC+4 RDM_XB                          MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]''
0000    100 ;                               'WB_RDM,WCTRL/PC_WB''
0000    101 ;PC_RDM                                 ''CCPSL/PSL_WB.CCBR_ALUS,WB_RDM''
0000    102 ;PSL_RDM                       ''ALPCTL/WX_R.Q_D''
0000    103 ;Q_D_WX_R                      'RSRC/LONLIT,ALPCTL/WX_S.Q_R''
0000    104 ;Q_LONLIT                      ''ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE''
0000    105 ;Q_M[].RL.PTE                  'DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND''
0000    106 ;Q_R[].AND.Q                   'DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR''
0000    107 ;Q_R[].OR.Q                    ''ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P''
0000    108 ;Q_R[].RL.P                    ''ALPCTL/WB_ALUF,RDM_WB''
0000    109 ;RDM_ALUF                              'FPA/WBUS_FPA,RDM_WB''
0000    110 ;RDM_FPA                       'WB_LONLIT,RDM_WB''
0000    111 ;RDM_LONLIT                    'RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB''
0000    112 ;RDM_LONLIT.Q_M                         'MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB''
0000    113 ;RDM_M[]                       'WB_M[a1].OR.R[a2],RDM_WB''
0000    114 ;RDM_M[].OR.R[]                'WB_M[a1]-R[a2],RDM_WB''
0000    115 ;RDM_M[]-R[]                   'MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB''
0000    116 ;RDM_PC
```

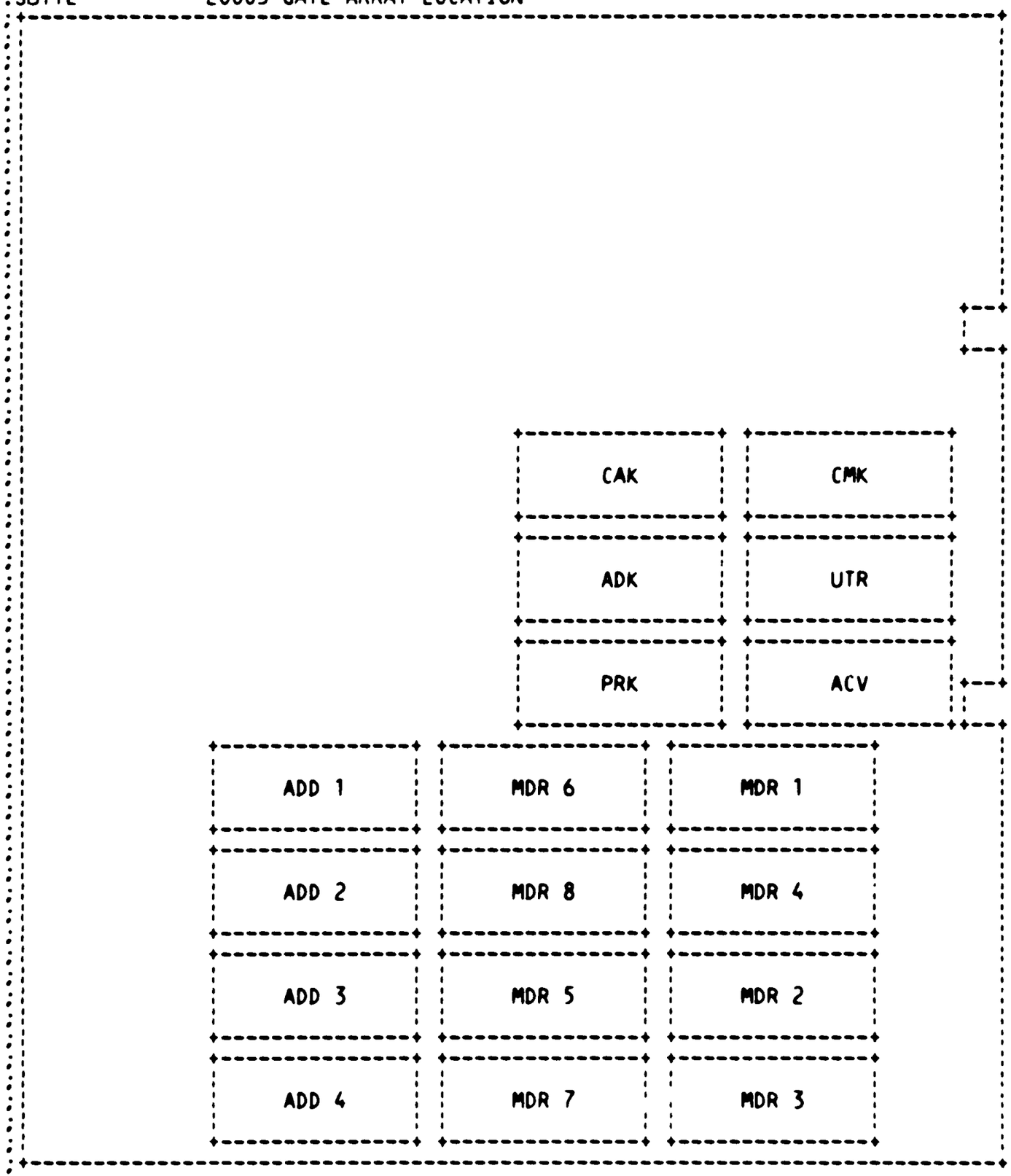
```
0000   117 ;RDM_PCBACK                     'RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB''
0000   118 ;RDM_Q                          'RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB''
0000   119 ;RDM_Q Q_D                      'ALPCTL/WX_Q.Q_D,RDM_WB''
0000   120 ;RDM_R[]                        'RSRC/a1,ROT7ZERO,MUX/R.S,ALU/OR,RDM_WB''
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/RSIZE,VSIZE/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-ZLIT0[]                  'WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,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''
```

```
                        151 .SBTTL  ''       L0001 MULTIPLIER AND GATE ARRAY LOCATION''
```



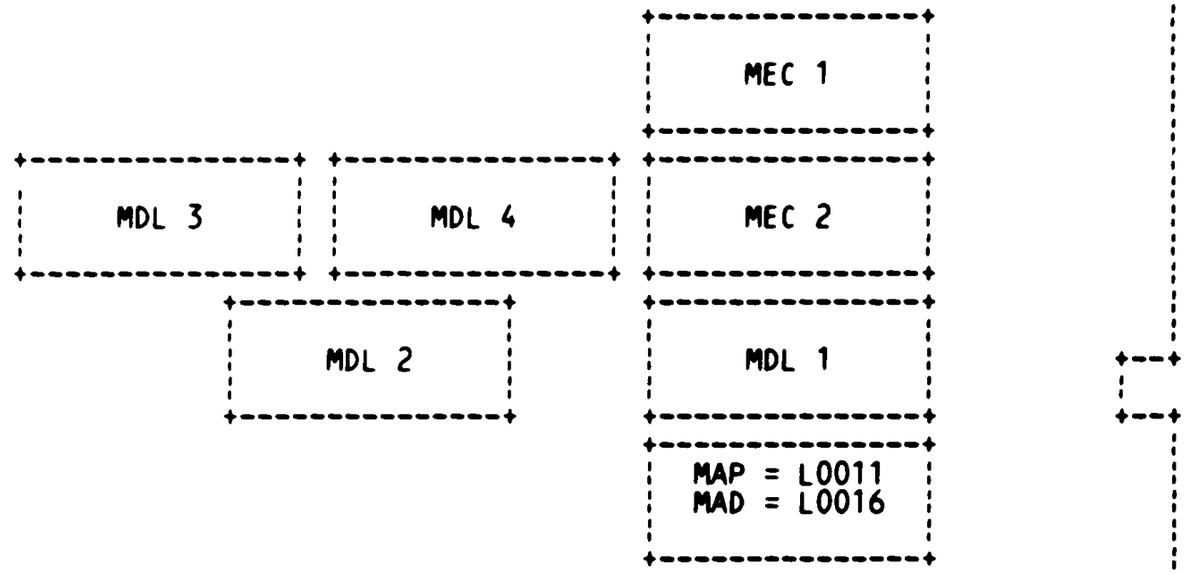
```
                        0000   208 .SBTTL  "       L0002 GATE ARRAY LOCATION"
                        0000   209 ;+-----------------------------------------------------------------------+
                        0000   210 ;|                                                                       |
                        0000   211 ;|                                                                       |
                        0000   212 ;|                                                                       |
                        0000   213 ;|                                                                       |
                        0000   214 ;|                                                                       |
                        0000   215 ;|                                                                       |
                        0000   216 ;|                                                                       |
                        0000   217 ;|                                                                       |
                        0000   218 ;|                    +---------------+ +---------------+              |
                        0000   219 ;|                    |               | |               |              |
                        0000   220 ;|                    |      PHB      | |      MSQ      |              |
                        0000   221 ;|                    |               | |               |              |
                        0000   222 ;|                    +---------------+ +---------------+              |
                        0000   223 ;|                    +---------------+ +---------------+              +---+
                        0000   224 ;|                    |               | |               |              |   |
                        0000   225 ;|                    |      CCC      | |      SAC      |              +---+
                        0000   226 ;|                    |               | |               |              |
                        0000   227 ;|                    +---------------+ +---------------+              |
                        0000   228 ;|  +---------------+ +---------------+ +---------------+              |
                        0000   229 ;|  |               | |               | |               |              |
                        0000   230 ;|  |      SPA      | |      SRK      | |      IRD      |              |
                        0000   231 ;|  |               | |               | |               |              |
                        0000   232 ;|  +---------------+ +---------------+ +---------------+              |
                        0000   233 ;|  +---------------+ +---------------+ +---------------+              |
                        0000   234 ;|  |               | |               | |               |              |
                        0000   235 ;|  |      TOK      | |      CLA      | |      ALK      |              |
                        0000   236 ;|  |               | |               | |               |              |
                        0000   237 ;|  +---------------+ +---------------+ +---------------+              |
                        0000   238 ;|                    +---------------+ +---------------+              |
                        0000   239 ;|                    |               | |               |              |
                        0000   240 ;|                    |     ALP 1     | |     ALP 5     |              |
                        0000   241 ;|                    |               | |               |              |
                        0000   242 ;|                    +---------------+ +---------------+              +---+
                        0000   243 ;|                    +---------------+ +---------------+              |   |
                        0000   244 ;|                    |               | |               |              +---+
                        0000   245 ;|                    |     ALP 2     | |     ALP 6     |              |
                        0000   246 ;|                    |               | |               |              |
                        0000   247 ;|                    +---------------+ +---------------+              |
                        0000   248 ;|                    +---------------+ +---------------+              |
                        0000   249 ;|                    |               | |               |              |
                        0000   250 ;|                    |     ALP 3     | |     ALP 7     |              |
                        0000   251 ;|                    |               | |               |              |
                        0000   252 ;|                    +---------------+ +---------------+              |
                        0000   253 ;|                    +---------------+ +---------------+              |
                        0000   254 ;|                    |               | |               |              |
                        0000   255 ;|                    |     ALP 4     | |     ALP 8     |              |
                        0000   256 ;|                    |               | |               |              |
                        0000   257 ;|                    +---------------+ +---------------+              |
                        0000   258 ;|  +---------------+ +---------------+ +---------------+ +-------------+|
                        0000   259 ;|  |               | |               | |               | |             ||
                        0000   260 ;|  |     SRM 1     | |     SRM 2     | |     SRM 3     | |    SRM 4    ||
                        0000   261 ;|  |               | |               | |               | |             ||
                        0000   262 ;|  +---------------+ +---------------+ +---------------+ +-------------+|
                        0000   263 ;+-----------------------------------------------------------------------+
```

