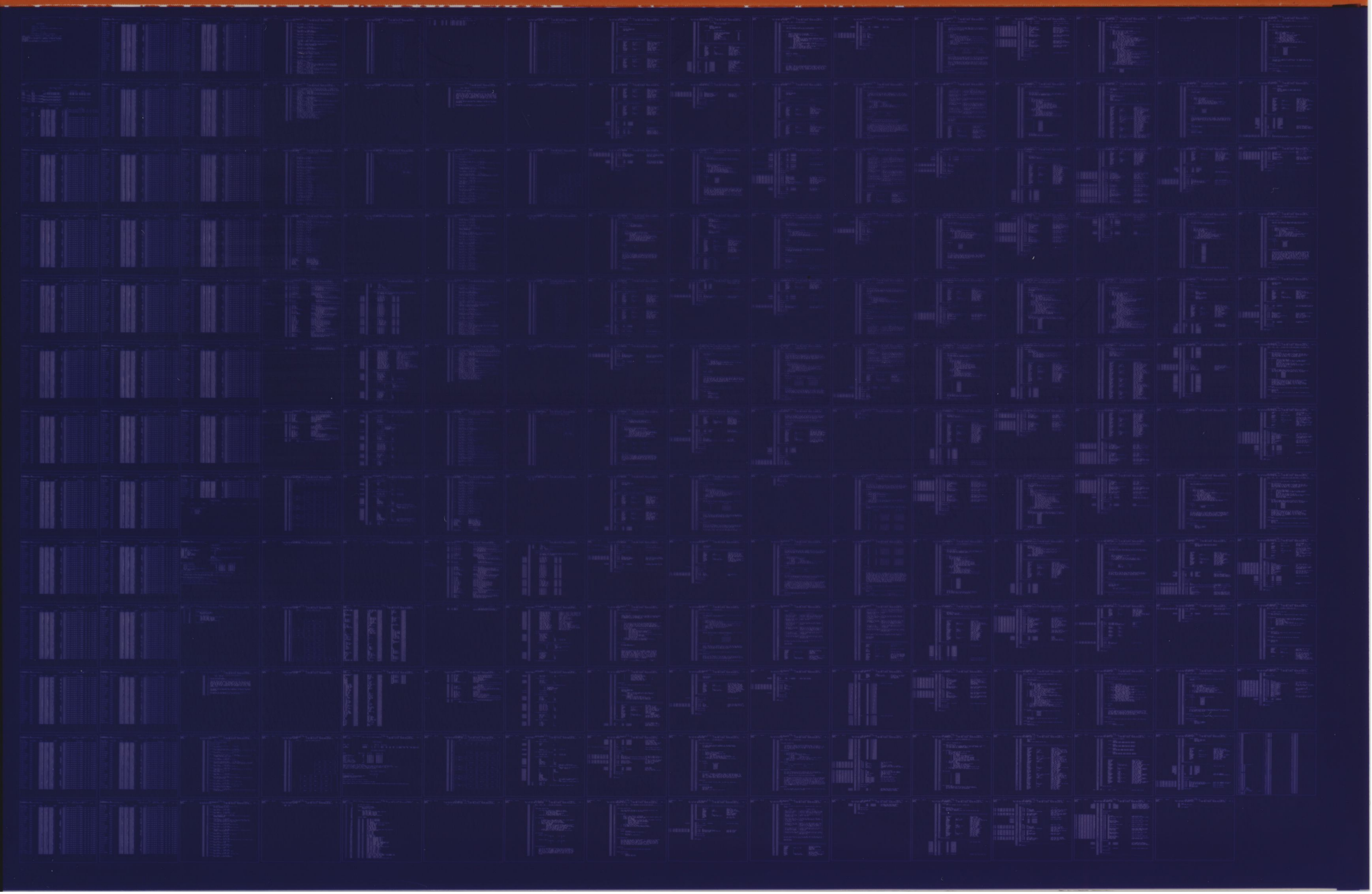
# ECKAC-8.0

VAX 11/750 MICRO MIC

EP-ECKAC-DL-8.0
1 OF 4 JUL 1986
COPYRIGHT© 1980-86

digital
MADE IN USA



# ECKAC-8.0

VAX 11/750 MICRO MIC

EP-ECKAC-DL-8.0
2 OF 4 JUL 1986
COPYRIGHT© 1980-86

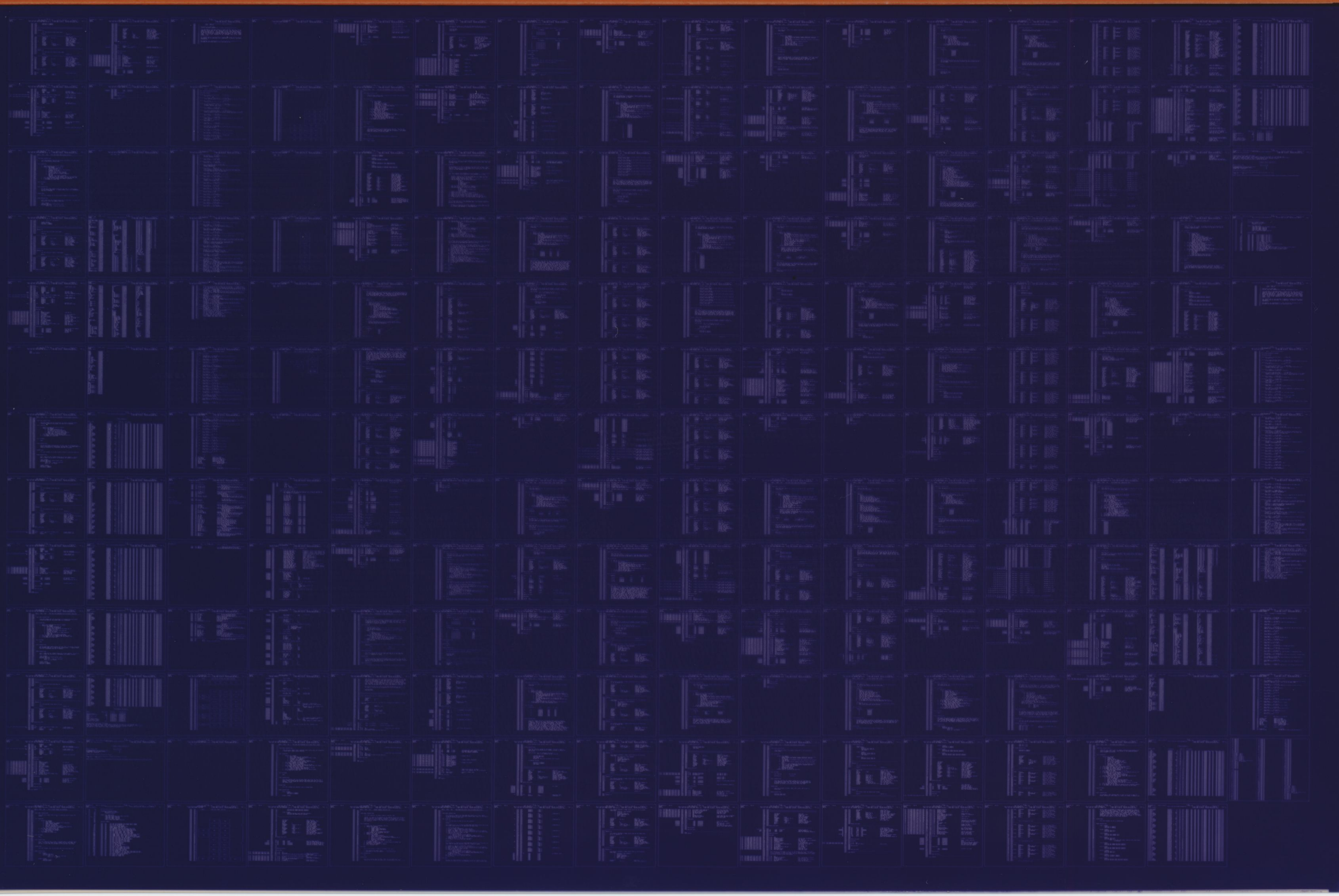
digital
MADE IN USA

# ECKAC-8.0

VAX 11/750 MICRO MIC

EP-ECKAC-DL-8.0
3 OF 4 JUL 1986
COPYRIGHT© 1980-86

digital
MADE IN USA

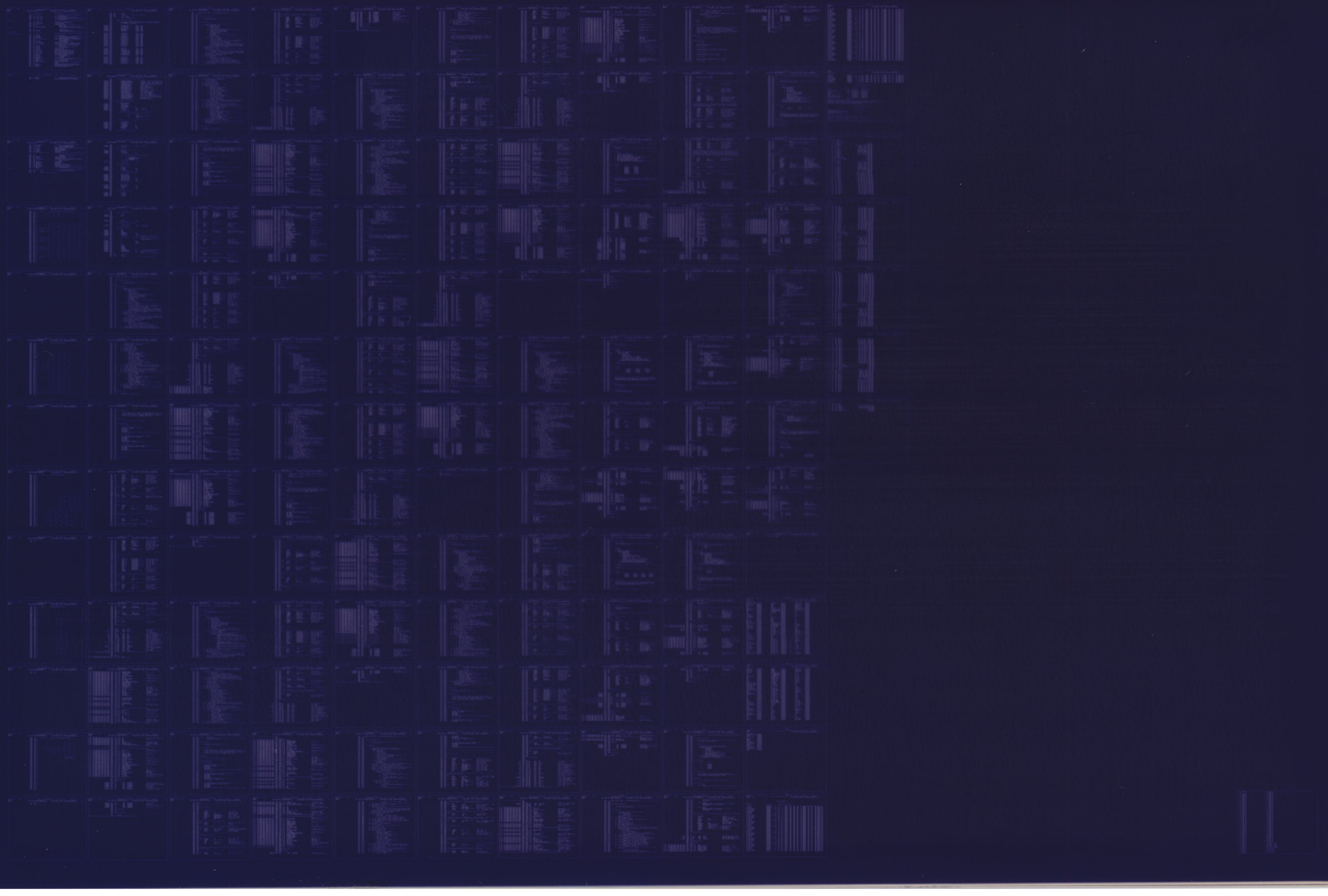


# ECKAC-8.0

VAX 11/750 MICRO MIC

EP-ECKAC-DL-8.0
4 OF 4 JUL 1986
COPYRIGHT© 1980-86

digital
MADE IN USA



## IDENTIFICATION

PRODUCT ID: ZZ-ECKAC-8.0

PRODUCT TITLE: ECKAC703 KA750 MICRO DIAGNOSTIC (L0003)

DECO/DEPO: 8.0

DATE: JULY 1982

MAINTAINED BY: VAX DIAGNOSTIC ENGINEERING

TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.
DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

## Object Module Synopsis

| Module Name | Ident | Bytes | File | Creation Date | Creator |
|---|---|---|---|---|---|
| ECKAC | V08.00 | 8 | [VAX750.MIC]ECKAC0.OBJ;3 | 17-FEB-1986 13:37 | VAX/VMS Macro V04-00 |
| ECKAC | V08.00 | 22064 | [VAX750.MIC]ECKAC1.OBJ;2 | 17-FEB-1986 13:38 | VAX/VMS Macro V04-00 |
| ECKAC | V08.00 | 29614 | [VAX750.MIC]ECKAC2.OBJ;2 | 17-FEB-1986 13:40 | VAX/VMS Macro V04-00 |
| %LINK-W-WRNERS, compilation warnings in module ECKAC file DISK$UCODPACK00:[VAX750.MIC]ECKAC2.OBJ;2 | | | | | |
| ECKAC | V08.00 | 22307 | [VAX750.MIC]ECKAC3.OBJ;8 | 3-MAR-1986 10:49 | VAX/VMS Macro V04-00 |
| %LINK-W-WRNERS, compilation warnings in module ECKAC file DISK$UCODPACK00:[VAX750.MIC]ECKAC3.OBJ;8 | | | | | |
| ECKAC | V08.00 | 20343 | [VAX750.MIC]ECKAC4.OBJ;4 | 27-FEB-1986 11:42 | VAX/VMS Macro V04-00 |
| %LINK-W-WRNERS, compilation warnings in module ECKAC file DISK$UCODPACK00:[VAX750.MIC]ECKAC4.OBJ;4 | | | | | |