```
                        265 .SBTTL  ''       L0003 GATE ARRAY LOCATION''
```



```
                        0000    323 .SBTTL  ''       L0004 GATE ARRAY LOCATION''
                        0000    324 ;+-----------------------------------------------------------------------+
                        0000    325 ;|                                                                       |
                        0000    326 ;|                                                                       |
                        0000    327 ;|                                                                       |
                        0000    328 ;|                                                                       |
                        0000    329 ;|                                                                       |
                        0000    330 ;|                                                                       |
                        0000    331 ;|                                                                       |
                        0000    332 ;|                                                                       |
                        0000    333 ;|                                                                       |
                        0000    334 ;|                                                                       |
                        0000    335 ;|                    +---------------+                                  |
                        0000    336 ;|                    |               |                                  |
                        0000    337 ;|                    |      UCN      |                                  |
                        0000    338 ;|                    |               |                                  |
                        0000    339 ;|                    +---------------+                                  |
                        0000    340 ;|                    +---------------+                                  |
                        0000    341 ;|                    |               |                                  |
                        0000    342 ;|                    |     UDP 1     |                                  |
                        0000    343 ;|                    |               |                                  |
                        0000    344 ;|                    +---------------+                                  |
                        0000    345 ;|                    +---------------+                                  |
                        0000    346 ;|                    |               |                                  |
                        0000    347 ;|                    |     UDP 2     |                                  |
                        0000    348 ;|                    |               |                                  |
                        0000    349 ;|                    +---------------+                                  |
                        0000    350 ;|                    +---------------+                                  |
                        0000    351 ;|                    |               |                                  |
                        0000    352 ;|                    |     UDP 4     |                                  |
                        0000    353 ;|                    |               |                                  |
                        0000    354 ;|                    +---------------+                                  |
                        0000    355 ;|                    +---------------+                                  |
                        0000    356 ;|                    |               |                                  |
                        0000    357 ;|                    |     UDP 3     |                                  |
                        0000    358 ;|                    |               |                                  |
                        0000    359 ;|                    +---------------+                                  |
                        0000    360 ;|                    +---------------+                                  |
                        0000    361 ;|                    |               |                                  |
                        0000    362 ;|                    |      INT      |                                  |
                        0000    363 ;|                    |               |                                  |
                        0000    364 ;|                    +---------------+                                  |
                        0000    365 ;|                    +---------------+                                  |
                        0000    366 ;|                    |               |                                  |
                        0000    367 ;|                    |     CON 1     |                                  |
                        0000    368 ;|                    |               |                                  |
                        0000    369 ;|                    +---------------+                                  |
                        0000    370 ;|                    +---------------+                                  |
                        0000    371 ;|                    |               |                                  |
                        0000    372 ;|                    |     CON 2     |                                  |
                        0000    373 ;|                    |               |                                  |
                        0000    374 ;|                    +---------------+                                  |
                        0000    375 ;|                                                                       |
                        0000    376 ;|                                                                       |
                        0000    377 ;|                                                                       |
                        0000    378 ;|                                                                       |
                        0000    379 ;+-----------------------------------------------------------------------+
```

```
                        0000    381 .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
                        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        /WRKD$0:[COMET]COMETMAC/
                     0000   442          .LIST   ME
                     0000   443          .NLIST  CND,MC
                     0000                .SBTTL  "EQUATED SYMBOLS
                     0000          ;
                     0000          ; RDM REGISTER DEFINITIONS:
                     0000          ;
                     0000          ; NOTE: THESE REGITER 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                CMI_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              CMICTL=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
          0000        ;
          0000        ; RDM REGISTER BIT DEFINITIONS:
          0000        ;
          0000        ; 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
          0000        ;
          0000        ; FOLLOWING IS A FUNCTION TABLE OF THE CLOCK CONTROL BITS
          0000        ;
          0000        ;       CLK_CTRL 1,0    FUNCTION
          0000        ;           11            HALT
          0000        ;           01            SINGLE MICRO INSTRUCTION
          0000        ;           10            SINGLE TICK
          0000        ;           00            RUN
          0000        ;
          0000        ; DCS CONTROL FILE      ALL OF THESE BITS ARE ACTIVE LOW
          0000        ;
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
          0000        ;
          0000        ; CS ADDRESS MATCH REGISTER (HIGH)
          0000        ;
00000040  0000              VBUS_SERIAL_IN   =      ^X40    ; ACTIVE LOW
00000080  0000              VBUS_CLOCK       =      ^X80
          0000        ;
```

```
                        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
                        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            CPU_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
          0000            ;
          0000            ;
          0000            ;
00000040  0000                    PB_INIT         =       ^X40    ; ACTIVE LOW
00000080  0000                    FRONT_PNL_LOCK  =       ^X80
          0000            ;
          0000            ; DCS ADDRESS REGISTER READ ONLY
          0000            ;
00000040  0000                    CLK_CTRL_0_RO   =       ^X40    ; ACTIVE HIGH
00000080  0000                    CLK_CTRL_1_RO   =       ^X80    ; ACTIVE HIGH
          0000            ;
          0000            ; MISCELLANEOUS EQUATES:
          0000            ;
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                    MSTEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003F3  0000                    MSTEST_SIZE_D   = MSTEST_SIZE - 960
          0000                                              ; MAXIMUM SIZE OF TEST OVERLAY WITH
          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
          0000                                              ; START ADDRESS OF MEMORY ERROR LOG
          0000    445       .NLIST  ME
          0000    446       .LIST   MC
          0000    447       .SBTTL  ""
          0000    448       .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' 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
```

```
                     003B    60          LOADREG         DCSAD,1$,,B                 ; Write the register
                     0043    61          CMPREGMSK       CLKDCS,1$,,2$,,B,           ; Read the data back and check
                     004F    62          IFERROR         ,,<<RDM>>                   ; RDM bad if compare failed
                     0055    63          ENDLOOP         1                           ; Terminate the loop
                     0058    64          SKIP                                        ; Go to end of test
                     005F    65
                     005F    66
                     005F    67
                     005F    68
                     005F    69          BEGIN_TEST_DATA
                     005F                ;-------------------------------------------------------------------------
                     005F    70
                00000000 005F    71 1$:     .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 ALGORITHIM:
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 succesfully 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          ;-----------------------------------------------------------------------
                     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_CHK_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          ;-----------------------------------------------------------------------
                     0096   165          ENDTEST
```

```
                        .SBTTL  ''TEST   003     CLOCK CONTROL BITS
                    ;*****************************************************************************
                    ;++
                    ;
                    ; FUNCTIONAL DESCRIPTION:
                    ;
                    ;       This test ensures that the clock control bits in the DCS CONTROL
                    ;       register and the CLOCK CONTROL register (read only) function
                    ;       correctly.
                    ;
                    ;
                    ; TEST ALGORITHM:
                    ;
                    ;       1.  Clear MASTER HALT ENABLE BIT to allow uniterrupted operation
                    ;       2.  Create a test loop of 4 iterations
                    ;       3.  Load test pattern into clock control bits of DCS CONTROL REGISTER
                    ;       4.  Read CLOCK CONTROL & DCS ADDRESS REGISTER for compliment of
                    ;           test pattern.
                    ;       5.  If no error go to step 3 for remaining test patterns.
                    ;
                    ;
                    ; TEST PATTERNS:
                    ;
                    ;       0
                    ;       1
                    ;       2
                    ;       3
                    ;
                    ;
                    ; LOGIC DESCRIPTION:
                    ;
                    ;       This test checks the ability of the 8085 to load the clock control bits
                    ;       in the DCS CONTROL REGISTER.  The bits are then read back via the
                    ;       CLOCK CONTROL & DCS ADDRESS REGISTER.
                    ;
                    ;
                    ; ASSUMPTIONS:
                    ;
                    ;       This test assumes the power-up self test has run successfully.
                    ;
                    ;
                    ; ERROR DESCRIPTION:
                    ;
                    ;       EXPECTED CLOCK CONTROL & DCS ADDRESS REGISTER CONTENTS
                    ;       RECEIVED CLOCK CONTROL & DCS ADDRESS REGISTER CONTENTS
                    ;       LOOP COUNT
                    ;
                    ;--
                    ;*****************************************************************************


                            INITIALIZE                              ;Init cpu
0019                        LOADREG         RDCTRL,4$,,B            ; Clear Master Halt Enable
0021                        LOOP            I,1,4                   ; Init the test loop
0028                        ERRLOOP                                 ; Set the error loop address
002A                        LOADREG         DCSCR,1$,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         I                       ; 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       ;-----------------------------------------------------------------------
                     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       ;-----------------------------------------------------------------------
                     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 remaing test patterns
001F    266 ;       9.  When loop J exhusted go to step 1 until loop I is exhusted
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$,J,B            ; Test CONTROL FILE register
                                005A    318         IFERROR         ,,<<RDM>>               ; Failure in control file data
                                0060    319         ENDLOOP         J                       ; Terminate the test looo
                                0063    320         ENDLOOP         I                       ; End address loop
                                0066    321         SKIP                                    ; Go to the next test
                                006D    322
                                006D    323
                                006D    324
                                006D    325         BEGIN_TEST_DATA
                                006D            ;-------------------------------------------------------------------
                                006D    326
                    0F 33 55 AA 006D    327 1$:     .BYTE   ^XAA,^X55,^X33,^X0F             ; Test patterns that get loaded
07 06 05 04 03 02 01 00         0071    328 2$:     .BYTE   ^X0,^X1,^X2,^X3,^X4,^X5,^X6,^X7
0F 0E 0D 0C 0B 0A 09 08         0079    329         .BYTE   ^X8,^X9,^XA,^XB,^XC,^XD,^XE,^XF
17 16 15 14 13 12 11 1C         0081    330         .BYTE   ^X10,^X11,^X12,^X13,^X14,^X15,^X16,^X17
1F 1E 1D 1C 1B 1A 19 18         0089    331         .BYTE   ^X18,^X19,^X1A,^X1B,^X1C,^X1D,^X1E,^X1F
27 26 25 24 23 22 21 20         0091    332         .BYTE   ^X20,^X21,^X22,^X23,^X24,^X25,^X26,^X27
2F 2E 2D 2C 2B 2A 29 28         0099    333         .BYTE   ^X28,^X29,^X2A,^X2B,^X2C,^X2D,^X2E,^X2F
37 36 35 34 33 32 31 30         00A1    334         .BYTE   ^X30,^X31,^X32,^X33,^X34,^X35,^X36,^X37
3F 3E 3D 3C 3B 3A 39 38         00A9    335         .BYTE   ^X38,^X39,^X3A,^X3B,^X3C,^X3D,^X3E,^X3F
                                00B1    336
                                00B1    337         END_TEST_DATA
                                00B1
                                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                                      ; Init the cpu
001D    389         LOADREG         DCSAD,1$,,B                     ; Select address 0
0025    390         LOADREG         DCSDA,2$,,B                     ; Load the Control File
002D    391         LOADREG         DCSAD,1$,,B                     ; Select address 0
0035    392         LOADREG         DCSAD,1$,,B                     ; Select address 0
003D    393         LOADREG         DCSCR,3$,,B                     ; Clear the control file
0045    394         LOADREG         DCSCR,4$,,B                     ;  ...
004D    395         CMPREG          DCSCF,5$,,B                     ; Check the control file
```

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

```
                        .SBTTL  ''TEST   006     DCS CHIP SELECT COUNTER
                ;*******************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       This test checks that the DCS chip select counter counts thru 11
                ;       states and then causes the DCS ADDRESS register to increment.
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  Load the DCS ADDRESS register with 0
                ;       2.  Load 10 bytes into the DCS.
                ;       3.  Check that DCS ADDRESS register is still 0.
                ;       4.  Load 1 more byte into the DCS.
                ;       5.  Check that DCS ADDRESS register incremented to 1.
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       none
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       This test checks the DCS CHIP SELECT COUNTER, DECODER logic to
                ;       insure that the dcs address is incremented after 11 bytes are
                ;       loaded into dcs.
                ;
                ;
                ; ASSUMPTIONS:
                ;
                ;       This test assumes the power up self test has run successfully
                ;
                ;
                ; ERROR DESCRIPTION:
                ;
                ;       EXPECTED DCS ADDRESS REGISTER DATA
                ;       RECEIVED DCS ADDRESS REGISTER DATA
                ;       LOOP COUNT
                ;
                ;--
                ;*******************************************************************************


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
```

```
                     0068   466
                     0068   467
                     0068   468
                     0068   469          BEGIN_TEST_DATA
                     0068
                     0068          ;-----------------------------------------------------------------------
                     0068   470
00                   0068   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          ;-----------------------------------------------------------------------
                     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         ,,<<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
```

```
                                        0064
                                        0064        ;-------------------------------------------------------------------------
                                        0064    534 i$:                                             ; u-word for DCS
U 00, 7700,C800,0364,0300,0470,1801     0064    535 ;% 00   NOP
                                        006F    539
                                        006F    540 ;+
                                        006F    541 ; Following is the expected contents of the ADDRESS register each iteration
                                        006F    542 ; of the 'I' loop.
                                        006F    543 ;-
0C 0B 0A 09 08 07 06 05 04 03 02 01     006F    544 2$:     .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
                              3F        00AF    548 3$:     .BYTE   ^X3F                            ; Mask for DCS ADDRESS register
                              00        0080    549 4$:     .BYTE   0                               ; Starting address in DCS
                                        0081    550
                                        0081    551         END_TEST_DATA
                                        0081
                                        0081        ;-------------------------------------------------------------------------
                                        0081    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 ALGORITHIM:
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  ;***********************************************************************
                                      0018
                                      0018
                                      0018
                                      0018          INITIALIZE                                  ; Init the cpu
                                      001A  ;+
                                      001A  ; Load the control file of each micro word with a unique pattern
                                      001A  ;-
                                      001A          ERRLOOP                                     ; Error loop point
                                      001C          LOOP            I,1,64                      ; Loop to load the control file
                                      0023          LOADREG         DCSAD,1$,I,B                ; Load the ADDRESS register
                                      002B          LOADREG         DCSDA,1$,I,B                ; Load a data pattern into the
                                      0033                                                      ; control file
                                      0033          ENDLOOP         I                           ; End loop to load control file
                                      0036  ;+
                                      0036  ; Read the control file of each micro word and check for the correct pattern
                                      0036  ;-
                                      0036          LOOP            I,1,64                      ; Loop to read the control file
                                      003D          LOADREG         DCSAD,1$,I,B                ; Load the ADDRESS register
                                      0045          LOADREG         DCSAD,1$,I,B                ; Load the ADDRESS register
                                      004D          CMPREG          DCSCF,1$,I,B,               ; Check the control file
                                      0056          IFERROR         ,,<<RDM>>                   ; Incorrect data
                                      005C          ENDLOOP         I                           ; END loop to read control file
                                      005F          SKIP                                        ; Go to the next test
                                      0066
                                      0066
                                      0066          BEGIN_TEST_DATA
                                      0066
                                      0066  ;------------------------------------------------------------------
                                      0066  ;
                                      0066  ; Data loaded into the control file
                                      0066  ;
0B 0A 09 08 07 06 05 04 03 02 01 00   0066  1$:     .BYTE   0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20
         14 13 12 11 10 0F 0E 0D 0C   0072
20 1F 1E 1D 1C 1B 1A 19 18 17 16 15   0078          .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   0087
34 33 32 31 30 2F 2E 2D 2C 2B 2A 29   008F          .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   009B
                          3F 3E 3D    00A3          .BYTE   61,62,63
                                      00A6
                                      00A6          END_TEST_DATA
                                      00A6
                                      00A6  ;------------------------------------------------------------------
                                      00A6          ENDTEST
```

```
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 ALGORITHIM:
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 CSAMR 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 ;       55555555555555555555
0017   676 ;       33333333333333333333
0017   677 ;       0F0F0F0F0F0F0F0F0F0F
0017   678 ;       00FF00FF00FF00FF00FF
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$,,B                     ; uADDR INH and Par. check off
0087   730         CMPREGMSK       FRONTI,30$,,30$,,B,             ; Check parity error light
0093   731         IFERROR         ,,<<RDM>>                       ; Bit stuck in DCS
0099   732         ENDLOOP         I                               ; Terminate the program loop
009C   733
009C   734
009C   735
009C   736 ;///////////////////////////////////////////////////////////////////////////////
009C   737 ;+
009C   738 ;SUBTEST #2
009C   739 ;
009C   740 ;       Test all DCS locations with test pattern #2
009C   741 ;-
009C   742         SUBTEST                                         ; SUBTEST #2
009F
009F       ;////////////////////////////////////////////////////////////////////////
009F   743         INITIALIZE                                      ; Init the cpu
00A1   744         LOADREG         RDCTRL,11$,,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       FRONTI,31$,,30$,,B,         ; Check parity error light
0118   760          IFERROR         ,,<<RDM>>                   ; Bit stuck in DCS
011E   761          ENDLOOP         I                           ; 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$,J,B               ; Load test pattern
015E   781          ENDLOOP         J                           ; ...
0161   782          LOADREG         DCSAD,7$,I,B                ; Reload DCS ADD register
0169   783          LOADREG         DCSCR,4$,,B                 ; uADDR INH and Parity Check
0171   784          LOADREG         CSAMR,3$,,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$,,B                 ; uADDR INH and Par. check off
0191   788          CMPREGMSK       FRONTI,32$,,30$,,B,         ; Check parity error light
019D   789          IFERROR         ,,<<RDM>>                   ; Bit stuck in DCS
01A3   790          ENDLOOP         I                           ; Terminate the program loop
01A6   791
01A6   792
01A6   793
01A6   794  ;////////////////////////////////////////////////////////////////////////////////
01A6   795  ;+
01A6   796  ;SUBTEST #4
01A6   797  ;
01A6   798  ;       Test all DCS locations with test pattern #4
01A6   799  ;-
01A6   800          SUBTEST                                     ; SUBTEST #4
01A9
01A9        ;////////////////////////////////////////////////////////////////////////////////
01A9   801          INITIALIZE                                  ; Init the cpu
01AB   802          LOADREG         RDCTRL,11$,,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         DCSAD,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
022B   820
022B   821
022B   822
022B   823  ;//////////////////////////////////////////////////////////////////////////////
022B   824  ;+
022B   825  ;SUBTEST #5
022B   826  ;
022B   827  ;       Test all DCS locations with test pattern #5
022B   828  ;-
022B   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$,,B           ; ...
0251   836          LOADREG         DCSAD,7$,I,B            ; Address to DCS ADD register
0259   837          LOOP            J,1,11                  ; Loop to load test pattern
0260   838          LOADREG         DCSDA,24$,J,B           ; Load test pattern
0268   839          ENDLOOP         J                       ; ...
0268   840          LOADREG         DCSAD,7$,I,B            ; Reload DCS ADD register
0273   841          LOADREG         DCSCR,4$,,B             ; uADDR INH and Parity Check
027B   842          LOADREG         CSAMR,3$,,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$,,B             ; uADDR INH and Par. check off
029B   846          CMPREGMSK       FRONT1,34$,,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
```

```
038D          ;////////////////////////////////////////////////////////////////////////////
038D    917           INITIALIZE                                      ; Init the cpu
038F    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                    ; Address to DCS ADD register
03E8    924           LOOP            J,1,11                          ; Loop to load test pattern
03EF    925           LOADREG         DCSDA,27$,J,B                   ; Load test pattern
03F7    926           ENDLOOP         J                               ;
03FA    927           LOADREG         DCSAD,7$,I,B                    ; Reload DCS ADD register
0402    928           LOADREG         DCSCR,4$,,B                     ; uADDR INH and Parity Check
040A    929           LOADREG         CSAMR,3$,,W                     ; 1800 to Match Register
0412    930           LOADREG         DCSCR,5$,,B                     ; Tick the clock
041A    931           LOADREG         DCSCR,4$,,B                     ; one micro instruction
0422    932           LOADREG         DCSCR,6$,,B                     ; uADDR INH and Par. check off
042A    933           CMPREGMSK       FRONT1,37$,,30$,,B,             ; Check parity error light
0436    934           IFERROR         ,,<<RDM>>                       ; Bit stuck in DCS
043C    935           ENDLOOP         I                               ; Terminate the program loop
043F    936
043F    937
043F    938
043F    939 ;////////////////////////////////////////////////////////////////////////////
043F    940 ;+
043F    941 ;SUBTEST #9
043F    942 ;
043F    943 ;       Test all DCS locations with test pattern #9
043F    944 ;-
043F    945           SUBTEST                                         ; SUBTEST #9
0442
0442          ;////////////////////////////////////////////////////////////////////////////
0442    946           INITIALIZE                                      ; Init the cpu
0444    947           LOADREG         RDCTRL,11$,,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$,I,B                    ; Address to DCS ADD register
046D    953           LOOP            J,1,11                          ; Loop to load test pattern
0474    954           LOADREG         DCSDA,28$,J,B                   ; Load test pattern
047C    955           ENDLOOP         J                               ;
047F    956           LOADREG         DCSAD,7$,I,B                    ; Reload DCS ADD register
0487    957           LOADREG         DCSCR,4$,,B                     ; uADDR INH and Parity Check
048F    958           LOADREG         CSAMR,3$,,W                     ; 1800 to Match Register
0497    959           LOADREG         DCSCR,5$,,B                     ; Tick the clock
049F    960           LOADREG         DCSCR,4$,,B                     ; one micro instruction
04A7    961           LOADREG         DCSCR,6$,,B                     ; uADDR INH and Par. check off
04AF    962           CMPREGMSK       FRONT1,38$,,30$,,B,             ; Check parity error light
04BB    963           IFERROR         ,,<<RDM>>                       ; Bit stuck in DCS
04C1    964           ENDLOOP         I                               ; Terminate the program loop
04C4    965
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$,,B,             ; Check parity error light
                        0540   992         IFERROR         ,,<<RDM>>                       ; Bit stuck in DCS
                        0546   993         ENDLOOP         I                               ; Terminate the program loop
                        0549   994         SKIP                                            ; Go to next test
                        0550   995
                        0550   996
                        0550   997
                        0550   998         BEGIN_TEST_DATA
                        0550
                        0550       ;-------------------------------------------------------------------------
                        0550    999
                    40  0550   1000 1$:     .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$:
U 00, 7700,C800,0364,0300,0470,1801  0597   1015 ;% 00  NOP                                 ; Good data for dcs
                    00  05A2   1019 11$:    .BYTE   0
 00 00 00 00 00 00 00 00 00 00 FF  05A3   1020 20$:    .BYTE   ^XFF,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00,^X00
 FF FF FF FF FF FF FF FF FF FF FF  05AE   1021 21$:    .BYTE   ^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF,^XFF
 AA AA AA AA AA AA AA AA AA AA FF  05B9   1022 22$:    .BYTE   ^XFF,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA,^XAA
 55 55 55 55 55 55 55 55 55 55 FF  05C4   1023 23$:    .BYTE   ^XFF,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55,^X55
 33 33 33 33 33 33 33 33 33 33 FF  05CF   1024 24$:    .BYTE   ^XFF,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33,^X33
 0F 0F 0F 0F 0F 0F 0F 0F 0F 0F FF  05DA   1025 25$:    .BYTE   ^XFF,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F,^X0F
 FF 00 FF 00 FF 00 FF 00 FF 00 FF  05E5   1026 26$:    .BYTE   ^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF,^X00,^XFF
```