## Program Section Synopsis

| Psect Name | Module Name | Base | End | Length | | Align | Attributes |
|---|---|---|---|---|---|---|---|
| A_DIRECTORY | | 00000000 | 00000001 | 00000002 | ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000000 | 00000001 | 00000002 | ( 2.) | BYTE 0 | |
| DIRECTORY | | 00000002 | 000000FF | 000000FE | ( 254.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000002 | 00000009 | 00000008 | ( 8.) | BYTE 0 | |
| | ECKAC | 0000000A | 00000039 | 00000030 | ( 48.) | BYTE 0 | |
| | ECKAC | 0000003A | 00000067 | 0000002E | ( 46.) | BYTE 0 | |
| | ECKAC | 00000068 | 0000008A | 00000023 | ( 35.) | BYTE 0 | |
| | ECKAC | 0000008B | 000000FF | 00000075 | ( 117.) | BYTE 0 | |
| X_0001_D | | 00000100 | 00000101 | 00000002 | ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000100 | 00000101 | 00000002 | ( 2.) | BYTE 0 | |
| X_0001_ERTEMP | | 00000102 | 00000112 | 00000011 | ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000102 | 00000112 | 00000011 | ( 17.) | BYTE 0 | |
| X_0001_FCACHE | | 00000113 | 00000113 | 00000001 | ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000113 | 00000113 | 00000001 | ( 1.) | BYTE 0 | |
| X_0001_GDCS | | 00000114 | 0000014B | 00000038 | ( 56.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000114 | 0000014B | 00000038 | ( 56.) | BYTE 0 | |
| X_0001_HFILL | | 0000014C | 0000017F | 00000034 | ( 52.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000014C | 0000017F | 00000034 | ( 52.) | BYTE 0 | |
| X_0001_T | | 00000180 | 000002FF | 00000180 | ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000180 | 000002FF | 00000180 | ( 384.) | BYTE 0 | |
| X_0002_D | | 00000300 | 00000301 | 00000002 | ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000300 | 00000301 | 00000002 | ( 2.) | BYTE 0 | |
| X_0002_ERTEMP | | 00000302 | 0000030A | 00000009 | ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000302 | 0000030A | 00000009 | ( 9.) | BYTE 0 | |
| X_0002_FCACHE | | 0000030B | 0000030B | 00000001 | ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000030B | 0000030B | 00000001 | ( 1.) | BYTE 0 | |
| X_0002_GDCS | | 0000030C | 00000343 | 00000038 | ( 56.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000030C | 00000343 | 00000038 | ( 56.) | BYTE 0 | |
| X_0002_HFILL | | 00000344 | 0000037F | 0000003C | ( 60.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0002_HFILL | | 00000344 | 0000037F | 0000003C ( 60.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000344 | 0000037F | 0000003C ( 60.) | BYTE 0 | |
| X_0002_T | | 00000380 | 0000047F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000380 | 0000047F | 00000100 ( 256.) | BYTE 0 | |
| X_0003_D | | 00000480 | 00000481 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000480 | 00000481 | 00000002 ( 2.) | BYTE 0 | |
| X_0003_ERTEMP | | 00000482 | 0000048A | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000482 | 0000048A | 00000009 ( 9.) | BYTE 0 | |
| X_0003_FCACHE | | 0000048B | 0000048B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000048B | 0000048B | 00000001 ( 1.) | BYTE 0 | |
| X_0003_GDCS | | 0000048C | 000004C3 | 00000038 ( 56.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000048C | 000004C3 | 00000038 ( 56.) | BYTE 0 | |
| X_0003_HFILL | | 000004C4 | 000004FF | 0000003C ( 60.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000004C4 | 000004FF | 0000003C ( 60.) | BYTE 0 | |
| X_0003_T | | 00000500 | 000005FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000500 | 000005FF | 00000100 ( 256.) | BYTE 0 | |
| X_0004_D | | 00000600 | 00000601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000600 | 00000601 | 00000002 ( 2.) | BYTE 0 | |
| X_0004_ERTEMP | | 00000602 | 00000612 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000602 | 00000612 | 00000011 ( 17.) | BYTE 0 | |
| X_0004_FCACHE | | 00000613 | 00000613 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000613 | 00000613 | 00000001 ( 1.) | BYTE 0 | |
| X_0004_GDCS | | 00000614 | 00000677 | 00000064 ( 100.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000614 | 00000677 | 00000064 ( 100.) | BYTE 0 | |
| X_0004_HFILL | | 00000678 | 0000067F | 00000008 ( 8.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000678 | 0000067F | 00000008 ( 8.) | BYTE 0 | |
| X_0004_T | | 00000680 | 0000077F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000680 | 0000077F | 00000100 ( 256.) | BYTE 0 | |
| X_0005_D | | 00000780 | 00000781 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000780 | 00000781 | 00000002 ( 2.) | BYTE 0 | |
| X_0005_ERTEMP | | 00000782 | 00000786 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000782 | 00000786 | 00000005 ( 5.) | BYTE 0 | |
| X_0005_FCACHE | | 00000787 | 00000787 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000787 | 00000787 | 00000001 ( 1.) | BYTE 0 | |
| X_0005_GDCS | | 00000788 | 000007CA | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000788 | 000007CA | 00000043 ( 67.) | BYTE 0 | |
| X_0005_HFILL | | 000007CB | 000007FF | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000007CB | 000007FF | 00000035 ( 53.) | BYTE 0 | |
| X_0005_T | | 00000800 | 000008FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000800 | 000008FF | 00000100 ( 256.) | BYTE 0 | |
| X_0006_D | | 00000900 | 00000901 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000900 | 00000901 | 00000002 ( 2.) | BYTE 0 | |
| X_0006_ERTEMP | | 00000902 | 00000902 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000902 | 00000902 | 00000001 ( 1.) | BYTE 0 | |
| X_0006_FCACHE | | 00000903 | 00000903 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000903 | 00000903 | 00000001 ( 1.) | BYTE 0 | |
| X_0006_GDCS | | 00000904 | 00000930 | 0000002D ( 45.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000904 | 00000930 | 0000002D ( 45.) | BYTE 0 | |
| X_0006_HFILL | | 00000931 | 0000097F | 0000004F ( 79.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000931 | 0000097F | 0000004F ( 79.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0006_T | | 00000980 | 00000A7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000980 | 00000A7F | 00000100 ( 256.) | BYTE 0 | |
| X_0007_D | | 00000A80 | 00000A81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000A80 | 00000A81 | 00000002 ( 2.) | BYTE 0 | |
| X_0007_ERTEMP | | 00000A82 | 00000A86 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000A82 | 00000A86 | 00000005 ( 5.) | BYTE 0 | |
| X_0007_FCACHE | | 00000A87 | 00000A87 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000A87 | 00000A87 | 00000001 ( 1.) | BYTE 0 | |
| X_0007_GDCS | | 00000A88 | 00000ABF | 00000038 ( 56.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000A88 | 00000ABF | 00000038 ( 56.) | BYTE 0 | |
| X_0007_HFILL | | 00000AC0 | 00000AFF | 00000040 ( 64.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000AC0 | 00000AFF | 00000040 ( 64.) | BYTE 0 | |
| X_0007_T | | 00000B00 | 00000B7F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000B00 | 00000B7F | 00000080 ( 128.) | BYTE 0 | |
| X_0008_D | | 00000B80 | 00000B81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000B80 | 00000B81 | 00000002 ( 2.) | BYTE 0 | |
| X_0008_ERTEMP | | 00000B82 | 00000B82 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000B82 | 00000B82 | 00000001 ( 1.) | BYTE 0 | |
| X_0008_FCACHE | | 00000B83 | 00000B83 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000B83 | 00000B83 | 00000001 ( 1.) | BYTE 0 | |
| X_0008_GDCS | | 00000B84 | 00000BB0 | 0000002D ( 45.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000B84 | 00000BB0 | 0000002D ( 45.) | BYTE 0 | |
| X_0008_HFILL | | 00000BB1 | 00000BFF | 0000004F ( 79.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000BB1 | 00000BFF | 0000004F ( 79.) | BYTE 0 | |
| X_0008_T | | 00000C00 | 00000C7F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000C00 | 00000C7F | 00000080 ( 128.) | BYTE 0 | |
| X_0009_D | | 00000C80 | 00000C81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000C80 | 00000C81 | 00000002 ( 2.) | BYTE 0 | |
| X_0009_ERTEMP | | 00000C82 | 00000C82 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000C82 | 00000C82 | 00000001 ( 1.) | BYTE 0 | |
| X_0009_FCACHE | | 00000C83 | 00000C83 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000C83 | 00000C83 | 00000001 ( 1.) | BYTE 0 | |
| X_0009_GDCS | | 00000C84 | 00000CB0 | 0000002D ( 45.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000C84 | 00000CB0 | 0000002D ( 45.) | BYTE 0 | |
| X_0009_HFILL | | 00000CB1 | 00000CFF | 0000004F ( 79.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000CB1 | 00000CFF | 0000004F ( 79.) | BYTE 0 | |
| X_0009_T | | 00000D00 | 00000D7F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000D00 | 00000D7F | 00000080 ( 128.) | BYTE 0 | |
| X_0010_D | | 00000D80 | 00000D81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000D80 | 00000D81 | 00000002 ( 2.) | BYTE 0 | |
| X_0010_ERTEMP | | 00000D82 | 00000D82 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000D82 | 00000D82 | 00000001 ( 1.) | BYTE 0 | |
| X_0010_FCACHE | | 00000D83 | 00000D83 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000D83 | 00000D83 | 00000001 ( 1.) | BYTE 0 | |
| X_0010_GDCS | | 00000D84 | 00000DB0 | 0000002D ( 45.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000D84 | 00000DB0 | 0000002D ( 45.) | BYTE 0 | |
| X_0010_HFILL | | 00000DB1 | 00000DFF | 0000004F ( 79.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000DB1 | 00000DFF | 0000004F ( 79.) | BYTE 0 | |
| X_0010_T | | 00000E00 | 00000E7F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000E00 | 00000E7F | 00000080 ( 128.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0011_D | | 00000E80 | 00000E81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000E80 | 00000E81 | 00000002 ( 2.) | BYTE 0 | |
| X_0011_ERTEMP | | 00000E82 | 00000E92 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000E82 | 00000E92 | 00000011 ( 17.) | BYTE 0 | |
| X_0011_FCACHE | | 00000E93 | 00000E93 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000E93 | 00000E93 | 00000001 ( 1.) | BYTE 0 | |
| X_0011_GDCS | | 00000E94 | 00000F02 | 0000006F ( 111.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000E94 | 00000F02 | 0000006F ( 111.) | BYTE 0 | |
| X_0011_HFILL | | 00000F03 | 00000F7F | 0000007D ( 125.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000F03 | 00000F7F | 0000007D ( 125.) | BYTE 0 | |
| X_0011_T | | 00000F80 | 0000107F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00000F80 | 0000107F | 00000100 ( 256.) | BYTE 0 | |
| X_0012_D | | 00001080 | 00001081 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001080 | 00001081 | 00000002 ( 2.) | BYTE 0 | |
| X_0012_ERTEMP | | 00001082 | 00001082 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001082 | 00001082 | 00000001 ( 1.) | BYTE 0 | |
| X_0012_FCACHE | | 00001083 | 00001083 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001083 | 00001083 | 00000001 ( 1.) | BYTE 0 | |
| X_0012_GDCS | | 00001084 | 000010B0 | 0000002D ( 45.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001084 | 000010B0 | 0000002D ( 45.) | BYTE 0 | |
| X_0012_HFILL | | 000010B1 | 000010FF | 0000004F ( 79.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000010B1 | 000010FF | 0000004F ( 79.) | BYTE 0 | |
| X_0012_T | | 00001100 | 0000117F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001100 | 0000117F | 00000080 ( 128.) | BYTE 0 | |
| X_0013_D | | 00001180 | 00001181 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001180 | 00001181 | 00000002 ( 2.) | BYTE 0 | |
| X_0013_ERTEMP | | 00001182 | 00001182 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001182 | 00001182 | 00000001 ( 1.) | BYTE 0 | |
| X_0013_FCACHE | | 00001183 | 00001183 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001183 | 00001183 | 00000001 ( 1.) | BYTE 0 | |
| X_0013_GDCS | | 00001184 | 000011C6 | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001184 | 000011C6 | 00000043 ( 67.) | BYTE 0 | |
| X_0013_HFILL | | 000011C7 | 000011FF | 00000039 ( 57.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000011C7 | 000011FF | 00000039 ( 57.) | BYTE 0 | |
| X_0013_T | | 00001200 | 000012FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001200 | 000012FF | 00000100 ( 256.) | BYTE 0 | |
| X_0014_D | | 00001300 | 00001301 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001300 | 00001301 | 00000002 ( 2.) | BYTE 0 | |
| X_0014_ERTEMP | | 00001302 | 00001302 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001302 | 00001302 | 00000001 ( 1.) | BYTE 0 | |
| X_0014_FCACHE | | 00001303 | 00001307 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001303 | 00001307 | 00000005 ( 5.) | BYTE 0 | |
| X_0014_GDCS | | 00001308 | 0000134A | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001308 | 0000134A | 00000043 ( 67.) | BYTE 0 | |
| X_0014_HFILL | | 0000134B | 0000137F | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000134B | 0000137F | 00000035 ( 53.) | BYTE 0 | |
| X_0014_T | | 00001380 | 000013FF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001380 | 000013FF | 00000080 ( 128.) | BYTE 0 | |
| X_0015_D | | 00001400 | 00001401 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001400 | 00001401 | 00000002 ( 2.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0015_ERTEMP | | 00001402 | 00001402 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001402 | 00001402 | 00000001 ( 1.) | BYTE 0 | |
| X_0015_FCACHE | | 00001403 | 00001407 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001403 | 00001407 | 00000005 ( 5.) | BYTE 0 | |
| X_0015_GDCS | | 00001408 | 0000144A | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001408 | 0000144A | 00000043 ( 67.) | BYTE 0 | |
| X_0015_HFILL | | 0000144B | 0000147F | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000144B | 0000147F | 00000035 ( 53.) | BYTE 0 | |
| X_0015_T | | 00001480 | 000014FF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001480 | 000014FF | 00000080 ( 128.) | BYTE 0 | |
| X_0016_D | | 00001500 | 00001501 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001500 | 00001501 | 00000002 ( 2.) | BYTE 0 | |
| X_0016_ERTEMP | | 00001502 | 00001502 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001502 | 00001502 | 00000001 ( 1.) | BYTE 0 | |
| X_0016_FCACHE | | 00001503 | 00001507 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001503 | 00001507 | 00000005 ( 5.) | BYTF 0 | |
| X_0016_GDCS | | 00001508 | 0000154A | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001508 | 0000154A | 00000043 ( 67.) | BYTE 0 | |
| X_0016_HFILL | | 0000154B | 0000157F | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000154B | 0000157F | 00000035 ( 53.) | BYTE 0 | |
| X_0016_T | | 00001580 | 000015FF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001580 | 000015FF | 00000080 ( 128.) | BYTE 0 | |
| X_0017_D | | 00001600 | 00001601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001600 | 00001601 | 00000002 ( 2.) | BYTE 0 | |
| X_0017_ERTEMP | | 00001602 | 00001602 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001602 | 00001602 | 00000001 ( 1.) | BYTE 0 | |
| X_0017_FCACHE | | 00001603 | 00001603 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001603 | 00001603 | 00000001 ( 1.) | BYTE 0 | |
| X_0017_GDCS | | 00001604 | 0000163B | 00000038 ( 56.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001604 | 0000163B | 00000038 ( 56.) | BYTE 0 | |
| X_0017_HFILL | | 0000163C | 0000167F | 00000044 ( 68.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000163C | 0000167F | 00000044 ( 68.) | BYTE 0 | |
| X_0017_T | | 00001680 | 000016FF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001680 | 000016FF | 00000080 ( 128.) | BYTE 0 | |
| X_0018_D | | 00001700 | 00001701 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001700 | 00001701 | 00000002 ( 2.) | BYTE 0 | |
| X_0018_ERTEMP | | 00001702 | 0000171A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001702 | 0000171A | 00000019 ( 25.) | BYTE 0 | |
| X_0018_FCACHE | | 0000171B | 0000171B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000171B | 0000171B | 00000001 ( 1.) | BYTE 0 | |
| X_0018_GDCS | | 0000171C | 0000171C | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000171C | 0000171C | 00000001 ( 1.) | BYTE 0 | |
| X_0018_HFILL | | 0000171D | 0000177F | 00000063 ( 99.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000171D | 0000177F | 00000063 ( 99.) | BYTE 0 | |
| X_0018_T | | 00001780 | 000019FF | 00000280 ( 640.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001780 | 000019FF | 00000280 ( 640.) | BYTE 0 | |
| X_0019_D | | 00001A00 | 00001A01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001A00 | 00001A01 | 00000002 ( 2.) | BYTE 0 | |
| X_0019_T | | 00001A02 | 00001AFF | 000000FE ( 254.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001A02 | 00001AFF | 000000FE ( 254.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0020_D | | 00001B00 | 00001B01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001B00 | 00001B01 | 00000002 ( 2.) | BYTE 0 | |
| X_0020_ERTEMP | | 00001B02 | 00001B02 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001B02 | 00001B02 | 00000001 ( 1.) | BYTE 0 | |
| X_0020_FCACHE | | 00001B03 | 00001B03 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001B03 | 00001B03 | 00000001 ( 1.) | BYTE 0 | |
| X_0020_GDCS | | 00001B04 | 00001B67 | 00000064 ( 100.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001B04 | 00001B67 | 00000064 ( 100.) | BYTE 0 | |
| X_0020_HFILL | | 00001B68 | 00001B7F | 00000018 ( 24.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001B68 | 00001B7F | 00000018 ( 24.) | BYTE 0 | |
| X_0020_T | | 00001B80 | 00001BFF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001B80 | 00001BFF | 00000080 ( 128.) | BYTE 0 | |
| X_0021_D | | 00001C00 | 00001C01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001C00 | 00001C01 | 00000002 ( 2.) | BYTE 0 | |
| X_0021_ERTEMP | | 00001C02 | 00001C06 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001C02 | 00001C06 | 00000005 ( 5.) | BYTE 0 | |
| X_0021_FCACHE | | 00001C07 | 00001C07 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001C07 | 00001C07 | 00000001 ( 1.) | BYTE 0 | |
| X_0021_GDCS | | 00001C08 | 00001C08 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001C08 | 00001C08 | 00000001 ( 1.) | BYTE 0 | |
| X_0021_HFILL | | 00001C09 | 00001C7F | 00000077 ( 119.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001C09 | 00001C7F | 00000077 ( 119.) | BYTE 0 | |
| X_0021_T | | 00001C80 | 00001E7F | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001C80 | 00001E7F | 00000200 ( 512.) | BYTE 0 | |
| X_0022_D | | 00001E80 | 00001E81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001E80 | 00001E81 | 00000002 ( 2.) | BYTE 0 | |
| X_0022_ERTEMP | | 00001E82 | 00001E86 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001E82 | 00001E86 | 00000005 ( 5.) | BYTE 0 | |
| X_0022_FCACHE | | 00001E87 | 00001E87 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001E87 | 00001E87 | 00000001 ( 1.) | BYTE 0 | |
| X_0022_GDCS | | 00001E88 | 00001E88 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001E88 | 00001E88 | 00000001 ( 1.) | BYTE 0 | |
| X_0022_HFILL | | 00001E89 | 00001EFF | 00000077 ( 119.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001E89 | 00001EFF | 00000077 ( 119.) | BYTE 0 | |
| X_0022_T | | 00001F00 | 000020FF | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00001F00 | 000020FF | 00000200 ( 512.) | BYTE 0 | |
| X_0023_D | | 00002100 | 00002101 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002100 | 00002101 | 00000002 ( 2.) | BYTE 0 | |
| X_0023_ERTEMP | | 00002102 | 00002106 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002102 | 00002106 | 00000005 ( 5.) | BYTE 0 | |
| X_0023_FCACHE | | 00002107 | 00002107 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002107 | 00002107 | 00000001 ( 1.) | BYTE 0 | |
| X_0023_GDCS | | 00002108 | 00002108 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002108 | 00002108 | 00000001 ( 1.) | BYTE 0 | |
| X_0023_HFILL | | 00002109 | 0000217F | 00000077 ( 119.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002109 | 0000217F | 00000077 ( 119.) | BYTE 0 | |
| X_0023_T | | 00002180 | 0000237F | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002180 | 0000237F | 00000200 ( 512.) | BYTE 0 | |
| X_0024_D | | 00002380 | 00002381 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002380 | 00002381 | 00000002 ( 2.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0024_ERTEMP | | 00002382 | 00002386 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002382 | 00002386 | 00000005 ( 5.) | BYTE 0 | |
| X_0024_FCACHE | | 00002387 | 00002387 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002387 | 00002387 | 00000001 ( 1.) | BYTE 0 | |
| X_0024_GDCS | | 00002388 | 00002388 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002388 | 00002388 | 00000001 ( 1.) | BYTE 0 | |
| X_0024_HFILL | | 00002389 | 000023FF | 00000077 ( 119.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002389 | 000023FF | 00000077 ( 119.) | BYTE 0 | |
| X_0024_T | | 00002400 | 000025FF | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002400 | 000025FF | 00000200 ( 512.) | BYTE 0 | |
| X_0025_D | | 00002600 | 00002601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002600 | 00002601 | 00000002 ( 2.) | BYTE 0 | |
| X_0025_T | | 00002602 | 000027FF | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002602 | 000027FF | 000001FE ( 510.) | BYTE 0 | |
| X_0026_D | | 00002800 | 00002801 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002800 | 00002801 | 00000002 ( 2.) | BYTE 0 | |
| X_0026_ERTEMP | | 00002802 | 00002816 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002802 | 00002816 | 00000015 ( 21.) | BYTE 0 | |
| X_0026_FCACHE | | 00002817 | 00002817 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002817 | 00002817 | 00000001 ( 1.) | BYTE 0 | |
| X_0026_GDCS | | 00002818 | 000028DE | 000000C7 ( 199.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002818 | 000028DE | 000000C7 ( 199.) | BYTE 0 | |
| X_0026_HFILL | | 000028DF | 000028FF | 00000021 ( 33.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000028DF | 000028FF | 00000021 ( 33.) | BYTE 0 | |
| X_0026_T | | 00002900 | 000029FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002900 | 000029FF | 00000100 ( 256.) | BYTE 0 | |
| X_0027_D | | 00002A00 | 00002A01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002A00 | 00002A01 | 00000002 ( 2.) | BYTE 0 | |
| X_0027_T | | 00002A02 | 00002C7F | 0000027E ( 638.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002A02 | 00002C7F | 0000027E ( 638.) | BYTE 0 | |
| X_0028_D | | 00002C80 | 00002C81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002C80 | 00002C81 | 00000002 ( 2.) | BYTE 0 | |
| X_0028_ERTEMP | | 00002C82 | 00002C92 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002C82 | 00002C92 | 00000011 ( 17.) | BYTE 0 | |
| X_0028_FCACHE | | 00002C93 | 00002C93 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002C93 | 00002C93 | 00000001 ( 1.) | BYTE 0 | |
| X_0028_GDCS | | 00002C94 | 00002C94 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002C94 | 00002C94 | 00000001 ( 1.) | BYTE 0 | |
| X_0028_HFILL | | 00002C95 | 00002CFF | 0000006B ( 107.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002C95 | 00002CFF | 0000006B ( 107.) | BYTE 0 | |
| X_0028_T | | 00002D00 | 00002FFF | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00002D00 | 00002FFF | 00000300 ( 768.) | BYTE 0 | |
| X_0029_D | | 00003000 | 00003001 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003000 | 00003001 | 00000002 ( 2.) | BYTE 0 | |
| X_0029_ERTEMP | | 00003002 | 0000300E | 0000000D ( 13.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003002 | 0000300E | 0000000D ( 13.) | BYTE 0 | |
| X_0029_FCACHE | | 0000300F | 0000300F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000300F | 0000300F | 00000001 ( 1.) | BYTE 0 | |
| X_0029_GDCS | | 00003010 | 00003010 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003010 | 00003010 | 00000001 ( 1.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0029_HFILL | | 00003011 | 0000307F | 0000006F ( 111.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003011 | 0000307F | 0000006F ( 111.) | BYTE 0 | |
| X_0029_T | | 00003080 | 000033FF | 00000380 ( 896.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003080 | 000033FF | 00000380 ( 896.) | BYTE 0 | |
| X_0030_D | | 00003400 | 00003401 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003400 | 00003401 | 00000002 ( 2.) | BYTE 0 | |
| X_0030_ERTEMP | | 00003402 | 00003402 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003402 | 00003402 | 00000001 ( 1.) | BYTE 0 | |
| X_0030_FCACHE | | 00003403 | 00003403 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003403 | 00003403 | 00000001 ( 1.) | BYTE 0 | |
| X_0030_GDCS | | 00003404 | 00003425 | 00000022 ( 34.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003404 | 00003425 | 00000022 ( 34.) | BYTE 0 | |
| X_0030_HFILL | | 00003426 | 0000347F | 0000005A ( 90.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003426 | 0000347F | 0000005A ( 90.) | BYTE 0 | |
| X_0030_T | | 00003480 | 0000357F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003480 | 0000357F | 00000100 ( 256.) | BYTE 0 | |
| X_0031_D | | 00003580 | 00003581 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003580 | 00003581 | 00000002 ( 2.) | BYTE 0 | |
| X_0031_ERTEMP | | 00003582 | 0000358E | 0000000D ( 13.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003582 | 0000358E | 0000000D ( 13.) | BYTE 0 | |
| X_0031_FCACHE | | 0000358F | 00003593 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000358F | 00003593 | 00000005 ( 5.) | BYTE 0 | |
| X_0031_GDCS | | 00003594 | 000036B2 | 0000011F ( 287.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003594 | 000036B2 | 0000011F ( 287.) | BYTE 0 | |
| X_0031_HFILL | | 000036B3 | 000036FF | 0000004D ( 77.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000036B3 | 000036FF | 0000004D ( 77.) | BYTE 0 | |
| X_0031_T | | 00003700 | 000037FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003700 | 000037FF | 00000100 ( 256.) | BYTE 0 | |
| X_0032_D | | 00003800 | 00003801 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003800 | 00003801 | 00000002 ( 2.) | BYTE 0 | |
| X_0032_ERTEMP | | 00003802 | 00003802 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003802 | 00003802 | 00000001 ( 1.) | BYTE 0 | |
| X_0032_FCACHE | | 00003803 | 00003803 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003803 | 00003803 | 00000001 ( 1.) | BYTE 0 | |
| X_0032_GDCS | | 00003804 | 00003851 | 0000004E ( 78.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003804 | 00003851 | 0000004E ( 78.) | BYTE 0 | |
| X_0032_HFILL | | 00003852 | 0000387F | 0000002E ( 46.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003852 | 0000387F | 0000002E ( 46.) | BYTE 0 | |
| X_0032_T | | 00003880 | 000038FF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003880 | 000038FF | 00000080 ( 128.) | BYTE 0 | |
| X_0033_D | | 00003900 | 00003901 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003900 | 00003901 | 00000002 ( 2.) | BYTE 0 | |
| X_0033_ERTEMP | | 00003902 | 00003902 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003902 | 00003902 | 00000001 ( 1.) | BYTE 0 | |
| X_0033_FCACHE | | 00003903 | 00003953 | 00000051 ( 81.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003903 | 00003953 | 00000051 ( 81.) | BYTE 0 | |
| X_0033_GDCS | | 00003954 | 000039A1 | 0000004E ( 78.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003954 | 000039A1 | 0000004E ( 78.) | BYTE 0 | |
| X_0033_HFILL | | 000039A2 | 000039FF | 0000005E ( 94.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000039A2 | 000039FF | 0000005E ( 94.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0033_T | | 00003A00 | 00003AFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003A00 | 00003AFF | 00000100 ( 256.) | BYTE 0 | |
| X_0034_D | | 00003B00 | 00003B01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003B00 | 00003B01 | 00000002 ( 2.) | BYTE 0 | |
| X_0034_ERTEMP | | 00003B02 | 00003B0A | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003B02 | 00003B0A | 00000009 ( 9.) | BYTE 0 | |
| X_0034_FCACHE | | 00003B0B | 00003B0B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003B0B | 00003B0B | 00000001 ( 1.) | BYTE 0 | |
| X_0034_GDCS | | 00003B0C | 00003BA6 | 0000009B ( 155.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003B0C | 00003BA6 | 0000009B ( 155.) | BYTE 0 | |
| X_0034_HFILL | | 00003BA7 | 00003BFF | 00000059 ( 89.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003BA7 | 00003BFF | 00000059 ( 89.) | BYTE 0 | |
| X_0034_T | | 00003C00 | 00003CFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003C00 | 00003CFF | 00000100 ( 256.) | BYTE 0 | |
| X_0035_D | | 00003D00 | 00003D01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003D00 | 00003D01 | 00000002 ( 2.) | BYTE 0 | |
| X_0035_ERTEMP | | 00003D02 | 00003D02 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003D02 | 00003D02 | 00000001 ( 1.) | BYTE 0 | |
| X_0035_FCACHE | | 00003D03 | 00003D03 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003D03 | 00003D03 | 00000001 ( 1.) | BYTE 0 | |
| X_0035_GDCS | | 00003D04 | 00003D51 | 0000004E ( 78.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003D04 | 00003D51 | 0000004E ( 78.) | BYTE 0 | |
| X_0035_HFILL | | 00003D52 | 00003D7F | 0000002E ( 46.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003D52 | 00003D7F | 0000002E ( 46.) | BYTE 0 | |
| X_0035_T | | 00003D80 | 00003E7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003D80 | 00003E7F | 00000100 ( 256.) | BYTE 0 | |
| X_0036_D | | 00003E80 | 00003E81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003E80 | 00003E81 | 00000002 ( 2.) | BYTE 0 | |
| X_0036_ERTEMP | | 00003E82 | 00003E86 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003E82 | 00003E86 | 00000005 ( 5.) | BYTE 0 | |
| X_0036_FCACHE | | 00003E87 | 00003E87 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003E87 | 00003E87 | 00000001 ( 1.) | BYTE 0 | |
| X_0036_GDCS | | 00003E88 | 00003F0C | 00000085 ( 133.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003E88 | 00003F0C | 00000085 ( 133.) | BYTE 0 | |
| X_0036_HFILL | | 00003F0D | 00003F7F | 00000073 ( 115.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003F0D | 00003F7F | 00000073 ( 115.) | BYTE 0 | |
| X_0036_T | | 00003F80 | 0000407F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00003F80 | 0000407F | 00000100 ( 256.) | BYTE 0 | |
| X_0037_D | | 00004080 | 00004081 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004080 | 00004081 | 00000002 ( 2.) | BYTE 0 | |
| X_0037_ERTEMP | | 00004082 | 0000408A | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004082 | 0000408A | 00000009 ( 9.) | BYTE 0 | |
| X_0037_FCACHE | | 0000408B | 0000408B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000408B | 0000408B | 00000001 ( 1.) | BYTE 0 | |
| X_0037_GDCS | | 0000408C | 000040D9 | 0000004E ( 78.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000408C | 000040D9 | 0000004E ( 78.) | BYTE 0 | |
| X_0037_HFILL | | 000040DA | 000040FF | 00000026 ( 38.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000040DA | 000040FF | 00000026 ( 38.) | BYTE 0 | |
| X_0037_T | | 00004100 | 0000417F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004100 | 0000417F | 00000080 ( 128.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0038_D | | 00004180 | 00004181 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004180 | 00004181 | 00000002 ( 2.) | BYTE 0 | |
| X_0038_T | | 00004182 | 000042FF | 0000017E ( 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004182 | 000042FF | 0000017E ( 382.) | BYTE 0 | |
| X_0039_D | | 00004300 | 00004301 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004300 | 00004301 | 00000002 ( 2.) | BYTE 0 | |
| X_0039_T | | 00004302 | 0000447F | 0000017E ( 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004302 | 0000447F | 0000017E ( 382.) | BYTE 0 | |
| X_0040_D | | 00004480 | 00004481 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004480 | 00004481 | 00000002 ( 2.) | BYTE 0 | |
| X_0040_T | | 00004482 | 000045FF | 0000017E ( 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004482 | 000045FF | 0000017E ( 382.) | BYTE 0 | |
| X_0041_D | | 00004600 | 00004601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004600 | 00004601 | 00000002 ( 2.) | BYTE 0 | |
| X_0041_T | | 00004602 | 0000477F | 0000017E ( 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004602 | 0000477F | 0000017E ( 382.) | BYTE 0 | |
| X_0042_D | | 00004780 | 00004781 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004780 | 00004781 | 00000002 ( 2.) | BYTE 0 | |
| X_0042_T | | 00004782 | 0000497F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004782 | 0000497F | 000001FE ( 510.) | BYTE 0 | |
| X_0043_D | | 00004980 | 00004981 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004980 | 00004981 | 00000002 ( 2.) | BYTE 0 | |
| X_0043_T | | 00004982 | 00004B7F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004982 | 00004B7F | 000001FE ( 510.) | BYTE 0 | |
| X_0044_D | | 00004B80 | 00004B81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004B80 | 00004B81 | 00000002 ( 2.) | BYTE 0 | |
| X_0044_ERTEMP | | 00004B82 | 00004B82 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004B82 | 00004B82 | 00000001 ( 1.) | BYTE 0 | |
| X_0044_FCACHE | | 00004B83 | 00004B83 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004B83 | 00004B83 | 00000001 ( 1.) | BYTE 0 | |
| X_0044_GDCS | | 00004B84 | 00004CD9 | 00000156 ( 342.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004B84 | 00004CD9 | 00000156 ( 342.) | BYTE 0 | |
| X_0044_HFILL | | 00004CDA | 00004CFF | 00000026 ( 38.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004CDA | 00004CFF | 00000026 ( 38.) | BYTE 0 | |
| X_0044_T | | 00004D00 | 00004DFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004D00 | 00004DFF | 00000100 ( 256.) | BYTE 0 | |
| X_0045_D | | 00004E00 | 00004E01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004E00 | 00004E01 | 00000002 ( 2.) | BYTE 0 | |
| X_0045_ERTEMP | | 00004E02 | 00004E02 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004E02 | 00004E02 | 00000001 ( 1.) | BYTE 0 | |
| X_0045_FCACHE | | 00004E03 | 00004E03 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004E03 | 00004E03 | 00000001 ( 1.) | BYTE 0 | |
| X_0045_GDCS | | 00004E04 | 00004F59 | 00000156 ( 342.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004E04 | 00004F59 | 00000156 ( 342.) | BYTE 0 | |
| X_0045_HFILL | | 00004F5A | 00004F7F | 00000026 ( 38.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004F5A | 00004F7F | 00000026 ( 38.) | BYTE 0 | |
| X_0045_T | | 00004F80 | 0000507F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00004F80 | 0000507F | 00000100 ( 256.) | BYTE 0 | |
| X_0046_D | | 00005080 | 00005081 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005080 | 00005081 | 00000002 ( 2.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0046_ERTEMP | | 00005082 | 00005082 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005082 | 00005082 | 00000001 ( 1.) | BYTE 0 | |
| X_0046_FCACHE | | 00005083 | 00005083 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005083 | 00005083 | 00000001 ( 1.) | BYTE 0 | |
| X_0046_GDCS | | 00005084 | 000051D9 | 00000156 ( 342.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005084 | 000051D9 | 00000156 ( 342.) | BYTE 0 | |
| X_0046_HFILL | | 000051DA | 000051FF | 00000026 ( 38.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000051DA | 000051FF | 00000026 ( 38.) | BYTE 0 | |
| X_0046_T | | 00005200 | 0000527F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005200 | 0000527F | 00000080 ( 128.) | BYTE 0 | |
| X_0047_D | | 00005280 | 00005281 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005280 | 00005281 | 00000002 ( 2.) | BYTE 0 | |
| X_0047_ERTEMP | | 00005282 | 00005282 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005282 | 00005282 | 00000001 ( 1.) | BYTE 0 | |
| X_0047_FCACHE | | 00005283 | 00005283 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005283 | 00005283 | 00000001 ( 1.) | BYTE 0 | |
| X_0047_GDCS | | 00005284 | 000053D9 | 00000156 ( 342.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005284 | 000053D9 | 00000156 ( 342.) | BYTE 0 | |
| X_0047_HFILL | | 000053DA | 000053FF | 00000026 ( 38.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000053DA | 000053FF | 00000026 ( 38.) | BYTE 0 | |
| X_0047_T | | 00005400 | 0000547F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005400 | 0000547F | 00000080 ( 128.) | BYTE 0 | |
| X_0048_D | | 00005480 | 00005481 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005480 | 00005481 | 00000002 ( 2.) | BYTE 0 | |
| X_0048_ERTEMP | | 00005482 | 00005482 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005482 | 00005482 | 00000001 ( 1.) | BYTE 0 | |
| X_0048_FCACHE | | 00005483 | 00005483 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005483 | 00005483 | 00000001 ( 1.) | BYTE 0 | |
| X_0048_GDCS | | 00005484 | 00005605 | 00000182 ( 386.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005484 | 00005605 | 00000182 ( 386.) | BYTE 0 | |
| X_0048_HFILL | | 00005606 | 0000567F | 0000007A ( 122.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005606 | 0000567F | 0000007A ( 122.) | BYTE 0 | |
| X_0048_T | | 00005680 | 000056FF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005680 | 000056FF | 00000080 ( 128.) | BYTE 0 | |
| X_0049_D | | 00005700 | 00005701 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005700 | 00005701 | 00000002 ( 2.) | BYTE 0 | |
| X_0049_ERTEMP | | 00005702 | 00005702 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005702 | 00005702 | 00000001 ( 1.) | BYTE 0 | |
| X_0049_FCACHE | | 00005703 | 00005703 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005703 | 00005703 | 00000001 ( 1.) | BYTE 0 | |
| X_0049_GDCS | | 00005704 | 00005885 | 00000182 ( 386.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005704 | 00005885 | 00000182 ( 386.) | BYTE 0 | |
| X_0049_HFILL | | 00005886 | 000058FF | 0000007A ( 122.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005886 | 000058FF | 0000007A ( 122.) | BYTE 0 | |
| X_0049_T | | 00005900 | 0000597F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005900 | 0000597F | 00000080 ( 128.) | BYTE 0 | |
| X_0050_D | | 00005980 | 00005981 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005980 | 00005981 | 00000002 ( 2.) | BYTE 0 | |
| X_0050_ERTEMP | | 00005982 | 00005982 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005982 | 00005982 | 00000001 ( 1.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0050_FCACHE | | 00005983 | 00005983 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005983 | 00005983 | 00000001 ( 1.) | BYTE 0 | |
| X_0050_GDCS | | 00005984 | 00005A13 | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005984 | 00005A13 | 00000090 ( 144.) | BYTE 0 | |
| X_0050_HFILL | | 00005A14 | 00005A7F | 0000006C ( 108.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005A14 | 00005A7F | 0000006C ( 108.) | BYTE 0 | |
| X_0050_T | | 00005A80 | 00005BFF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005A80 | 00005BFF | 00000180 ( 384.) | BYTE 0 | |
| X_0051_D | | 00005C00 | 00005C01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005C00 | 00005C01 | 00000002 ( 2.) | BYTE 0 | |
| X_0051_ERTEMP | | 00005C02 | 00005C02 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005C02 | 00005C02 | 00000001 ( 1.) | BYTE 0 | |
| X_0051_FCACHE | | 00005C03 | 00005C03 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005C03 | 00005C03 | 00000001 ( 1.) | BYTE 0 | |
| X_0051_GDCS | | 00005C04 | 00005C93 | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005C04 | 00005C93 | 00000090 ( 144.) | BYTE 0 | |
| X_0051_HFILL | | 00005C94 | 00005CFF | 0000006C ( 108.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005C94 | 00005CFF | 0000006C ( 108.) | BYTE 0 | |
| X_0051_T | | 00005D00 | 00005E7F | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005D00 | 00005E7F | 00000180 ( 384.) | BYTE 0 | |
| X_0052_D | | 00005E80 | 00005E81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005E80 | 00005E81 | 00000002 ( 2.) | BYTE 0 | |
| X_0052_ERTEMP | | 00005E82 | 00005E82 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005E82 | 00005E82 | 00000001 ( 1.) | BYTE 0 | |
| X_0052_FCACHE | | 00005E83 | 00005E83 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005E83 | 00005E83 | 00000001 ( 1.) | BYTE 0 | |
| X_0052_GDCS | | 00005E84 | 00005F1E | 0000009B ( 155.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005E84 | 00005F1E | 0000009B ( 155.) | BYTE 0 | |
| X_0052_HFILL | | 00005F1F | 00005F7F | 00000061 ( 97.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005F1F | 00005F7F | 00000061 ( 97.) | BYTE 0 | |
| X_0052_T | | 00005F80 | 000060FF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00005F80 | 000060FF | 00000180 ( 384.) | BYTE 0 | |
| X_0053_D | | 00006100 | 00006101 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006100 | 00006101 | 00000002 ( 2.) | BYTE 0 | |
| X_0053_ERTEMP | | 00006102 | 00006102 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006102 | 00006102 | 00000001 ( 1.) | BYTE 0 | |
| X_0053_FCACHE | | 00006103 | 00006103 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006103 | 00006103 | 00000001 ( 1.) | BYTE 0 | |
| X_0053_GDCS | | 00006104 | 0000619E | 0000009B ( 155.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006104 | 0000619E | 0000009B ( 155.) | BYTE 0 | |
| X_0053_HFILL | | 0000619F | 000061FF | 00000061 ( 97.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000619F | 000061FF | 00000061 ( 97.) | BYTE 0 | |
| X_0053_T | | 00006200 | 0000637F | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006200 | 0000637F | 00000180 ( 384.) | BYTE 0 | |
| X_0054_D | | 00006380 | 00006381 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006380 | 00006381 | 00000002 ( 2.) | BYTE 0 | |
| X_0054_ERTEMP | | 00006382 | 00006382 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006382 | 00006382 | 00000001 ( 1.) | BYTE 0 | |
| X_0054_FCACHE | | 00006383 | 00006383 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006383 | 00006383 | 00000001 ( 1.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | | Align | Attributes |
|---|---|---|---|---|---|---|---|
| X_0054_GDCS | | 00006384 | 00006413 | 00000090 ( | 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006384 | 00006413 | 00000090 ( | 144.) | BYTE 0 | |
| X_0054_HFILL | | 00006414 | 0000647F | 0000006C ( | 108.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006414 | 0000647F | 0000006C ( | 108.) | BYTE 0 | |
| X_0054_T | | 00006480 | 000065FF | 00000180 ( | 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006480 | 000065FF | 00000180 ( | 384.) | BYTE 0 | |
| X_0055_D | | 00006600 | 00006601 | 00000002 ( | 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006600 | 00006601 | 00000002 ( | 2.) | BYTE 0 | |
| X_0055_ERTEMP | | 00006602 | 00006602 | 00000001 ( | 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006602 | 00006602 | 00000001 ( | 1.) | BYTE 0 | |
| X_0055_FCACHE | | 00006603 | 00006603 | 00000001 ( | 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006603 | 00006603 | 00000001 ( | 1.) | BYTE 0 | |
| X_0055_GDCS | | 00006604 | 00006693 | 00000090 ( | 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006604 | 00006693 | 00000090 ( | 144.) | BYTE 0 | |
| X_0055_HFILL | | 00006694 | 000066FF | 0000006C ( | 108.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006694 | 000066FF | 0000006C ( | 108.) | BYTE 0 | |
| X_0055_T | | 00006700 | 0000687F | 00000180 ( | 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006700 | 0000687F | 00000180 ( | 384.) | BYTE 0 | |
| X_0056_D | | 00006880 | 00006881 | 00000002 ( | 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006880 | 00006881 | 00000002 ( | 2.) | BYTE 0 | |
| X_0056_ERTEMP | | 00006882 | 00006882 | 00000001 ( | 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006882 | 00006882 | 00000001 ( | 1.) | BYTE 0 | |
| X_0056_FCACHE | | 00006883 | 00006883 | 00000001 ( | 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006883 | 00006883 | 00000001 ( | 1.) | BYTE 0 | |
| X_0056_GDCS | | 00006884 | 0000691E | 0000009B ( | 155.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006884 | 0000691E | 0000009B ( | 155.) | BYTE 0 | |
| X_0056_HFILL | | 0000691F | 0000697F | 00000061 ( | 97.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000691F | 0000697F | 00000061 ( | 97.) | BYTE 0 | |
| X_0056_T | | 00006980 | 00006AFF | 00000180 ( | 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006980 | 00006AFF | 00000180 ( | 384.) | BYTE 0 | |
| X_0057_D | | 00006B00 | 00006B01 | 00000002 ( | 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006B00 | 00006B01 | 00000002 ( | 2.) | BYTE 0 | |
| X_0057_ERTEMP | | 00006B02 | 00006B02 | 00000001 ( | 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006B02 | 00006B02 | 00000001 ( | 1.) | BYTE 0 | |
| X_0057_FCACHE | | 00006B03 | 00006B03 | 00000001 ( | 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006B03 | 00006B03 | 00000001 ( | 1.) | BYTE 0 | |
| X_0057_GDCS | | 00006B04 | 00006B9E | 0000009B ( | 155.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006B04 | 00006B9E | 0000009B ( | 155.) | BYTE 0 | |
| X_0057_HFILL | | 00006B9F | 00006BFF | 00000061 ( | 97.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006B9F | 00006BFF | 00000061 ( | 97.) | BYTE 0 | |
| X_0057_T | | 00006C00 | 00006D7F | 00000180 ( | 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006C00 | 00006D7F | 00000180 ( | 384.) | BYTE 0 | |
| X_0058_D | | 00006D80 | 00006D81 | 00000002 ( | 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006D80 | 00006D81 | 00000002 ( | 2.) | BYTE 0 | |
| X_0058_T | | 00006D82 | 00006EFF | 0000017E ( | 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006D82 | 00006EFF | 0000017E ( | 382.) | BYTE 0 | |
| X_0059_D | | 00006F00 | 00006F01 | 00000002 ( | 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006F00 | 00006F01 | 00000002 ( | 2.) | BYTE 0 | |
| X_0059_T | | 00006F02 | 0000707F | 0000017E ( | 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00006F02 | 0000707F | 0000017E ( | 382.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0060_D | | 00007080 | 00007081 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007080 | 00007081 | 00000002 ( 2.) | BYTE 0 | |
| X_0060_T | | 00007082 | 0000727F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007082 | 0000727F | 000001FE ( 510.) | BYTE 0 | |
| X_0061_D | | 00007280 | 00007281 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007280 | 00007281 | 00000002 ( 2.) | BYTE 0 | |
| X_0061_T | | 00007282 | 0000747F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007282 | 0000747F | 000001FE ( 510.) | BYTE 0 | |
| X_0062_D | | 00007480 | 00007481 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007480 | 00007481 | 00000002 ( 2.) | BYTE 0 | |
| X_0062_T | | 00007482 | 0000767F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007482 | 0000767F | 000001FE ( 510.) | BYTE 0 | |
| X_0063_D | | 00007680 | 00007681 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007680 | 00007681 | 00000002 ( 2.) | BYTE 0 | |
| X_0063_T | | 00007682 | 0000787F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007682 | 0000787F | 000001FE ( 510.) | BYTE 0 | |
| X_0064_D | | 00007880 | 00007881 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007880 | 00007881 | 00000002 ( 2.) | BYTE 0 | |
| X_0064_ERTEMP | | 00007882 | 00007892 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007882 | 00007892 | 00000011 ( 17.) | BYTE 0 | |
| X_0064_FCACHE | | 00007893 | 00007893 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007893 | 00007893 | 00000001 ( 1.) | BYTE 0 | |
| X_0064_GDCS | | 00007894 | 00007894 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007894 | 00007894 | 00000001 ( 1.) | BYTE 0 | |
| X_0064_HFILL | | 00007895 | 000078FF | 0000006B ( 107.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007895 | 000078FF | 0000006B ( 107.) | BYTE 0 | |
| X_0064_T | | 00007900 | 00007BFF | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007900 | 00007BFF | 00000300 ( 768.) | BYTE 0 | |
| X_0065_D | | 00007C00 | 00007C01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007C00 | 00007C01 | 00000002 ( 2.) | BYTE 0 | |
| X_0065_ERTEMP | | 00007C02 | 00007C12 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007C02 | 00007C12 | 00000011 ( 17.) | BYTE 0 | |
| X_0065_FCACHE | | 00007C13 | 00007C13 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007C13 | 00007C13 | 00000001 ( 1.) | BYTE 0 | |
| X_0065_GDCS | | 00007C14 | 00007C14 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007C14 | 00007C14 | 00000001 ( 1.) | BYTE 0 | |
| X_0065_HFILL | | 00007C15 | 00007C7F | 0000006B ( 107.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007C15 | 00007C7F | 0000006B ( 107.) | BYTE 0 | |
| X_0065_T | | 00007C80 | 00007F7F | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007C80 | 00007F7F | 00000300 ( 768.) | BYTE 0 | |
| X_0066_D | | 00007F80 | 00007F81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007F80 | 00007F81 | 00000002 ( 2.) | BYTE 0 | |
| X_0066_ERTEMP | | 00007F82 | 00007F9E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007F82 | 00007F9E | 0000001D ( 29.) | BYTE 0 | |
| X_0066_FCACHE | | 00007F9F | 00007F9F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007F9F | 00007F9F | 00000001 ( 1.) | BYTE 0 | |
| X_0066_GDCS | | 00007FA0 | 00007FA0 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007FA0 | 00007FA0 | 00000001 ( 1.) | BYTE 0 | |
| X_0066_HFILL | | 00007FA1 | 00007FFF | 0000005F ( 95.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00007FA1 | 00007FFF | 0000005F ( 95.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0066_T | | 00008000 | 000082FF | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008000 | 000082FF | 00000300 ( 768.) | BYTE 0 | |
| X_0067_D | | 00008300 | 00008301 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008300 | 00008301 | 00000002 ( 2.) | BYTE 0 | |
| X_0067_ERTEMP | | 00008302 | 0000831E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008302 | 0000831E | 0000001D ( 29.) | BYTE 0 | |
| X_0067_FCACHE | | 0000831F | 0000831F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000831F | 0000831F | 00000001 ( 1.) | BYTE 0 | |
| X_0067_GDCS | | 00008320 | 00008320 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008320 | 00008320 | 00000001 ( 1.) | BYTE 0 | |
| X_0067_HFILL | | 00008321 | 0000837F | 0000005F ( 95.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008321 | 0000837F | 0000005F ( 95.) | BYTE 0 | |
| X_0067_T | | 00008380 | 0000867F | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008380 | 0000867F | 00000300 ( 768.) | BYTE 0 | |
| X_0068_D | | 00008680 | 00008681 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008680 | 00008681 | 00000002 ( 2.) | BYTE 0 | |
| X_0068_ERTEMP | | 00008682 | 0000869E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008682 | 0000869E | 0000001D ( 29.) | BYTE 0 | |
| X_0068_FCACHE | | 0000869F | 0000869F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000869F | 0000869F | 00000001 ( 1.) | BYTE 0 | |
| X_0068_GDCS | | 000086A0 | 000086A0 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000086A0 | 000086A0 | 00000001 ( 1.) | BYTE 0 | |
| X_0068_HFILL | | 000086A1 | 000086FF | 0000005F ( 95.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000086A1 | 000086FF | 0000005F ( 95.) | BYTE 0 | |