```
   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;
```

```
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            I,1,4                   ; Loop for 4 patterns
0021  1098          LOADREG         RDCTRL,4$,,B            ; 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$,,W,               ; Check the CS Address Bus
                                0043  1103          IFERROR         ,,<<RDM>>                       ; Check if there is an error
                                0049  1104          ENDLOOP         I                               ; End test loop
                                004C  1105          SKIP                                            ; Go to next test
                                0053  1106
                                0053  1107          BEGIN_TEST_DATA
                                0053
                                0053          ;-----------------------------------------------------------------------
                                0053  1108  ;
                                0053  1109  ; The following micro words contain the test patterns for the HIGH NEXT field
                                0053  1110  ;
                                0053  1111  1$:
U 00, 7700,C800,0364,0300,0470,2AAA  0053  1112  ;% 00    NEXT/2AAA
U 00, 7700,C800,0364,0300,0470,1555  005E  1116  ;% 00    NEXT/1555
U 00, 7700,4800,0364,0300,0470,3333  0069  1120  ;% 00    NEXT/3333
U 00, 7700,4800,0364,0300,0470,0F0F  0074  1124  ;% 00    NEXT/0F0F
                                007F  1128
                  0F0F 3333 1555 2AAA  007F  1129  2$:     .WORD   ^X2AAA,^X1555,^X3333,^X0F0F     ; Expected CS ADDRESS
                                0087  1130
                               1FC0  0087  1131  3$:     .WORD   ^X1FC0                          ; CS ADDRESS mask
                                 00  0089  1132  4$:     .BYTE   0
                                008A  1133
                                008A  1134          END_TEST_DATA
                                008A
                                008A          ;-----------------------------------------------------------------------
                                008A  1135          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            I,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         ,,<<RDM>>               ; 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          ;-----------------------------------------------------------------------
                                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          ;-----------------------------------------------------------------------
                                      009C  1229          ENDTEST
```

```
                        0015       .SBTTL  ''TEST   00C     TRAP HALT ENABLE
                        0015  1231 ;*******************************************************************************
                        0015  1232 ;++
                        0015  1233 ;
                        0015  1234 ; FUNCTIONAL DESCRIPTION:
                        0015  1235 ;
                        0015  1236 ;       This test checks the functions of the TRAP HALT ENABLE bit and
                        0015  1237 ;       the MASTER HALT ENABLE bit.
                        0015  1238 ;
                        0015  1239 ;
                        0015  1240 ; TEST ALGORITHM:
                        0015  1241 ;
                        0015  1242 ;       SUBTEST #1
                        0015  1243 ;       1.  Get control of CS ADDRESS BUS and load a 0 address
                        0015  1244 ;       2.  Clear the control file in all 64 DCS locations
                        0015  1245 ;       3.  Remove 0 address from CS ADDRESS BUS
                        0015  1246 ;
                        0015  1247 ;       SUBTEST #2
                        0015  1248 ;       1.  Get control of CS ADDRESS BUS and load a 0 address
                        0015  1249 ;       2.  Set MASTER HALT ENABLE & TRAP ENABLE
                        0015  1250 ;       3.  Let the clock free run
                        0015  1251 ;       4.  Test the CLKDCS register to see that clock stopped
                        0015  1252 ;       5.  Turn clock off manually to be sure its off
                        0015  1253 ;
                        0015  1254 ;       SUBTEST #3
                        0015  1255 ;       1.  Get control of CS ADDRESS BUS and load a 0 address
                        0015  1256 ;       2.  Set MASTER HALT ENABLE but not TRAP ENABLE
                        0015  1257 ;       3.  Let the clock free run
                        0015  1258 ;       4.  Test the CLKDCS register to see that the clock is running
                        0015  1259 ;       5.  Turn clock off
                        0015  1260 ;
                        0015  1261 ;       SUBTEST #4
                        0015  1262 ;       1.  Get control of CS ADDRESS BUS and load 1800 for address
                        0015  1263 ;       2.  Tick the clock to enable the first dcs u-inst on the bus
                        0015  1264 ;       3.  Set MASTER HALT ENABLE & TRAP ENABLE
                        0015  1265 ;       4.  Let the clock free run
                        0015  1266 ;       5.  Test the CLKDCS register to see that the clock is running
                        0015  1267 ;       6.  Turn clock off
                        0015  1268 ;
                        0015  1269 ;       SUBTEST #5
                        0015  1270 ;       1.  Get control of CS ADDRESS BUS and load a 0 address
                        0015  1271 ;       2.  Clear MASTER HALT ENABLE, set TRAP ENABLE
                        0015  1272 ;       3.  Let the clock free run
                        0015  1273 ;       4.  Test the CLKDCS register to see that the clock is running
                        0015  1274 ;       5.  Turn clock off
                        0015  1275 ;
                        0015  1276 ;
                        0015  1277 ; TEST PATTERNS:
                        0015  1278 ;
                        0015  1279 ;       NONE
                        0015  1280 ;
                        0015  1281 ;
                        0015  1282 ; LOGIC DESCRIPTION:
                        0015  1283 ;
                        0015  1284 ;       This test attempts to check the MASTER HALT ENABLE & TRAP HALT
                        0015  1285 ;       ENABLE bits to ensure they can control the clock properly.
                        0015  1286 ;       Any error found should be in RDCTRL register or the clock control
```

```
                     0015  1287 ;       logic.
                     0015  1288 ;
                     0015  1289 ; ASSUMPTIONS:
                     0015  1290 ;
                     0015  1291 ;       This test assumes the power up self test has run successfully
                     0015  1292 ;
                     0015  1293 ;
                     0015  1294 ; ERROR DESCRIPTION:
                     0015  1295 ;
                     0015  1296 ;       EXPECTED CLKDCS REGISTER CONTENTS
                     0015  1297 ;       RECEIVED CLKDCS REGISTER CONTENTS
                     0015  1298 ;
                     0015  1299 ;--
                     0015  1300 ;*****************************************************************************
                     0015  1301 ;/////////////////////////////////////////////////////////////////////////////
                     0015  1302 ;+
                     0015  1303 ;SUBTEST #1
                     0015  1304 ;
                     0015  1305 ;       Clear the stop clock bit in all 64 words of DCS
                     0015  1306 ;-
                     0015  1307         SUBTEST                                 ; SUBTEST #1
                     0018
                     0018       ;/////////////////////////////////////////////////////////////////////////////
                     0018  1308         INITIALIZE                              ; Init the cpu
                     001A  1309         LOADREG     DCSCR,1$,,B                 ; Make sure DCS is not selected
                     0022  1310         LOADREG     CSAMR,2$,,W                 ; Address 0 on CS BUS
                     002A  1311         LOOP        I,1,64                      ; Create a test loop of 64
                     0031  1312         LOADREG     DCSAD,40$,I,B               ; Select a DCS address
                     0039  1313         LOADREG     DCSDA,41$,,B                ; Clear the control file
                     0041  1314         ENDLOOP     I                           ; End test loop
                     0044  1315         LOADREG     DCSCR,5$,,B                 ; Get off CS BUS
                     004C  1316
                     004C  1317
                     004C  1318
                     004C  1319 ;/////////////////////////////////////////////////////////////////////////////
                     004C  1320 ;+
                     004C  1321 ;SUBTEST #2
                     004C  1322 ;
                     004C  1323 ;       Check that the clock stops if the DCS is not selected
                     004C  1324 ;-
                     004C  1325         SUBTEST                                 ; SUBTEST #2
                     004F
                     004F       ;/////////////////////////////////////////////////////////////////////////////
                     004F  1326         LOADREG     DCSCR,1$,,B                 ; Set parity check
                     0057  1327         LOADREG     CSAMR,2$,,W                 ; 0 on CS BUS
                     005F  1328         ERRLOOP                                 ; Error loop point
                     0061  1329         LOADREG     RDCTRL,2$,,B                ; Clear Master Halt Enable
                     0069  1330         LOADREG     RDCTRL,3$,,B                ; Master Hlt Enabl and Trap Enable
                     0071  1331         LOADREG     DCSCR,6$,,B                 ; Start the clock
                     0079  1332         CMPREGMSK   CLKDCS,4$,,4$,,B,           ; Check that clock stopped
                     0085  1333         LOADREG     DCSCR,5$,,B                 ; Stop the clock
                     008D  1334         IFERROR     ,,<<RDM>>                   ; Clock didn't stop
                     0093  1335
                     0093  1336
                     0093  1337
                     0093  1338 ;/////////////////////////////////////////////////////////////////////////////
                     0093  1339 ;+
```

```
                        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
                        00A8  1348          LOADREG         RDCTRL,2$,,B                    ; Clear MASTER HALT ENABLE
                        00B0  1349          LOADREG         RDCTRL,10$,,B                   ; Set MASTER HALT ENABLE
                        00B8  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
                        0128  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$,,B             ; Stop the clock
                                0172  1393          IFERROR         ,,<<RDM>>               ; Clock had stopped
                                0178  1394          SKIP                                    ; Go to next test
                                017F  1395
                                017F  1396          BEGIN_TEST_DATA
                                017F
                                017F          ;-------------------------------------------------------------------------
                                017F  1397
                             E8 017F  1398 1$:      .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$:     .BYTE   TRAP_HALT_EN
                                0188  1411
08 0A 09 08 07 06 05 04 03 02 01 00  0188  1412 40$:     .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
         2A 29 28 27 26 25 24 23     01AE
36 35 34 33 32 31 30 2F 2E 2D 2C 28  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
```

```
                        .SBTTL  ''TEST   00D     DISABLE HIGH NEXT
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       Check that DISABLE HIGH NEXT disables the HIGH NEXT drivers.
                ;
                ; TEST ALGORITHM:
                ;
                ;       1. Load the HIGH NEXT latch with a known test pattern
                ;       2. Assert uADDR INHIBIT.
                ;       3. Check that CS ADDRESS is not the contents of the CS ADDR Buffer
                ;
                ; TEST PATTERNS:
                ;
                ;       2AAA
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       This test detects whether or not the DISABLE HIGH NEXT signal coming
                ;       from COMET is able to shut of the HIGH NEXT FIELD bus drivers.
                ;
                ; ASSUMPTIONS:
                ;
                ;       This test assumes that COMET is sending the DISABLE HIGH NEXT signal
                ;       when it receives uADDR INHIBIT from the RDM.
                ;
                ; ERROR DESCRIPTION:
                ;
                ;       NOT EXPECTED CSBUF DATA
                ;       RECEIVED CSBUF DATA
                ;
                ;--
                ;*****************************************************************************


                        INITIALIZE                                      ; Init the cpu
0018            1464    LOADDCS         1$,,0,1                         ; Load test micro word
001F            1465    ERRLOOP                                         ; Error loop
0021            1466    FETCH           0                               ; Latch the micro word
0026            1467    LOADREG         DCSCR,2$,,B                     ; Ascert uADDR INHIBIT
002E            1468    CMPREGMSK       CSBUF,5$,,3$,,W,NE              ; Test for not the test pattern
003A            1469    LOADREG         DCSCR,4$,,B                     ; Deascert uADDR INHIBIT
0042            1470    IFERROR         ,,<<RDM>>                       ; Check if error
0048            1471    SKIP                                            ; Go to next test
004F            1472
004F            1473    BEGIN_TEST_DATA
004F
                ;-------------------------------------------------------------------------
004F            1474
```

```
                                 004F  1475 1$:
U 00, 7700,C800,0364,0300,0470,2AAA  004F  1476 ;% 00   NEXT/2AAA
                                 005A  1480
                                  E0  005A  1481 2$:     .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  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 instrution.
                        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$,,B,       ; Test the ADDRESS REG
                        003B  1536          IFERROR         ,,<<RDM>>               ; Check if error
                        0041  1537          SKIP                                    ; Go to the next test
                        0048  1538
                        0048  1539          BEGIN_TEST_DATA
                        0048        ;-----------------------------------------------------------------
                        0048  1540
                        0048  1541 1$:
U 00, 7700,C800,0364,0300,0470,1801     0048  1542 ;% 00    NOP                     ; NOP Micro Instruction
```

```
                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   00F     DCS CYCLE TEST
0013  1552 ;*****************************************************************************
0013  1553 ;++
0013  1554 ;
0013  1555 ; FUNCTIONAL DESCRIPTION:
0013  1556 ;
0013  1557 ;       This test checks that NOP micro instrucitons 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
0013  1600         ERRLOOP                                 ; Error loop point
0015  1601         INITIALIZE                              ; INIT THE CPU
0017  1602         LOADDCS         1$,,0,64                ; Load the DCS
001E  1603         FETCH           0                       ; Fetch location 0
0023  1604         LOADREG         DCSCR,2$,,B             ; Start the clock
002B  1605         CMPREGMSK       CLKDCS,4$,,3$,,B,       ; Test clock stop bits
0037  1606         LOADREG         DCSCR,5$,,B             ; Stop the clock
003F  1607         IFERROR         ,,<<RDM>>               ; Unexpected clock stop
```

```
                                0045  1608          SKIP                                    ; Go to next text
                                004C  1609
                                004C  1610          BEGIN_TEST_DATA
                                004C
                                004C          ;-------------------------------------------------------------------------
                                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
```

```
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$:      .BYTE   PAR_STOP_ENABL
                                  C0 030D  1871 3$:      .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         2$,,0,63                ; Load nops into DCS
0030  1927         FETCH           0                       ; Start DCS program
0035  1928         BRSTCLK         62                      ; End DCS program
0039  1929         CMPREG          CLKDCS,1$,,B,           ; Test stop address
0042  1930         IFERROR         ,,<<RDM>>               ; 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
```

```
U 00, 7700,C800,0364,0300,0470,1801  0276  2139 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  0281  2143 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  028C  2147 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  0297  2151 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  02A2  2155 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  02AD  2159 ;% C0   NOP
U 00, 7700,C800,0364,0300,0470,1801  02B8  2163 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  02C3  2167 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  02CE  2171 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  02D9  2175 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  02E4  2179 ;% 00   NOP
U 00, 7700,C800,0364,0300,0470,1801  02EF  2183 ;% 00   NOP
U 00, 7300,C800,0364,0300,0470,1801  02FA  2187 ;% 00   NOP,STOP.CLOCK
                                     0305  2191
                                     0305  2192         END_TEST_DATA
                                     0305
                                     0305        ;-------------------------------------------------------------------------
                                     0305  2193         ENDTEST
```

```
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                                  ; Init the cpu
001F  2240          LOADREG         RDCTRL,6$,,B                ; Clear MASTER HALT ENABLE
0027  2241          LOADDCS         1$,,0,3                     ; Load NOP micro instructions
002E  2242          FETCH           0                           ; Start execution
0033  2243          LOOP            1,1,2                       ; Loop to tick the clock
003A  2244          LOADREG         DCSCR,2$,,B                 ; Tick the clock
0042  2245          LOADREG         DCSCR,3$,,B                 ; ...
004A  2246          ENDLOOP         1                           ; End loop
004D  2247          CMPREGMSK       CLKDCS,4$,,5$,,B,           ; Test DCS ADDRESS
0059  2248          IFERROR         ,,<<RDM>>                   ; Incorrect address
005F  2249          SKIP                                        ; Go to next test
0066  2250
```

```
                                0066  2251          BEGIN_TEST_DATA
                                0066
                                0066          ;-------------------------------------------------------------------------
                                0066  2252
                                0066  2253 1$:                                                  ; Micro code
U 00, 7700,4800,0364,0300,0470,0000  0066  2254 ;% C0    NOP,NEXT/0
U 01, 7700,C800,0364,0300,0470,1802  0071  2258 ;% 01    NOP
U 02, 7700,4800,0364,0300,0470,1803  007C  2262 ;% 02    NOP
                                0087  2266
                             20 0087  2267 2$:       .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          ;-------------------------------------------------------------------------
                                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 ;       AAAAAAAA
0026  2305 ;       55555555
0026  2306 ;       33333333
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                                    ; Go to next test
                        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
```

```
                     ;       are operational.
                     ;
                     ; ERROR DESCRIPTION:
                     ;
                     ;       SUBTEST #1 & 2
                     ;               EXPECTED STATUS REGISTER CONTENTS
                     ;               RECEIVED STATUS REGISTER CONTENTS
                     ;               LOOP COUNT
                     ;
                     ;       SUBTEST #3
                     ;               EXPECTED VBUS BIT 1D
                     ;               RECEIVED VBUS BIT 1D
                     ;                       Bit<8> contains the value
                     ;                       Bits <7:0> contain the bit position
                     ;
                     ;--
                     ;*****************************************************************************
                     ;/////////////////////////////////////////////////////////////////////////////
                     ;+
                     ; SUBTEST #1
                     ;
                     ;       Fill the VBUS with all 0's and check the output
                     ;-
0018  2448                   SUBTEST                                    ; SUBTEST #1
001B
001B                 ;/////////////////////////////////////////////////////////////////////////////
001B  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                                  ; End test loop
0039  2455           LOOP            I,1,64                             ; Create a test loop of 64
0040  2456           CMPREGMSK       STATUS,2$,,3$,,B,                  ; Check for zero's coming out
004C  2457           IFERROR         ,,<<RDM>>                          ; Incorrect data
0052  2458           LOADREG         CSAMR,4$,,W                        ; Clock the VBUS
005A  2459           LOADREG         CSAMR,5$,,W                        ;  ...
0062  2460           ENDLOOP         I                                  ; End test loop
0065  2461
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                 ;/////////////////////////////////////////////////////////////////////////////
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                                  ; End test loop
0084  2475           LOOP            I,1,64                             ; Check 64 bits
0088  2476           CMPREGMSK       STATUS,3$,,3$,,B                   ; 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
```

```
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
```

```
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         I                       ; End test loop
                     007A  2630         SKIP                                    ; Go to next test
                     0081  2631
                     0081  2632         BEGIN_TEST_DATA
                     0081
                     0081               ;-------------------------------------------------------------------------
                     0081  2633
                       E8  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
          0FFF 07FF 03FF 01FF 00FF  0090
                     3FFF 1FFF  009A  2636          .WORD   ^X1FFF,^X3FFF
                       A8  009E  2637 3$:      .BYTE   CLK_CTRL_0!PAR_CHK_ENABL!MICRO_ADDR_INH
0080 0040 0020 0010 0008 0004 0002  009F  2638 4$:      .WORD   2,4,8,^X10, ^X20, ^X40, ^X80,^X100,^X200,^X400,^X800
               0800 0400 0200 0100  00AD
                    0000 2000 1000  00B5  2639          .WORD   ^X1000,^X2000,0
                     1FFF  00B8  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               ;-------------------------------------------------------------------------
                     00BF  2645         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                              ; Init the cpu
                        0024   2698         LDFIELD         8$,,B,11$,78,78,,,,NP,  ; Force parity error
                        0035   2699         LOADDCS         10$,,0,6                ; Load test u-code
                        003C   2700         FETCH           0,INH                   ; Start DCS program
                        0041   2701         LOADREG         CSAMR,7$,,W             ; Load 1800 into CSAMR
                        0049   2702         LOADREG         DCSCR,1$,,B             ; 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
                        008A        ;-----------------------------------------------------------------------
                        008A  2713
                     E8 008A  2714 1$:      .BYTE   CLK_CTRL_0!CLK_CTRL_1!-         ; Enable CSAMR onto CS
                        0088  2715                  PAR_CHK_ENABL!MICRO_ADDR_INH    ; BUS with clock halted
                     88 0088  2716 2$:      .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
                        00D7        ;-----------------------------------------------------------------------
                        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  ;
0014  2764  ; TEST ALGORITHM:
0014  2765  ;
0014  2766  ;       1.  Create a test loop of 10
0014  2767  ;       2.  Load the MAIN32 register with not the TRPOFF bit
0014  2768  ;       3.  Perform step 2 ten times
0014  2769  ;       4.  Test STATUS register to see that TRPOFF bit is clear
0014  2770  ;       5.  Create a test loop of 10
0014  2771  ;       6.  Set the TRPOFF bit in the MAIN32 register
0014  2772  ;       7.  Perform step 6 ten times
0014  2773  ;       8.  Test STATUS register to see that TRPOFF bit is set
0014  2774  ;       9.  If no error go to next test
0014  2775  ;
0014  2776  ;
0014  2777  ; TEST PATTERNS:
0014  2778  ;
0014  2779  ;       none
0014  2780  ;
0014  2781  ;
0014  2782  ; LOGIC DESCRIPTION:
0014  2783  ;
0014  2784  ;       This test clears and sets the TRPOFF bit in the MAIN32 register 10
0014  2785  ;       times.  This is done to allow you to scope out the 8085 trap logic
0014  2786  ;       should you suspect errors.  Otherwise the only indication of failure
0014  2787  ;       you will get with this test is if the STATUS register is unable to read
0014  2788  ;       the TRPOFF bit back.
0014  2789  ;
0014  2790  ;
0014  2791  ; ASSUMPTIONS:
0014  2792  ;
0014  2793  ;       This test assumes the power up self test has run successfully
0014  2794  ;
0014  2795  ;
0014  2796  ; ERROR DESCRIPTION:
0014  2797  ;
0014  2798  ;       EXPECTED STATUS REGISTER CONTENTS
0014  2799  ;       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
```

```
                     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$,,B,               ; Test for TRPOFF set
                     005A  2816          IFERROR         ,,<<RDM>>                       ; Bit not set
                     0060  2817          SKIP                                            ; Go to next test
                     0067  2818
                     0067  2819
                     0067  2820          BEGIN_TEST_DATA
                     0067
                     0067          ;-------------------------------------------------------------------------
                     0067  2821
                05   0067  2822 1$:      .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
                     006B          ;------------------------------------------------------- -------------
                     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
0038  2974         CMPREG          CLKDCS,3$,,B            ; Check for clock stopped and
0044  2975                                                 ; correct address
```

```
                                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
```

```
                        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         ,,<<RDM>>               ; See if error
                        005F  3069          ENDLOOP         I                       ; End test loop
                        0062  3070          SKIP                                    ; Go to next test
                        0069  3071
                        0069  3072          BEGIN_TEST_DATA
                        0069
                        0069          ;-----------------------------------------------------------------------
                        0069  3073
                        0069  3074 1$:                                              ; Micro code
U 00, 7700,7800,7FFF,FFFF,8470,1801     0069  3075 ;% 00    LONLIT [0]
U 01, 6700,8800,0047,03D4,0470,1802     0074  3079 ;% 01    RDM_LONLIT.Q_M
U 02, 7300,4800,0364,0300,0470,1803     007F  3083 ;% 02    STOP.CLOCK
                        008A  3087
                    00000000  008A  3088 2$:      .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          ;-----------------------------------------------------------------------
                        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 patte   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         1$,,0,4                 ; Load the micro instructions
001C  3150         LOOP            I,1,6                   ; Create a test loop of 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         1                       ; End test loop
                                006C  3162          SKIP                                    ; Go to next test
                                0073  3163
                                0073  3164          BEGIN_TEST_DATA
                                0073
                                0073          ;-----------------------------------------------------------------------
                                0073  3165
                                0073  3166 1$:                                              ; Micro code
U 00, 7700,C800,0364,0300,0470,1801  0073  3167 ;% 00     NOP
U 01, 5700,8840,0364,0300,0470,1802  007E  3171 ;% 01     R[0]_RDM
U 02, 6700,4800,5BE4,0300,0470,1803  0089  3175 ;% 02     RDM_WB,WB_R[0]
U 03, 7300,C800,0364,0300,0470,1804  0094  3179 ;% 03     STOP.CLOCK
                                     009F  3183
                            00000000 009F  3184 2$:       .LONG   0
                                  00 00A3  3185 3$:       .BYTE   0
                                  08 00A4  3186 4$:       .BYTE   MASTER_HALT_EN
                                  60 00A5  3187 5$:       .BYTE   CLK_CTRL_1!CLK_CTRL_0
                                     00A6  3188
                                     00A6  3189          END_TEST_DATA
                                     00A6
                                     00A6          ;-----------------------------------------------------------------------
                                     00A6
                                     00A6  3190          ENDTEST
```