| X_0068_T | | 00008700 | 000089FF | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008700 | 000089FF | 00000300 ( 768.) | BYTE 0 | |
| X_0069_D | | 00008A00 | 00008A01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008A00 | 00008A01 | 00000002 ( 2.) | BYTE 0 | |
| X_0069_ERTEMP | | 00008A02 | 00008A1E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008A02 | 00008A1E | 0000001D ( 29.) | BYTE 0 | |
| X_0069_FCACHE | | 00008A1F | 00008A1F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008A1F | 00008A1F | 00000001 ( 1.) | BYTE 0 | |
| X_0069_GDCS | | 00008A20 | 00008A20 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008A20 | 00008A20 | 00000001 ( 1.) | BYTE 0 | |
| X_0069_HFILL | | 00008A21 | 00008A7F | 0000005F ( 95.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008A21 | 00008A7F | 0000005F ( 95.) | BYTE 0 | |
| X_0069_T | | 00008A80 | 00008D7F | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008A80 | 00008D7F | 00000300 ( 768.) | BYTE 0 | |
| X_0070_D | | 00008D80 | 00008D81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008D80 | 00008D81 | 00000002 ( 2.) | BYTE 0 | |
| X_0070_ERTEMP | | 00008D82 | 00008D8A | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008D82 | 00008D8A | 00000009 ( 9.) | BYTE 0 | |
| X_0070_FCACHE | | 00008D8B | 00008D8B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008D8B | 00008D8B | 00000001 ( 1.) | BYTE 0 | |
| X_0070_GDCS | | 00008D8C | 00008D8C | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008D8C | 00008D8C | 00000001 ( 1.) | BYTE 0 | |
| X_0070_HFILL | | 00008D8D | 00008DFF | 00000073 ( 115.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008D8D | 00008DFF | 00000073 ( 115.) | BYTE 0 | |
| X_0070_T | | 00008E00 | 00008FFF | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00008E00 | 00008FFF | 00000200 ( 512.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0071_D | | 00009000 | 00009001 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009000 | 00009001 | 00000002 ( 2.) | BYTE 0 | |
| X_0071_T | | 00009002 | 000090FF | 000000FE ( 254.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009002 | 000090FF | 000000FE ( 254.) | BYTE 0 | |
| X_0072_D | | 00009100 | 00009101 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009100 | 00009101 | 00000002 ( 2.) | BYTE 0 | |
| X_0072_ERTEMP | | 00009102 | 0000911E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009102 | 0000911E | 0000001D ( 29.) | BYTE 0 | |
| X_0072_FCACHE | | 0000911F | 0000911F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000911F | 0000911F | 00000001 ( 1.) | BYTE 0 | |
| X_0072_GDCS | | 00009120 | 00009233 | 00000114 ( 276.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009120 | 00009233 | 00000114 ( 276.) | BYTE 0 | |
| X_0072_HFILL | | 00009234 | 0000927F | 0000004C ( 76.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009234 | 0000927F | 0000004C ( 76.) | BYTE 0 | |
| X_0072_T | | 00009280 | 0000937F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009280 | 0000937F | 00000100 ( 256.) | BYTE 0 | |
| X_0073_D | | 00009380 | 00009381 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009380 | 00009381 | 00000002 ( 2.) | BYTE 0 | |
| X_0073_ERTEMP | | 00009382 | 0000939E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009382 | 0000939E | 0000001D ( 29.) | BYTE 0 | |
| X_0073_FCACHE | | 0000939F | 0000939F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000939F | 0000939F | 00000001 ( 1.) | BYTE 0 | |
| X_0073_GDCS | | 000093A0 | 000094B3 | 00000114 ( 276.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000093A0 | 000094B3 | 00000114 ( 276.) | BYTE 0 | |
| X_0073_HFILL | | 000094B4 | 000094FF | 0000004C ( 76.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000094B4 | 000094FF | 0000004C ( 76.) | BYTE 0 | |
| X_0073_T | | 00009500 | 000095FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009500 | 000095FF | 00000100 ( 256.) | BYTE 0 | |
| X_0074_D | | 00009600 | 00009601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009600 | 00009601 | 00000002 ( 2.) | BYTE 0 | |
| X_0074_ERTEMP | | 00009602 | 0000961A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009602 | 0000961A | 00000019 ( 25.) | BYTE 0 | |
| X_0074_FCACHE | | 0000961B | 0000961B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000961B | 0000961B | 00000001 ( 1.) | BYTE 0 | |
| X_0074_GDCS | | 0000961C | 000096D7 | 000000BC ( 188.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000961C | 000096D7 | 000000BC ( 188.) | BYTE 0 | |
| X_0074_HFILL | | 000096D8 | 000096FF | 00000028 ( 40.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000096D8 | 000096FF | 00000028 ( 40.) | BYTE 0 | |
| X_0074_T | | 00009700 | 000097FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009700 | 000097FF | 00000100 ( 256.) | BYTE 0 | |
| X_0075_D | | 00009800 | 00009801 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009800 | 00009801 | 00000002 ( 2.) | BYTE 0 | |
| X_0075_ERTEMP | | 00009802 | 0000981A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009802 | 0000981A | 00000019 ( 25.) | BYTE 0 | |
| X_0075_FCACHE | | 0000981B | 0000981B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000981B | 0000981B | 00000001 ( 1.) | BYTE 0 | |
| X_0075_GDCS | | 0000981C | 0000990E | 000000F3 ( 243.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000981C | 0000990E | 000000F3 ( 243.) | BYTE 0 | |
| X_0075_HFILL | | 0000990F | 0000997F | 00000071 ( 113.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000990F | 0000997F | 00000071 ( 113.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0075_T | | 00009980 | 00009A7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009980 | 00009A7F | 00000100 ( 256.) | BYTE 0 | |
| X_0076_D | | 00009A80 | 00009A81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009A80 | 00009A81 | 00000002 ( 2.) | BYTE 0 | |
| X_0076_ERTEMP | | 00009A82 | 00009A9A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009A82 | 00009A9A | 00000019 ( 25.) | BYTE 0 | |
| X_0076_FCACHE | | 00009A9B | 00009A9B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009A9B | 00009A9B | 00000001 ( 1.) | BYTE 0 | |
| X_0076_GDCS | | 00009A9C | 00009B8E | 000000F3 ( 243.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009A9C | 00009B8E | 000000F3 ( 243.) | BYTE 0 | |
| X_0076_HFILL | | 00009B8F | 00009BFF | 00000071 ( 113.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009B8F | 00009BFF | 00000071 ( 113.) | BYTE 0 | |
| X_0076_T | | 00009C00 | 00009CFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009C00 | 00009CFF | 00000100 ( 256.) | BYTE 0 | |
| X_0077_D | | 00009D00 | 00009D01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009D00 | 00009D01 | 00000002 ( 2.) | BYTE 0 | |
| X_0077_ERTEMP | | 00009D02 | 00009D1E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009D02 | 00009D1E | 0000001D ( 29.) | BYTE 0 | |
| X_0077_FCACHE | | 00009D1F | 00009D1F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009D1F | 00009D1F | 00000001 ( 1.) | BYTE 0 | |
| X_0077_GDCS | | 00009D20 | 00009DDB | 000000BC ( 188.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009D20 | 00009DDB | 000000BC ( 188.) | BYTE 0 | |
| X_0077_HFILL | | 00009DDC | 00009DFF | 00000024 ( 36.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009DDC | 00009DFF | 00000024 ( 36.) | BYTE 0 | |
| X_0077_T | | 00009E00 | 00009EFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009E00 | 00009EFF | 00000100 ( 256.) | BYTE 0 | |
| X_0078_D | | 00009F00 | 00009F01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009F00 | 00009F01 | 00000002 ( 2.) | BYTE 0 | |
| X_0078_ERTEMP | | 00009F02 | 00009F12 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009F02 | 00009F12 | 00000011 ( 17.) | BYTE 0 | |
| X_0078_FCACHE | | 00009F13 | 00009F13 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009F13 | 00009F13 | 00000001 ( 1.) | BYTE 0 | |
| X_0078_GDCS | | 00009F14 | 00009FA3 | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009F14 | 00009FA3 | 00000090 ( 144.) | BYTE 0 | |
| X_0078_HFILL | | 00009FA4 | 00009FFF | 0000005C ( 92.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00009FA4 | 00009FFF | 0000005C ( 92.) | BYTE 0 | |
| X_0078_T | | 0000A000 | 0000A0FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A000 | 0000A0FF | 00000100 ( 256.) | BYTE 0 | |
| X_0079_D | | 0000A100 | 0000A101 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A100 | 0000A101 | 00000002 ( 2.) | BYTE 0 | |
| X_0079_ERTEMP | | 0000A102 | 0000A102 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A102 | 0000A102 | 00000001 ( 1.) | BYTE 0 | |
| X_0079_FCACHE | | 0000A103 | 0000A107 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A103 | 0000A107 | 00000005 ( 5.) | BYTE 0 | |
| X_0079_GDCS | | 0000A108 | 0000A108 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A108 | 0000A108 | 00000001 ( 1.) | BYTE 0 | |
| X_0079_HFILL | | 0000A109 | 0000A17F | 00000077 ( 119.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A109 | 0000A17F | 00000077 ( 119.) | BYTE 0 | |
| X_0079_T | | 0000A180 | 0000A37F | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A180 | 0000A37F | 00000200 ( 512.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0080_D | | 0000A380 | 0000A381 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A380 | 0000A381 | 00000002 ( 2.) | BYTE 0 | |
| X_0080_ERTEMP | | 0000A382 | 0000A396 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A382 | 0000A396 | 00000015 ( 21.) | BYTE 0 | |
| X_0080_FCACHE | | 0000A397 | 0000A397 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A397 | 0000A397 | 00000001 ( 1.) | BYTE 0 | |
| X_0080_GDCS | | 0000A398 | 0000A453 | 000000BC ( 188.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A398 | 0000A453 | 000000BC ( 188.) | BYTE 0 | |
| X_0080_HFILL | | 0000A454 | 0000A47F | 0000002C ( 44.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A454 | 0000A47F | 0000002C ( 44.) | BYTE 0 | |
| X_0080_T | | 0000A480 | 0000A5FF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A480 | 0000A5FF | 00000180 ( 384.) | BYTE 0 | |
| X_0081_D | | 0000A600 | 0000A601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A600 | 0000A601 | 00000002 ( 2.) | BYTE 0 | |
| X_0081_T | | 0000A602 | 0000AA7F | 0000047E ( 1150.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000A602 | 0000AA7F | 0000047E ( 1150.) | BYTE 0 | |
| X_0082_D | | 0000AA80 | 0000AA81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000AA80 | 0000AA81 | 00000002 ( 2.) | BYTE 0 | |
| X_0082_T | | 0000AA82 | 0000AEFF | 0000047E ( 1150.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000AA82 | 0000AEFF | 0000047E ( 1150.) | BYTE 0 | |
| X_0083_D | | 0000AF00 | 0000AF01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000AF00 | 0000AF01 | 00000002 ( 2.) | BYTE 0 | |
| X_0083_ERTEMP | | 0000AF02 | 0000AF22 | 00000021 ( 33.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000AF02 | 0000AF22 | 00000021 ( 33.) | BYTE 0 | |
| X_0083_FCACHE | | 0000AF23 | 0000AF23 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000AF23 | 0000AF23 | 00000001 ( 1.) | BYTE 0 | |
| X_0083_GDCS | | 0000AF24 | 0000B00B | 000000E8 ( 232.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000AF24 | 0000B00B | 000000E8 ( 232.) | BYTE 0 | |
| X_0083_HFILL | | 0000B00C | 0000B07F | 00000074 ( 116.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B00C | 0000B07F | 00000074 ( 116.) | BYTE 0 | |
| X_0083_T | | 0000B080 | 0000B2FF | 00000280 ( 640.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B080 | 0000B2FF | 00000280 ( 640.) | BYTE 0 | |
| X_0084_D | | 0000B300 | 0000B301 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B300 | 0000B301 | 00000002 ( 2.) | BYTE 0 | |
| X_0084_ERTEMP | | 0000B302 | 0000B31E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B302 | 0000B31E | 0000001D ( 29.) | BYTE 0 | |
| X_0084_FCACHE | | 0000B31F | 0000B31F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B31F | 0000B31F | 00000001 ( 1.) | BYTE 0 | |
| X_0084_GDCS | | 0000B320 | 0000B407 | 000000E8 ( 232.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B320 | 0000B407 | 000000E8 ( 232.) | BYTE 0 | |
| X_0084_HFILL | | 0000B408 | 0000B47F | 00000078 ( 120.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B408 | 0000B47F | 00000078 ( 120.) | BYTE 0 | |
| X_0084_T | | 0000B480 | 0000B57F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B480 | 0000B57F | 00000100 ( 256.) | BYTE 0 | |
| X_0085_D | | 0000B580 | 0000B581 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B580 | 0000B581 | 00000002 ( 2.) | BYTE 0 | |
| X_0085_ERTEMP | | 0000B582 | 0000B5AA | 00000029 ( 41.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B582 | 0000B5AA | 00000029 ( 41.) | BYTE 0 | |
| X_0085_FCACHE | | 0000B5AB | 0000B5AB | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B5AB | 0000B5AB | 00000001 ( 1.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0085_GDCS | | 0000B5AC | 0000B6A9 | 000000FE ( 254.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B5AC | 0000B6A9 | 000000FE ( 254.) | BYTE 0 | |
| X_0085_HFILL | | 0000B6AA | 0000B6FF | 00000056 ( 86.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B6AA | 0000B6FF | 00000056 ( 86.) | BYTE 0 | |
| X_0085_T | | 0000B700 | 0000B7FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B700 | 0000B7FF | 00000100 ( 256.) | BYTE 0 | |
| X_0086_D | | 0000B800 | 0000B801 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B800 | 0000B801 | 00000002 ( 2.) | BYTE 0 | |
| X_0086_ERTEMP | | 0000B802 | 0000B81A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B802 | 0000B81A | 00000019 ( 25.) | BYTE 0 | |
| X_0086_FCACHE | | 0000B81B | 0000B81B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B81B | 0000B81B | 00000001 ( 1.) | BYTE 0 | |
| X_0086_GDCS | | 0000B81C | 0000B903 | 000000E8 ( 232.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B81C | 0000B903 | 000000E8 ( 232.) | BYTE 0 | |
| X_0086_HFILL | | 0000B904 | 0000B97F | 0000007C ( 124.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B904 | 0000B97F | 0000007C ( 124.) | BYTE 0 | |
| X_0086_T | | 0000B980 | 0000BAFF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000B980 | 0000BAFF | 00000180 ( 384.) | BYTE 0 | |
| X_0087_D | | 0000BB00 | 0000BB01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BB00 | 0000BB01 | 00000002 ( 2.) | BYTE 0 | |
| X_0087_ERTEMP | | 0000BB02 | 0000BB22 | 00000021 ( 33.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BB02 | 0000BB22 | 00000021 ( 33.) | BYTE 0 | |
| X_0087_FCACHE | | 0000BB23 | 0000BB23 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BB23 | 0000BB23 | 00000001 ( 1.) | BYTE 0 | |
| X_0087_GDCS | | 0000BB24 | 0000BC42 | 0000011F ( 287.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BB24 | 0000BC42 | 0000011F ( 287.) | BYTE 0 | |
| X_0087_HFILL | | 0000BC43 | 0000BC7F | 0000003D ( 61.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BC43 | 0000BC7F | 0000003D ( 61.) | BYTE 0 | |
| X_0087_T | | 0000BC80 | 0000BDFF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BC80 | 0000BDFF | 00000180 ( 384.) | BYTE 0 | |
| X_0088_D | | 0000BE00 | 0000BE01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BE00 | 0000BE01 | 00000002 ( 2.) | BYTE 0 | |
| X_0088_ERTEMP | | 0000BE02 | 0000BE12 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BE02 | 0000BE12 | 00000011 ( 17.) | BYTE 0 | |
| X_0088_FCACHE | | 0000BE13 | 0000BE1B | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BE13 | 0000BE1B | 00000009 ( 9.) | BYTE 0 | |
| X_0088_GDCS | | 0000BE1C | 0000BEAB | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BE1C | 0000BEAB | 00000090 ( 144.) | BYTE 0 | |
| X_0088_HFILL | | 0000BEAC | 0000BEFF | 00000054 ( 84.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BEAC | 0000BEFF | 00000054 ( 84.) | BYTE 0 | |
| X_0088_T | | 0000BF00 | 0000BFFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000BF00 | 0000BFFF | 00000100 ( 256.) | BYTE 0 | |
| X_0089_D | | 0000C000 | 0000C001 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C000 | 0000C001 | 00000002 ( 2.) | BYTE 0 | |
| X_0089_ERTEMP | | 0000C002 | 0000C012 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C002 | 0000C012 | 00000011 ( 17.) | BYTE 0 | |
| X_0089_FCACHE | | 0000C013 | 0000C027 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C013 | 0000C027 | 00000015 ( 21.) | BYTE 0 | |
| X_0089_GDCS | | 0000C028 | 0000C0EE | 000000C7 ( 199.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C028 | 0000C0EE | 000000C7 ( 199.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0089_HFILL | ECKAC | 0000C0EF | 0000C0FF | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0089_T | ECKAC | 0000C100 | 0000C1FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0090_D | ECKAC | 0000C200 | 0000C201 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0090_ERTEMP | ECKAC | 0000C202 | 0000C20E | 0000000D ( 13.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0090_FCACHE | ECKAC | 0000C20F | 0000C217 | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0090_GDCS | ECKAC | 0000C218 | 0000C270 | 00000059 ( 89.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0090_HFILL | ECKAC | 0000C271 | 0000C27F | 0000000F ( 15.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0090_T | ECKAC | 0000C280 | 0000C37F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0091_D | ECKAC | 0000C380 | 0000C381 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0091_ERTEMP | ECKAC | 0000C382 | 0000C396 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0091_FCACHE | ECKAC | 0000C397 | 0000C397 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0091_GDCS | ECKAC | 0000C398 | 0000C43D | 000000A6 ( 166.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0091_HFILL | ECKAC | 0000C43E | 0000C47F | 00000042 ( 66.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0091_T | ECKAC | 0000C480 | 0000C57F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0092_D | ECKAC | 0000C580 | 0000C581 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0092_ERTEMP | ECKAC | 0000C582 | 0000C58E | 0000000D ( 13.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0092_FCACHE | ECKAC | 0000C58F | 0000C597 | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0092_GDCS | ECKAC | 0000C598 | 0000C5F0 | 00000059 ( 89.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0092_HFILL | ECKAC | 0000C5F1 | 0000C5FF | 0000000F ( 15.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0092_T | ECKAC | 0000C600 | 0000C6FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0093_D | ECKAC | 0000C700 | 0000C701 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0093_ERTEMP | ECKAC | 0000C702 | 0000C716 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0093_FCACHE | ECKAC | 0000C717 | 0000C717 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0093_GDCS | ECKAC | 0000C718 | 0000C7BD | 000000A6 ( 166.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| X_0093_HFILL | ECKAC | 0000C7BE | 0000C7FF | 00000042 ( 66.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0093_T | | 0000C800 | 0000C8FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C800 | 0000C8FF | 00000100 ( 256.) | BYTE 0 | |
| X_0094_D | | 0000C900 | 0000C901 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C900 | 0000C901 | 00000002 ( 2.) | BYTE 0 | |
| X_0094_ERTEMP | | 0000C902 | 0000C912 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C902 | 0000C912 | 00000011 ( 17.) | BYTE 0 | |
| X_0094_FCACHE | | 0000C913 | 0000C91B | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C913 | 0000C91B | 00000009 ( 9.) | BYTE 0 | |
| X_0094_GDCS | | 0000C91C | 0000C9AB | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C91C | 0000C9AB | 00000090 ( 144.) | BYTE 0 | |
| X_0094_HFILL | | 0000C9AC | 0000C9FF | 00000054 ( 84.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000C9AC | 0000C9FF | 00000054 ( 84.) | BYTE 0 | |
| X_0094_T | | 0000CA00 | 0000CA7F | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CA00 | 0000CA7F | 00000080 ( 128.) | BYTE 0 | |
| X_0095_D | | 0000CA80 | 0000CA81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CA80 | 0000CA81 | 00000002 ( 2.) | BYTE 0 | |
| X_0095_ERTEMP | | 0000CA82 | 0000CA8A | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CA82 | 0000CA8A | 00000009 ( 9.) | BYTE 0 | |
| X_0095_FCACHE | | 0000CA8B | 0000CA8B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CA8B | 0000CA8B | 00000001 ( 1.) | BYTE 0 | |
| X_0095_GDCS | | 0000CA8C | 0000CB5D | 000000D2 ( 210.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CA8C | 0000CB5D | 000000D2 ( 210.) | BYTE 0 | |
| X_0095_HFILL | | 0000CB5E | 0000CB7F | 00000022 ( 34.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CB5E | 0000CB7F | 00000022 ( 34.) | BYTE 0 | |
| X_0095_T | | 0000CB80 | 0000CC7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CB80 | 0000CC7F | 00000100 ( 256.) | BYTE 0 | |
| X_0096_D | | 0000CC80 | 0000CC81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CC80 | 0000CC81 | 00000002 ( 2.) | BYTE 0 | |
| X_0096_ERTEMP | | 0000CC82 | 0000CC96 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CC82 | 0000CC96 | 00000015 ( 21.) | BYTE 0 | |
| X_0096_FCACHE | | 0000CC97 | 0000CC97 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CC97 | 0000CC97 | 00000001 ( 1.) | BYTE 0 | |
| X_0096_GDCS | | 0000CC98 | 0000CDA0 | 00000109 ( 265.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CC98 | 0000CDA0 | 00000109 ( 265.) | BYTE 0 | |
| X_0096_HFILL | | 0000CDA1 | 0000CDFF | 0000005F ( 95.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CDA1 | 0000CDFF | 0000005F ( 95.) | BYTE 0 | |
| X_0096_T | | 0000CE00 | 0000CEFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CE00 | 0000CEFF | 00000100 ( 256.) | BYTE 0 | |
| X_0097_D | | 0000CF00 | 0000CF01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CF00 | 0000CF01 | 00000002 ( 2.) | BYTE 0 | |
| X_0097_ERTEMP | | 0000CF02 | 0000CF12 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CF02 | 0000CF12 | 00000011 ( 17.) | BYTE 0 | |
| X_0097_FCACHE | | 0000CF13 | 0000CF13 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CF13 | 0000CF13 | 00000001 ( 1.) | BYTE 0 | |
| X_0097_GDCS | | 0000CF14 | 0000CFA3 | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CF14 | 0000CFA3 | 00000090 ( 144.) | BYTE 0 | |
| X_0097_HFILL | | 0000CFA4 | 0000CFFF | 0000005C ( 92.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000CFA4 | 0000CFFF | 0000005C ( 92.) | BYTE 0 | |
| X_0097_T | | 0000D000 | 0000D1FF | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D000 | 0000D1FF | 00000200 ( 512.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0098_D | | 0000D200 | 0000D201 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D200 | 0000D201 | 00000002 ( 2.) | BYTE 0 | |
| X_0098_T | | 0000D202 | 0000D37F | 0000017E ( 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D202 | 0000D37F | 0000017E ( 382.) | BYTE 0 | |
| X_0099_D | | 0000D380 | 0000D381 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D380 | 0000D381 | 00000002 ( 2.) | BYTE 0 | |
| X_0099_ERTEMP | | 0000D382 | 0000D38A | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D382 | 0000D38A | 00000009 ( 9.) | BYTE 0 | |
| X_0099_FCACHE | | 0000D38B | 0000D38B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D38B | 0000D38B | 00000001 ( 1.) | BYTE 0 | |
| X_0099_GDCS | | 0000D38C | 0000D38C | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D38C | 0000D38C | 00000001 ( 1.) | BYTE 0 | |
| X_0099_HFILL | | 0000D38D | 0000D3FF | 00000073 ( 115.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D38D | 0000D3FF | 00000073 ( 115.) | BYTE 0 | |
| X_0099_T | | 0000D400 | 0000D5FF | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D400 | 0000D5FF | 00000200 ( 512.) | BYTE 0 | |
| X_0100_D | | 0000D600 | 0000D601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D600 | 0000D601 | 00000002 ( 2.) | BYTE 0 | |
| X_0100_ERTEMP | | 0000D602 | 0000D606 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D602 | 0000D606 | 00000005 ( 5.) | BYTE 0 | |
| X_0100_FCACHE | | 0000D607 | 0000D607 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D607 | 0000D607 | 00000001 ( 1.) | BYTE 0 | |
| X_0100_GDCS | | 0000D608 | 0000D608 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D608 | 0000D608 | 00000001 ( 1.) | BYTE 0 | |
| X_0100_HFILL | | 0000D609 | 0000D67F | 00000077 ( 119.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D609 | 0000D67F | 00000077 ( 119.) | BYTE 0 | |
| X_0100_T | | 0000D680 | 0000D97F | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D680 | 0000D97F | 00000300 ( 768.) | BYTE 0 | |
| X_0101_D | | 0000D980 | 0000D981 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D980 | 0000D981 | 00000002 ( 2.) | BYTE 0 | |
| X_0101_T | | 0000D982 | 0000DB7F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000D982 | 0000DB7F | 000001FE ( 510.) | BYTE 0 | |
| X_0102_D | | 0000DB80 | 0000DB81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DB80 | 0000DB81 | 00000002 ( 2.) | BYTE 0 | |
| X_0102_T | | 0000DB82 | 0000DD7F | 000001FE ( 510.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DB82 | 0000DD7F | 000001FE ( 510.) | BYTE 0 | |
| X_0103_D | | 0000DD80 | 0000DD81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DD80 | 0000DD81 | 00000002 ( 2.) | BYTE 0 | |
| X_0103_ERTEMP | | 0000DD82 | 0000DD96 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DD82 | 0000DD96 | 00000015 ( 21.) | BYTE 0 | |
| X_0103_FCACHE | | 0000DD97 | 0000DD97 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DD97 | 0000DD97 | 00000001 ( 1.) | BYTE 0 | |
| X_0103_GDCS | | 0000DD98 | 0000DDF0 | 00000059 ( 89.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DD98 | 0000DDF0 | 00000059 ( 89.) | BYTE 0 | |
| X_0103_HFILL | | 0000DDF1 | 0000DDFF | 0000000F ( 15.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DDF1 | 0000DDFF | 0000000F ( 15.) | BYTE 0 | |
| X_0103_T | | 0000DE00 | 0000DEFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DE00 | 0000DEFF | 00000100 ( 256.) | BYTE 0 | |
| X_0104_D | | 0000DF00 | 0000DF01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DF00 | 0000DF01 | 00000002 ( 2.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0104_ERTEMP | | 0000DF02 | 0000DF12 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DF02 | 0000DF12 | 00000011 ( 17.) | BYTE 0 | |
| X_0104_FCACHE | | 0000DF13 | 0000DF13 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DF13 | 0000DF13 | 00000001 ( 1.) | BYTE 0 | |
| X_0104_GDCS | | 0000DF14 | 0000DF6C | 00000059 ( 89.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DF14 | 0000DF6C | 00000059 ( 89.) | BYTE 0 | |
| X_0104_HFILL | | 0000DF6D | 0000DF7F | 00000013 ( 19.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DF6D | 0000DF7F | 00000013 ( 19.) | BYTE 0 | |
| X_0104_T | | 0000DF80 | 0000E07F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000DF80 | 0000E07F | 00000100 ( 256.) | BYTE 0 | |
| X_0105_D | | 0000E080 | 0000E081 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E080 | 0000E081 | 00000002 ( 2.) | BYTE 0 | |
| X_0105_ERTEMP | | 0000E082 | 0000E096 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E082 | 0000E096 | 00000015 ( 21.) | BYTE 0 | |
| X_0105_FCACHE | | 0000E097 | 0000E097 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E097 | 0000E097 | 00000001 ( 1.) | BYTE 0 | |
| X_0105_GDCS | | 0000E098 | 0000E0F0 | 00000059 ( 89.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E098 | 0000E0F0 | 00000059 ( 89.) | BYTE 0 | |
| X_0105_HFILL | | 0000E0F1 | 0000E0FF | 0000000F ( 15.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E0F1 | 0000E0FF | 0000000F ( 15.) | BYTE 0 | |
| X_0105_T | | 0000E100 | 0000E1FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E100 | 0000E1FF | 00000100 ( 256.) | BYTE 0 | |
| X_0106_D | | 0000E200 | 0000E201 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E200 | 0000E201 | 00000002 ( 2.) | BYTE 0 | |
| X_0106_ERTEMP | | 0000E202 | 0000E226 | 00000025 ( 37.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E202 | 0000E226 | 00000025 ( 37.) | BYTE 0 | |
| X_0106_FCACHE | | 0000E227 | 0000E227 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E227 | 0000E227 | 00000001 ( 1.) | BYTE 0 | |
| X_0106_GDCS | | 0000E228 | 0000E2AC | 00000085 ( 133.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E228 | 0000E2AC | 00000085 ( 133.) | BYTE 0 | |
| X_0106_HFILL | | 0000E2AD | 0000E2FF | 00000053 ( 83.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E2AD | 0000E2FF | 00000053 ( 83.) | BYTE 0 | |
| X_0106_T | | 0000E300 | 0000E5FF | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E300 | 0000E5FF | 00000300 ( 768.) | BYTE 0 | |
| X_0107_D | | 0000E600 | 0000E601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E600 | 0000E601 | 00000002 ( 2.) | BYTE 0 | |
| X_0107_ERTEMP | | 0000E602 | 0000E632 | 00000031 ( 49.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E602 | 0000E632 | 00000031 ( 49.) | BYTE 0 | |
| X_0107_FCACHE | | 0000E633 | 0000E633 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E633 | 0000E633 | 00000001 ( 1.) | BYTE 0 | |
| X_0107_GDCS | | 0000E634 | 0000E6A2 | 0000006F ( 111.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E634 | 0000E6A2 | 0000006F ( 111.) | BYTE 0 | |
| X_0107_HFILL | | 0000E6A3 | 0000E6FF | 0000005D ( 93.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E6A3 | 0000E6FF | 0000005D ( 93.) | BYTE 0 | |
| X_0107_T | | 0000E700 | 0000EA7F | 00000380 ( 896.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000E700 | 0000EA7F | 00000380 ( 896.) | BYTE 0 | |
| X_0108_D | | 0000EA80 | 0000EA81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EA80 | 0000EA81 | 00000002 ( 2.) | BYTE 0 | |
| X_0108_ERTEMP | | 0000EA82 | 0000EAA6 | 00000025 ( 37.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EA82 | 0000EAA6 | 00000025 ( 37.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0108_FCACHE | | 0000EAA7 | 0000EAA7 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EAA7 | 0000EAA7 | 00000001 ( 1.) | BYTE 0 | |
| X_0108_GDCS | | 0000EAA8 | 0000EB2C | 00000085 ( 133.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EAA8 | 0000EB2C | 00000085 ( 133.) | BYTE 0 | |
| X_0108_HFILL | | 0000EB2D | 0000EB7F | 00000053 ( 83.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EB2D | 0000EB7F | 00000053 ( 83.) | BYTE 0 | |
| X_0108_T | | 0000EB80 | 0000EE7F | 00000300 ( 768.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EB80 | 0000EE7F | 00000300 ( 768.) | BYTE 0 | |
| X_0109_D | | 0000EE80 | 0000EE81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EE80 | 0000EE81 | 00000002 ( 2.) | BYTE 0 | |
| X_0109_ERTEMP | | 0000EE82 | 0000EE9A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EE82 | 0000EE9A | 00000019 ( 25.) | BYTE 0 | |
| X_0109_FCACHE | | 0000EE9B | 0000EE9B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EE9B | 0000EE9B | 00000001 ( 1.) | BYTE 0 | |
| X_0109_GDCS | | 0000EE9C | 0000EF4C | 000000B1 ( 177.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EE9C | 0000EF4C | 000000B1 ( 177.) | BYTE 0 | |
| X_0109_HFILL | | 0000EF4D | 0000EF7F | 00000033 ( 51.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EF4D | 0000EF7F | 00000033 ( 51.) | BYTE 0 | |
| X_0109_T | | 0000EF80 | 0000F07F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000EF80 | 0000F07F | 00000100 ( 256.) | BYTE 0 | |
| X_0110_D | | 0000F080 | 0000F081 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F080 | 0000F081 | 00000002 ( 2.) | BYTE 0 | |
| X_0110_ERTEMP | | 0000F082 | 0000F096 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F082 | 0000F096 | 00000015 ( 21.) | BYTE 0 | |
| X_0110_FCACHE | | 0000F097 | 0000F09F | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F097 | 0000F09F | 00000009 ( 9.) | BYTE 0 | |
| X_0110_GDCS | | 0000F0A0 | 0000F15B | 000000BC ( 188.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F0A0 | 0000F15B | 000000BC ( 188.) | BYTE 0 | |
| X_0110_HFILL | | 0000F15C | 0000F17F | 00000024 ( 36.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F15C | 0000F17F | 00000024 ( 36.) | BYTE 0 | |
| X_0110_T | | 0000F180 | 0000F27F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F180 | 0000F27F | 00000100 ( 256.) | BYTE 0 | |
| X_0111_D | | 0000F280 | 0000F281 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F280 | 0000F281 | 00000002 ( 2.) | BYTE 0 | |
| X_0111_ERTEMP | | 0000F282 | 0000F2A2 | 00000021 ( 33.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F282 | 0000F2A2 | 00000021 ( 33.) | BYTE 0 | |
| X_0111_FCACHE | | 0000F2A3 | 0000F2A7 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F2A3 | 0000F2A7 | 00000005 ( 5.) | BYTE 0 | |
| X_0111_GDCS | | 0000F2A8 | 0000F379 | 000000D2 ( 210.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F2A8 | 0000F379 | 000000D2 ( 210.) | BYTE 0 | |
| X_0111_HFILL | | 0000F37A | 0000F37F | 00000006 ( 6.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F37A | 0000F37F | 00000006 ( 6.) | BYTE 0 | |
| X_0111_T | | 0000F380 | 0000F4FF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F380 | 0000F4FF | 00000180 ( 384.) | BYTE 0 | |
| X_0112_D | | 0000F500 | 0000F501 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F500 | 0000F501 | 00000002 ( 2.) | BYTE 0 | |
| X_0112_ERTEMP | | 0000F502 | 0000F51A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F502 | 0000F51A | 00000019 ( 25.) | BYTE 0 | |
| X_0112_FCACHE | | 0000F51B | 0000F51F | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F51B | 0000F51F | 00000005 ( 5.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0112_GDCS | | 0000F520 | 0000F5F1 | 000000D2 ( 210.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F520 | 0000F5F1 | 000000D2 ( 210.) | BYTE 0 | |
| X_0112_HFILL | | 0000F5F2 | 0000F5FF | 0000000E ( 14.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F5F2 | 0000F5FF | 0000000E ( 14.) | BYTE 0 | |
| X_0112_T | | 0000F600 | 0000F6FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F600 | 0000F6FF | 00000100 ( 256.) | BYTE 0 | |
| X_0113_D | | 0000F700 | 0000F701 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F700 | 0000F701 | 00000002 ( 2.) | BYTE 0 | |
| X_0113_ERTEMP | | 0000F702 | 0000F71E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F702 | 0000F71E | 0000001D ( 29.) | BYTE 0 | |
| X_0113_FCACHE | | 0000F71F | 0000F727 | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F71F | 0000F727 | 00000009 ( 9.) | BYTE 0 | |
| X_0113_GDCS | | 0000F728 | 0000F80F | 000000E8 ( 232.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F728 | 0000F80F | 000000E8 ( 232.) | BYTE 0 | |
| X_0113_HFILL | | 0000F810 | 0000F87F | 00000070 ( 112.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F810 | 0000F87F | 00000070 ( 112.) | BYTE 0 | |
| X_0113_T | | 0000F880 | 0000F8FF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F880 | 0000F8FF | 00000080 ( 128.) | BYTE 0 | |
| X_0114_D | | 0000F900 | 0000F901 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F900 | 0000F901 | 00000002 ( 2.) | BYTE 0 | |
| X_0114_ERTEMP | | 0000F902 | 0000F91A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F902 | 0000F91A | 00000019 ( 25.) | BYTE 0 | |
| X_0114_FCACHE | | 0000F91B | 0000F923 | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F91B | 0000F923 | 00000009 ( 9.) | BYTE 0 | |
| X_0114_GDCS | | 0000F924 | 0000F9DF | 000000BC ( 188.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F924 | 0000F9DF | 000000BC ( 188.) | BYTE 0 | |
| X_0114_HFILL | | 0000F9E0 | 0000F9FF | 00000020 ( 32.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000F9E0 | 0000F9FF | 00000020 ( 32.) | BYTE 0 | |
| X_0114_T | | 0000FA00 | 0000FAFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FA00 | 0000FAFF | 00000100 ( 256.) | BYTE 0 | |
| X_0115_D | | 0000FB00 | 0000FB01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FB00 | 0000FB01 | 00000002 ( 2.) | BYTE 0 | |
| X_0115_ERTEMP | | 0000FB02 | 0000FB02 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FB02 | 0000FB02 | 00000001 ( 1.) | BYTE 0 | |
| X_0115_FCACHE | | 0000FB03 | 0000FB07 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FB03 | 0000FB07 | 00000005 ( 5.) | BYTE 0 | |
| X_0115_GDCS | | 0000FB08 | 0000FB55 | 0000004E ( 78.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FB08 | 0000FB55 | 0000004E ( 78.) | BYTE 0 | |
| X_0115_HFILL | | 0000FB56 | 0000FB7F | 0000002A ( 42.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FB56 | 0000FB7F | 0000002A ( 42.) | BYTE 0 | |
| X_0115_T | | 0000FB80 | 0000FBFF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FB80 | 0000FBFF | 00000080 ( 128.) | BYTE 0 | |
| X_0116_D | | 0000FC00 | 0000FC01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FC00 | 0000FC01 | 00000002 ( 2.) | BYTE 0 | |
| X_0116_T | | 0000FC02 | 0000FD7F | 0000017E ( 382.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FC02 | 0000FD7F | 0000017E ( 382.) | BYTE 0 | |
| X_0117_D | | 0000FD80 | 0000FD81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FD80 | 0000FD81 | 00000002 ( 2.) | BYTE 0 | |
| X_0117_ERTEMP | | 0000FD82 | 0000FD82 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FD82 | 0000FD82 | 00000001 ( 1.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0117_FCACHE | | 0000FD83 | 0000FD83 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FD83 | 0000FD83 | 00000001 ( 1.) | BYTE 0 | |
| X_0117_GDCS | | 0000FD84 | 0000FD9A | 00000017 ( 23.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FD84 | 0000FD9A | 00000017 ( 23.) | BYTE 0 | |
| X_0117_HFILL | | 0000FD9B | 0000FDFF | 00000065 ( 101.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FD9B | 0000FDFF | 00000065 ( 101.) | BYTE 0 | |
| X_0117_T | | 0000FE00 | 0000FEFF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FE00 | 0000FEFF | 00000100 ( 256.) | BYTE 0 | |
| X_0118_D | | 0000FF00 | 0000FF01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FF00 | 0000FF01 | 00000002 ( 2.) | BYTE 0 | |
| X_0118_ERTEMP | | 0000FF02 | 0000FF12 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FF02 | 0000FF12 | 00000011 ( 17.) | BYTE 0 | |
| X_0118_FCACHE | | 0000FF13 | 0000FF13 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FF13 | 0000FF13 | 00000001 ( 1.) | BYTE 0 | |
| X_0118_GDCS | | 0000FF14 | 0000FF77 | 00000064 ( 100.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FF14 | 0000FF77 | 00000064 ( 100.) | BYTE 0 | |
| X_0118_HFILL | | 0000FF78 | 0000FF7F | 00000008 ( 8.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FF78 | 0000FF7F | 00000008 ( 8.) | BYTE 0 | |
| X_0118_T | | 0000FF80 | 0000FFFF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0000FF80 | 0000FFFF | 00000080 ( 128.) | BYTE 0 | |
| X_0119_D | | 00010000 | 00010001 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010000 | 00010001 | 00000002 ( 2.) | BYTE 0 | |
| X_0119_ERTEMP | | 00010002 | 0001000A | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010002 | 0001000A | 00000009 ( 9.) | BYTE 0 | |
| X_0119_FCACHE | | 0001000B | 0001000B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001000B | 0001000B | 00000001 ( 1.) | BYTE 0 | |
| X_0119_GDCS | | 0001000C | 0001000C | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001000C | 0001000C | 00000001 ( 1.) | BYTE 0 | |
| X_0119_HFILL | | 0001000D | 0001007F | 00000073 ( 115.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001000D | 0001007F | 00000073 ( 115.) | BYTE 0 | |
| X_0119_T | | 00010080 | 0001027F | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010080 | 0001027F | 00000200 ( 512.) | BYTE 0 | |
| X_0120_D | | 00010280 | 00010281 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010280 | 00010281 | 00000002 ( 2.) | BYTE 0 | |
| X_0120_ERTEMP | | 00010282 | 00010282 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010282 | 00010282 | 00000001 ( 1.) | BYTE 0 | |
| X_0120_FCACHE | | 00010283 | 00010283 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010283 | 00010283 | 00000001 ( 1.) | BYTE 0 | |
| X_0120_GDCS | | 00010284 | 0001029A | 00000017 ( 23.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010284 | 0001029A | 00000017 ( 23.) | BYTE 0 | |
| X_0120_HFILL | | 0001029B | 000102FF | 00000065 ( 101.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001029B | 000102FF | 00000065 ( 101.) | BYTE 0 | |
| X_0120_T | | 00010300 | 000103FF | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010300 | 000103FF | 00000100 ( 256.) | BYTE 0 | |
| X_0121_D | | 00010400 | 00010401 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010400 | 00010401 | 00000002 ( 2.) | BYTE 0 | |
| X_0121_ERTEMP | | 00010402 | 00010416 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010402 | 00010416 | 00000015 ( 21.) | BYTE 0 | |
| X_0121_FCACHE | | 00010417 | 00010417 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010417 | 00010417 | 00000001 ( 1.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0121_GDCS | | 00010418 | 0001044F | 00000038 ( 56.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010418 | 0001044F | 00000038 ( 56.) | BYTE 0 | |
| X_0121_HFILL | | 00010450 | 0001047F | 00000030 ( 48.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010450 | 0001047F | 00000030 ( 48.) | BYTE 0 | |
| X_0121_T | | 00010480 | 000105FF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010480 | 000105FF | 00000180 ( 384.) | BYTE 0 | |
| X_0122_D | | 00010600 | 00010601 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010600 | 00010601 | 00000002 ( 2.) | BYTE 0 | |
| X_0122_ERTEMP | | 00010602 | 00010622 | 00000021 ( 33.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010602 | 00010622 | 00000021 ( 33.) | BYTE 0 | |
| X_0122_FCACHE | | 00010623 | 0001062B | 00000009 ( 9.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010623 | 0001062B | 00000009 ( 9.) | BYTE 0 | |
| X_0122_GDCS | | 0001062C | 00010760 | 00000135 ( 309.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001062C | 00010760 | 00000135 ( 309.) | BYTE 0 | |
| X_0122_HFILL | | 00010761 | 0001077F | 0000001F ( 31.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010761 | 0001077F | 0000001F ( 31.) | BYTE 0 | |
| X_0122_T | | 00010780 | 0001087F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010780 | 0001087F | 00000100 ( 256.) | BYTE 0 | |
| X_0123_D | | 00010880 | 00010881 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010880 | 00010881 | 00000002 ( 2.) | BYTE 0 | |
| X_0123_ERTEMP | | 00010882 | 00010896 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010882 | 00010896 | 00000015 ( 21.) | BYTE 0 | |
| X_0123_FCACHE | | 00010897 | 00010897 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010897 | 00010897 | 00000001 ( 1.) | BYTE 0 | |
| X_0123_GDCS | | 00010898 | 00010906 | 0000006F ( 111.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010898 | 00010906 | 0000006F ( 111.) | BYTE 0 | |
| X_0123_HFILL | | 00010907 | 0001097F | 00000079 ( 121.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010907 | 0001097F | 00000079 ( 121.) | BYTE 0 | |
| X_0123_T | | 00010980 | 00010AFF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010980 | 00010AFF | 00000180 ( 384.) | BYTE 0 | |
| X_0124_D | | 00010B00 | 00010B01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010B00 | 00010B01 | 00000002 ( 2.) | BYTE 0 | |
| X_0124_ERTEMP | | 00010B02 | 00010B12 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010B02 | 00010B12 | 00000011 ( 17.) | BYTE 0 | |
| X_0124_FCACHE | | 00010B13 | 00010B13 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010B13 | 00010B13 | 00000001 ( 1.) | BYTE 0 | |
| X_0124_GDCS | | 00010B14 | 00010B77 | 00000064 ( 100.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010B14 | 00010B77 | 00000064 ( 100.) | BYTE 0 | |
| X_0124_HFILL | | 00010B78 | 00010B7F | 00000008 ( 8.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010B78 | 00010B7F | 00000008 ( 8.) | BYTE 0 | |
| X_0124_T | | 00010B80 | 0001107F | 00000500 ( 1280.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00010B80 | 0001107F | 00000500 ( 1280.) | BYTE 0 | |
| X_0125_D | | 00011080 | 00011081 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011080 | 00011081 | 00000002 ( 2.) | BYTE 0 | |
| X_0125_ERTEMP | | 00011082 | 00011092 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011082 | 00011092 | 00000011 ( 17.) | BYTE 0 | |
| X_0125_FCACHE | | 00011093 | 00011093 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011093 | 00011093 | 00000001 ( 1.) | BYTE 0 | |
| X_0125_GDCS | | 00011094 | 000110F7 | 00000064 ( 100.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011094 | 000110F7 | 00000064 ( 100.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0125_HFILL | | 000110F8 | 000110FF | 00000008 ( 8.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000110F8 | 000110FF | 00000008 ( 8.) | BYTE 0 | |
| X_0125_T | | 00011100 | 0001157F | 00000480 ( 1152.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011100 | 0001157F | 00000480 ( 1152.) | BYTE 0 | |
| X_0126_D | | 00011580 | 00011581 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011580 | 00011581 | 00000002 ( 2.) | BYTE 0 | |
| X_0126_ERTEMP | | 00011582 | 00011592 | 00000011 ( 17.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011582 | 00011592 | 00000011 ( 17.) | BYTE 0 | |
| X_0126_FCACHE | | 00011593 | 00011593 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011593 | 00011593 | 00000001 ( 1.) | BYTE 0 | |
| X_0126_GDCS | | 00011594 | 00011594 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011594 | 00011594 | 00000001 ( 1.) | BYTE 0 | |
| X_0126_HFILL | | 00011595 | 000115FF | 0000006B ( 107.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011595 | 000115FF | 0000006B ( 107.) | BYTE 0 | |
| X_0126_T | | 00011600 | 000117FF | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011600 | 000117FF | 00000200 ( 512.) | BYTE 0 | |
| X_0127_D | | 00011800 | 00011801 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011800 | 00011801 | 00000002 ( 2.) | BYTE 0 | |
| X_0127_ERTEMP | | 00011802 | 0001181A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011802 | 0001181A | 00000019 ( 25.) | BYTE 0 | |
| X_0127_FCACHE | | 0001181B | 0001181B | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001181B | 0001181B | 00000001 ( 1.) | BYTE 0 | |
| X_0127_GDCS | | 0001181C | 0001181C | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001181C | 0001181C | 00000001 ( 1.) | BYTE 0 | |
| X_0127_HFILL | | 0001181D | 0001187F | 00000063 ( 99.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001181D | 0001187F | 00000063 ( 99.) | BYTE 0 | |
| X_0127_T | | 00011880 | 00011AFF | 00000280 ( 640.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011880 | 00011AFF | 00000280 ( 640.) | BYTE 0 | |
| X_0128_D | | 00011B00 | 00011B01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011B00 | 00011B01 | 00000002 ( 2.) | BYTE 0 | |
| X_0128_ERTEMP | | 00011B02 | 00011B2E | 0000002D ( 45.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011B02 | 00011B2E | 0000002D ( 45.) | BYTE 0 | |
| X_0128_FCACHE | | 00011B2F | 00011B2F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011B2F | 00011B2F | 00000001 ( 1.) | BYTE 0 | |
| X_0128_GDCS | | 00011B30 | 00011CD2 | 000001A3 ( 419.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011B30 | 00011CD2 | 000001A3 ( 419.) | BYTE 0 | |
| X_0128_HFILL | | 00011CD3 | 00011CFF | 0000002D ( 45.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011CD3 | 00011CFF | 0000002D ( 45.) | BYTE 0 | |
| X_0128_T | | 00011D00 | 00011E7F | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011D00 | 00011E7F | 00000180 ( 384.) | BYTE 0 | |
| X_0129_D | | 00011E80 | 00011E81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011E80 | 00011E81 | 00000002 ( 2.) | BYTE 0 | |
| X_0129_ERTEMP | | 00011E82 | 00011EA6 | 00000025 ( 37.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011E82 | 00011EA6 | 00000025 ( 37.) | BYTE 0 | |
| X_0129_FCACHE | | 00011EA7 | 00011EA7 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011EA7 | 00011EA7 | 00000001 ( 1.) | BYTE 0 | |
| X_0129_GDCS | | 00011EA8 | 00012013 | 0000016C ( 364.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00011EA8 | 00012013 | 0000016C ( 364.) | BYTE 0 | |
| X_0129_HFILL | | 00012014 | 0001207F | 0000006C ( 108.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012014 | 0001207F | 0000006C ( 108.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0129_T | | 00012080 | 0001217F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012080 | 0001217F | 00000100 ( 256.) | BYTE 0 | |
| X_0130_D | | 00012180 | 00012181 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012180 | 00012181 | 00000002 ( 2.) | BYTE 0 | |
| X_0130_ERTEMP | | 00012182 | 000121BA | 00000039 ( 57.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012182 | 000121BA | 00000039 ( 57.) | BYTE 0 | |
| X_0130_FCACHE | | 000121BB | 000121BB | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000121BB | 000121BB | 00000001 ( 1.) | BYTE 0 | |
| X_0130_GDCS | | 000121BC | 000121BC | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000121BC | 000121BC | 00000001 ( 1.) | BYTE 0 | |
| X_0130_HFILL | | 000121BD | 000121FF | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000121BD | 000121FF | 00000043 ( 67.) | BYTE 0 | |
| X_0130_T | | 00012200 | 0001287F | 00000680 ( 1664.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012200 | 0001287F | 00000680 ( 1664.) | BYTE 0 | |
| X_0131_D | | 00012880 | 00012881 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012880 | 00012881 | 00000002 ( 2.) | BYTE 0 | |
| X_0131_ERTEMP | | 00012882 | 000128BA | 00000039 ( 57.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012882 | 000128BA | 00000039 ( 57.) | BYTE 0 | |
| X_0131_FCACHE | | 000128BB | 000128BB | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000128BB | 000128BB | 00000001 ( 1.) | BYTE 0 | |
| X_0131_GDCS | | 000128BC | 000128BC | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000128BC | 000128BC | 00000001 ( 1.) | BYTE 0 | |
| X_0131_HFILL | | 000128BD | 000128FF | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000128BD | 000128FF | 00000043 ( 67.) | BYTE 0 | |
| X_0131_T | | 00012900 | 00012F7F | 00000680 ( 1664.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012900 | 00012F7F | 00000680 ( 1664.) | BYTE 0 | |
| X_0132_D | | 00012F80 | 00012F81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012F80 | 00012F81 | 00000002 ( 2.) | BYTE 0 | |
| X_0132_ERTEMP | | 00012F82 | 00012FBA | 00000039 ( 57.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012F82 | 00012FBA | 00000039 ( 57.) | BYTE 0 | |
| X_0132_FCACHE | | 00012FBB | 00012FBB | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012FBB | 00012FBB | 00000001 ( 1.) | BYTE 0 | |
| X_0132_GDCS | | 00012FBC | 00012FBC | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012FBC | 00012FBC | 00000001 ( 1.) | BYTE 0 | |
| X_0132_HFILL | | 00012FBD | 00012FFF | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00012FBD | 00012FFF | 00000043 ( 67.) | BYTE 0 | |
| X_0132_T | | 00013000 | 0001367F | 00000680 ( 1664.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013000 | 0001367F | 00000680 ( 1664.) | BYTE 0 | |
| X_0133_D | | 00013680 | 00013681 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013680 | 00013681 | 00000002 ( 2.) | BYTE 0 | |
| X_0133_ERTEMP | | 00013682 | 000136BA | 00000039 ( 57.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013682 | 000136BA | 00000039 ( 57.) | BYTE 0 | |
| X_0133_FCACHE | | 000136BB | 000136BB | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000136BB | 000136BB | 00000001 ( 1.) | BYTE 0 | |
| X_0133_GDCS | | 000136BC | 000136BC | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000136BC | 000136BC | 00000001 ( 1.) | BYTE 0 | |
| X_0133_HFILL | | 000136BD | 000136FF | 00000043 ( 67.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000136BD | 000136FF | 00000043 ( 67.) | BYTE 0 | |
| X_0133_T | | 00013700 | 00013D7F | 00000680 ( 1664.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013700 | 00013D7F | 00000680 ( 1664.) | BYTE 0 | |