```
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                              ; Init the cpu
0022    3242            LOADDCS         1$,,0,3                 ; Load the Micro words
0029    3243            LOADREG         WDREG,2$,,L             ; Load initial test pattern
0031    3244            LOADREG         MAIN32,6$,,B            ; Enable WD reg onto the WBUS
0039    3245            LOADREG         MAIN32,7$,,B            ; Strobe the WH reg
0041    3246            LOADREG         MAIN32,6$,,B            ; ...
0049    3247            LOADREG         MAIN32,8$,,B            ; Disable WD reg from the WBUS
```

```
                                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$,,L                     ; Test the WH register contents
                                0079  3254          IFERROR         ,,<<RDM>>                       ; Incorrect data
                                007F  3255          SKIP                                            ; Go to next test
                                0086  3256
                                0086  3257          BEGIN_TEST_DATA
                                0086
                                0086        ;-----------------------------------------------------------------------
                                0086  3258
                                0086  3259 1$:                                                      ; Micro code
U 00, 7700,7800,7FFF,FFFF,8470,1801  0086  3260 ;% 00    LONLIT [0]                                  ; Init the LONLIT register
U 01, 6700,0800,0047,03D4,0470,0000  0091  3264 ;% 01    RDM_LONLIT,Q_M,NEXT/0                       ; Try to read the LONLIT reg
U 02, 7700,4800,0364,0300,0470,1803  009C  3268 ;% 02    NOP
                                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
```

```
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                          ; Init the cpu
001C  3329          LOADDCS         1$,,0,4             ; Load the DCS
0023  3330          ERRLOOP                             ; Error loop point
0025  3331          FETCH           1                   ; Start execution
002A  3332          LOADREG         DCSCR,3$,,B         ; Tick the clock
0032  3333          LOADREG         DCSCR,4$,,B         ; ...
003A  3334          CMPREG          CLKDCS,2$,,B        ; Test DCS ADDRESS register
0043  3335          IFERROR         ,,<<RDM>>           ; Incorrect data
0049  3336          SKIP                                ; Go to next test
0050  3337
0050  3338          BEGIN_TEST_DATA
```

```
                                0050
                                0050        ;-------------------------------------------------------------------------
                                0050  3339
                                0050  3340 1$:                                                  ; Micro code
U 00, 7700,C800,0364,0300,0470,1801  0050  3341 ;% 00   NOP
U 01, 7700,C800,0364,0300,0470,1802  005B  3345 ;% 01   NOP
U 02, 7500,4800,0364,0300,0470,1803  0066  3349 ;% 02   DCS.ADDR_0
U 03, 7700,C800,0364,0300,0470,1804  0071  3353 ;% 03   NOP
                                007C  3357
                              C0  007C  3358 2$:      .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        ;-------------------------------------------------------------------------
                                007F  3363          ENDTEST
```

```
                        .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)
```

```
                                001E  3421 ;--
                                001E  3422 ;*******************************************************************************
                                001E  3423 ;///////////////////////////////////////////////////////////////////////////////
                                001E  3424 ;+
                                001E  3425 ;SUBTEST #1
                                001E  3426 ;
                                001E  3427 ;       Test that CSTRP register is constantly loaded
                                001E  3428 ;-
                                001E  3429         SUBTEST                                     ; SUBTEST #1
                                0021
                                0021       ;///////////////////////////////////////////////////////////////////////////
                                0021  3430         INITIALIZE                                  ; Init the cpu
                                0023  3431         LOADDCS         1$,,1,1                     ; Load a NOP in location 1
                                002A  3432         LOOP            I,1,5                       ; 5 data patterns
                                0031  3433         LOADDCS         2$,I,0,1                    ; Load the micro instruction
                                0038  3434         ERRLOOP                                     ; Error loop point
                                003A  3435         FETCH           0                           ; Start execution
                                003F  3436         LOADREG         DCSCR,3$,,B                 ; Tick the clock one
                                0047  3437         LOADREG         DCSCR,4$,,B                 ; micro instruction
                                004F  3438         CMPREGMSK       CSTRP,5$,I,6$,,W            ; Check the TRAP register
                                005B  3439         IFERROR         ,,<<RDM>>                   ; Incorrect data
                                0061  3440         ENDLOOP         I                           ; End test loop
                                0064  3441
                                0064  3442
                                0064  3443
                                0064  3444 ;///////////////////////////////////////////////////////////////////////////////
                                0064  3445 ;+
                                0064  3446 ;SUBTEST #2
                                0064  3447 ;
                                0064  3448 ;       Test that CSTRP does not clock when ENTRCE bit is set
                                0064  3449 ;-
                                0064  3450         SUBTEST                                     ; SUBTEST #2
                                0067
                                0067       ;///////////////////////////////////////////////////////////////////////////
                                0067  3451         INITIALIZE                                  ; Init the cpu
                                0069  3452         LOADDCS         10$,,0,3                    ; Load the micro instructions
                                0070  3453         ERRLOOP                                     ; Error loop point
                                0072  3454         FETCH           0                           ; Start execution
                                0077  3455         BRSTCLK         2                           ; Execute the sequence
                                007B  3456         CMPREGMSK       CSTRP,11$,,6$,,W            ; Check the TRAP register
                                0087  3457         IFERROR         ,,<<RDM>>                   ; Incorrect data
                                008D  3458         SKIP                                        ; Go to next test
                                0094  3459
                                0094  3460         BEGIN_TEST_DATA
                                0094
                                0094       ;---------------------------------------------------------------------------
                                0094  3461
                                0094  3462 1$:
U 01, 7700,C800,0364,0300,0470,1802  0094  3463 :% 01    NOP
                                009F  3467 2$:                                               ; Test micro instructions
U 00, 7700,C800,0364,0300,0470,1AAA  009F  3468 :% 00    NEXT/1AAA
U 00, 7700,4800,0364,0300,0470,1D55  00AA  3472 :% 00    NEXT/1D55
U 00, 7700,C800,0364,0300,0470,1CCC  00B5  3476 :% 00    NEXT/1CCC
U 00, 7700,4800,0364,0300,0470,1BC3  00C0  3480 :% 00    NEXT/1BC3
U 00, 7700,4800,0364,0300,0470,183F  00CB  3484 :% 00    NEXT/183F
                                     00D6  3488
                                  20 00D6  3489 3$:      .BYTE   CLK_CTRL_0                  ; Single Micro Instruction
```

```
                      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 ;
```

```
                        ;--
                        ;*****************************************************************************
                        ;/////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       Test VBUS load with VSTB bit set
                        ;-
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
```

```
                        00DB  3629 1$:
U 00, 7600,C800,0364,0300,0470,1801  00DB  3630 ;% 00   NOP,STROBE.V.BUS                    ; Reverse a parity field
                        00E6  3634 12$:
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$:     .BYTE   1
                              00F8  3645         VBUSDATA        <^X1D>,HIGH              ; CS Parity Error H
                              00F9  3646
                              00F9  3647 ;+
                              00F9  3648 ; Subtest 2 data
                              00F9  3649 ;-
                              00F9  3650 10$:
U 00, 7700,C800,0364,0300,0470,1801  00F9  3651 ;% 00   NOP
                          01  0104  3655 11$:    .BYTE   1
                              0105  3656         VBUSDATA        <^X1D>,LOW               ; CS Parity Error H
                          00  0106  3657 13$:    .BYTE   0
                              0107  3658
                              0107  3659         END_TEST_DATA
                              0107
                              0107        ;-----------------------------------------------------------------------
                              0107  3660
                              0107  3661         .SBTTL  ''
                              0107  3662         .SBTTL  ''STOP CLOCK FUNCTIONS''
                              0107  3663         .SBTTL  ''
                              0107  3664         ENDTEST
```

```
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
```

```
                        ;       RECEIVED CONTENTS OF CLKDCS REGISTER
                        ;
                        ;--
                        ;*******************************************************************************
                        ;///////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       Test clock control logic with PARSTP set
                        ;-
                                SUBTEST                                 ; SUBTEST #1

                        ;///////////////////////////////////////////////////////////////////////////////
                                LOADDCS         1$,,2,1                 ; Good parity for DCS address 2
                                INITIALIZE                              ; Init the cpu
                                LDFIELD         11$,,B,2$,78,78,,,,NP,  ; Force parity error
                                LOADDCS         2$,,2,1                 ; Bad parity for DCS address 2
                                ERRLOOP                                 ; Error loop point
                                FETCH           0,INH                   ; Set DCS add 0 into latches
                                LOADREG         CSAMR,3$,,W             ; Load 1800 into CSAMR
                                LOADREG         DCSCR,4$,,B             ; Set PARSTP bit
                                LOADREG         RDCTRL,5$,,B            ; Set MSTP bit
                                LOADREG         DCSCR,6$,,B             ; Start the clock
                                CMPREG          CLKDCS,7$,,B,           ; Test for clock stop at DCS 03
                                LOADREG         DCSCR,4$,,B             ; Stop the clock
                                IFERROR         ,,<<RDM>>               ; Incorrect clock stop


                        ;///////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #2
                        ;
                        ;       Test clock control logic with PARSTP not set
                        ;-
                                SUBTEST                                 ; SUBTEST #2

                        ;///////////////////////////////////////////////////////////////////////////////
                                LOADDCS         1$,,2,1                 ; Good parity for DCS address 2
                                INITIALIZE                              ; Init the cpu
                                LOADDCS         2$,,2,1                 ; Bad parity for DCS address 2
                                ERRLOOP                                 ; Error loop point
                                FETCH           0,INH                   ; Set DCS add 0 into latches
                                LOADREG         CSAMR,3$,,W             ; Load 1800 into CSAMR
                                LOADREG         DCSCR,8$,,B             ; Clear PARSTP bit
                                LOADREG         RDCTRL,5$,,B            ; Set MSTP bit
                                LOADREG         DCSCR,9$,,B             ; Start the clock
                                CMPREG          CLKDCS,10$,,B,          ; Test clock stop at DCS 05
                                LOADREG         DCSCR,8$,,B             ; Stop the clock
                                IFERROR         ,,<<RDM>>               ; Incorrect clock stop
                                SKIP                                    ; Go to next test


                                BEGIN_TEST_DATA

                        ;-------------------------------------------------------------------------
```

```
                                00E7  3773
                                00E7  3774 1$:
U 02, 7700,4800,0364,0300,0470,1803  00E7  3775 ;% 02   NOP                                     ; Good parity for DCS add 2
                                00F2  3779 2$:
U 02, 7700,4800,0364,0300,0470,1803  00F2  3780 ;% 02   NOP                                     ; Bad parity for DCS add 2
                              1800  00FD  3784 3$:      .WORD   DCS_START                       ; 1800 to CSAMR
                                F8  00FF  3785 4$:      .BYTE   PAR_CHK_ENABL!PAR_STOP_ENABL!-
                                0100  3786                      CLK_CTRL_0!CLK_CTRL_1!-
                                0100  3787                      MICRO_ADDR_INH
                                08  0100  3788 5$:      .BYTE   MASTER_HALT_EN
                                98  0101  3789 6$:      .BYTE   PAR_CHK_ENABL!PAR_STOP_ENABL!-
                                0102  3790                      MICRO_ADDR_INH
                                C4  0102  3791 7$:      .BYTE   ^XC4                            ; Clock stopped at DCS add 03
                                E8  0103  3792 8$:      .BYTE   PAR_CHK_ENABL!MICRO_ADDR_INH!-
                                0104  3793                      CLK_CTRL_0!CLK_CTRL_1
                                88  0104  3794 9$:      .BYTE   PAR_CHK_ENABL!MICRO_ADDR_INH
                                C5  0105  3795 10$:     .BYTE   ^XC5                            ; Clock stopped at DCS add 05
                                00  0106  3796 11$:     .BYTE   0
                                0107  3797
                                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 ;
```