F 3

ZZ-ECKAC-8.0 Map Fiche 1 Frame F3 Sequence 31
DISK$UCODPACK00:[VAX750.MIC]ECKAC.EXE;4 3-MAR-1986 10:59 VAX-11 Linker V04-00 Page 30

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0134_D | | 00013D80 | 00013D81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013D80 | 00013D81 | 00000002 ( 2.) | BYTE 0 | |
| X_0134_ERTEMP | | 00013D82 | 00013DB6 | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013D82 | 00013DB6 | 00000035 ( 53.) | BYTE 0 | |
| X_0134_FCACHE | | 00013DB7 | 00013DB7 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013DB7 | 00013DB7 | 00000001 ( 1.) | BYTE 0 | |
| X_0134_GDCS | | 00013DB8 | 00013DB8 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013DB8 | 00013DB8 | 00000001 ( 1.) | BYTE 0 | |
| X_0134_HFILL | | 00013DB9 | 00013DFF | 00000047 ( 71.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013DB9 | 00013DFF | 00000047 ( 71.) | BYTE 0 | |
| X_0134_T | | 00013E00 | 000144FF | 00000700 ( 1792.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00013E00 | 000144FF | 00000700 ( 1792.) | BYTE 0 | |
| X_0135_D | | 00014500 | 00014501 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014500 | 00014501 | 00000002 ( 2.) | BYTE 0 | |
| X_0135_ERTEMP | | 00014502 | 00014536 | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014502 | 00014536 | 00000035 ( 53.) | BYTE 0 | |
| X_0135_FCACHE | | 00014537 | 00014537 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014537 | 00014537 | 00000001 ( 1.) | BYTE 0 | |
| X_0135_GDCS | | 00014538 | 00014538 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014538 | 00014538 | 00000001 ( 1.) | BYTE 0 | |
| X_0135_HFILL | | 00014539 | 0001457F | 00000047 ( 71.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014539 | 0001457F | 00000047 ( 71.) | BYTE 0 | |
| X_0135_T | | 00014580 | 00014BFF | 00000680 ( 1664.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014580 | 00014BFF | 00000680 ( 1664.) | BYTE 0 | |
| X_0136_D | | 00014C00 | 00014C01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014C00 | 00014C01 | 00000002 ( 2.) | BYTE 0 | |
| X_0136_ERTEMP | | 00014C02 | 00014C36 | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014C02 | 00014C36 | 00000035 ( 53.) | BYTE 0 | |
| X_0136_FCACHE | | 00014C37 | 00014C37 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014C37 | 00014C37 | 00000001 ( 1.) | BYTE 0 | |
| X_0136_GDCS | | 00014C38 | 00014C38 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014C38 | 00014C38 | 00000001 ( 1.) | BYTE 0 | |
| X_0136_HFILL | | 00014C39 | 00014C7F | 00000047 ( 71.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014C39 | 00014C7F | 00000047 ( 71.) | BYTE 0 | |
| X_0136_T | | 00014C80 | 0001537F | 00000700 ( 1792.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00014C80 | 0001537F | 00000700 ( 1792.) | BYTE 0 | |
| X_0137_D | | 00015380 | 00015381 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015380 | 00015381 | 00000002 ( 2.) | BYTE 0 | |
| X_0137_ERTEMP | | 00015382 | 000153B6 | 00000035 ( 53.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015382 | 000153B6 | 00000035 ( 53.) | BYTE 0 | |
| X_0137_FCACHE | | 000153B7 | 000153B7 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000153B7 | 000153B7 | 00000001 ( 1.) | BYTE 0 | |
| X_0137_GDCS | | 000153B8 | 000153B8 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000153B8 | 000153B8 | 00000001 ( 1.) | BYTE 0 | |
| X_0137_HFILL | | 000153B9 | 000153FF | 00000047 ( 71.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000153B9 | 000153FF | 00000047 ( 71.) | BYTE 0 | |
| X_0137_T | | 00015400 | 00015AFF | 00000700 ( 1792.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015400 | 00015AFF | 00000700 ( 1792.) | BYTE 0 | |
| X_0138_D | | 00015B00 | 00015B01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015B00 | 00015B01 | 00000002 ( 2.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0138_ERTEMP | | 00015B02 | 00015B02 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015B02 | 00015B02 | 00000001 ( 1.) | BYTE 0 | |
| X_0138_FCACHE | | 00015B03 | 00015B03 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015B03 | 00015B03 | 00000001 ( 1.) | BYTE 0 | |
| X_0138_GDCS | | 00015B04 | 00015B1A | 00000017 ( 23.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015B04 | 00015B1A | 00000017 ( 23.) | BYTE 0 | |
| X_0138_HFILL | | 00015B1B | 00015B7F | 00000065 ( 101.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015B1B | 00015B7F | 00000065 ( 101.) | BYTE 0 | |
| X_0138_T | | 00015B80 | 00015C7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015B80 | 00015C7F | 00000100 ( 256.) | BYTE 0 | |
| X_0139_D | | 00015C80 | 00015C81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015C80 | 00015C81 | 00000002 ( 2.) | BYTE 0 | |
| X_0139_ERTEMP | | 00015C82 | 00015C96 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015C82 | 00015C96 | 00000015 ( 21.) | BYTE 0 | |
| X_0139_FCACHE | | 00015C97 | 00015C97 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015C97 | 00015C97 | 00000001 ( 1.) | BYTE 0 | |
| X_0139_GDCS | | 00015C98 | 00015D11 | 0000007A ( 122.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015C98 | 00015D11 | 0000007A ( 122.) | BYTE 0 | |
| X_0139_HFILL | | 00015D12 | 00015D7F | 0000006E ( 110.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015D12 | 00015D7F | 0000006E ( 110.) | BYTE 0 | |
| X_0139_T | | 00015D80 | 00015E7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015D80 | 00015E7F | 00000100 ( 256.) | BYTE 0 | |
| X_0140_D | | 00015E80 | 00015E81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015E80 | 00015E81 | 00000002 ( 2.) | BYTE 0 | |
| X_0140_ERTEMP | | 00015E82 | 00015E96 | 00000015 ( 21.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015E82 | 00015E96 | 00000015 ( 21.) | BYTE 0 | |
| X_0140_FCACHE | | 00015E97 | 00015E97 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015E97 | 00015E97 | 00000001 ( 1.) | BYTE 0 | |
| X_0140_GDCS | | 00015E98 | 00015F27 | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015E98 | 00015F27 | 00000090 ( 144.) | BYTE 0 | |
| X_0140_HFILL | | 00015F28 | 00015F7F | 00000058 ( 88.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015F28 | 00015F7F | 00000058 ( 88.) | BYTE 0 | |
| X_0140_T | | 00015F80 | 000160FF | 00000180 ( 384.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00015F80 | 000160FF | 00000180 ( 384.) | BYTE 0 | |
| X_0141_D | | 00016100 | 00016101 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016100 | 00016101 | 00000002 ( 2.) | BYTE 0 | |
| X_0141_ERTEMP | | 00016102 | 0001611E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016102 | 0001611E | 0000001D ( 29.) | BYTE 0 | |
| X_0141_FCACHE | | 0001611F | 0001611F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001611F | 0001611F | 00000001 ( 1.) | BYTE 0 | |
| X_0141_GDCS | | 00016120 | 00016120 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016120 | 00016120 | 00000001 ( 1.) | BYTE 0 | |
| X_0141_HFILL | | 00016121 | 0001617F | 0000005F ( 95.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016121 | 0001617F | 0000005F ( 95.) | BYTE 0 | |
| X_0141_T | | 00016180 | 000164FF | 00000380 ( 896.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016180 | 000164FF | 00000380 ( 896.) | BYTE 0 | |
| X_0142_D | | 00016500 | 00016501 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016500 | 00016501 | 00000002 ( 2.) | BYTE 0 | |
| X_0142_ERTEMP | | 00016502 | 0001650E | 0000000D ( 13.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016502 | 0001650E | 0000000D ( 13.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0142_FCACHE | | 0001650F | 0001650F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001650F | 0001650F | 00000001 ( 1.) | BYTE 0 | |
| X_0142_GDCS | | 00016510 | 00016510 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016510 | 00016510 | 00000001 ( 1.) | BYTE 0 | |
| X_0142_HFILL | | 00016511 | 0001657F | 0000006F ( 111.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016511 | 0001657F | 0000006F ( 111.) | BYTE 0 | |
| X_0142_T | | 00016580 | 0001677F | 00000200 ( 512.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016580 | 0001677F | 00000200 ( 512.) | BYTE 0 | |
| X_0143_D | | 00016780 | 00016781 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016780 | 00016781 | 00000002 ( 2.) | BYTE 0 | |
| X_0143_ERTEMP | | 00016782 | 0001679E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016782 | 0001679E | 0000001D ( 29.) | BYTE 0 | |
| X_0143_FCACHE | | 0001679F | 000167A3 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001679F | 000167A3 | 00000005 ( 5.) | BYTE 0 | |
| X_0143_GDCS | | 000167A4 | 00016833 | 00000090 ( 144.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000167A4 | 00016833 | 00000090 ( 144.) | BYTE 0 | |
| X_0143_HFILL | | 00016834 | 0001687F | 0000004C ( 76.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016834 | 0001687F | 0000004C ( 76.) | BYTE 0 | |
| X_0143_T | | 00016880 | 0001697F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016880 | 0001697F | 00000100 ( 256.) | BYTE 0 | |
| X_0144_D | | 00016980 | 00016981 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016980 | 00016981 | 00000002 ( 2.) | BYTE 0 | |
| X_0144_ERTEMP | | 00016982 | 0001699A | 00000019 ( 25.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016982 | 0001699A | 00000019 ( 25.) | BYTE 0 | |
| X_0144_FCACHE | | 0001699B | 0001699F | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 0001699B | 0001699F | 00000005 ( 5.) | BYTE 0 | |
| X_0144_GDCS | | 000169A0 | 00016A24 | 00000085 ( 133.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 000169A0 | 00016A24 | 00000085 ( 133.) | BYTE 0 | |
| X_0144_HFILL | | 00016A25 | 00016A7F | 0000005B ( 91.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016A25 | 00016A7F | 0000005B ( 91.) | BYTE 0 | |
| X_0144_T | | 00016A80 | 00016B7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016A80 | 00016B7F | 00000100 ( 256.) | BYTE 0 | |
| X_0145_D | | 00016B80 | 00016B81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016B80 | 00016B81 | 00000002 ( 2.) | BYTE 0 | |
| X_0145_ERTEMP | | 00016B82 | 00016B9E | 0000001D ( 29.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016B82 | 00016B9E | 0000001D ( 29.) | BYTE 0 | |
| X_0145_FCACHE | | 00016B9F | 00016BA3 | 00000005 ( 5.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016B9F | 00016BA3 | 00000005 ( 5.) | BYTE 0 | |
| X_0145_GDCS | | 00016BA4 | 00016C5F | 000000BC ( 188.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016BA4 | 00016C5F | 000000BC ( 188.) | BYTE 0 | |
| X_0145_HFILL | | 00016C60 | 00016C7F | 00000020 ( 32.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016C60 | 00016C7F | 00000020 ( 32.) | BYTE 0 | |
| X_0145_T | | 00016C80 | 00016D7F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016C80 | 00016D7F | 00000100 ( 256.) | BYTE 0 | |
| X_0146_D | | 00016D80 | 00016D81 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016D80 | 00016D81 | 00000002 ( 2.) | BYTE 0 | |
| X_0146_ERTEMP | | 00016D82 | 00016D82 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016D82 | 00016D82 | 00000001 ( 1.) | BYTE 0 | |
| X_0146_FCACHE | | 00016D83 | 00016D83 | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016D83 | 00016D83 | 00000001 ( 1.) | BYTE 0 | |

| Psect Name | Module Name | Base | End | Length | Align | Attributes |
|---|---|---|---|---|---|---|
| X_0146_GDCS | | 00016D84 | 00016E29 | 000000A6 ( 166.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016D84 | 00016E29 | 000000A6 ( 166.) | BYTE 0 | |
| X_0146_HFILL | | 00016E2A | 00016E7F | 00000056 ( 86.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016E2A | 00016E7F | 00000056 ( 86.) | BYTE 0 | |
| X_0146_T | | 00016E80 | 00016EFF | 00000080 ( 128.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016E80 | 00016EFF | 00000080 ( 128.) | BYTE 0 | |
| X_0147_D | | 00016F00 | 00016F01 | 00000002 ( 2.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016F00 | 00016F01 | 00000002 ( 2.) | BYTE 0 | |
| X_0147_ERTEMP | | 00016F02 | 00016F0E | 0000000D ( 13.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016F02 | 00016F0E | 0000000D ( 13.) | BYTE 0 | |
| X_0147_FCACHE | | 00016F0F | 00016F0F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016F0F | 00016F0F | 00000001 ( 1.) | BYTE 0 | |
| X_0147_GDCS | | 00016F10 | 00016F7E | 0000006F ( 111.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016F10 | 00016F7E | 0000006F ( 111.) | BYTE 0 | |
| X_0147_HFILL | | 00016F7F | 00016F7F | 00000001 ( 1.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016F7F | 00016F7F | 00000001 ( 1.) | BYTE 0 | |
| X_0147_T | | 00016F80 | 0001707F | 00000100 ( 256.) | BYTE 0 | NOPIC,USR,CON,REL,LCL,NOSHR, EXE, RD, WRT,NOVEC |
| | ECKAC | 00016F80 | 0001707F | 00000100 ( 256.) | BYTE 0 | |

## Symbols By Name

| Symbol | Value | Symbol | Value | Symbol | Value | Symbol | Value |
|---|---|---|---|---|---|---|---|

Key for special characters above:

| | |
|---|---|
| * | - Undefined |
| U | - Universal |
| R | - Relocatable |
| X | - External |
| V | - Vectored |

```
                                                  +-----------------+
                                                  ! Image Synopsis !
                                                  +-----------------+

Virtual memory allocated:                          00000000 000171FF 00017200 (94720. bytes, 185. pages)
Stack size:                                               0. pages
Image binary virtual block limits:                        1.      185. (  185. blocks)
Image name and identification:                     ECKAC V08.00
Number of files:                                          5.
Number of modules:                                        5.
Number of program sections:                             798.
Number of global symbols:                                 0.
Number of image sections:                                 1.
Image type:                                        SYSTEM.
Map format:                                        DEFAULT in file DISK$UCODPACK00:[VAX750.MIC]ECKAC0.MAP;12
Estimated map length:                              806. blocks
                                                +----------------------+
                                                ! Link Run Statistics !
                                                +----------------------+

Performance Indicators                             Page Faults   CPU Time        Elapsed Time
----------------------                             -----------   --------        ------------
    Command processing:                                     56   00:00:00.45     00:00:00.73
    Pass 1:                                                181   00:00:01.66     00:00:03.69
    Allocation/Relocation:                                  43   00:00:00.71     00:00:02.55
    Pass 2:                                                186   00:00:03.37     00:00:07.46
    Map data after object module synopsis:                   3   00:00:03.62     00:00:05.33
    Symbol table output:                                     0   00:00:00.02     00:00:00.12
Total run values:                                          469   00:00:09.83     00:00:19.88

Using a working set limited to 1050 pages and 525 pages of data storage (excluding image)

Total number object records read (both passes):   3602
    of which 0 were in libraries and 50 were DEBUG data records containing 19185 bytes

Number of modules extracted explicitly              = 0
    with 0 extracted to resolve undefined symbols

0 library searches were for symbols not in the library searched

A total of 0 global symbol table records was written

LINK/SYSTEM=0/MAP/EXE=ECKAC ECKAC0+ECKAC1+ECKAC2+ECKAC3+ECKAC4
```

Table of contents

(1)  24  "MIC Revision History"
(1)  1  ""
(1)  3  "DIAGNOSTIC MICROCODE MACROS"
(1)  98  "  RDM Control File Macros"
(1)  109  "  Diagnostic Macros"
(1)  194  ""
(1)  195  "MODULE GATE ARRAY MAPS"
(1)  197  "  L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1)  254  "  L0002 GATE ARRAY LOCATION"
(1)  311  "  L0003 GATE ARRAY LOCATION"
(1)  369  "  L0004 GATE ARRAY LOCATION"
(1)  427  "  L0011 & L0016 GATE ARRAY LOCATION"
(1)  485  ""
(1)  490  "EQUATED SYMBOLS
(1)  493  ""

```
0000     1          .TITLE  ECKAC VAX-11/750 MIC MICRO DIAGNOSTICS
0000     2          .IDENT  /V08.00/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1982
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "MIC Revision History"
0000     25 ;++
0000     26 ; MIC MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Bill Landry and Dave Spain      4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Bill Landry and Dave Spain      5-FEB-1980
0000     34 ;       Fixed numerous tests to invalidate both groups of TB before using them.
0000     35 ;
0000     36 ; 00-03 Bill Landry                     6-FEB-1980
0000     37 ;       Fixed my typing errors in test 4C.
0000     38 ;
0000     39 ; 00-04 Bill Landry and Dave Spain      7-FEB-1980
0000     40 ;       Skipped subtest 7 - 10 in test 4C. Also added cache data to PC
0000     41 ;       increment and ADD adder tests.
0000     42 ;
0000     43 ; 00-05 Bill Landry     21-FEB-1980
0000     44 ;       Fixed CMPREGMSK psuedo-ops in TB tests.
0000     45 ;
0000     46 ; 00-06 Bill Landry and Dave Spain      27-FEB-1980
0000     47 ;       Changed a CLRTB.VA_VA macro in test 4C to CLRTB.VA_R[]. Also added
0000     48 ;       cache validates to test 12 to prevent random cache parity errors.
0000     49 ;
0000     50 ; 00-07 Bill Landry     3-MAR-1980
0000     51 ;       Added changes for revised SKIP pseudo op.
0000     52 ;
0000     53 ; 00-08 Bill Landry     13-MAR-1980
0000     54 ;       Fixed tests that didnot use a CMPREGMSK pseudo op for testing the
0000     55 ;       CSTRP register.
0000     56 ; 00-09 Bill Landry     25-MAR-1980
0000     57 ;       Added cache tests.
0000     58 ;
0000     59 ; 01-10 Dave Spain       4-APR-1980
0000     60 ;       Added IRD tests to official release version.
0000     61 ;
0000     62 ; 01-11 Bill Landry     4-APR-1980
0000     63 ;       Moved XB, PREFETCH and IRD tests to ECKAB.SRC so manufacturing can
0000     64 ;       run one tape for DPM.
0000     65 ;
0000     66 ; 00-12 Bill Landry     4-APR-1980
0000     67 ;       Let's save version 1.0 for our first SDC release
0000     68 ;
0000     69 ; 00-13 Bill Landry     8-APR-1980
0000     70 ;       Revised cache tests to survive parity error traps.
0000     71 ;
0000     72 ; 00-14 Bill Landry and Dave Spain      15-APR-1980
0000     73 ;       Reassembled using new COMETDEF.
0000     74 ;
0000     75 ; 00-15 Dave Spain      18-APR-1980
0000     76 ;       Changed tests 4F and 50 to call out the ACV and UTR gate arrays instead
0000     77 ;       of the MIC module.
0000     78 ;
```

```
0000     79 ; 00-16 Bill Landry     1-MAY-1980
0000     80 ;       Revised TB tests to loop on error nicely.
0000     81 ;       Deleted redundent TB tests.
0000     82 ;
0000     83 ; 00-17 Bill Landry     14-MAY-1980
0000     84 ;       Added TB parity error tests
0000     85 ;
0000     86 ; 00-18 Bill Landry     10-JUN-1980
0000     87 ;       Added RDM tests to beginning of tape.
0000     88 ;
0000     89 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     90 ;       Added diagnostic macro file to title section and submitted to
0000     91 ;       RELEASE ENGINEERING.
0000     92 ;
0000     93 ; 01-01 Dave Spain      19-JUN-1980
0000     94 ;       Fixed Prefetch Incrementer test to clear out cache data after
0000     95 ;       every prefetch so that old data from cache will not cause test
0000     96 ;       to mistake bad addresses for good ones.
0000     97 ;
0000     98 ; 01-02 Bill Landry     26-JUN-1980
0000     99 ;       Added tests that check XB trap conditions
0000    100 ;
0000    101 ; 01-03 Bill Landry     30-JUN-1980
0000    102 ;       Added tests that check XB access violations
0000    103 ;
0000    104 ; 01-04 Dave Spain      11-JUL-1980
0000    105 ;       Changed location of BEGINSA pseudo-op to fix signature analysis
0000    106 ;       problem in PC_PC+WBUS test.
0000    107 ;
0000    108 ; 01-05 Bill Landry     8-AUG-1980
0000    109 ;       Added CMI tests in honor of my vacation.
0000    110 ;
0000    111 ; 01-06 Bill Landry     25-AUG-1980
0000    112 ;       Remodeled RDM to CMI tests.
0000    113 ;
0000    114 ; 01-07 Bill Landry     23-SEP-1980
0000    115 ;       Added byte mask testing and improved testing of first memory array.
0000    116 ;
0000    117 ; 02-00 Bill Landry     29-SEP-1980
0000    118 ;       Second release to SDC
0000    119 ;
0000    120 ; 02-01 Bill Landry     5-NOV-1980
0000    121 ;       Added miscellaneous enhancements
0000    122 ;
0000    123 ; 03-00 Bill Landry     17-NOV-1980
0000    124 ;       Third release to SDC
0000    125 ;
0000    126 ; 03-01 Bill Landry     24-NOV-1980
0000    127 ;       Added tests for cache parity logic
0000    128 ;
0000    129 ; 03-02 Bill Landry     30-DEC-1980
0000    130 ;       Separated RDM and MIC tests into two files
0000    131 ;
0000    132 ; 04-00 Bill Landry     05-JAN-1981
0000    133 ;       Submitted to RELEASE ENGINEERING
```

```
0000    134 ;
0000    135 ; 04-01 Bill Landry     14-JAN-1981
0000    136 ;       Revised ECC test to fix bug caused by ECO to INT PEND logic.
0000    137 ;
0000    138 ; 04-02 Bill Landry     27-FEB-1981
0000    139 ;       Udated MDR test to include branching logic.
0000    140 ;       Fixed PC_* tests to fix bug found by field service.
0000    141 ;
0000    142 ; 05-00 Bill Landry     2-MAR-1981
0000    143 ;       Added WRITE BUS ERROR INTERRUPT test
0000    144 ;       Submitted to RELEASE ENGINEERING
0000    145 ;
0000    146 ; 05-01 Bill Landry     18-MAR-1981
0000    147 ;       Made minor changes to various tests as suggested by BT manufacturing.
0000    148 ;
0000    149 ; 05-02 Bill Landry     8-JUN-1981
0000    150 ;       Fixed bug found by Paul Magnet in TB TAG PARITY TESTS.
0000    151 ;
0000    152 ; 05-03 Bill Landry     20-AUG-1981
0000    153 ;       Fixed bugs in memory tests caused by eco to interrupt pending logic.
0000    154 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    155 ;       logic.
0000    156 ;       Replaced all references to COMET with the word CPU.
0000    157 ;       Fixed bug in CACHE TAG MEMORY DUAL ADDRESSING TEST.
0000    158 ;       Submitted to RELEASE ENGINEERING.
0000    159 ;
0000    160 ; 05-04 Bill Landry     4-NOV-1981
0000    161 ;       Moved WCS test from DPM tape to this tape
0000    162 ;       Submitted to RELEASE ENGINEERING.
0000    163 ;
0000    164 ; 05-05 Bill Landry     10-NOV-1981
0000    165 ;       Changed all references of CLRTB.VA_WB to CLRTB.VA_VA.  This fixes a
0000    166 ;       problem caused by plugging in the FPA.
0000    167 ;       Re-released to SDC. Stopped release of 5.4
0000    168 ;
0000    169 ; 05-06 Bill Landry     6-JAN-1982
0000    170 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    171 ;       logic (same as 5.3 above).
0000    172 ;
0000    173 ; 05-07 Bill Landry     23-FEB-1982
0000    174 ;       Fixed more bugs in single bit error tests found by PAUL NATUSH.
0000    175 ;       Same as revision 5.3.
0000    176 ; 05-08 Dave Shull      9-Mar-1982
0000    177 ;       Changed the the IFERROR callouts so when L0011 (MC0) is encountered
0000    178 ;       L0016 (MC1) will be included, when M8728 (MA0) is encountered M8750
0000    179 ;       (MA1) will be included, and when the MAP gate array is encountered
0000    180 ;       MAD gate array will be included.
0000    181 ; 05-09 Dave Shull      10-Mar-1982
0000    182 ;       Added a test to check MS750 controller CSR2 and determine the
0000    183 ;       controller type (ie., L0011 or L0016) and to also print a configuration
0000    184 ;       map of the array's that are installed.
0000    185 ; 06-00 Dave Shull      22-mar-1982
0000    186 ;       Changed the 4 existing memory tests to check for array 0 type then
0000    187 ;       adjust addresses accordingly and test all of the first array versus
0000    188 ;       only testing the first 256kb.
```

```
                     0000    189 ;       Added 4 new tests that will check to see if array 1 is present and if
                     0000    190 ;       it is then determine the size and test accordingly.  It looks at both
                     0000    191 ;       array 0 and array 1, array 0 to determine the starting address of array
                     0000    192 ;       1 and array 1 to determine the starting and ending address to be tested
                     0000    193 ; 06-01 Dave Shull      26-Mar-1982
                     0000    194 ;       Changed READ.SEC MICRO ORDER test so the test could ran in an infinite
                     0000    195 ;       loop by itself.  Added code to reload Cache everytime the test is
                     0000    196 ;       restarted in any of the test that are using cache data that could get
                     0000    197 ;       blown away.
                     0000    198 ; 06-02 Dave Shull      05-Apr-1982
                     0000    199 ;       Removed all of the BEGINSA and ENDSA pseudo's
                     0000    200 ;       Submitted to RELEASE ENGINEERING
                     0000    201 ; 07-00 Dave Shull      10-May-1982
                     0000    202 ;       Added new test ( name <Test PCBACK While PC Increment Logic is Active>)
                     0000    203 ;       to check for side-affects on PCBACK register while incrementin the PC
                     0000    204 ;       through the execution buffer.
                     0000    205 ; 07-01 Dave Shull      14-May-1982
                     0000    206 ;       Changed all CLRTB.VA_VA macro's to prevent translation buffer
                     0000    207 ;       invalidate failures when manufacturing runs the micro's with an
                     0000    208 ;       external clock set to fast margins.
                     0000    209 ; 07-02 Dave Shull      17-May-1982
                     0000    210 ;       Added coverage in the Read Lock Function Test for the CML gate array.
                     0000    211 ; 07-03 Dave Shull      14-Jun-1982
                     0000    212 ;       Changed CMK callouts to CML for new gate array.
                     0000    213 ; 08-00 Joe Zagarella   11-Feb-1986
                     0000    214 ;       Enhanced the following tests to be compatible with the new memory
                     0000    215 ;       controller(L0022) and array card(M7199):
                     0000    216 ;       Test write bus error interrupt
                     0000    217 ;       Test for controller type and array configuration
                     0000    218 ;       Test data integrity for address test
                     0000    219 ;       Main memory dual addressing test
                     0000    220 ;       Array 0 memory test (part1 thru part4)
                     0000    221 ;       Array 1 memory test (part1 thru part4)
                     0000    222 ;
                     0000    223 ;--
                     0000      1         .SBTTL  ""
```

```
                        0000     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
                        0000     4 ;++
                        0000     5 ; Revision History
                        0000     6 ;
                        0000     7 ; 28    Dave Spain      22-Jun-82
                        0000     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
                        0000     9 ;       Decided not to use them.
                        0000    10 ;
                        0000    11 ; 27    Dave Spain      15-Jun-82
                        0000    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
                        0000    13 ;
                        0000    14 ; 26    Dave Spain      19-May-82
                        0000    15 ;       Added M[]_FPA macro.
                        0000    16 ;
                        0000    17 ; 25    Rich Lewis      23-Feb-82
                        0000    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
                        0000    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
                        0000    20 ;
                        0000    21 ; 24    Dave Spain      02-Feb-82
                        0000    22 ;       Added FPA_R[]+M[] macro.
                        0000    23 ;
                        0000    24 ; 23    Bill Landry     22-OCT-81
                        0000    25 ;       Added PSL_RDM macro
                        0000    26 ;
                        0000    27 ; 22    Dave Spain       5 Oct 81
                        0000    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
                        0000    29 ;
                        0000    30 ; 21    Bill Landry     29 Sept 81
                        0000    31 ;       Added FPA macro, FPA_M[] FPA_R[].
                        0000    32 ;
                        0000    33 ; 20    Dave Spain      14 Aug 81
                        0000    34 ;       Added the TESTFPA [] macro.
                        0000    35 ;
                        0000    36 ; 19    Dave Spain      18 July 81
                        0000    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
                        0000    38 ;
                        0000    39 ; 18    Dave Spain      24 Sept 80
                        0000    40 ;       Added RDM_Q macro. This macro uses the ALU.
                        0000    41 ;
                        0000    42 ; 17    Bill Landry     23 Sept 80
                        0000    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
                        0000    44 ;
                        0000    45 ; 16    Dave Spain      27 August 80
                        0000    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
                        0000    47 ;
                        0000    48 ; 15    Bill Landry     4 June 80
                        0000    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
                        0000    50 ;       satisfy validity checks.
                        0000    51 ;
                        0000    52 ; 14    Dave Spain      23 May 80
                        0000    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
                        0000    54 ;       Bill.
                        0000    55 ;
                        0000    56 ; 13    Dave Spain      14 Apr 80
                        0000    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

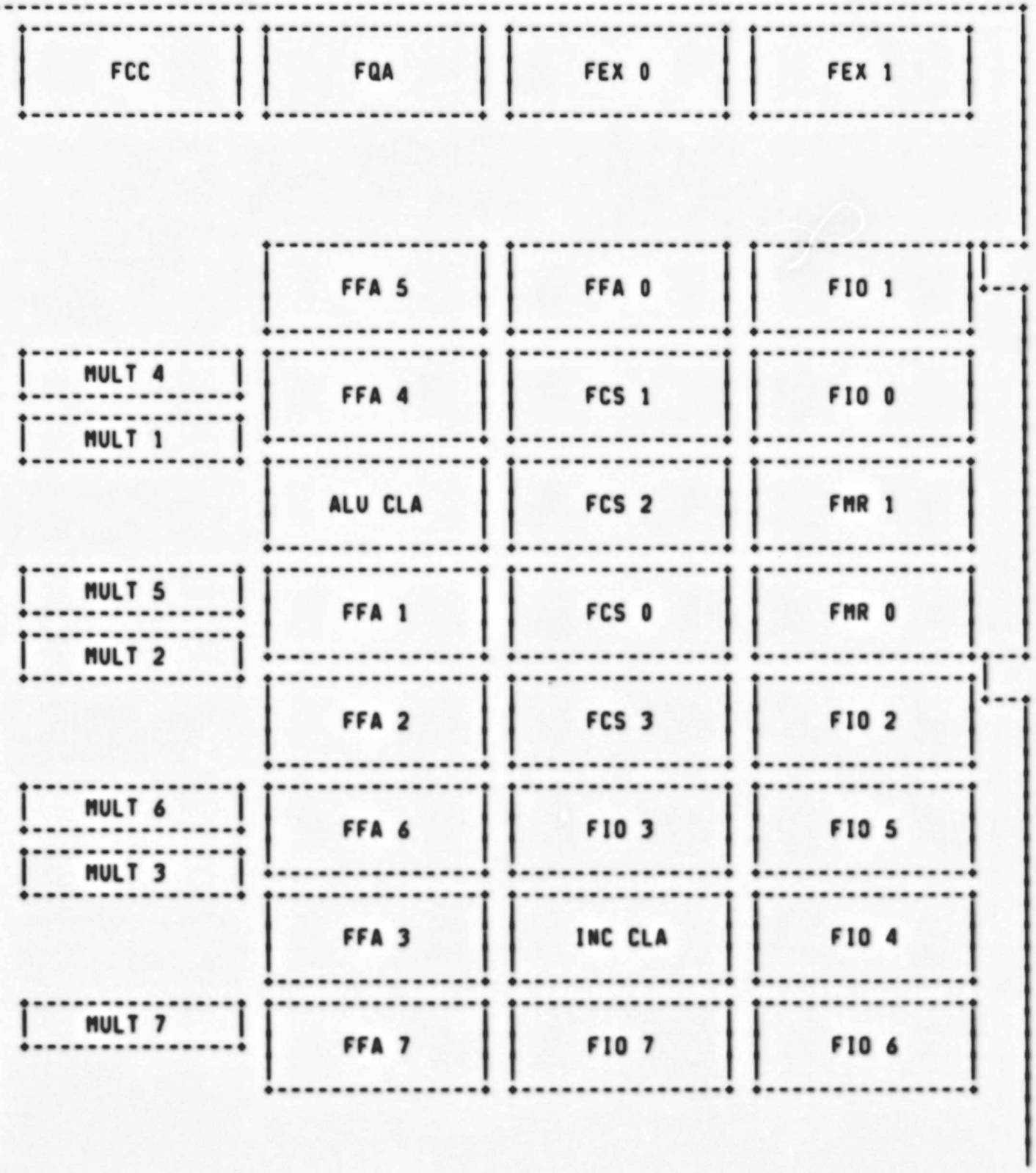
```
0000     58 ;       the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain      27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry     19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry     19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry     18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry     07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain      06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain      03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain      22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain      30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain      30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain      29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                                "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                              "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                           "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
                0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_ -
cont> WB"
                0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   115 ;                                       BUS/PRB.RD.PTE"
                0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   118 ;                                       BUS/PRB.RD.PTE"
                0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_ -
cont> WB"
                0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
                0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
                0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
                0000   124 ;                                       WCTRL/VA_WB"
                0000   125 ;FPA_M[] FPA_RDM                       "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
                0000   126 ;FPA_M[] FPA_R[]                       "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX -
cont> /R.S,
                0000   127 ;                                       ROT/ZERO,ALU/OR"
                0000   128 ;FPA_RDM                               "WB_RDM,FPA/FPA_DATA.WBUS"
                0000   129 ;FPA_R[]                               "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_D -
cont> ATA.WBUS"
                0000   130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FP -
cont> A/FPA_DATA.WBUS"
                0000   131 ;FPA_WB                         "FPA/FPA_DATA.WBUS"
                0000   132 ;MEMSCAR_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
                0000   133 ;M[]_FPA                               "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
                0000   134 ;M[]_LONLIT                    "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
                0000   135 ;M[]_R[].OR.NIB0(M)            "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
                0000   136 ;                                       MUX/R.S,DQ1/NOP"
                0000   137 ;M[]_RDM                               "WB_RDM,SPW/MLONG,MSRC/a1"
                0000   138 ;PC_PC+RDM                     "WCTRL/PC_PC+WB,WB_RDM"
                0000   139 ;PC_PC+4 MB_XB                 "MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
                0000   140 ;PC_PC+4 RDM_XB                "RSRC/ZERO,MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,
                0000   141 ;                                       MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
                0000   142 ;PC_RDM                        "WB_RDM,WCTRL/PC_WB"
                0000   143 ;PSL_RDM                               "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
                0000   144 ;PSL_WB                        "CCPSL/PSL_WB.CCBR_ALUS"
                0000   145 ;Q_D WX_R                      "ALPCTL/WX_R.Q_D"
                0000   146 ;Q_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
                0000   147 ;Q_M[].RL.PTE                  "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
                0000   148 ;Q_R[].AND.Q                   "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
                0000   149 ;Q_R[].ASL.P                   "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
                0000   150 ;Q_R[].OR.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
                0000   151 ;Q_R[].RL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
                0000   152 ;RDM_ALUF                      "ALPCTL/WB_ALUF,RDM_WB"
                0000   153 ;RDM_FPA                               "FPA/WBUS_FPA,RDM_WB"
                0000   154 ;RDM_LONLIT                    "WB_LONLIT,RDM_WB"
                0000   155 ;RDM_LONLIT.Q_M                "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
                0000   156 ;RDM_M[]                               "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
                0000   157 ;RDM_M[].OR.R[]                "WB_M[a1].OR.R[a2],RDM_WB"
                0000   158 ;RDM_M[]-R[]                   "WB_M[a1]-R[a2],RDM_WB"
                0000   159 ;RDM_M[].XZ.OR.R[]             "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                0000   160 ;                              RDM_WB"
                0000   161 ;RDM_PC                        "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
                0000   162 ;RDM_PCBACK                    "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
                0000   163 ;RDM_Q                         "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
                0000   164 ;RDM_Q Q_D                     "ALPCTL/WX_Q.Q_D,RDM_WB"
```

```
0000   165 ;RDM_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                     "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                     "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

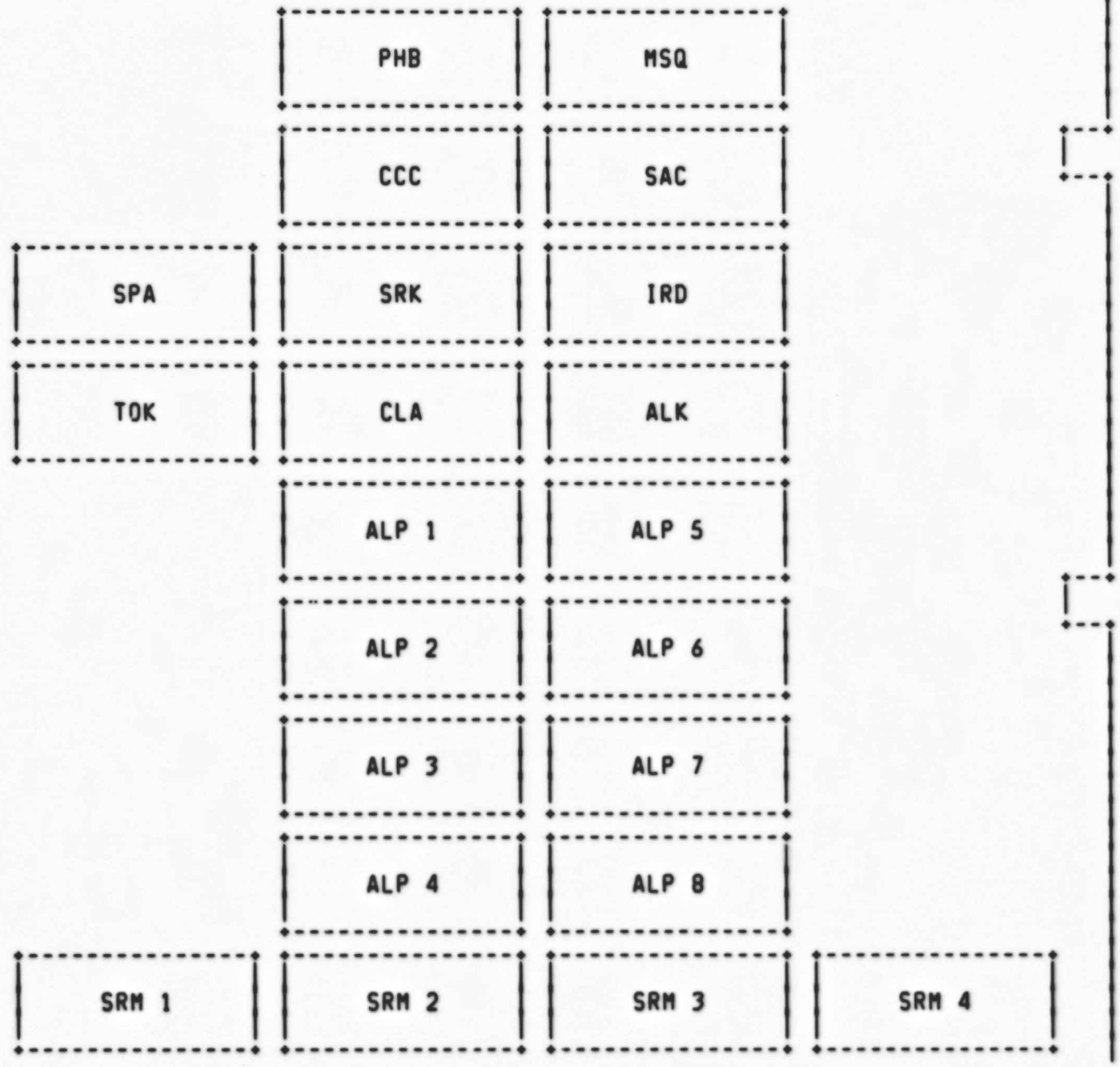
```
                0000    168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
                0000    169 ;R[]_RDM                                "WB_RDM,SPW/RLONG,RSRC/a1"
                0000    170 ;R[].DT_Q                       "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
                0000    171 ;R[]_VA                         "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
                0000    172 ;R[]_WB                         "RSRC/a1,SPW/RLONG"
                0000    173 ;TB_RDM                         "WCTRL/TB_WB,WB_RDM,MSRC/VA"
                0000    174 ;TB_WB                          "WCTRL/TB_WB,MSRC/VA"
                0000    175 ;TESTFPA []                     "TESTFPA/a1"
                0000    176 ;VA_PC+LONLIT+I.PC_PC+I         "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/ -
cont> OR,
                0000    177 ;                                       ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
                0000    178 ;                                       SIZE[LONG]"
                0000    179 ;VA_Q.OR.R[]                    "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
                0000    180 ;VA_RDM                         "WB_RDM,WCTRL/VA_WB"
                0000    181 ;VA_R[].RL.P                    "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
                0000    182 ;VA_VA-ZLIT0[]                  "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
                0000    183 ;                                       MUX/M.S,ALU/A-B-CI"
                0000    184 ;VA_WB                          "WCTRL/VA_WB"
                0000    185 ;WB_FPA CCR                     "FPA/WBUS_FPA.CC"
                0000    186 ;WB_FPA TCR                     "WCTRL/FPTCR"
                0000    187 ;WB_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000    188 ;WB_LONLIT.Q_M                  "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000    189 ;WB_M[]+R[]                     "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
                0000    190 ;WB_R[].Q_D                     "RSRC/a1,ALPCTL/WX_R.Q_D"
                0000    191 ;WDR_R[]                                "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/Z -
cont> ERO"
                0000    192 ;WRITE.LONG R[]                 "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
                0000    193 ;                                       MUX/R.S,ALU/OR,SIZE[LONG]"
                0000    194                 .SBTTL  ""
                0000    195                 .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                        .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```



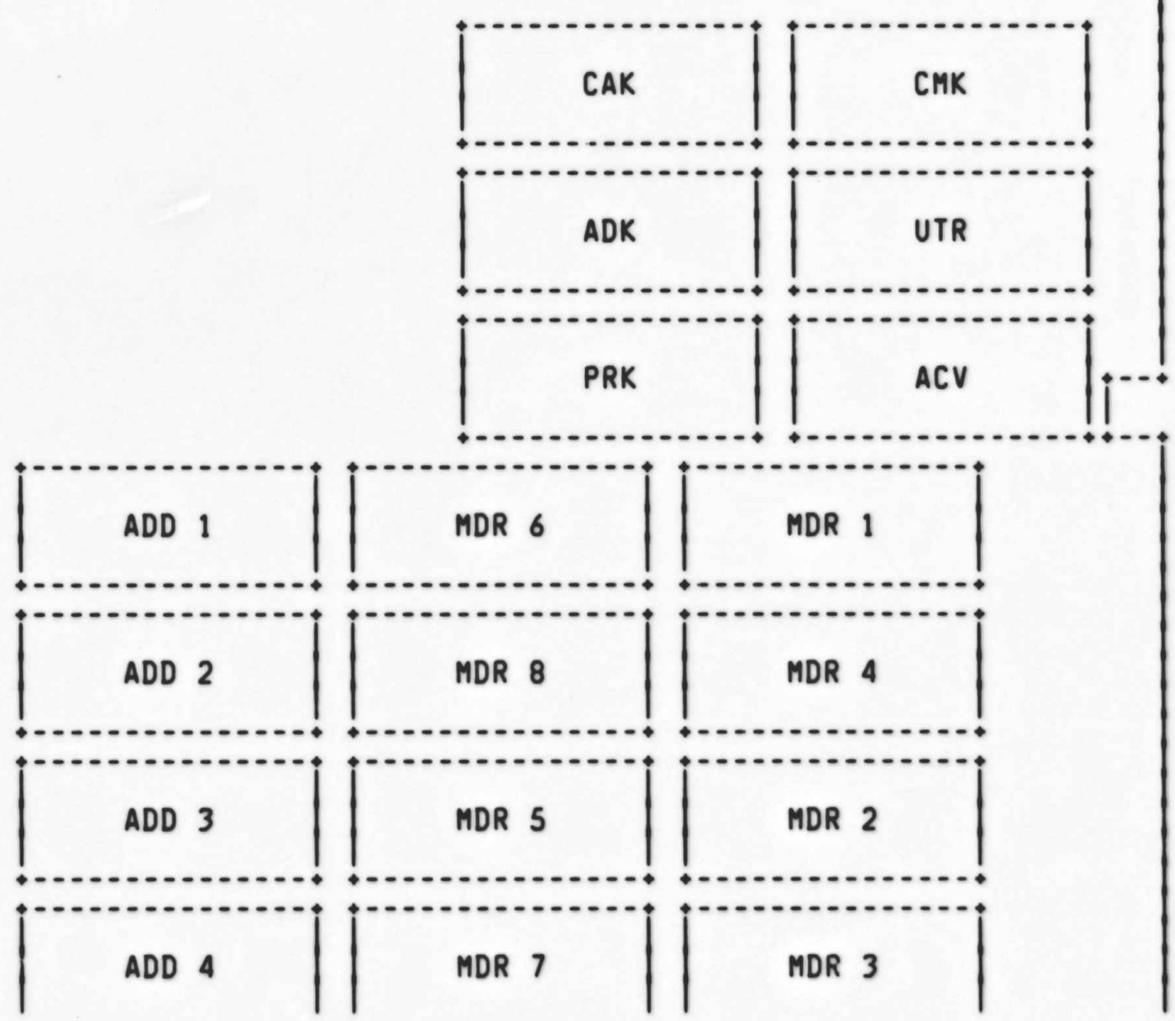
```
                0000   252 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0002 GATE ARRAY LOCATION"
```



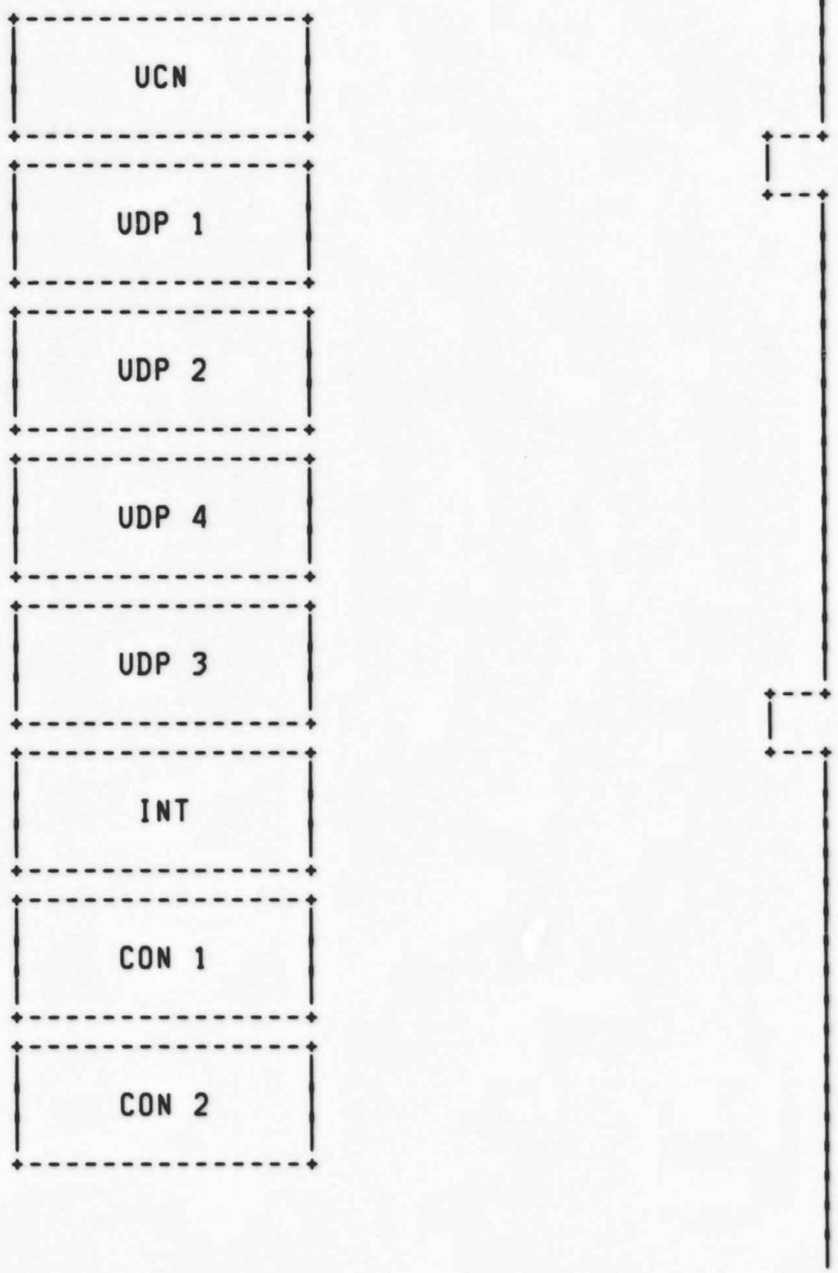
```
0000   309 ;+-----------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0003 GATE ARRAY LOCATION"
```



```
0000   366 ;|              +---------------+ +---------------+ +---------------+          |
0000   367 ;+-------------------------------------------------------------------------------+
```

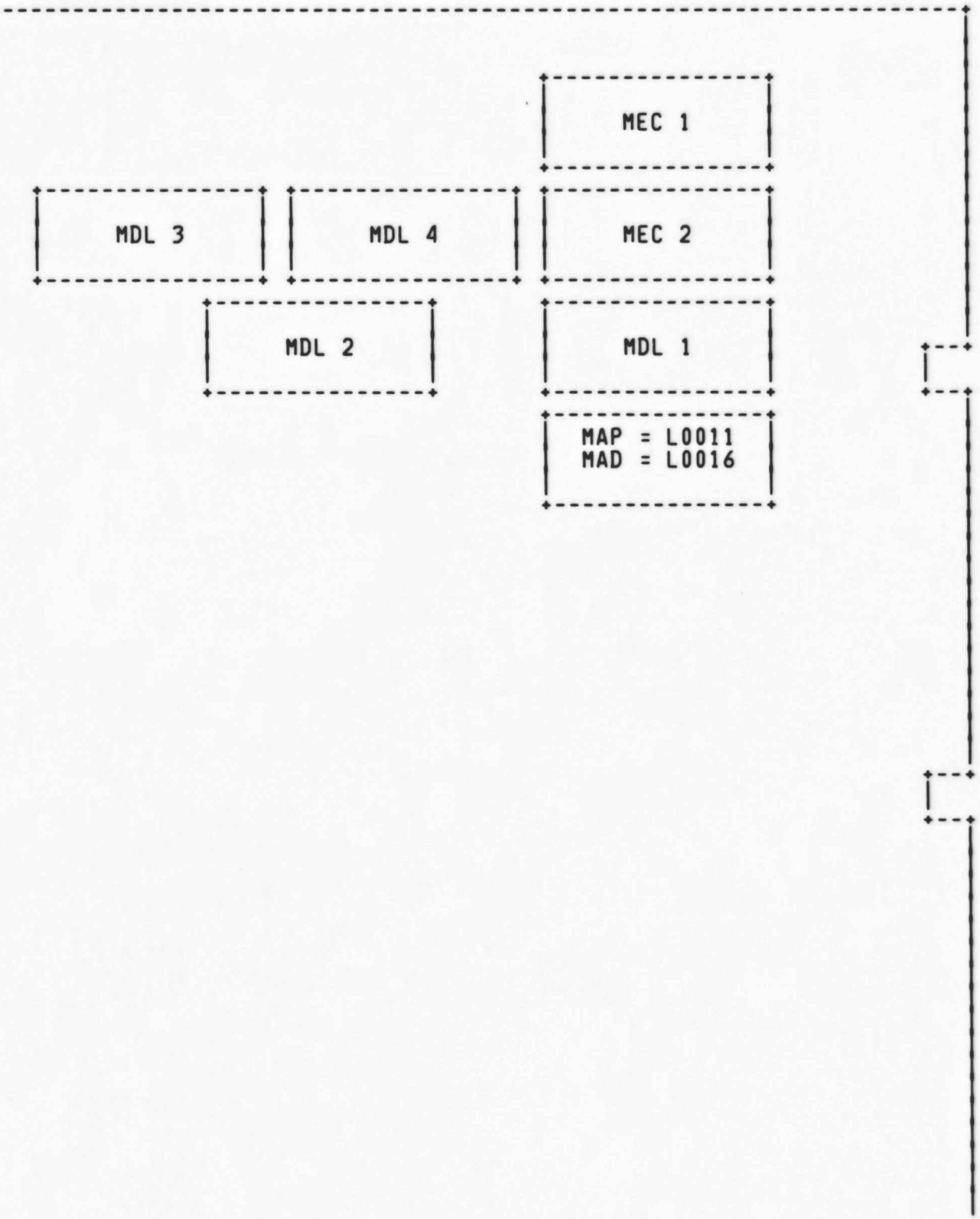
```
                        .SBTTL  "       L0004 GATE ARRAY LOCATION"
```



```
ZZ-ECKAC-8.0    V08.00                  " L0004 GATE ARR17-FEB-1986      Fiche 1  Frame C5        Sequence 54
ECKAC                       VAX-11/750 MIC MICRO DIAGNOSTICS         17-FEB-1986 13:37:43  VAX/VMS Macro V04-00        Page  1
V08.00                      " L0004 GATE ARRAY LOCATION"             14-AUG-1984 15:55:16  [VAX750]COMETASM.MAR;221          (1

                    0000    424 ;|                                                                                      |
                    0000    425 ;+--------------------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
```



```
0000   482 ;|                                                                                |
0000   483 ;+--------------------------------------------------------------------------------+
```

```
                        0000    485         .SBTTL  ""
                        0000    486         .LIST   MC
                        0000    487         .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                        0000    488         .LIST   ME
                        0000    489         .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
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_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    491   .NLIST  ME
          0000    492   .LIST   MC
          0000    493   .SBTTL  ""
          0000    494   .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
                0008     2 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
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
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
CML             = 0000000E
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
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_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
END_SLICE       = 00000025
EN_TRACE_REG    = 00000008
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
L               = 00000004
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     = 000007A2
M$TEST_SIZE_D   = 000003E2
MA0             = 00000008
MA1             = 0000000A
MA2             = 0000000C
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
MC2             = 0000000B
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
MEC             = 0000001A
MHREG           = 00007418
MH_REG_BYTE_0   = 00007418
MH_REG_BYTE_1   = 00007419
MH_REG_BYTE_2   = 0000741A
MH_REG_BYTE_3   = 0000741B
MIC             = 00000001
MICROWORD       = 0000000A
```

```
MICRO_ADDR_INH = 00000080
MONITOR_SIZE   = 00002C5E
MSQ            = 00000008
MT_KEY         = 0000002E
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
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
RT_KEY         = 0000002C
SAC            = 00000009
SA_CLOCK       = 00000080
SA_FLAG        = 00000010
SA_KEY         = 00000010
SA_START_STOP  = 00000080
SBE_FLAG       = 00000020
SBE_KEY        = 00000042
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           = 00000002
TEMP1          = 00000002
TEST           = 00000004
TEST_KEY       = 00000000
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 synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| DIRECTORY | 00000008 ( 8.) | 01 ( 1.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |

## Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 48 | 00:00:00.11 | 00:00:00.43 |
| Command processing | 879 | 00:00:00.94 | 00:00:01.48 |
| Pass 1 | 611 | 00:00:03.21 | 00:00:04.44 |
| Symbol table sort | 0 | 00:00:00.33 | 00:00:00.33 |
| Pass 2 | 9 | 00:00:01.76 | 00:00:02.20 |
| Symbol table output | 2 | 00:00:00.22 | 00:00:00.50 |
| Psect synopsis output | 0 | 00:00:00.02 | 00:00:00.02 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 1549 | 00:00:06.59 | 00:00:09.40 |

The working set limit was 2400 pages.
28063 bytes (55 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 285 non-local and 0 local symbols.
720 source lines were read in Pass 1, producing 10 object records in Pass 2.
23 pages of virtual memory were used to define 2 macros.