```
0015  3886 ;       SUBTEST #2
0015  3887 ;               ITERATION       CSAMR & NEXT FIELD
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
0015  3912 ;
0015  3913 ;       SUBTEST #3
0015  3914 ;               ITERATION       CSAMR           NEXT FIELD
0015  3915 ;                   1               1802            1800
0015  3916 ;                   2               1804            1800
0015  3917 ;                   3               1808            1800
0015  3918 ;                   4               1810            1800
0015  3919 ;                   5               1820            1800
0015  3920 ;                   6               1840            1800
0015  3921 ;                   7               1880            1800
0015  3922 ;                   8               1900            1800
0015  3923 ;                   9               1A00            1800
0015  3924 ;                  10               1C00            1800
0015  3925 ;                  11               1FFF            1800
0015  3926 ;
0015  3927 ;       SUBTEST #4
0015  3928 ;               ITERATION       CSAMR           NEXT FIELD
0015  3929 ;                   1               1800            1802
0015  3930 ;                   2               1800            1804
0015  3931 ;                   3               1800            1808
0015  3932 ;                   4               1800            1810
0015  3933 ;                   5               1800            1820
0015  3934 ;                   6               1800            1840
0015  3935 ;                   7               1800            1880
0015  3936 ;                   8               1800            1900
0015  3937 ;                   9               1800            1A00
0015  3938 ;                  10               1800            1C00
0015  3939 ;                  11               1800            1FFF
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 ;
                     0015  3955 ; ERROR DESCRIPTION:
                     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       ;////////////////////////////////////////////////////////////////////////////
                        00A5  4009          LDFIELD         20$,,W,11$,0,13,        ; Set next field to 1800
                        00B6  4010          LOADDCS         11$,,1,1                ; Load the microword
                        00BD  4011          LOOP            I,1,11                  ; Set loop of 11 test patterns
                        00C4  4012          ERRLOOP                                 ; Error loop point
                        00C6  4013          FETCH           0                       ; Start execution
                        00CB  4014          LOADREG         CSAMR,10$+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$,,B             ; Stop the clock
                        00EC  4018          IFERROR         ,,<<RDM>>               ; Incorrect clock stop
                        00F2  4019          ENDLOOP         I                       ; End test loop
                        00F5  4020
                        00F5  4021
                        00F5  4022 ;//////////////////////////////////////////////////////////////////////////////
                        00F5  4023 ;+
                        00F5  4024 ;SUBTEST #4
                        00F5  4025 ;
                        00F5  4026 ;       Test more patterns for a not match condition
                        00F5  4027 ;-
                        00F5  4028          SUBTEST                                 ; SUBTEST #4
                        00F8
                        00F8       ;////////////////////////////////////////////////////////////////////////////
                        00F8  4029          LOOP            I,1,11                  ; Set loop for 11 test patterns
                        00FF  4030          LDFIELD         10$+2,I,W,11$,0,13,     ; Set next field of a microword
                        0110  4031          LOADDCS         11$,,1,1                ; Load the microword
                        0117  4032          ERRLOOP                                 ; 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                                    ; Go to next test
                        014F  4041
                        014F  4042          BEGIN_TEST_DATA
                        014F
                        014F       ;-----------------------------------------------------------------------------
                        014F  4043
                        014F  4044 1$:
U 00, 7700,4800,0364,0300,0470,18AA  014F  4045 ;% 00    NEXT/18AA
U 01, 7700,C800,0364,0300,0470,1820  015A  4049 ;% 01    NEXT/1820
```

```
U 02, 7700,4800,0364,0300,0470,1803  0165  4053 ;% 02   NOP
U 03, 7300,C800,0364,0300,0470,1804  0170  4057 ;% 03   STOP.CLOCK
                                     017B  4061
                                1820 017B  4062 2$:     .WORD   ^X1820
                                  01 017D  4063 3$:     .BYTE   HALT_ON_MATCH
                                  C3 017E  4064 4$:     .BYTE   3 ! CLK_CTRL_0_RO!CLK_CTRL_1_RO
                                  60 017F  4065 5$:     .BYTE   CLK_CTRL_0!CLK_CTRL_1
                                     0180  4066
 1820 1810 1808 1804 1802 1801 1800  0180  4067 10$:    .WORD   ^X1800,^X1801,^X1802,^X1804,^X1808,^X1810,^X1820,^X1840,^X1880
                           1880 1840  018E
                      1C00 1A00 1900  0192  4068         .WORD   ^X1900,^X1A00,^X1C00
 1FDF 1FEF 1FF7 1FFB 1FFD 1FFE 1FFF  0198  4069         .WORD   ^X1FFF,^X1FFE,^X1FFD,^X1FFB,^X1FF7,^X1FEF,^X1FDF,^X1FBF,^X1F7F
                           1F7F 1FBF  01A6
                      1BFF 1DFF 1EFF  01AA  4070         .WORD   ^X1EFF,^X1DFF,^X1BFF
                                     01B0  4071 11$:
U 01, 7700,4800,0364,0300,0470,1800  01B0  4072 ;% 01   NEXT/1800
                                     01B8  4076
                                1800 01B8  4077 20$:    .WORD   ^X1800
                                  C4 01BD  4078 21$:    .BYTE   4 ! CLK_CTRL_0_RO!CLK_CTRL_1_RO
                                     01BE  4079
                                     01BE  4080         END_TEST_DATA
                                     01BE
                                     01BE       ;-----------------------------------------------------------------------
                                     01BE  4081         ENDTEST
```

```
                        0021        .SBTTL  "TEST   021     TEST CLOCK DETECTION CIRCUIT
                        0021  4083 ;*******************************************************************************
                        0021  4084 ;++
                        0021  4085 ;
                        0021  4086 ; FUNCTIONAL DESCRIPTION:
                        0021  4087 ;
                        0021  4088 ;       THIS TEST VERIFIES THAT THE B CLOCK DETECTION CIRCUIT ON THE RDM IS
                        0021  4089 ;       OPERATIONAL.
                        0021  4090 ;
                        0021  4091 ;
                        0021  4092 ; TEST ALGORITHM:
                        0021  4093 ;
                        0021  4094 ;       1.  INITIALIZE THE CPU
                        0021  4095 ;       2.  LOAD THE DCSCR TO STOP THE CLOCK
                        0021  4096 ;       3.  TEST THE STATUS REGISTER TO SEE IF B CLOCK IS HALTED
                        0021  4097 ;       4.  IF NO ERROR SET DC LOW IN THE RDCTRL REGISTER
                        0021  4098 ;       5.  TEST THE STATUS REGISTER TO SEE THAT B CLOCK IS RUNNING
                        0021  4099 ;       6.  IF NO ERROR DISABLE DC LOW AND GO TO NEXT TEST
                        0021  4100 ;
                        0021  4101 ;
                        0021  4102 ; TEST PATTERNS:
                        0021  4103 ;
                        0021  4104 ;       NONE
                        0021  4105 ;
                        0021  4106 ;
                        0021  4107 ; LOGIC DESCRIPTION:
                        0021  4108 ;
                        0021  4109 ;       THIS TEST CHECKS THE J/K FLIP FLOP THAT DETECTS WHETHER THE COMET B
                        0021  4110 ;       CLOCK IS RUNNING.  ANY FAILURES SHOULD BE IN THE FLIP FLOP CIRCUIT
                        0021  4111 ;       OR THE STATUS REGISTER ITSELF.
                        0021  4112 ;
                        0021  4113 ;
                        0021  4114 ; ASSUMPTIONS:
                        0021  4115 ;
                        0021  4116 ;       THIS TEST ASSUMES THE POWER UP SELF TEST HAS RUN AND THE COMET CLOCK
                        0021  4117 ;       IS OPERATIONAL.
                        0021  4118 ;
                        0021  4119 ;
                        0021  4120 ; ERROR DESCRIPTION:
                        0021  4121 ;
                        0021  4122 ;       EXPECTED CONTENTS OF THE STATUS REGISTER
                        0021  4123 ;       RECEIVED CONTENTS OF THE STATUS REGISTER
                        0021  4124 ;
                        0021  4125 ;--
                        0021  4126 ;*******************************************************************************
                        0021  4127
                        0021  4128
                        0021  4129          INITIALIZE                                  ;INIT THE CPU
                        0023  4130          LOADREG         DCSCR,1$,,B                 ;HALT THE CLOCK
                        0028  4131          CMPREGMSK       STATUS,2$,,3$,,B,           ;TEST THE CLOCK DETECTION BIT
                        0037  4132          IFERROR         ,,<<RDM>>                   ;BIT INCORRECT
                        003D  4133          LOADREG         RDCTRL,4$,,B                ;HOLD DC LOW TO RUN CLOCK
                        0045  4134          CMPREGMSK       STATUS,5$,,3$,,B,           ;TEST THE CLOCK DETECTION BIT
                        0051  4135          IFERROR         ,,<<RDM>>                   ;BIT INCORRECT
                        0057  4136          LOADREG         RDCTRL,5$,,B                ;FREE DC LOW
                        005F  4137          SKIP                                        ;GO TO NEXT TEST
                        0066  4138
```

```
                     0066  4139
                     0066  4140 BEGIN_TEST_DATA
                     0066
                     0066       ;-------------------------------------------------------------------------
                     0066  4141
            60       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         ENDTEST
```

```
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 ;*****************************************************************************
```

```
                     0021  4211
                     0021  4212
                     0021  4213
                     0021  4214          INITIALIZE                              ; Init the cpu
                     0023  4215          ERRLOOP                                 ; Error loop point
                     0025  4216 ;
                     0025  4217 ; Make sure DCS is not selected
                     0025  4218 ;
                     0025  4219          LOADREG         DCSCR,6$,,B             ; Par Chk and Micro Addr Inh
                     002D  4220          LOADREG         CSAMR,1$,,W             ; Address 0 on CS BUS
                     0035  4221 ;
                     0035  4222 ; Load DCS address 0 with 0's
                     0035  4223 ;
                     0035  4224          LOADREG         DCSAD,1$,,B             ; Select address 0
                     003D  4225          LOOP            I,1,2                   ; Loop of 2
                     0044  4226          LOADREG         DCSDA,2$,,B             ; Load first 2 bytes with 0
                     004C  4227          ENDLOOP         I                       ; ...
                     004F  4228          LOADREG         RDCTRL,1$,,B            ; Clear Master Halt Enable
                     0057  4229 ;
                     0057  4230 ; Load the TRAP register with zero's
                     0057  4231 ;
                     0057  4232          LOADREG         DCSCR,7$,,B             ; Set trace enable
                     005F  4233          LOADREG         DCSAD,2$,,B             ; Select DCS address ^X3F
                     0067  4234          LOADREG         DCSCR,8$,,B             ; Tick the clock
                     006F  4235          LOADREG         DCSCR,7$,,B             ; one micro instruction
                     0077  4236 ;
                     0077  4237 ; Set trace enable and tick the clock again
                     0077  4238 ;
                     0077  4239          LOADREG         CSAMR,2$,,W             ; Load 1's into CSAMR
                     007F  4240          LOADREG         DCSAD,1$,,B             ; Select DCS Address 0
                     0087  4241          LOADREG         DCSCR,7$,,B             ; Set trace
                     008F  4242          LOADREG         DCSCR,8$,,B             ; Tick the clock
                     0097  4243          LOADREG         DCSCR,7$,,B             ; one micro instruction
                     009F  4244 ;
                     009F  4245 ; Check that the DCS address register incremented
                     009F  4246 ;
                     009F  4247          CMPREGMSK       CLKDCS,10$,,9$,,B,      ; Test DCS address = 1
                     00AB  4248          IFERROR         ,,<<RDM>>               ; Incorrect address
                     00B1  4249 ;
                     00B1  4250 ; Now read the DCS and check that zero's were written
                     00B1  4251 ;
                     00B1  4252          LOADREG         DCSCR,6$,,B             ; Turn trace off
                     00B9  4253          LOADREG         DCSAD,1$,,B             ; Reload the address register
                     00C1  4254          LOADREG         DCSAD,1$,,B             ; Load the Control File Latch
                     00C9  4255          CMPREGMSK       DCSCF,1$,,9$,,B,        ; Check for zero
                     00D5  4256          IFERROR         ,,<<RDM>>               ; Incorrect data
                     00DB  4257          CMPREG          BADD1,2$,,B             ; Test BADD1 for 0
                     00E4  4258          IFERROR         ,,<<RDM>>               ; Incorrect data
                     00EA  4259          SKIP                                    ; Go to next test
                     00F1  4260
                     00F1  4261          BEGIN_TEST_DATA
                     00F1
                     00F1          ;-----------------------------------------------------------------------
                     00F1  4262
0000                 00F1  4263 1$:      .WORD   0
  FF                 00F3  4264 2$:      .BYTE   -1
  E8                 00F4  4265 6$:      .BYTE   CLK_CTRL_1!CLK_CTRL_0!PAR_CHK_ENABL!MICRO_ADDR_INH
```