## Macro library statistics

| Macro library name | Macros defined |
|---|---|
| DISK$UCODPACK00:[VAX750]COMETMAC.MLB;1 | 2 |
| SYS$COMMON:[SYSLIB]STARLET.MLB;2 | 0 |
| TOTALS (all libraries) | 2 |

552 GETS were required to define 2 macros.

There were no errors, warnings or information messages.

MACRO UCOD$0:[VAX750.MIC]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.MIC]ECKAC0/LIS=ECKAC0/OBJ=ECKAC0

(1)  24  "MIC Revision History"
(1)  1  ""
(1)  3  "DIAGNOSTIC MICROCODE MACROS"
(1)  98  "  RDM Control File Macros"
(1)  109  "  Diagnostic Macros"
(1)  194  ""
(1)  195  "MODULE GATE ARRAY MAPS"
(1)  197  "  L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1)  254  "  L0002 GATE ARRAY LOCATION"
(1)  311  "  L0003 GATE ARRAY LOCATION"
(1)  369  "  L0004 GATE ARRAY LOCATION"
(1)  427  "  L0011 & L0016 GATE ARRAY LOCATION"
(1)  485  ""
(1)  490  "EQUATED SYMBOLS
(1)  493  ""
(1)  2  "TEST  001  TEST ADK S/C REGISTERS
(1)  193  "TEST  002  TEST CAK S/C REGISTERS
(1)  339  "TEST  003  TEST UTR S/C REGISTERS
(1)  492  "TEST  004  TEST MEMSCAR
(1)  669  "TEST  005  TEST PROCESSOR INITIALIZE BUS FUNCTION
(1)  816  "TEST  006  TEST MDR REGISTER
(1)  958  "TEST  007  TEST "WCTRL/MDR_0" MICRO ORDER
(1)  1066  "TEST  008  VA REGISTER TEST
(1)  1171  "TEST  009  PC REGISTER TEST
(1)  1277  "TEST  00A  TEST WDR REGISTER
(1)  1378  "TEST  00B  TEST WDR ROTATED
(1)  1565  "TEST  00C  TEST ASYNCHRONOUS SYSTEM TRAP LEVEL REGISTER
(1)  1663  "TEST  00D  PC BACKUP LATCH TEST
(1)  1795  "TEST  00E  PC_PC+1 TEST
(1)  1948  "TEST  00F  PC_PC+2 TEST
(1)  2102  "TEST  010  PC_PC+4 TEST
(1)  2255  "TEST  011  VA_VA+4 TEST
(1)  2382  "TEST  012  ADD ADDER TEST
(1)  2727  "TEST  013  PC_PC+WBUS TEST
(1)  2885  "TEST  014  TEST MBUS_CACHE_WBUS DATA PATH
(1)  3011  "TEST  015  CACHE HIT COMPARATOR TEST 1
(1)  3219  "TEST  016  CACHE HIT COMPARATOR TEST 2
(1)  3425  "TEST  017  CACHE HIT COMPARATOR TEST 3
(1)  3636  "TEST  018  CACHE HIT COMPARATOR TEST 4
(1)  3843  "TEST  019  CACHE DATA MEMORY DUAL ADDRESSING TEST
(1)  4025  "TEST  01A  CACHE TAG MEMORY DUAL ADDRESSING TEST
(1)  4230  "TEST  01B  CACHE DATA MEMORY MARCH TEST
(1)  4444  "TEST  01C  CACHE TAG MEMORY MARCH TEST
(1)  4722  "TEST  01D  CACHE VALID BIT MARCH TEST
(1)  5029  "TEST  01E  TEST FORCE PARITY ERROR MICRO FUNCTION
(1)  5144  "TEST  01F  CACHE DATA PARITY TEST 1
(1)  5384  "TEST  020  CACHE DATA PARITY TEST 2
(1)  5500  "TEST  021  CACHE DATA PARITY TEST 3
(1)  5683  "TEST  022  CACHE TAG PARITY TEST 1
(1)  5846  "TEST  023  CACHE TAG PARITY TEST 2
(1)  5978  "TEST  024  CACHE TAG PARITY TEST 3
(1)  6142  "TEST  025  TEST MBUS_TB_WBUS DATA PATH
(1)  6266  "TEST  026  TB DATA PATH TO AND FROM GROUP 0 TAG MEMORY TEST
(1)  6407  "TEST  027  TB DATA PATH TO AND FROM GROUP 1 TAG MEMORY TEST
(1)  6548  "TEST  028  TB GROUP 0 FORCE MISS TEST
(1)  6688  "TEST  029  TB GROUP 1 FORCE MISS TEST
(1)  6828  "TEST  02A  TB GROUP 0 VALID BIT AND TB MISS UTRAP TEST

```
(1)     7023        "TEST   02B     TB GROUP 1 VALID BIT AND TB MISS UTRAP TEST
(1)     7196        "TEST   02C     TB WRITE PULSE LOGIC TEST 1
(1)     7410        "TEST   02D     TB WRITE PULSE LOGIC TEST 2
(1)     7623        "TEST   02E     TB WRITE PULSE LOGIC TEST 3
(1)     7824        "TEST   02F     TB WRITE PULSE LOGIC TEST 4
(1)     8025        "TEST   030     TB WRITE PULSE LOGIC TEST 5
```

```
0000     1          .TITLE  ECKAC VAX-11/750 MIC MICRO DIAGNOSTICS
0000     2          .IDENT  /V08.00/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1982
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "MIC Revision History"
0000     25 ;++
0000     26 ; MIC MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Bill Landry and Dave Spain      4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Bill Landry and Dave Spain      5-FEB-1980
0000     34 ;       Fixed numerous tests to invalidate both groups of TB before using them.
0000     35 ;
0000     36 ; 00-03 Bill Landry                     6-FEB-1980
0000     37 ;       Fixed my typing errors in test 4C.
0000     38 ;
0000     39 ; 00-04 Bill Landry and Dave Spain      7-FEB-1980
0000     40 ;       Skipped subtest 7 - 10 in test 4C. Also added cache data to PC
0000     41 ;       increment and ADD adder tests.
0000     42 ;
0000     43 ; 00-05 Bill Landry     21-FEB-1980
0000     44 ;       Fixed CMPREGMSK psuedo-ops in TB tests.
0000     45 ;
0000     46 ; 00-06 Bill Landry and Dave Spain      27-FEB-1980
0000     47 ;       Changed a CLRTB.VA_VA macro in test 4C to CLRTB.VA_R[]. Also added
0000     48 ;       cache validates to test 12 to prevent random cache parity errors.
0000     49 ;
0000     50 ; 00-07 Bill Landry     3-MAR-1980
0000     51 ;       Added changes for revised SKIP pseudo op.
0000     52 ;
0000     53 ; 00-08 Bill Landry     13-MAR-1980
0000     54 ;       Fixed tests that didnot use a CMPREGMSK pseudo op for testing the
0000     55 ;       CSTRP register.
0000     56 ; 00-09 Bill Landry     25-MAR-1980
0000     57 ;       Added cache tests.
0000     58 ;
0000     59 ; 01-10 Dave Spain       4-APR-1980
0000     60 ;       Added IRD tests to official release version.
0000     61 ;
0000     62 ; 01-11 Bill Landry     4-APR-1980
0000     63 ;       Moved XB, PREFETCH and IRD tests to ECKAB.SRC so manufacturing can
0000     64 ;       run one tape for DPM.
0000     65 ;
0000     66 ; 00-12 Bill Landry     4-APR-1980
0000     67 ;       Let's save version 1.0 for our first SDC release
0000     68 ;
0000     69 ; 00-13 Bill Landry     8-APR-1980
0000     70 ;       Revised cache tests to survive parity error traps.
0000     71 ;
0000     72 ; 00-14 Bill Landry and Dave Spain      15-APR-1980
0000     73 ;       Reassembled using new COMETDEF.
0000     74 ;
0000     75 ; 00-15 Dave Spain      18-APR-1980
0000     76 ;       Changed tests 4F and 50 to call out the ACV and UTR gate arrays instead
0000     77 ;       of the MIC module.
0000     78 ;
```

```
0000     79 ; 00-16 Bill Landry     1-MAY-1980
0000     80 ;       Revised TB tests to loop on error nicely.
0000     81 ;       Deleted redundent TB tests.
0000     82 ;
0000     83 ; 00-17 Bill Landry     14-MAY-1980
0000     84 ;       Added TB parity error tests
0000     85 ;
0000     86 ; 00-18 Bill Landry     10-JUN-1980
0000     87 ;       Added RDM tests to beginning of tape.
0000     88 ;
0000     89 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     90 ;       Added diagnostic macro file to title section and submitted to
0000     91 ;       RELEASE ENGINEERING.
0000     92 ;
0000     93 ; 01-01 Dave Spain      19-JUN-1980
0000     94 ;       Fixed Prefetch Incrementer test to clear out cache data after
0000     95 ;       every prefetch so that old data from cache will not cause test
0000     96 ;       to mistake bad addresses for good ones.
0000     97 ;
0000     98 ; 01-02 Bill Landry     26-JUN-1980
0000     99 ;       Added tests that check XB trap conditions
0000    100 ;
0000    101 ; 01-03 Bill Landry     30-JUN-1980
0000    102 ;       Added tests that check XB access violations
0000    103 ;
0000    104 ; 01-04 Dave Spain      11-JUL-1980
0000    105 ;       Changed location of BEGINSA pseudo-op to fix signature analysis
0000    106 ;       problem in PC_PC+WBUS test.
0000    107 ;
0000    108 ; 01-05 Bill Landry     8-AUG-1980
0000    109 ;       Added CMI tests in honor of my vacation.
0000    110 ;
0000    111 ; 01-06 Bill Landry     25-AUG-1980
0000    112 ;       Remodeled RDM to CMI tests.
0000    113 ;
0000    114 ; 01-07 Bill Landry     23-SEP-1980
0000    115 ;       Added byte mask testing and improved testing of first memory array.
0000    116 ;
0000    117 ; 02-00 Bill Landry     29-SEP-1980
0000    118 ;       Second release to SDC
0000    119 ;
0000    120 ; 02-01 Bill Landry     5-NOV-1980
0000    121 ;       Added miscellaneous enhancements
0000    122 ;
0000    123 ; 03-00 Bill Landry     17-NOV-1980
0000    124 ;       Third release to SDC
0000    125 ;
0000    126 ; 03-01 Bill Landry     24-NOV-1980
0000    127 ;       Added tests for cache parity logic
0000    128 ;
0000    129 ; 03-02 Bill Landry     30-DEC-1980
0000    130 ;       Separated RDM and MIC tests into two files
0000    131 ;
0000    132 ; 04-00 Bill Landry     05-JAN-1981
0000    133 ;       Submitted to RELEASE ENGINEERING
```

```
0000    134 ;
0000    135 ; 04-01 Bill Landry     14-JAN-1981
0000    136 ;       Revised ECC test to fix bug caused by ECO to INT PEND logic.
0000    137 ;
0000    138 ; 04-02 Bill Landry     27-FEB-1981
0000    139 ;       Udated MDR test to include branching logic.
0000    140 ;       Fixed PC_* tests to fix bug found by field service.
0000    141 ;
0000    142 ; 05-00 Bill Landry     2-MAR-1981
0000    143 ;       Added WRITE BUS ERROR INTERRUPT test
0000    144 ;       Submitted to RELEASE ENGINEERING
0000    145 ;
0000    146 ; 05-01 Bill Landry     18-MAR-1981
0000    147 ;       Made minor changes to various tests as suggested by BT manufacturing.
0000    148 ;
0000    149 ; 05-02 Bill Landry     8-JUN-1981
0000    150 ;       Fixed bug found by Paul Magnet in TB TAG PARITY TESTS.
0000    151 ;
0000    152 ; 05-03 Bill Landry     20-AUG-1981
0000    153 ;       Fixed bugs in memory tests caused by eco to interrupt pending logic.
0000    154 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    155 ;       logic.
0000    156 ;       Replaced all references to COMET with the word CPU.
0000    157 ;       Fixed bug in CACHE TAG MEMORY DUAL ADDRESSING TEST.
0000    158 ;       Submitted to RELEASE ENGINEERING.
0000    159 ;
0000    160 ; 05-04 Bill Landry     4-NOV-1981
0000    161 ;       Moved WCS test from DPM tape to this tape
0000    162 ;       Submitted to RELEASE ENGINEERING.
0000    163 ;
0000    164 ; 05-05 Bill Landry     10-NOV-1981
0000    165 ;       Changed all references of CLRTB.VA_WB to CLRTB.VA_VA.  This fixes a
0000    166 ;       problem caused by plugging in the FPA.
0000    167 ;       Re-released to SDC. Stopped release of 5.4
0000    168 ;
0000    169 ; 05-06 Bill Landry     6-JAN-1982
0000    170 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    171 ;       logic (same as 5.3 above).
0000    172 ;
0000    173 ; 05-07 Bill Landry     23-FEB-1982
0000    174 ;       Fixed more bugs in single bit error tests found by PAUL NATUSH.
0000    175 ;       Same as revision 5.3.
0000    176 ; 05-08 Dave Shull      9-Mar-1982
0000    177 ;       Changed the the IFERROR callouts so when L0011 (MC0) is encountered
0000    178 ;       L0016 (MC1) will be included, when M8728 (MA0) is encountered M8750
0000    179 ;       (MA1) will be included, and when the MAP gate array is encountered
0000    180 ;       MAD gate array will be included.
0000    181 ; 05-09 Dave Shull      10-Mar-1982
0000    182 ;       Added a test to check MS750 controller CSR2 and determine the
0000    183 ;       controller type (ie., L0011 or L0016) and to also print a configuration
0000    184 ;       map of the array's that are installed.
0000    185 ; 06-00 Dave Shull      22-mar-1982
0000    186 ;       Changed the 4 existing memory tests to check for array 0 type then
0000    187 ;       adjust addresses accordingly and test all of the first array versus
0000    188 ;       only testing the first 256kb.
```

```
0000    189 ;       Added 4 new tests that will check to see if array 1 is present and if
0000    190 ;       it is then determine the size and test accordingly.  It looks at both
0000    191 ;       array 0 and array 1, array 0 to determine the starting address of array
0000    192 ;       1 and array 1 to determine the starting and ending address to be tested
0000    193 ; 06-01 Dave Shull      26-Mar-1982
0000    194 ;       Changed READ.SEC MICRO ORDER test so the test could ran in an infinite
0000    195 ;       loop by itself.  Added code to reload Cache everytime the test is
0000    196 ;       restarted in any of the test that are using cache data that could get
0000    197 ;       blown away.
0000    198 ; 06-02 Dave Shull      05-Apr-1982
0000    199 ;       Removed all of the BEGINSA and ENDSA pseudo's
0000    200 ;       Submitted to RELEASE ENGINEERING
0000    201 ; 07-00 Dave Shull      10-May-1982
0000    202 ;       Added new test ( name <Test PCBACK While PC Increment Logic is Active>)
0000    203 ;       to check for side-affects on PCBACK register while incrementin the PC
0000    204 ;       through the execution buffer.
0000    205 ; 07-01 Dave Shull      14-May-1982
0000    206 ;       Changed all CLRTB.VA_VA macro's to prevent translation buffer
0000    207 ;       invalidate failures when manufacturing runs the micro's with an
0000    208 ;       external clock set to fast margins.
0000    209 ; 07-02 Dave Shull      17-May-1982
0000    210 ;       Added coverage in the Read Lock Function Test for the CML gate array.
0000    211 ; 07-03 Dave Shull      14-Jun-1982
0000    212 ;       Changed CMK callouts to CML for new gate array.
0000    213 ; 08-00 Joe Zagarella   11-Feb-1986
0000    214 ;       Enhanced the following tests to be compatible with the new memory
0000    215 ;       controller(L0022) and array card(M7199):
0000    216 ;       Test write bus error interrupt
0000    217 ;       Test for controller type and array configuration
0000    218 ;       Test data integrity for address test
0000    219 ;       Main memory dual addressing test
0000    220 ;       Array 0 memory test (part1 thru part4)
0000    221 ;       Array 1 memory test (part1 thru part4)
0000    222 ;
0000    223 ;--
0000      1         .SBTTL  ""
```

```
0000     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000     4 ;++
0000     5 ; Revision History
0000     6 ;
0000     7 ; 28    Dave Spain      22-Jun-82
0000     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     9 ;       Decided not to use them.
0000    10 ;
0000    11 ; 27    Dave Spain      15-Jun-82
0000    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000    13 ;
0000    14 ; 26    Dave Spain      19-May-82
0000    15 ;       Added M[]_FPA macro.
0000    16 ;
0000    17 ; 25    Rich Lewis      23-Feb-82
0000    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000    20 ;
0000    21 ; 24    Dave Spain      02-Feb-82
0000    22 ;       Added FPA_R[]+M[] macro.
0000    23 ;
0000    24 ; 23    Bill Landry     22-OCT-81
0000    25 ;       Added PSL_RDM macro
0000    26 ;
0000    27 ; 22    Dave Spain       5 Oct 81
0000    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000    29 ;
0000    30 ; 21    Bill Landry     29 Sept 81
0000    31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000    32 ;
0000    33 ; 20    Dave Spain      14 Aug 81
0000    34 ;       Added the TESTFPA [] macro.
0000    35 ;
0000    36 ; 19    Dave Spain      18 July 81
0000    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000    38 ;
0000    39 ; 18    Dave Spain      24 Sept 80
0000    40 ;       Added RDM_Q macro. This macro uses the ALU.
0000    41 ;
0000    42 ; 17    Bill Landry     23 Sept 80
0000    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000    44 ;
0000    45 ; 16    Dave Spain      27 August 80
0000    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000    47 ;
0000    48 ; 15    Bill Landry     4 June 80
0000    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000    50 ;       satisfy validity checks.
0000    51 ;
0000    52 ; 14    Dave Spain      23 May 80
0000    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000    54 ;       Bill.
0000    55 ;
0000    56 ; 13    Dave Spain      14 Apr 80
0000    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

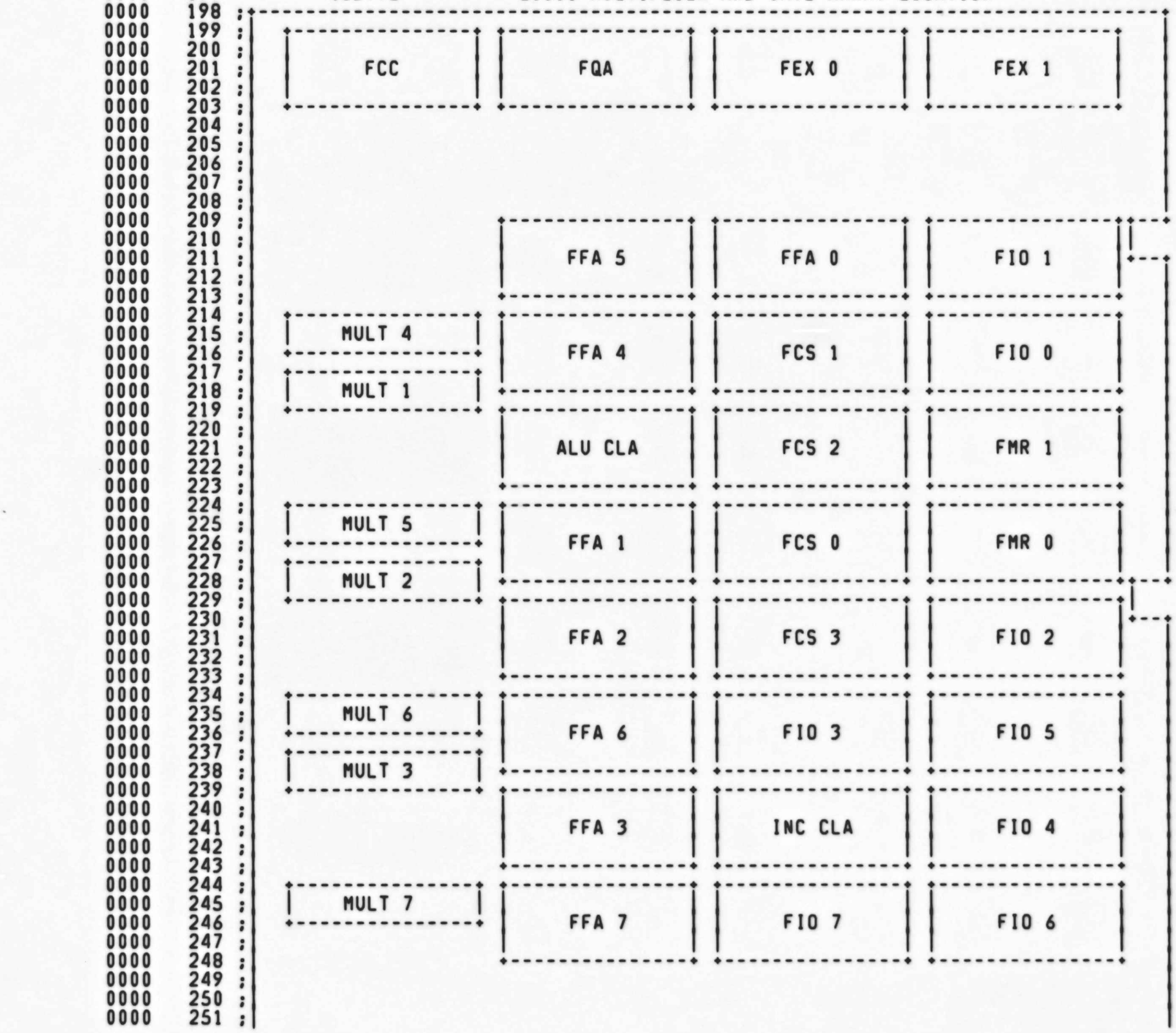
```
                     0000     58 ;       the COMET micro-code defines to run.
                     0000     59 ;
                     0000     60 ; 12    Bill Landry      1 Apr 80
                     0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
                     0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
                     0000     63 ;
                     0000     64 ; 11    Dave Spain      27 Mar 80
                     0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
                     0000     66 ;
                     0000     67 ; 10    Bill Landry     19 Mar 80
                     0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
                     0000     69 ;
                     0000     70 ; 09    Bill Landry     19 Mar 80
                     0000     71 ;       Deleted TB_R[].RL.P.OR.D
                     0000     72 ;
                     0000     73 ; 08    Bill Landry     18 Mar 80
                     0000     74 ;       Added VA_R[].RL.P
                     0000     75 ;
                     0000     76 ; 07    Bill Landry     07 Mar 80
                     0000     77 ;       Added Q_M[].RL.PTE
                     0000     78 ;
                     0000     79 ; 06    Dave Spain      06 Mar 80
                     0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
                     0000     81 ;
                     0000     82 ; 05    Dave Spain      03 Mar 80
                     0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
                     0000     84 ;
                     0000     85 ; 04    Dave Spain      22 Feb 80
                     0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
                     0000     87 ;
                     0000     88 ; 03    Dave Spain      30 Jan 80
                     0000     89 ;       Added Tom Groetzinger's macro definitions.
                     0000     90 ;
                     0000     91 ; 02    Dave Spain      30 Jan 80
                     0000     92 ;       Added Tony Vezza's macro definitions.
                     0000     93 ;
                     0000     94 ; 01    Dave Spain      29 Jan 80
                     0000     95 ;       Macros initiated.
                     0000     96 ;--
                     0000     97
                     0000     98          .SBTTL  "       RDM Control File Macros"
                     0000     99
                     0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
                     0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
                     0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
                     0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
                     0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
                     0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
                     0000    106 ;RDM_CMI                          "RDMCF6/STROBE.CMI"
                     0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
                     0000    108
                     0000    109          .SBTTL  "       Diagnostic Macros"
                     0000    110
                     0000    111 ;CLOCK.EXT                        "CLKX/XTND"
                     0000    112 ;CLRCH.VA_RDM                     "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
                0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_ -
cont> WB"
                0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   115 ;                                        BUS/PRB.RD.PTE"
                0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   118 ;                                        BUS/PRB.RD.PTE"
                0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_ -
cont> WB"
                0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
                0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
                0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
                0000   124 ;                                        WCTRL/VA_WB"
                0000   125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
                0000   126 ;FPA_M[] FPA_R[]                        "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX -
cont> /R.S,
                0000   127 ;                                        ROT/ZERO,ALU/OR"
                0000   128 ;FPA_RDM                                "WB_RDM,FPA/FPA_DATA.WBUS"
                0000   129 ;FPA_R[]                                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_D -
cont> ATA.WBUS"
                0000   130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FP -
cont> A/FPA_DATA.WBUS"
                0000   131 ;FPA_WB                         "FPA/FPA_DATA.WBUS"
                0000   132 ;MEMSCAR_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
                0000   133 ;M[]_FPA                                "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
                0000   134 ;M[]_LONLIT                     "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
                0000   135 ;M[]_R[].OR.NIB0(M)             "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
                0000   136 ;                                        MUX/R.S,DQ1/NOP"
                0000   137 ;M[]_RDM                                "WB_RDM,SPW/MLONG,MSRC/a1"
                0000   138 ;PC_PC+RDM                      "WCTRL/PC_PC+WB,WB_RDM"
                0000   139 ;PC_PC+4 MB_XB                  "MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
                0000   140 ;PC_PC+4 RDM_XB                 "RSRC/ZERO,MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,
                0000   141 ;                                        MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
                0000   142 ;PC_RDM                         "WB_RDM,WCTRL/PC_WB"
                0000   143 ;PSL_RDM                                "CCPSL/PSL_WB,CCBR_ALUS,WB_RDM"
                0000   144 ;PSL_WB                         "CCPSL/PSL_WB,CCBR_ALUS"
                0000   145 ;Q_D WX_R                       "ALPCTL/WX_R.Q_D"
                0000   146 ;Q_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
                0000   147 ;Q_M[].RL.PTE                   "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
                0000   148 ;Q_R[].AND.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
                0000   149 ;Q_R[].ASL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
                0000   150 ;Q_R[].OR.Q                     "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
                0000   151 ;Q_R[].RL.P                     "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
                0000   152 ;RDM_ALUF                       "ALPCTL/WB_ALUF,RDM_WB"
                0000   153 ;RDM_FPA                                "FPA/WBUS_FPA,RDM_WB"
                0000   154 ;RDM_LONLIT                     "WB_LONLIT,RDM_WB"
                0000   155 ;RDM_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
                0000   156 ;RDM_M[]                                "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
                0000   157 ;RDM_M[].OR.R[]                 "WB_M[a1].OR.R[a2],RDM_WB"
                0000   158 ;RDM_M[]-R[]                    "WB_M[a1]-R[a2],RDM_WB"
                0000   159 ;RDM_M[].XZ.OR.R[]              "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                0000   160 ;                               RDM_WB"
                0000   161 ;RDM_PC                         "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
                0000   162 ;RDM_PCBACK                     "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
                0000   163 ;RDM_Q                          "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
                0000   164 ;RDM_Q Q_D                      "ALPCTL/WX_Q.Q_D,RDM_WB"
```

```
0000   165 ;RDM_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                     "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                     "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

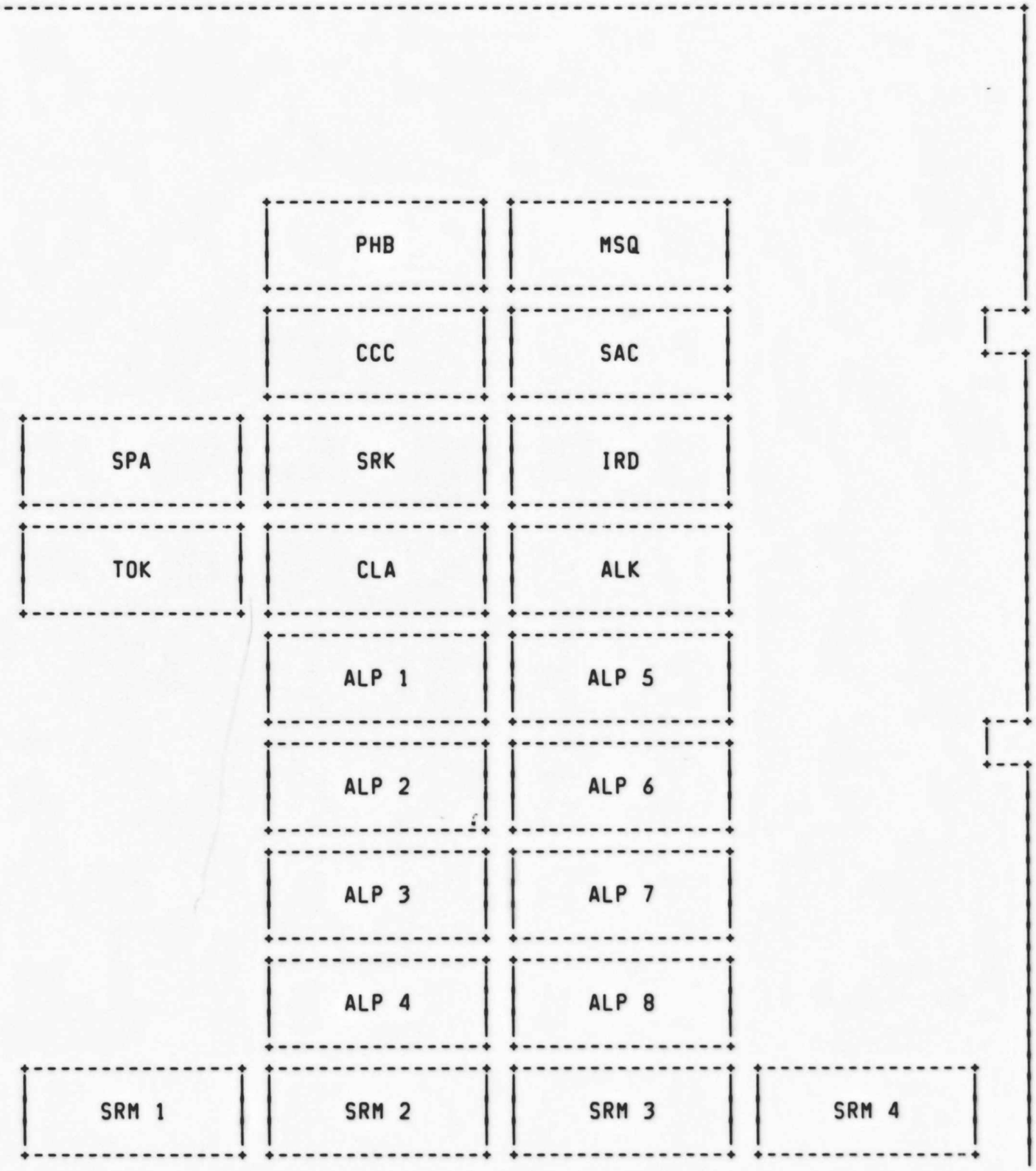
```
                0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
                0000   169 ;R[]_RDM                                "WB_RDM,SPW/RLONG,RSRC/a1"
                0000   170 ;R[].DT_Q                        "RSRC/a1,D_Q_Q_D,SPW/RSIZE,VSIZE/1"
                0000   171 ;R[]_VA                         "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
                0000   172 ;R[]_WB                         "RSRC/a1,SPW/RLONG"
                0000   173 ;TB_RDM                         "WCTRL/TB_WB,WB_RDM,MSRC/VA"
                0000   174 ;TB_WB                          "WCTRL/TB_WB,MSRC/VA"
                0000   175 ;TESTFPA []                     "TESTFPA/a1"
                0000   176 ;VA_PC+LONLIT+I.PC_PC+I         "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/ -
cont> OR,
                0000   177 ;                                       ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
                0000   178 ;                                       SIZE[LONG]"
                0000   179 ;VA_Q.OR.R[]                    "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
                0000   180 ;VA_RDM                         "WB_RDM,WCTRL/VA_WB"
                0000   181 ;VA_R[].RL.P                    "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
                0000   182 ;VA_VA-ZLIT0[]                  "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
                0000   183 ;                                       MUX/M.S,ALU/A-B-CI"
                0000   184 ;VA_WB                          "WCTRL/VA_WB"
                0000   185 ;WB_FPA CCR                     "FPA/WBUS_FPA.CC"
                0000   186 ;WB_FPA TCR                     "WCTRL/FPTCR"
                0000   187 ;WB_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000   188 ;WB_LONLIT.Q_M                  "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000   189 ;WB_M[]+R[]                     "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
                0000   190 ;WB_R[].Q_D                     "RSRC/a1,ALPCTL/WX_R.Q_D"
                0000   191 ;WDR_R[]                                "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/Z -
cont> ERO"
                0000   192 ;WRITE.LONG R[]                  "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
                0000   193 ;                                       MUX/R.S,ALU/OR,SIZE[LONG]"
                0000   194          .SBTTL  ""
                0000   195          .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                        .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```



```
0000   252 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0002 GATE ARRAY LOCATION"
```

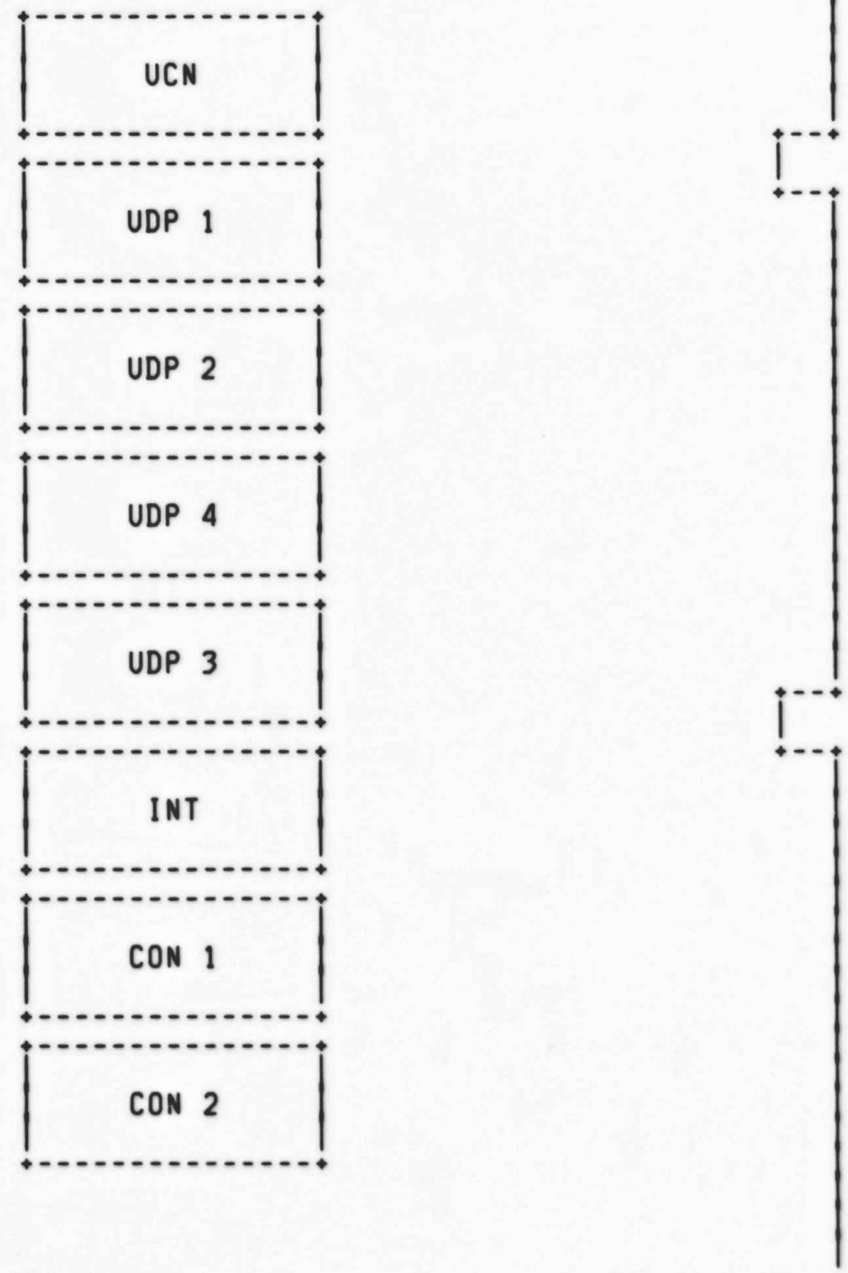


```
                0000   309 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0003 GATE ARRAY LOCATION"
;+-------------------------------------------------------------------------------
;|
;|
;|
```

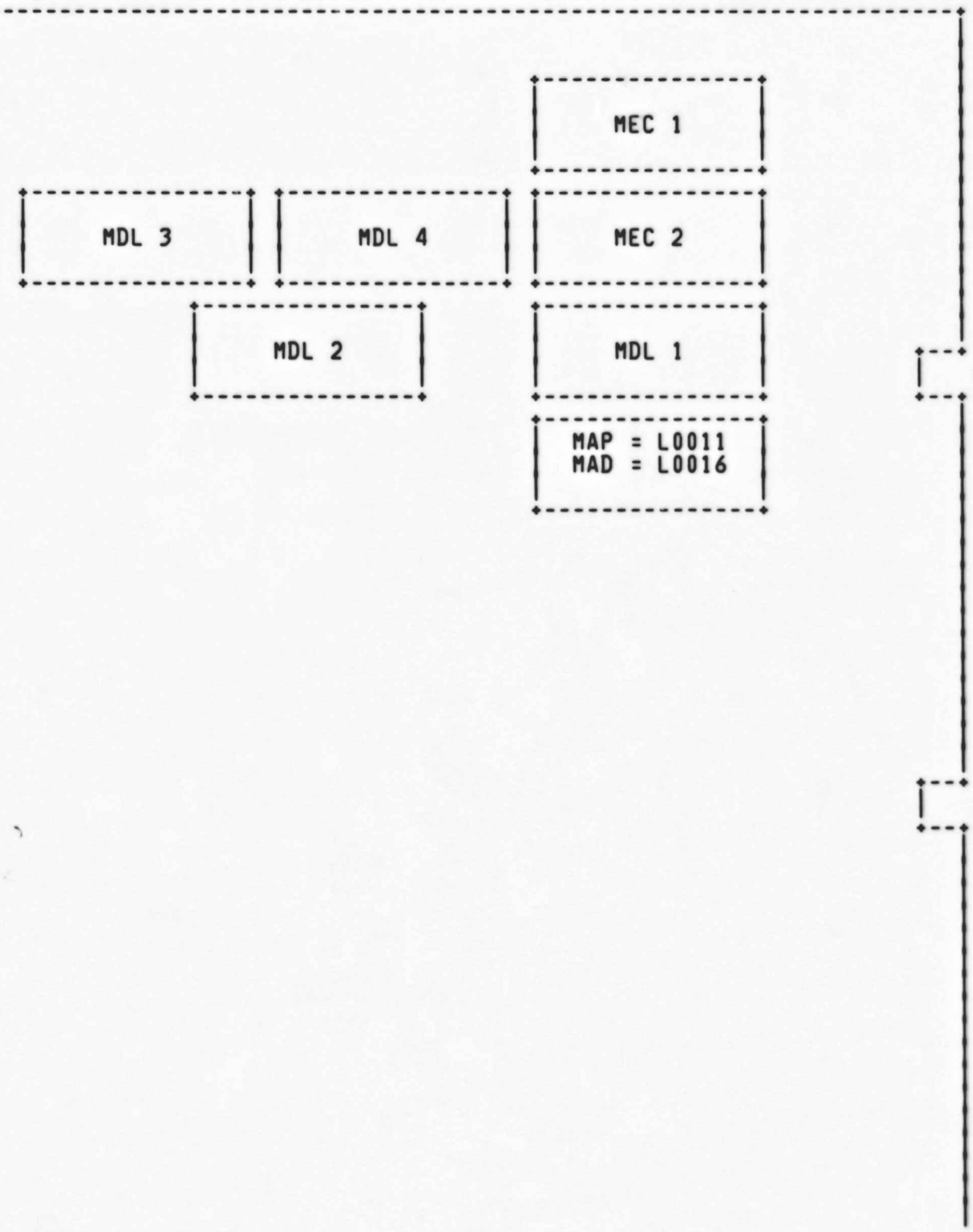
```
0000   366 ;|              +---------------+ +---------------+ +---------------+          |
0000   367 ;+-------------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0004 GATE ARRAY LOCATION"
```



```
0000   424 ;|                                                                    |
0000   425 ;+-------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
```



```
ZZ-ECKAC-8.0    V08.00                  " L0011 & L0016 17-FEB-1986      Fiche 1  Frame I7        Sequence 86
ECKAC                       VAX-11/750 MIC MICRO DIAGNOSTICS          17-FEB-1986 13:38:57  VAX/VMS Macro V04-00        Page  1
V08.00                      " L0011 & L0016 GATE ARRAY LOCATION"      14-AUG-1984 15:55:16  [VAX750]COMETASM.MAR;221           (1

                         0000    482 ;|                                                                                  |
                         0000    483 ;+----------------------------------------------------------------------------------+
```

```
                     0000    485          .SBTTL  ""
                     0000    486          .LIST   MC
                     0000    487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                     0000    488          .LIST   ME
                     0000    489          .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
00002C5E  0000              MONITOR_SIZE    =       ^X2C5E
00000400  0000              STACK_SIZE      =       1024
000007A2  0000              M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                        ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000              M$TEST_SIZE_D   = M$TEST_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    491       .NLIST  ME
          0000    492       .LIST   MC
          0000    493       .SBTTL  ""
          0000    494       .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
                        .SBTTL  "TEST   001     TEST ADK S/C REGISTERS
                    3 ;*******************************************************************************
                    4 ;++
                    5 ;
                    6 ; FUNCTIONAL DESCRIPTION:
                    7 ;
                    8 ;       THIS TEST WILL CONSIST OF 4 SUBTESTS TO CHECK:
                    9 ;               1.  PHYSICAL/VIRTUAL ADDRESSING REGISTER
                   10 ;               2.  SAVED MODE REGISTER
                   11 ;               3.  WRITE VECTOR OCCURRED REGISTER
                   12 ;               4.  TB GROUP DISABLE REGISTER
                   13 ;       THESE ARE ALL READ/WRITE REGISTERS IN THE ADK CHIP.
                   14 ;
                   15 ;
                   16 ; TEST ALGORITHM:
                   17 ;
                   18 ;       1.  IN SUBTESTS 2, 3 AND 4 MODIFY RSRC FIELD IN DCS ADDRESS 1
                   19 ;           TO SELECT PROPER TEMPORARY REGISTER
                   20 ;       2.  SET UP A LOOP TO SELECT TEST PATTERNS
                   21 ;       3.  LOAD D REGISTER (RDM) WITH TEST PATTERN
                   22 ;       4.  EXECUTE DCS PROGRAM
                   23 ;               1.  LOAD MEMORY STATUS AND CONTROL ADDRESS REGISTER
                   24 ;                   (MEMSCAR) WITH ADDRESS IN TEMPORARY REGISTER
                   25 ;               2.  WRITE TEST PATTERN IN D REGISTER (RDM) TO MEMORY
                   26 ;                   STATUS AND CONTROL REGISTER (MEMSCR)
                   27 ;               3.  READ TEST PATTERN BACK TO H REGISTER (RDM)
                   28 ;       5.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                   29 ;       6.  IF NO ERROR GO TO STEP 3 FOR REMAINING TEST PATTERNS
                   30 ;
                   31 ;
                   32 ; TEST PATTERNS:
                   33 ;
                   34 ;       SUBTEST 1 AND 3:
                   35 ;                           ITERATION       TEST PATTERN
                   36 ;                               1               1
                   37 ;                               2               0
                   38 ;
                   39 ;       SUBTEST 2 AND 4:
                   40 ;                           ITERATION       TEST PATTERN
                   41 ;                               1               A
                   42 ;                               2               5
                   43 ;                               3               3
                   44 ;
                   45 ;
                   46 ; LOGIC DESCRIPTION:
                   47 ;
                   48 ;       THIS TEST IS DESIGNED TO CHECK ONLY THE S/C REGISTERS THAT
                   49 ;       LIVE IN THE ADK GATE ARRAY.  THE REGISTERS ARE ACCESSED VIA
                   50 ;       WBUS<27:24>.  IF ON ERROR, REPLACING THE ADK DOES NOT WORK,
                   51 ;       THEN YOU MIGHT TRY RUNNING THE MEMSCAR TEST WHICH WILL CHECK
                   52 ;       FOR A MULTIPLE ADDRESSING PROBLEM.  IF THAT FAILS THEN ONE
                   53 ;       OF THE OTHER GATE ARRAYS WITH S/C REGISTERS IS GIVING YOU
                   54 ;       TROUBLE.
                   55 ;
                   56 ;
```

```
                     ; ASSUMPTIONS:
                     ;
                     ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
                     ;
                     ;
                     ; ERROR DESCRIPTION:
                     ;
                     ;       EXPECTED MEMSCR DATA
                     ;       RECEIVED MEMSCR DATA
                     ;       LOOP COUNT
                     ;
                     ;--
                     ;*****************************************************************************

                             SUBTEST                                         ;PHYSICAL/VIRTUAL ADD REG

                     ;/////////////////////////////////////////////////////////////////////////////
                     ;+
                     ;SUBTEST #1 PHYSICAL/VIRTUAL ADDRESS REGISTER
                     ;-
                             INITIALIZE                                      ;INIT CPU
                             LDFIELD         8$,,B,3$,34,39                  ;MODIFY U-WORD TO RSRC/TEMP0
                             LOADDCS         3$,,01,1                        ;CHANGE DCS ADDRESS 01
                             LOOP            I,1,2                           ;CREATE TEST LOOP
                             LOADREG         WDREG,1$,I,L                    ;TEST PATTERN TO D REG
                             ERRLOOP                                         ;ERROR LOOP
                             FETCH           0                               ;START DCS PROGRAM
                             BRSTCLK         4                               ;END DCS PROGRAM
                             CMPREGMSK       WHREG,1$,I,7$,,L,               ;COMPARE RESULTS
                             IFERROR         ,<<ADK>>,                       ;POSSIBLE FAILING ARRAY
                             ENDLOOP         I                               ;END TEST LOOP

                             SUBTEST                                         ;SAVED MODE REGISTER

                     ;/////////////////////////////////////////////////////////////////////////////
                     ;+
                     ;SUBTEST #2 SAVED MODE REGISTER
                     ;-
                             INITIALIZE                                      ;INIT CPU
                             LDFIELD         4$,,B,3$,34,39                  ;MODIFY U-WORD TO RSRC/TEMP1
                             LOADDCS         3$,,01,1                        ;CHANGE DCS ADDRESS 1
                             LOOP            I,1,3                           ;CREATE TEST LOOP
                             LOADREG         WDREG,2$,I,L                    ;TEST PATTERN TO D REG
                             ERRLOOP                                         ;ERROR LOOP
                             FETCH           0                               ;START DCS PROGRAM
                             BRSTCLK         4                               ;END DCS PROGRAM
                             CMPREGMSK       WHREG,2$,I,7$,,L,               ;COMPARE RESULTS
                             IFERROR         ,<<ADK>>,                       ;POSSIBLE FAILING ARRAY
                             ENDLOOP         I                               ;END TEST LOOP

                             SUBTEST                                         ;WRITE VECTOR OCCURRED REG

                     ;/////////////////////////////////////////////////////////////////////////////
                     ;+
                     ;SUBTEST #3 WRITE VECTOR OCCURRED REGISTER
```

```
                00B6   106 ;-
                00B6   107         INITIALIZE                                  ;INIT CPU
                00B8   108         LDFIELD         5$,,B,3$,34,39              ;MODIFY U-WORD TO RSRC/TEMP2
                00C9   109         LOADDCS         3$,,01,1                    ;CHANGE DCS ADDRESS 1
                00D0   110         LOOP            I,1,2                       ;CREATE TEST LOOP
                00D7   111         LOADREG         WDREG,1$,I,L                ;TEST PATTERN TO D REG
                00DF   112         ERRLOOP                                     ;ERROR LOOP
                00E1   113         FETCH           0                           ;START DCS PROGRAM
                00E6   114         BRSTCLK         4                           ;END DCS PROGRAM
                00EA   115         CMPREGMSK       WHREG,1$,I,7$,,L,           ;COMPARE RESULTS
                00F6   116         IFERROR         ,<<ADK>>,                   ;POSSIBLE FAILING ARRAY
                00FC   117         ENDLOOP         I                           ;END TEST LOOP
                00FF   118
                00FF   119         SUBTEST                                     ;TB GROUP DISABLE REG
                0102
                0102               ;////////////////////////////////////////////////////////////////////////////////
                0102   120 ;+
                0102   121 ;SUBTEST #4 TB GROUP DISABLE REGISTER
                0102   122 ;-
                0102   123         INITIALIZE                                  ;INIT CPU
                0104   124         LDFIELD         6$,,B,3$,34,39              ;MODIFY U-WORD TO RSRC/TEMP3
                0115   125         LOADDCS         3$,,01,1                    ;CHANGE DCS ADDRESS 1
                011C   126         LOOP            I,1,3                       ;CREATE TEST LOOP
                0123   127         LOADREG         WDREG,2$,I,L                ;TEST PATTERN TO D REG
                012B   128         ERRLOOP                                     ;ERROR LOOP
                012D   129         FETCH           0                           ;START DCS PROGRAM
                0132   130         BRSTCLK         4                           ;END DCS PROGRAM
                0136   131         CMPREGMSK       WHREG,2$,I,7$,,L,           ;COMPARE RESULTS
                0142   132         IFERROR         ,<<ADK>>,                   ;POSSIBLE FAILING ARRAY
                0148   133         ENDLOOP         I                           ;END TEST LOOP
                014B   134         SKIP
                0152   135
                0152   136 BEGIN_TEST_DATA
                0152
                0152               ;-------------------------------------------------------------------------------
                0152   137
    01000000    0152   138 1$:      .LONG   ^X01000000                          ;TEST PATTERNS
    00000000    0156   139         .LONG   ^X00000000                          ;  "       "
    0A000000    015A   140 2$:      .LONG   ^X0A000000                          ;  "       "
    05000000    015E   141         .LONG   ^X05000000                          ;  "       "
    03000000    0162   142         .LONG   ^X03000000                          ;
                0166   143
                0166   144 3$:                                                 ;USED TO UPDATE RSRC FIELD IN
U 01, 7700,C800,5BE4,0300,6870,1802 0166   145 ;% 01   MEMSCAR_R[TEMP0]        ;DCS ADDRESS 01
                0171   149
          01    0171   150 4$:      .BYTE   ^X01                                ;ADDRESS OF RTEMP 01
          02    0172   151 5$:      .BYTE   ^X02                                ;ADDRESS OF RTEMP 02
          03    0173   152 6$:      .BYTE   ^X03                                ;ADDRESS OF RTEMP 03
    0F000000    0174   153 7$:      .LONG   ^X0F000000                          ;MASK FOR CMPREGMSK PSEUDO OP
          00    0178   154 8$:      .BYTE   ^X00                                ;ADDRESS OF RTEMP 00
                0179   155
                0179   156 BEGIN_CPU_DATA
                0002   157
                0002   158 DCS_DATA
                0001   159
```

```
ZZ-ECKAC-8.0    V08.00                    "TEST 001 TEST A17-FEB-1986      Fiche 1  Frame D8        Sequence 94
ECKAC                        VAX-11/750 MIC MICRO DIAGNOSTICS          17-FEB-1986 13:38:57  VAX/VMS Macro V04-00        Page  2
V08.00                       "TEST 001 TEST ADK S/C REGISTERS          17-FEB-1986 13:38:39  [VAX750.MIC]ECKAC1.MAR;1          (1

U 00, 7700,C800,0364,0300,0470,1801   0001    160 ;* 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C    164 ;* 01    MEMSCAR_R[TEMP0]                          ;ADDRESS OF MEMSCR TO MEMSCAR
U 02, 5700,C800,0364,0300,6070,1803   0017    168 ;* 02    WB_RDM,WCTRL/MEMSCR_WB                    ;WRITE TEST PATTERN TO MEMSCR
U 03, 6700,C800,0364,0300,6470,1804   0022    172 ;* 03    RDM_WB,WCTRL/MEMSCR                       ;READ TEST PATTERN BACK TO RDM
U 04, 7300,4800,0364,0300,0470,1805   002D    176 ;* 04    NOP,STOP.CLOCK                            ;STOP CPU CLOCK
                                      0038    180
                                      0038    181 RTEMP_DATA
                                      0001    182
                            00000000  0001    183         .LONG   ^X00000000                        ;PHYSICAL/VIRTUAL REG ADDRESS
                            01000000  0005    184         .LONG   ^X01000000                        ;SAVED MODE REGISTER ADDRESS
                            02000000  0009    185         .LONG   ^X02000000                        ;WRITE VECTOR OCC REG ADDRESS
                            03000000  000D    186         .LONG   ^X03000000                        ;TB GROUP DISABLE REG ADDRESS
                                      0011    187
                                      0011    188 END_CPU_DATA
                                      0179    189
                                      0179    190 END_TEST_DATA
                                      0179
                                      0179        ;-------------------------------------------------------------------------
                                      0179    191
                                      0179    192 ENDTEST
```

```
001B        .SBTTL  "TEST   002     TEST CAK S/C REGISTERS
001B    194 ;*****************************************************************************
001B    195 ;++
001B    196 ;
001B    197 ; FUNCTIONAL DESCRIPTION:
001B    198 ;
001B    199 ;       THIS TEST WILL CONSIST OF 2 SUBTESTS TO CHECK:
001B    200 ;               1.  CACHE ERROR REGISTER
001B    201 ;               2.  CACHE CONTROL REGISTER
001B    202 ;       THESE ARE ALL READ/WRITE REGISTERS IN THE CAK CHIP.
001B    203 ;
001B    204 ;
001B    205 ; TEST ALGORITHM:
001B    206 ;
001B    207 ;       1.  IN SUBTEST 2 CHANGE DCS ADDRESS 01 TO SELECT TEMPORARY REGISTER 1
001B    208 ;       2.  SET UP A LOOP TO SELECT TEST PATTERNS
001B    209 ;       3.  LOAD D REGISTER (RDM) WITH TEST PATTERN
001B    210 ;       4.  EXECUTE DCS PROGRAM
001B    211 ;               1.  LOAD MEMORY STATUS AND CONTROL ADDRESS REGISTER
001B    212 ;                   (MEMSCAR) WITH ADDRESS IN TEMPORARY REGISTER
001B    213 ;               2.  WRITE TEST PATTERN IN D REGISTER (RDM) TO MEMORY
001B    214 ;                   STATUS AND CONTROL REGISTER (MEMSCR)
001B    215 ;               3.  READ TEST PATTERN BACK TO H REGISTER (RDM)
001B    216 ;       5.  COMPARE RESULTS (EXPECTED AND RECEIVED)
001B    217 ;       6.  IF NO ERROR GO TO STEP 3 FOR REMAINING TEST PATTERNS
001B    218 ;
001B    219 ;
001B    220 ; TEST PATTERNS
001B    221 ;
001B    222 ;       ITERATION       TEST PATTERN
001B    223 ;           1               A
001B    224 ;           2               5
001B    225 ;           3               3
001B    226 ;
001B    227 ;
001B    228 ; LOGIC DESCRIPTION:
001B    229 ;
001B    230 ;       THIS TEST IS DESIGNED TO CHECK ONLY THE S/C REGISTERS THAT
001B    231 ;       LIVE IN THE CAK GATE ARRAY.  THE REGISTERS ARE ACCESSED VIA
001B    232 ;       WBUS<27:24>.  IF ON ERROR, REPLACING THE CAK DOES NOT WORK,
001B    233 ;       THEN YOU MIGHT TRY RUNNING THE MEMSCAR TEST WHICH WILL CHECK
001B    234 ;       FOR A MULTIPLE ADDRESSING PROBLEM.  IF THAT FAILS THEN ONE
001B    235 ;       OF THE OTHER GATE ARRAYS WITH S/C REGISTERS IS GIVING YOU
001B    236 ;       TROUBLE.
001B    237 ;
001B    238 ;
001B    239 ; ASSUMPTIONS:
001B    240 ;
001B    241 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
001B    242 ;
001B    243 ;
001B    244 ; ERROR DESCRIPTION:
001B    245 ;
001B    246 ;       EXPECTED MEMSCR DATA
001B    247 ;       RECEIVED MEMSCR DATA
```

```
                     001B   248 ;       LOOP COUNT
                     001B   249 ;
                     001B   250 ;--
                     001B   251 ;*******************************************************************************
                     001B   252
                     001B   253         SUBTEST                                         ;CACHE ERROR REG
                     001E
                     001E       ;////////////////////////////////////////////////////////////////////////////
                     001E   254 ;+
                     001E   255 ;SUBTEST #1 CACHE ERROR REGISTER
                     001E   256 ;-
                     001E   257         INITIALIZE                                      ;INIT CPU
                     0020   258         LOADDCS         4$,,01,1                        ;MODIFY DCS ADDRESS 01
                     0027   259         LOOP            I,1,3                           ;CREATE TEST LOOP
                     002E   260         LOADREG         WDREG,1$,I,L                    ;TEST PATTERN TO D REG
                     0036   261         ERRLOOP                                         ;ERROR LOOP
                     0038   262         FETCH           0                               ;START DCS PROGRAM
                     003D   263         BRSTCLK         4                               ;END DCS PROGRAM
                     0041   264         CMPREGMSK       WHREG,1$,I,3$,,L,               ;COMPARE RESULTS
                     004D   265         IFERROR         ,<<CAK>>,                       ;POSSIBLE FAILING ARRAY
                     0053   266         ENDLOOP         I                               ;END TEST LOOP
                     0056   267
                     0056   268         SUBTEST                                         ;CACHE CONTROL REG
                     0059
                     0059       ;////////////////////////////////////////////////////////////////////////////
                     0059   269 ;+
                     0059   270 ;SUBTEST #2 CACHE CONTROL REGISTER
                     0059   271 ;-
                     0059   272         INITIALIZE                                      ;INIT CPU
                     005B   273         LOADDCS         2$,,01,1                        ;CHANGE DCS ADDRESS 01
                     0062   274         LOOP            I,1,3                           ;CREATE TEST LOOP
                     0069   275         LOADREG         WDREG,1$,I,L                    ;TEST PATTERN TO D REG
                     0071   276         ERRLOOP                                         ;ERROR LOOP
                     0073   277         FETCH           0                               ;START DCS PROGRAM
                     0078   278         BRSTCLK         4                               ;END DCS PROGRAM
                     007C   279         CMPREGMSK       WHREG,1$,I,3$,,L,               ;COMPARE RESULTS
                     0088   280         IFERROR         ,<<CAK>>,                       ;POSSIBLE FAILING ARRAY
                     008E   281         ENDLOOP         I                               ;END TEST LOOP
                     0091   282         SKIP
                     0098   283
                     0098   284 BEGIN_TEST_DATA
                     0098
                     0098       ;-------------------------------------------------------------------------
                     0098   285
          0A000000   0098   286 1$:     .LONG   ^X0A000000                             ;TEST PATTERNS
          05000000   009C   287         .LONG   ^X05000000                              ;  "       "
          03000000   00A0   288         .LONG   ^X03000000                              ;  "       "
                     00A4   289
                     00A4   290 2$:
U 01, 7700,8800,5BE4,0304,6870,1802  00A4   291 ;% 01    MEMSCAR_R[TEMP1]                ;RTEMP FOR SUBTEST 2
                     00AF   295
          0F000000   00AF   296 3$:     .LONG   ^X0F000000                             ;MASK FOR CMPREGMSK PSEUDO OP
                     00B3   297
                     00B3   298 4$:
U 01, 7700,C800,5BE4,0300,6870,1802  00B3   299 ;% 01    MEMSCAR_R[TEMP0]                ;RTEMP FOR SUBTEST 1
```

```
                                00BE   303
                                00BE   304 BEGIN_CPU_DATA
                                0002   305
                                0002   306 DCS_DATA
                                0001   307
U 00. 7700.C800.0364.0300.0470.1801   0001   308 ;% 00   NOP
U 01. 7700.C800.5BE4.0300.6870.1802   000C   312 ;% 01   MEMSCAR_R[TEMP0]                    ;WRITE ADD OF S/C TO MEMSCAR
U 02. 5700.C800.0364.0300.6070.1803   0017   316 ;% 02   WB_RDM,WCTRL/MEMSCR_WB             ;WRITE TEST PATTERN TO MEMSCR
U 03. 6700.C800.0364.0300.6470.1804   0022   320 ;% 03   RDM_WB,WCTRL/MEMSCR               ;READ TEST PATTERN BACK TO RDM
U 04. 7300.4800.0364.0300.0470.1805   002D   324 ;% 04   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
                                0038   328
                                0038   329 RTEMP_DATA
                                0001   330
                       04000000 0001   331         .LONG   ^X04000000                    ;CACHE ERROR REGISTER ADDRESS
                       06000000 0005   332         .LONG   ^X06000000                    ;CACHE CONTROL REGISTER ADD
                                0009   333
                                0009   334 END_CPU_DATA
                                00BE   335
                                00BE   336 END_TEST_DATA
                                00BE
                                00BE       ;-----------------------------------------------------------------------
                                00BE   337
                                00BE   338 ENDTEST
```

```
                        001B        .SBTTL  "TEST   003     TEST UTR S/C REGISTERS
                        001B    340 ;*******************************************************************************
                        001B    341 ;++
                        001B    342 ;
                        001B    343 ; FUNCTIONAL DESCRIPTION:
                        001B    344 ;
                        001B    345 ;       THIS TEST WILL CONSIST OF 2 SUBTESTS TO CHECK:
                        001B    346 ;               1.  MACHINE CHECK ERROR SUMMARY REGISTER
                        001B    347 ;               2.  REFERENCE CACHE ONLY REGISTER
                        001B    348 ;       THESE ARE ALL READ/WRITE REGISTERS IN THE UTR CHIP.
                        001B    349 ;
                        001B    350 ;
                        001B    351 ; TEST ALGORITHM:
                        001B    352 ;
                        001B    353 ;       1.  IN SUBTEST 2 CHANGE DCS ADDRESS 01 TO SELECT RTEMP1
                        001B    354 ;       2.  SET UP A LOOP TO SELECT TEST PATTERNS
                        001B    355 ;       3.  LOAD D REGISTER (RDM) WITH TEST PATTERN
                        001B    356 ;       4.  EXECUTE DCS PROGRAM
                        001B    357 ;               1.  LOAD MEMORY STATUS AND CONTROL ADDRESS REGISTER
                        001B    358 ;                   (MEMSCAR) WITH ADDRESS IN TEMPORARY REGISTER
                        001B    359 ;               2.  WRITE TEST PATTERN IN D REGISTER (RDM) TO MEMORY
                        001B    360 ;                   STATUS AND CONTROL REGISTER (MEMSCR)
                        001B    361 ;               3.  READ TEST PATTERN BACK TO H REGISTER (RDM)
                        001B    362 ;       5.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                        001B    363 ;       6.  IF NO ERROR GO TO STEP 3 FOR REMAINING TEST PATTERNS
                        001B    364 ;
                        001B    365 ;
                        001B    366 ; TEST PATTERNS:
                        001B    367 ;
                        001B    368 ;       SUBTEST 1:
                        001B    369 ;                           ITERATION       TEST PATTERN
                        001B    370 ;                               1               A
                        001B    371 ;                               2               5
                        001B    372 ;                               3               3
                        001B    373 ;
                        001B    374 ;       SUBTEST 2
                        001B    375 ;                           ITERATION       TEST PATTERN
                        001B    376 ;                               1               1
                        001B    377 ;                               2               0
                        001B    378 ;
                        001B    379 ;
                        001B    380 ; LOGIC DESCRIPTION:
                        001B    381 ;
                        001B    382 ;       THIS TEST IS DESIGNED TO CHECK ONLY THE S/C REGISTERS THAT
                        001B    383 ;       LIVE IN THE UTR GATE ARRAY.  THE REGISTERS ARE ACCESSED VIA
                        001B    384 ;       WBUS<27:24>.  IF ON ERROR, REPLACING THE UTR DOES NOT WORK,
                        001B    385 ;       THEN YOU MIGHT TRY RUNNING THE MEMSCAR TEST WHICH WILL CHECK
                        001B    386 ;       FOR A MULTIPLE ADDRESSING PROBLEM.  IF THAT FAILS THEN ONE
                        001B    387 ;       OF THE OTHER GATE ARRAYS WITH S/C REGISTERS IS GIVING YOU
                        001B    388 ;       TROUBLE.
                        001B    389 ;
                        001B    390 ;
                        001B    391 ; ASSUMPTIONS:
                        001B    392 ;
                        001B    393 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
```