```
                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
```

```
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             AAAAAAAA
0023  4325 ;           2             55555555
0023  4326 ;           3             33333333
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  4340 ;
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$,ONERROR                    ;TRY COMET TO MAIN MEMORY
0067  4371         ENDLOOP         I                               ;END TEST LOOP
006A  4372
006A  4373
006A  4374
006A  4375 ;///////////////////////////////////////////////////////////////////////////////
006A  4376 ;+
006A  4377 ;SUBTEST #2
006A  4378 ;
006A  4379 ;       TEST MDREG TO CMI
006A  4380 ;-
006A  4381         SUBTEST         2                               ;SUBTEST #2
006D
006D          ;///////////////////////////////////////////////////////////////////////////
006D  4382         INITIALIZE                                      ;INIT THE CPU
006F  4383         LOOP            I,1,6                           ;CREATE A TEST LOOP OF 6
0076  4384         SA01PAT         I,1$,,                          ;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         ,,<<RDM>>               ;MDREG ON RDM IS BAD
                     00AF  4392          ENDLOOP         I                       ;END TEST LOOP
                     00B2  4393          SKIP                                    ;GO TO NEXT TEST
                     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                                 ;ERROR LOOP POINT
                     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$,,L,            ;TEST FOR CORRECT PATTERN
                     00F8  4412          IFERROR         ,,<<CMI>>               ;SOMETHING IS TIED TO CMI
                     00FE  4413          ENDLOOP         I                       ;END TEST LOOP
                     0101  4414          CMPREG          MHREG,1$,,L,NE          ;FORCE AN ERROR
                     010A  4415          IFERROR                                 ;MAREG ON RDM IS BAD
                     010F  4416          SKIP                                    ;GO TO NEXT TEST
                     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 ;
```

```
                     4488 ;       SUBTEST #1:
                     4489 ;       FIRST IFERROR
                     4490 ;               EXPECTED DATA FROM CMI
                     4491 ;               RECEIVED DATA FROM CMI
                     4492 ;               LOOP COUNT I
                     4493 ;
                     4494 ;       SECOND IFERROR
                     4495 ;               EXPECTED STATUS FROM CMI
                     4496 ;               RECEIVED STATUS FROM CMI
                     4497 ;               LOOP COUNT I
                     4498 ;
                     4499 ;       SUBTEST #2:
                     4500 ;               EXPECTED STATUS FROM CMI
                     4501 ;               RECEIVED STATUS FROM CMI
                     4502 ;
                     4503 ;--
                     4504 ;*******************************************************************************
                     4505 ;+
                     4506 ;SUBTEST #1
                     4507 ;
                     4508 ;       TEST CMI CONTROL AND STATUS LINES
                     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$,,                                  ;GENERATE A TEST PATTERN
0038 4514            ERRLOOP                                                 ;ERROR LOOP POINT
003D 4515            FETCH           0                                       ;START DCS PROGRAM
0042 4516            LOADREG         DCSCR,2$,,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            ENDLOOP         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
                     00E7          ;-----------------------------------------------------------------------
                     00E7  4553
          00         00E7  4554 2$:      .BYTE   ^X00                            ;CLEAR REGISTERS
          01         00E8  4555 3$:      .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
                     00FA          ;-----------------------------------------------------------------------
                     00FA  4584 ENDTEST
```

```
                        0002  4587 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN            = 00000000
$TEST_NUMBER    = 00000025
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 00000008
ALK             = 00000000
ALP             = 00000016
ARG_CNT         = 000000D9 R      4E
A_REG_BYTE_0    = 00007410
A_REG_BYTE_1    = 00007411
A_REG_BYTE_2    = 00007412
A_REG_BYTE_3    = 00007413
B               = 00000001
BADD1           = 00007426
BELL_FLAG       = 00000008
BELL_KEY        = 0000000E
BUFF_ADDR_LOW   = 00007426
BURST_STOP_FLAG= 00000080
BYTE            = 00000001
CACHE_FLAG      = 00000000
CACHE_FLAG1     = 00000000
CACHE_LENGTH    = 00000001
CAK             = 0000000A
CCC             = 00000004
CCS             = 00000005
CF_KEY          = 00000024
CLA             = 00000003
CLEAR_CTRL_FILE= 00000002
CLKDCS          = 00007427
CLK_CTRL_0      = 00000020
CLK_CTRL_0_RO   = 00000040
CLK_CTRL_1      = 00000040
CLK_CTRL_1_RO   = 00000080
CLOCK_STOPED    = 00000080
CMI             = 00000007
CMICTL          = 0000741C
CMI_COMPLETE    = 00000020
CMI_CTRL_CLEAR  = 00000001
CMI_CTRL_REG    = 0000741C
CMI_GO          = 00000008
CMI_READ        = 00000040
CMI_STATUS_0    = 00000008
CMI_STATUS_1    = 00000010
CMI_WRITE       = 00000020
CMK             = 0000000E
COMPOSIT_LENGTH= 00000003
CON             - 00000011
CONTINUE_FLAG   = 00000002
CONTROL_C_FLAG  = 00000001
CONT_FLAG       = 00000020
CONT_KEY        = 0000001C
CPU_RUN         = 00000010
CSAMR           = 00007404
CSBUF           = 00007424
CSTRP           = 00007422
CS_ADD_BUFF_HGH= 00007425
CS_ADD_BUFF_LOW= 00007424
CS_ADD_TRAP_HGH= 00007423
CS_ADD_TRAP_LOW= 00007422
CYCLE_FLAG      = 00000008
CYCLE_KEY       = 00000020
D0              = 00000004
D1              = 00000002
D2              = 00000000
DATA_FLAG       = 00000000
DATA_LENGTH     = 00000001
DCSAD           = 00007401
DCSCF           = 00007403
DCSCR           = 00007402
DCSDA           = 00007400
DCS_ADDR_CLEAR  = 00000002
DCS_ADDR_REG_RO= 00007427
DCS_ADDR_REG_WO= 00007401
DCS_CTRL_FILE   = 00007403
DCS_CTRL_REG    = 00007402
DCS_DATA_REG    = 00007400
DCS_FLAG        = 00000000
DCS_FLAG1       = 00000000
DCS_HEAD        = 00000000 R      4F
DCS_LENGTH      = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
DUM1            = FFFFFFF4
ENDTEST_LOC     = 000000FF R      4E
END_SLICE       = 00000025
END_TEST_1        00000064 R      04
END_TEST_10       0000008A R      16
END_TEST_11       0000009C R      18
END_TEST_12       000001CC R      1A
END_TEST_13       00000060 R      1C
END_TEST_14       00000055 R      1E
END_TEST_15       00000310 R      20
END_TEST_16       00000305 R      22
END_TEST_17       0000008C R      24
END_TEST_18       00000090 R      26
END_TEST_19       00000148 R      28
END_TEST_2        00000096 R      06
END_TEST_20       000000BF R      2A
END_TEST_21       000000D7 R      2C
END_TEST_22       0000006B R      2E
END_TEST_23       00000096 R      30
END_TEST_24       0000007E R      32
END_TEST_25       00000090 R      34
END_TEST_26       000000A6 R      36
END_TEST_27       000000B0 R      38
END_TEST_28       0000007F R      3A
END_TEST_29       00000107 R      3C
END_TEST_3        00000058 R      08
END_TEST_30       00000107 R      3E
END_TEST_31       00000107 R      40
END_TEST_32       000001BE R      46
END_TEST_33       0000006B R      48
END_TEST_34       000000F9 R      4A
END_TEST_35       0000011F R      4C
END_TEST_36       000000FA R      4E
END_TEST_4        000000B1 R      0A
END_TEST_5        00000068 R      0C
END_TEST_6        0000006E R      0E
END_TEST_7        000000B1 R      10
END_TEST_8        000000A6 R      12
END_TEST_9        0000061B R      14
EN_TRACE_REG    = 00000008
ERRLOG_FLAG     = 00000000
ERROR_FLAG      = 00000010
ERROR_LOG_ADR   = 000032FC
ERR_LOG_INIT    = 00000001
EXP_STALL_FLAG = 00000010
EXP_UTRAP_FLAG = 00000040
FAC             = 00000020
FAULT           = 00000002
FCC             = 00000015
FCS             = 00000021
FETCH_FLAG      = 00000008
FEX             = 0000001E
FFH             = 00000024
FFL             = 00000023
FIC             = 0000001F
FIO             = 0000001D
FLAG_KEY        = 00000004
FMR             = 00000022
FPA             = 00000002
FP_BOOT_0       = 00000001
FP_BOOT_1       = 00000002
FP_START_0      = 00000004
FP_START_1      = 00000008
FQA             = 00000014
FRONT1          = 00007420
FRONT2          = 00007421
FRONT_PNL_1     = 00007420
FRONT_PNL_2     = 00007421
FRONT_PNL_LOCK  = 00000080
HALT_FLAG       = 00000001
HALT_KEY        = 00000008
HALT_ON_MATCH   = 00000001
HEAD            = FFFF7C00
HIGH            = 00000001
H_FROM_WBUS     = 00000010
IB_FLAG         = 00000020
IB_KEY          = 00000012
INSTR_FLAG      = 00000004
INSTR_KEY       = 00000018
INT             = 00000010
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
M$TEST_SIZE     = 000007B3
M$TEST_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
MC0             = 00000003
MC1             = 00000009
MDL             = 0000001B
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
MET             = 0000001A
MHREG           = 00007418
MH_REG_BYTE_0   = 00007418
MH_REG_BYTE_1   = 00007419
MH_REG_BYTE_2   = 0000741A
MH_REG_BYTE_3   = 0000741B
MIT             = 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_CHK_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
VBUS_CLOCK      = 00000080
```

```
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 synop

PSECT name	Allocation	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	000000FE (254.)	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	000000FE (254.)	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	000000FE (254.)	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	000000FE (254.)	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	0000067E (1662.)	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	000000FE (254.)	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	000000FE (254.)	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	000000FE (254.)	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	000000FE (254.)	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	000000FE (254.)	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	000000FE (254.)	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	000000FE (254.)	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

+-------------------------+
! Performan

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.

Macro library name	Macro lib Macros defined
WRKD$0:[COMET]COMETMAC.MLB;801	45
SYS$SYSROOT:[SYSLIB]STARLET.MLB;1	0
TOTALS (all libraries)	45

1887 GETS were required to define 45 macros.

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

WRKD$0:[COMET.RDM]TITLE+__DRB0:[COMET]COMETASM+WRKD$0:[COMET.RDM]ECKAF/LIS=ECKAF/OBJ=ECKAF