```
                     001B   394 ;
                     001B   395 ;
                     001B   396 ; ERROR DESCRIPTION:
                     001B   397 ;
                     001B   398 ;       EXPECTED MEMSCR DATA
                     001B   399 ;       RECEIVED MEMSCR DATA
                     001B   400 ;       LOOP COUNT
                     001B   401 ;
                     001B   402 ;--
                     001B   403 ;*******************************************************************************
                     001B   404
                     001B   405         SUBTEST                                     ;MACHINE CHECK ERROR SUM REG
                     001E
                     001E       ;////////////////////////////////////////////////////////////////////////////////
                     001E   406 ;+
                     001E   407 ;SUBTEST #1 MACHINE CHECK ERROR SUMMARY REGISTER
                     001E   408 ;-
                     001E   409         INITIALIZE                                  ;INIT CPU
                     0020   410         LOADDCS         5$,,01,1                    ;MODIFY DCS ADDRESS 01
                     0027   411         LOOP            I,1,3                       ;CREATE TEST LOOP
                     002E   412         LOADREG         WDREG,1$,I,L                ;TEST PATTERN TO D REG
                     0036   413         ERRLOOP                                     ;ERROR LOOP
                     0038   414         FETCH           0                           ;START DCS PROGRAM
                     003D   415         BRSTCLK         4                           ;END DCS PROGRAM
                     0041   416         CMPREGMSK       WHREG,1$,I,4$,,L,           ;COMPARE RESULTS
                     004D   417         IFERROR         ,<<UTR>>,                   ;POSSIBLE FAILING ARRAY
                     0053   418         ENDLOOP         I                           ;END TEST LOOP
                     0056   419
                     0056   420         SUBTEST                                     ;REFERENCE CACHE ONLY REG
                     0059
                     0059       ;////////////////////////////////////////////////////////////////////////////////
                     0059   421 ;+
                     0059   422 ;SUBTEST #2 REFERENCE CACHE ONLY REGISTER
                     0059   423 ;-
                     0059   424         INITIALIZE                                  ;INIT CPU
                     005B   425         LOADDCS         3$,,01,1                    ;CHANGE DCS ADDRESS 01
                     0062   426         LOOP            I,1,2                       ;CREATE TEST LOOP
                     0069   427         LOADREG         WDREG,2$,I,L                ;TEST PATTERN TO D REG
                     0071   428         ERRLOOP                                     ;ERROR LOOP
                     0073   429         FETCH           0                           ;START DCS PROGRAM
                     0078   430         BRSTCLK         4                           ;END DCS PROGRAM
                     007C   431         CMPREGMSK       WHREG,2$,I,4$,,L,           ;COMPARE RESULTS
                     0088   432         IFERROR         ,<<UTR>>,                   ;POSSIBLE FAILING ARRAY
                     008E   433         ENDLOOP         I                           ;END TEST LOOP
                     0091   434         SKIP
                     0098   435
                     0098   436 BEGIN_TEST_DATA
                     0098
                     0098       ;-------------------------------------------------------------------------------
                     0098   437
0A000000             0098   438 1$:     .LONG   ^X0A000000                          ;TEST PATTERNS
05000000             009C   439         .LONG   ^X05000000                          ;   "       "
03000000             00A0   440         .LONG   ^X03000000                          ;   "       "
01000000             00A4   441 2$:     .LONG   ^X01000000                          ;   "       "
00000000             00A8   442         .LONG   ^X00000000                          ;
```

```
                                00AC   443 3$:
U 01, 7700,8800,5BE4,0304,6870,1802  00AC   444 ;* 01   MEMSCAR_R[TEMP1]                    ;RTEMP1 FOR SUBTEST 2
                                00B7   448
                     0F000000   00B7   449 4$:     .LONG   ^X0F000000                    ;MASK FOR CMPREGMSK PSEUDO OP
                                00BB   450
                                00BB   451 5$:
U 01, 7700,C800,5BE4,0300,6870,1802  00BB   452 ;* 01   MEMSCAR_R[TEMP0]                    ;RTEMP0 FOR SUBTEST 1
                                00C6   456
                                00C6   457 BEGIN_CPU_DATA
                                0002   458
                                0002   459 DCS_DATA
                                0001   460
U 00, 7700,C800,0364,0300,0470,1801  0001   461 ;* 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C   465 ;* 01   MEMSCAR_R[TEMP0]                    ;WRITE ADD OF S/C TO MEMSCAR
U 02, 5700,C800,0364,0300,6070,1803  0017   469 ;* 02   WB_RDM,WCTRL/MEMSCR_WB             ;WRITE TEST PATTERN TO MEMSCR
U 03, 6700,C800,0364,0300,6470,1804  0022   473 ;* 03   RDM_WB,WCTRL/MEMSCR               ;READ TEST PATTERN BACK TO RDM
U 04, 7300,4800,0364,0300,0470,1805  002D   477 ;* 04   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
                                0038   481
                                0038   482 RTEMP_DATA
                                0001   483
                     08000000   0001   484         .LONG   ^X08000000                    ;MACHINE CHECK ERROR SUM ADD
                     0E000000   0005   485         .LONG   ^X0E000000                    ;REFERENCE CACHE ONLY REG ADD
                                0009   486
                                0009   487 END_CPU_DATA
                                00C6   488
                                00C6   489 END_TEST_DATA
                                00C6
                                00C6       ;-------------------------------------------------------------------------
                                00C6   490
                                00C6   491 ENDTEST
```

```
0011        .SBTTL  "TEST   004     TEST MEMSCAR
0011    493 ;*******************************************************************************
0011    494 ;++
0011    495 ;
0011    496 ; FUNCTIONAL DESCRIPTION:
0011    497 ;
0011    498 ;       SINCE THERE ARE 3 IDENTICAL COPIES OF THE MEMORY STATUS AND CONTROL
0011    499 ;       ADDRESS REGISTER IT IS NECESSARY TO BE SURE EACH IS SELECTING ONLY
0011    500 ;       S/C REGISTERS IT IS RESPONSIBLE FOR.  THIS TEST WILL LOOK FOR
0011    501 ;       AN INCORRECT DECODE BY THE MEMSCAR'S.
0011    502 ;
0011    503 ;
0011    504 ; TEST ALGORITHM:
0011    505 ;
0011    506 ;       1.  SET UP A LOOP OF 8 TO SELECT S/C REGISTER ADDRESSES FROM A TABLE
0011    507 ;           (THIS LOOP WILL POINT TO THE REGISTER UNDER TEST [RUT])
0011    508 ;       2.  LOAD THIS ADDRESS INTO THE LONGLIT FIELD OF DCS ADDRESS 01
0011    509 ;       3.  SET UP A LOOP OF 8 TO SELECT S/C REGISTER ADDRESSES FROM A TABLE
0011    510 ;       4.  IF LOOPS AT STEPS 1 AND 3 ARE POINTING TO SAME REGISTER THEN
0011    511 ;           SKIP TO STEP 8
0011    512 ;       5.  LOAD D REGISTER (RDM) WITH ADDRESS POINTED TO BY LOOP AT STEP 3
0011    513 ;       6.  EXECUTE MICRO PROGRAM
0011    514 ;               1.  LOAD LONGLIT REGISTER WITH RUT ADDRESS
0011    515 ;               2.  LOAD D REGISTER WITH RUT ADDRESS
0011    516 ;               3.  LOAD MEMSCAR AND TEMP 0 WITH RUT ADDRESS
0011    517 ;               4.  WRITE 0'S TO RUT
0011    518 ;               5.  LOAD MEMSCAR AND TEMP 1 WITH ANOTHER S/C REGISTER ADDRESS
0011    519 ;               6.  WRITE 1'S TO THIS REGISTER
0011    520 ;               7.  LOAD MEMSCAR WITH RUT ADDRESS
0011    521 ;               8.  READ RUT TO RDM
0011    522 ;       7.  COMPARE H REGISTER WITH 0'S TEST PATTERN
0011    523 ;       8.  IF NO ERROR GO TO STEP 3 FOR REMAINING S/C REGISTERS
0011    524 ;       9.  GO TO STEP 1 FOR REMAINIG S/C REGISTERS
0011    525 ;
0011    526 ;
0011    527 ; TEST PATTERNS:
0011    528 ;
0011    529 ;       REGISTER UNDER TEST=0'S
0011    530 ;       ALL OTHER REGISTERS=1'S
0011    531 ;
0011    532 ;
0011    533 ; LOGIC DESCRIPTION:
0011    534 ;
0011    535 ;       THIS TEST IS DESIGNED TO CHECK THE MEMORY STATUS AND CONTROL
0011    536 ;       ADDRESS REGISTER BY USING THE 8 READ/WRITE MEMORY STATUS AND
0011    537 ;       CONTROL REGISTERS.  THERE ARE 3 COPIES OF MEMSCAR HIDDEN IN
0011    538 ;       THREE DIFFERENT GATE ARRAYS (ADK, CAK, UTR).  IF THE DECODE
0011    539 ;       LOGIC IN ONE OF THESE REGISTERS FAILS, THEN YOU COULD END UP
0011    540 ;       ENABLING MORE THAN ONE MEMSCR AT A TIME.  THIS TEST ATTEMPTS
0011    541 ;       TO CATCH THIS FAILURE MODE BY WRITING 0'S TO ONE REGISTER
0011    542 ;       AND WRITING 1'S TO ALL OTHERS.  IF THE ADDRESS DECODE FAILS,
0011    543 ;       THE REGISTER WITH 0'S SHOULD GET CLOBBERED.
0011    544 ;
0011    545 ;       REG ADDRESS                 NAME                        LOCATION
0011    546 ;            0          PHYSICAL/VIRTUAL ADD REG                  ADK
```

```
                 0011   ;            1          SAVED MODE REGISTER                 "
                 0011   ;            2          WRITE VECTOR OCCURED REG            "
                 0011   ;            3          TB GROUP DISABLE REGISTER           "
                 0011   ;            4          CACHE ERROR REGISTER               CAK
                 0011   ;            6          CACHE CONTROL REGISTER              "
                 0011   ;            8          MACHINE CHECK ERROR SUM REG        UTR
                 0011   ;            E          REFERENCE CACHE ONLY REG            "
                 0011   ;
                 0011   ;
                 0011   ; ASSUMPTIONS:
                 0011   ;
                 0011   ;       THIS TEST ASSUMES THE DPM AND WBUS ARE OPERATIONAL.
                 0011   ;
                 0011   ;
                 0011   ; ERROR DESCRIPTION:
                 0011   ;
                 0011   ;       EXPECTED MEMSCR DATA
                 0011   ;       RECEIVED MEMSCR DATA
                 0011   ;       LOOP COUNT:
                 0011   ;               I=INDEX TO REGISTER UNDER TEST (SEE TABLE AT 1$)
                 0011   ;               J=INDEX TO S/C REGISTER THAT IS BEING COMPARED
                 0011   ;                 (SEE TABLE AT 1$)
                 0011   ;       TEMPORARY REGISTERS:
                 0011   ;               0=ADDRESS OF REGISTER UNDER TEST
                 0011   ;               1=ADDRESS OF REGISTER THAT HAS BAD DECODE LOGIC
                 0011   ;               2=0'S TEST PATTERN FOR REGISTER UNDER TEST
                 0011   ;               3=1'S TEST PATTERN FOR OTHER S/C REGISTERS
                 0011   ;
                 0011   ;--
                 0011   ;*******************************************************************************
                 0011   ;///////////////////////////////////////////////////////////////////////////////
                 0011
                 0011           INITIALIZE                                  ;INIT CPU
                 0013           LOOP            I,1,8                       ;REG UNDER TEST LOOP
                 001A           LDFIELD         1$,I,L,4$,31,62,LON         ;SELECT AN S/C REG
                 002B           LOADDCS         4$,,01,1                    ;MODIFY DCS ADDRESS 01
                 0032           LOOP            J,1,8                       ;LOOP TO WRITE OTHER S/C REG
                 0039           SKIP            2$,ONEQUAL,I,J              ;SKIP OVER REG UNDER TEST
                 0040           LOADREG         WDREG,1$,J,L                ;LOAD D WITH REG ADDRESS
                 0048           ERRLOOP                                     ;ERROR LOOP
                 004A           FETCH           0                           ;START DCS PROGRAM
                 004F           BRSTCLK         8                           ;END DCS PROGRAM
                 0053           CMPREGMSK       WHREG,3$,,5$,,L,                    ;COMPARE RESULTS
                 005F           IFERROR         4,<<ADK><CAK><UTR>>,        ;POSSIBLE FAILING ARRAYS
                 0067   2$:     ENDLOOP         J                           ;END WRITE AND COMPARE LOOP
                 006A           ENDLOOP         I                           ;END REG UNDER TEST LOOP
                 006D           SKIP
                 0074
                 0074   BEGIN_TEST_DATA
                 0074
                 0074   ;-----------------------------------------------------------------------
                 0074                                                       ;S/C REG ADDRESS TABLE
00000000         0074   1$:     .LONG   ^X00000000                          ;PHYSICAL/VIRTUAL ADD REG
01000000         0078           .LONG   ^X01000000                          ;SAVED MODE REG
02000000         007C           .LONG   ^X02000000                          ;WRITE VECTOR OCCURED REG
```

```
                    03000000  0080   600          .LONG   ^X03000000                      ;TB GROUP DISABLE REG
                    04000000  0084   601          .LONG   ^X04000000                      ;CACHE ERROR REG
                    06000000  0088   602          .LONG   ^X06000000                      ;CACHE CONTROL REG
                    08000000  008C   603          .LONG   ^X08000000                      ;MACHINE CHECK ERROR SUM REG
                    0E000000  0090   604          .LONG   ^X0E000000                      ;REFERENCE CACHE ONLY REG
                    00000000  0094   605 3$:      .LONG   ^X00000000                      ;COMPARE DATA PATTERN
                              0098   606
                              0098   607 4$:
U 01, 7700,7800,7FFF,FFFF,8470,1802   0098   608 ;* 01    LONLIT_[0]                      ;USED TO MODIFY DCS ADD 01
                              00A3   612
                    0F000000  00A3   613 5$:      .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                              00A7   614
                              00A7   615 BEGIN_CPU_DATA
                              0002   616
                              0002   617 DCS_DATA
                              0001   618
U 00, 7700,C800,0364,0300,0470,1801   0001   619 ;* 00    NOP
U 01, 7700,7800,7FFF,FFFF,8470,1802   000C   623 ;* 01    LONLIT_[0]                      ;RUT ADD TO LONGLIT REG
U 02, 7700,0840,5BE4,03D4,6870,1803   0017   627 ;* 02    WB_R[LONLIT],SPW/RLONG,WCTRL/MEMSCAR_WB ;RUT ADD TO TEMP0 AND MEMSCAR
U 03, 7700,0800,5BE4,0308,6070,1804   0022   631 ;* 03    MEMSCR_R[TEMP2]                 ;0'S TEST PATTERN TO RUT
U 04, 5700,4840,0364,0304,6870,1805   002D   635 ;* 04    WB_RDM,RSRC/TEMP1,SPW/RLONG,WCTRL/MEMSCAR_WB
                              0038   639                                                  ;S/C ADD TO TEMP0 AND MEMSCAR
U 05, 7700,C800,5BE4,030C,6070,1806   0038   640 ;* 05    MEMSCR_R[TEMP3]                 ;1'S TEST PATTERN TO MEMSCR
U 06, 7700,C800,5BE4,0300,6870,1807   0043   644 ;* 06    MEMSCAR_R[TEMP0]                ;RUT ADDRESS TO MEMSCAR
U 07, 6700,C800,0364,0300,6470,1808   004E   648 ;* 07    RDM_WB,WCTRL/MEMSCR             ;RUT DATA TO RDM
U 08, 7300,4800,0364,0300,0470,1809   0059   652 ;* 08    NOP,STOP.CLOCK                  ;STOP CPU CLOCK
                              0064   656
                              0064   657 RTEMP_DATA
                              0001   658
                    00000000  0001   659          .LONG   ^X00000000                      ;CLEAR TEMP0
                    00000000  0005   660          .LONG   ^X00000000                      ;CLEAR TEMP1
                    00000000  0009   661          .LONG   ^X00000000                      ;RUT TEST PATTERN
                    0F000000  000D   662          .LONG   ^X0F000000                      ;OTHER S/C REG TEST PATTERN
                              0011   663
                              0011   664 END_CPU_DATA
                              00A7   665
                              00A7   666 END_TEST_DATA
                              00A7
                              00A7        ;-------------------------------------------------------------------------
                              00A7   667
                              00A7   668 ENDTEST
```

```
                        .SBTTL  "TEST   005     TEST PROCESSOR INITIALIZE BUS FUNCTION
                    ;*****************************************************************************
                    ;++
                    ;
                    ; FUNCTIONAL DESCRIPTION:
                    ;
                    ;       THIS TEST CHECKS THE ABILITY OF THE "PROCESSOR INITIALIZE" BUS
                    ;       MICRO ORDER TO CLEAR THE S/C REGISTERS.
                    ;
                    ;
                    ; TEST ALGORITHM:
                    ;
                    ;       1.  CREATE A TEST LOOP OF 9 ITERATIONS
                    ;       2.  SELECT A MEMORY STATUS AND CONTROL REGISTER (MEMSCR) ADDRESS FROM
                    ;           A TABLE AND LOAD INTO D REGISTER (RDM)
                    ;       3.  EXECUTE DCS PROGRAM
                    ;               a.  WRITE ADDRESS IN D REGISTER (RDM) INTO MEMORY STATUS AND
                    ;                   CONTROL ADDRESS REGISTER (MEMSCAR)
                    ;               b.  WRITE ALL ONES INTO SELECTED MEMSCR
                    ;               c.  PERFORM "BUS/PRINIT"
                    ;               d.  READ SELECTED MEMSCR BACK TO RDM
                    ;       4.  TEST VBUS FOR PROC INIT L
                    ;       5.  IF NO ERROR TEST FOR CORRECT DATA IN MEMSCR
                    ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING MEMSCR'S
                    ;
                    ; TEST PATTERNS:
                    ;
                    ;       BEFORE "BUS/PRINIT"
                    ;               SELECTED WRITABLE S/C REGISTER=1'S
                    ;
                    ;       AFTER "BUS/PRINIT"
                    ;               SELECTED S/C REGISTER=0'S
                    ;
                    ;
                    ; LOGIC DESCRIPTION:
                    ;
                    ;       THIS TEST VERIFIES THAT THE GATE ARRAYS TH  CONTAIN S/C REGISTERS
                    ;       CAN DECODE AND PERFORM A "BUS/PRINIT".  BY LOOKING AT THE LOOP
                    ;       COUNT AND CONSULTING THE TABLE IN THE ERROR DESCRIPTION SECTION
                    ;       YOU CAN FIND WHICH GATE ARRAY TO REPLACE.  ONE THING TO NOTE IS
                    ;       THAT IN ITERATION #9 (TB HIT REGISTER) THE WRITE PORTION OF THE
                    ;       TEST DOES NOTHING SINCE THE REGISTER IS READ ONLY.
                    ;
                    ;
                    ; ASSUMPTIONS:
                    ;
                    ;       THIS TEST ASSUMES THE WBUS AND MBUS ARE OPERATIONAL.
                    ;
                    ;
                    ; ERROR DESCRIPTION:
                    ;
                    ;       FIRST IFERROR:
                    ;               EXPECTED VBUS DATA
                    ;               RECEIVED VBUS DATA
```

```
                002B   ;                LOOP COUNT I
                002B   ;                        (BITS <7:0> = BIT POSITION)
                002B   ;                        (BIT <8> = BIT VALUE)
                002B   ;
                002B   ;        SECOND IFERROR:
                002B   ;                EXPECTED MEMSCR DATA
                002B   ;                RECEIVED MEMSCR DATA
                002B   ;                LOOP COUNT I
                002B   ;
                002B   ;        LOOP COUNT      REGISTER UNDER TEST                     GATE ARRAY
                002B   ;
                002B   ;            1           PHYSICAL/VIRTUAL ADDRESSING                ADK
                002B   ;            2           MACHINE CHECK ERROR SUMMARY                UTR
                002B   ;            3           TB GROUP DISABLE                           ADK
                002B   ;            4           CACHE ERROR                                CAK
                002B   ;            5           CACHE CONTROL                              CAK
                002B   ;            6           REFERENCE CACHE ONLY                       UTR
                002B   ;            7           SAVED MODE                                 ADK
                002B   ;            8           WRITE VECTOR OCCURRED                      ADK
                002B   ;            9           TB HIT                                     UTR
                002B   ;
                002B   ;--
                002B   ;*******************************************************************************
                002B   ;///////////////////////////////////////////////////////////////////////////////
                002B
                002B           INITIALIZE                                   ;INIT CPU
                002D           LOOP            I,1,9                        ;CREATE A TEST LOOP
                0034           LOADREG         WDREG,1$,I,L                 ;LOAD MEMSCR ADDRESS TO D REG
                003C           ERRLOOP                                      ;ERROR LOOP
                003E           FETCH           0                            ;START DCS PROGRAM
                0043           BRSTCLK         5                            ;END DCS PROGRAM
                0047           CMPVBUS         4$                           ;TEST FOR PROC INIT
                004C           IFERROR         ,<ACV>,                      ;NO PROC INIT ON VBUS
                0052           CMPREGMSK       WHREG,2$,,3$,,L,             ;COMPARE RESULTS
                005E           IFERROR         ,<<ADK><UTR><CAK>>,          ;POSSIBLE FAILING ARRAYS
                0066           ENDLOOP         I                            ;END TEST LOOP
                0069           SKIP
                0070
                0070   BEGIN_TEST_DATA
                0070
                0070   ;-----------------------------------------------------------------------
                0070                                                        ;MEMSCR ADDRESS TABLE
00000000        0070   1$:      .LONG   ^X00000000                          ;PHY/VIR ADDRESSING
08000000        0074           .LONG   ^X08000000                          ;MACHINE CHECK ERROR SUMMARY
03000000        0078           .LONG   ^X03000000                          ;TB GROUP DISABLE
04000000        007C           .LONG   ^X04000000                          ;CACHE ERROR
06000000        0080           .LONG   ^X06000000                          ;CACHE CONTROL
0E000000        0084           .LONG   ^X0E000000                          ;REFERENCE CACHE ONLY
01000000        0088           .LONG   ^X01000000                          ;SAVED MODE
02000000        008C           .LONG   ^X02000000                          ;WRITE VECTOR OCCURRED
0C000000        0090           .LONG   ^X0C000000                          ;TB HIT
00000000        0094   2$:      .LONG   ^X00000000                          ;COMPARE DATA
0F000000        0098   3$:      .LONG   ^X0F000000                          ;MASK FOR CMPREGMSK PSEUDO OP
      01        009C   4$:      .BYTE   ^X1
                009D           VBUSDATA        <^X0F>,0                     ;PROC INIT ON VBUS
```

```
                                009E   777
                                009E   778 BEGIN_CPU_DATA
                                0002   779
                                0002   780 DCS_DATA
                                0001   781
U 00, 7700,C800,0364,0300,0470,1801   0001   782 ;% 00   NOP
U 01, 5700,C800,0364,0300,6870,1802   000C   786 ;% 01   WB_RDM,WCTRL/MEMSCAR_WB                ;MEMSCR ADDRESS TO MEMSCAR
U 02, 7700,C800,5BE4,0300,6070,1803   0017   790 ;% 02   MEMSCR_R[TEMP0]                       ;WRITE ALL 1'S
U 03, 7600,C800,0364,0300,0410,1804   0022   794 ;% 03   BUS/PRINIT,STROBE.V.BUS               ;PROCESSOR INIT
U 04, 6700,4800,0364,0300,6470,1805   002D   798 ;% 04   RDM_WB,WCTRL/MEMSCR                   ;READ MEMSCR TO RDM
U 05, 7300,4800,0364,0300,0470,1806   0038   802 ;% 05   NOP,STOP.CLOCK                       ;STOP CPU CLOCK
                                0043   806
                                0043   807 RTEMP_DATA
                                0001   808
                      0F000000  0001   809         .LONG   ^X0F000000                      ;DATA
                                0005   810
                                0005   811 END_CPU_DATA
                                009E   812
                                009E   813 END_TEST_DATA
                                009E
                                009E       ;-------------------------------------------------------------------------
                                009E   814
                                009E   815 ENDTEST
```

```
0016        .SBTTL  "TEST   006     TEST MDR REGISTER
0016   817 ;*******************************************************************************
0016   818 ;++
0016   819 ;
0016   820 ; FUNCTIONAL DESCRIPTION:
0016   821 ;
0016   822 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE MDR REGISTER
0016   823 ;       BY LOADING AND READING 6 DATA PATTERNS THROUGH THE REGISTER.
0016   824 ;
0016   825 ;
0016   826 ; TEST ALGORITHM:
0016   827 ;
0016   828 ;       1.  SETUP A LOOP OF SIX ITERATIONS
0016   829 ;       2.  GENERATE A TEST PATTERN
0016   830 ;       3.  LOAD TEST PATTERN INTO WD REGISTER (RDM)
0016   831 ;       4.  EXECUTE DCS PROGRAM
0016   832 ;               a.  LOAD TEST PATTERN INTO MDR REGISTER
0016   833 ;               b.  READ TEST PATTERN BACK TO WH REGISTER (RDM)
0016   834 ;       5.  COMPARE DATA (EXPECTED AND RECEIVED)
0016   835 ;       6.  IF NO ERROR TEST VBUS FOR LATCHED MBUS 15 L
0016   836 ;       7.  IF NO ERROR TEST CS ADDRESS FOR BRANCH ON MBUS 15
0016   837 ;       8.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0016   838 ;
0016   839 ;
0016   840 ; TEST PATTERNS:
0016   841 ;
0016   842 ;       ITERATION       TEST PATTERN
0016   843 ;           1             AAAAAAAA
0016   844 ;           2             55555555
0016   845 ;           3             33333333
0016   846 ;           4             0F0F0F0F
0016   847 ;           5             00FF00FF
0016   848 ;           6             0000FFFF
0016   849 ;
0016   850 ;
0016   851 ; LOGIC DESCRIPTION:
0016   852 ;
0016   853 ;       THIS TEST IS DESIGNED TO PRIMARILY CHECK THE MDR REGISTER ON THE
0016   854 ;       MIC MODULE.  IN DOING SO OTHER GATE ARRAYS, USED FOR CONTROL, WILL
0016   855 ;       COME INTO PLAY.  ADK CONTROLS THE MDR AND WBUS SELECT, PRK CONTROLS
0016   856 ;       THE MMUX, AND CAK CONTROLS THE ROT LOGIC.  IF ERRORS OCCUR, AND
0016   857 ;       REPLACING MDR DOES NOT FIX THE PROBLEM, THEN THESE OTHER ARRAYS
0016   858 ;       SHOULD BE TAKEN INTO CONSIDERATION.
0016   859 ;       ALSO TESTED IS A PIECE OF DISCRETE LOGIC THAT LATCHES THE MBUS BIT
0016   860 ;       15.  THIS IS EXAMINED VIA THE VBUS.
0016   861 ;       FINALLY THE BRANCHING LOGIC ON DPM IS TESTED FOR A BRANCH ON MBUS 15.
0016   862 ;
0016   863 ;
0016   864 ; ASSUMPTIONS:
0016   865 ;
0016   866 ;       THIS TEST ASSUMES THE WBUS AND MBUS ARE OPERATIONAL.
0016   867 ;
0016   868 ;
0016   869 ; ERROR DESCRIPTION:
0016   870 ;
```

```
                        0016   871 ;       FIRST IFERROR:
                        0016   872 ;               EXPECTED MDR DATA
                        0016   873 ;               RECEIVED MDR DATA
                        0016   874 ;               LOOP COUNT I
                        0016   875 ;               ANALYSIS OF WHICH MDR BIT SLICE FAILED.
                        0016   876 ;
                        0016   877 ;       SECOND IFERROR:
                        0016   878 ;               EXPECTED VBUS DATA (LATCHED MBUS 15 L)
                        0016   879 ;               RECEIVED VBUS DATA
                        0016   880 ;               LOOP COUNT I
                        0016   881 ;                       (BITS <7:0> = BIT POSITION)
                        0016   882 ;                       (BITS <8> = BIT STATE)
                        0016   883 ;
                        0016   884 ;       THIRD IFERROR:
                        0016   885 ;               EXPECTED CS ADDRESS
                        0016   886 ;               RECEIVED CS ADDRESS
                        0016   887 ;               LOOP COUNT I
                        0016   888 ;
                        0016   889 ;--
                        0016   890 ;*******************************************************************************
                        0016   891 ;///////////////////////////////////////////////////////////////////////////////
                        0016   892
                        0016   893         INITIALIZE                                      ;INIT CPU
                        0018   894         LOOP            I,1,6                           ;CREATE TEST LOOP
                        001F   895         SA01PAT         I,1$,,                          ;GENERATE A TEST PATTERN
                        0027   896         LOADREG         WDREG,1$,,L                     ;TEST PATTERN TO WD REG
                        002F   897         ERRLOOP                                         ;ERROR LOOP
                        0031   898         FETCH           0                               ;START DCS PROGRAM
                        0036   899         BRSTCLK         3                               ;END DCS PROGRAM
                        003A   900         CMPREG          WHREG,1$,,L,                    ;COMPARE RESULTS
                        0043   901         IFERROR         ,<<MDR><ADK><PRK><CAK>>,
                        004C   902         CMPVBUS         2$,I                            ;TEST MBUS 15
                        0051   903         IFERROR                                         ;BIT INCORRECT
                        0056   904         CMPREGMSK       CSTRP,3$,I,4$,,W,               ;TEST CS ADDRESS
                        0062   905         IFERROR         ,,<DPM>                         ;INCORRECT BRANCH ADDRESS
                        0068   906         ENDLOOP         I                               ;END TEST LOOP
                        006B   907         SKIP                                            ;GO TO NEXT TEST
                        0072   908
                        0072   909 BEGIN_TEST_DATA
                        0072
                        0072       ;-----------------------------------------------------------------------------
                        0072   910
00000000                0072   911 1$:     .LONG   ^X00000000                              ;TEST PATTERN STORAGE
              01        0076   912 2$:     .BYTE   ^X1                                     ;MBUS 15 LATCH
                        0077   913         VBUSDATA        <^X0E>,0
              01        0078   914         .BYTE   ^X1
                        0079   915         VBUSDATA        <^X0E>,1
              01        007A   916         .BYTE   ^X1
                        007B   917         VBUSDATA        <^X0E>,1
              01        007C   918         .BYTE   ^X1
                        007D   919         VBUSDATA        <^X0E>,1
              01        007E   920         .BYTE   ^X1
                        007F   921         VBUSDATA        <^X0E>,1
              01        0080   922         .BYTE   ^X1
                        0081   923         VBUSDATA        <^X0E>,0
```

```
ZZ-ECKAC-8.0    V08.00                    "TEST 006 TEST M17-FEB-1986      Fiche 1  Frame F9        Sequence 109
ECKAC                      VAX-11/750 MIC MICRO DIAGNOSTICS          17-FEB-1986 13:38:57  VAX/VMS Macro V04-00        Page  4
V08.00                     "TEST 006 TEST MDR REGISTER               17-FEB-1986 13:38:39  [VAX750.MIC]ECKAC1.MAR;1          (1

                     1801  0082    924 3$:     .WORD   ^X1801                          ;BRANCH ADDRESS TABLE
                     1803  0084    925         .WORD   ^X1803
                     1803  0086    926         .WORD   ^X1803
                     1803  0088    927         .WORD   ^X1803
                     1803  008A    928         .WORD   ^X1803
                     1801  008C    929         .WORD   ^X1801
                     3FFF  008E    930 4$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                           0090    931
                           0090    932 BEGIN_CPU_DATA
                           0002    933
                           0002    934 DCS_DATA
                           0001    935
U 00, 7700,C800,0364,0300,0470,1801   0001    936 ;% 00   NOP
U 01, 5700,C800,0364,0300,4670,1802   000C    940 ;% 01   WB_RDM,WCTRL/MDR_WB                     ;LOAD MDR FROM RDM
U 02, 6600,4812,5924,AB00,0470,1800   0017    944 ;% 02   RDM_WB,WB_M[MDR],BUT/FRO.FLTZ,NEXT/1800,STROBE.V.BUS;READ MDR TO RDM
U 03, 7300,C800,0364,0300,0470,1804   0022    948 ;% 03   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
                           002D    952
                           002D    953 END_CPU_DATA
                           0090    954
                           0090    955 END_TEST_DATA
                           0090
                           0090        ;-----------------------------------------------------------------------------
                           0090    956
                           0090    957 ENDTEST
```

```
0023        .SBTTL  "TEST   007     TEST "WCTRL/MDR_0" MICRO ORDER
0023    959 ;*******************************************************************************
0023    960 ;++
0023    961 ;
0023    962 ; FUNCTIONAL DESCRIPTION:
0023    963 ;
0023    964 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE "WCTRL/MDR_0"
0023    965 ;       MICRO ORDER.
0023    966 ;
0023    967 ;
0023    968 ; TEST ALGORITHM:
0023    969 ;
0023    970 ;       1.  EXECUTE DCS PROGRAM
0023    971 ;               a.  WRITE AN ALL 1'S PATTERN TO MEMORY DATA REGISTER (MDR)
0023    972 ;               b.  PERFORM "WCTRL/MDR_0" FUNCTION
0023    973 ;               c.  READ MDR TO RDM WHILE BRANCHING ON WBUS = 0 AND MBUS 15.
0023    974 ;       2.  TEST MDR REGISTER CONTENTS
0023    975 ;       3.  IF NO ERROR TEST CS ADDRESS FOR CORRECT BRANCH.
0023    976 ;
0023    977 ;
0023    978 ; TEST PATTERNS:
0023    979 ;
0023    980 ;                       MDR
0023    981 ;
0023    982 ;       BEFORE  FFFFFFFF
0023    983 ;       AFTER   00000000
0023    984 ;
0023    985 ;
0023    986 ; LOGIC DESCRIPTION:
0023    987 ;
0023    988 ;       THIS TEST CHECKS THE WCTRL FUNCTION THAT LOADS THE MDR WITH ALL 0'S.
0023    989 ;       IN SO DOING THE LOGIC THAT DRIVES THE DBUS ROTATOR AND THE DBUS SELECT
0023    990 ;       LINES ARE EXERCISED.  THE "ADK" GATE ARRAY DRIVES "DBUS SEL S1 AND S0"
0023    991 ;       WHILE THE "CAK" GATE ARRAY GENERATES "DBUS ROT S1 AND S0".  THE "MDR"
0023    992 ;       GATE ARRAY IS THE HOME OF THE MDR REGISTER.
0023    993 ;       ALSO TESTED HERE IS THE BRANCHING LOGIC ON DPM THAT LOOKS FOR WBUS = 0
0023    994 ;       AND MBUS 15.
0023    995 ;
0023    996 ;
0023    997 ; ASSUMPTIONS:
0023    998 ;
0023    999 ;       THIS TEST ASSUMES THE WBUS AND MBUS ARE OPERATIONAL.
0023   1000 ;
0023   1001 ;
0023   1002 ; ERROR DESCRIPTION:
0023   1003 ;
0023   1004 ;       FIRST IFERROR:
0023   1005 ;               EXPECTED MDR DATA
0023   1006 ;               RECEIVED MDR DATA
0023   1007 ;
0023   1008 ;       SECOND IFERROR:
0023   1009 ;               EXPECTED CS ADDRESS
0023   1010 ;               RECEIVED CS ADDRESS
0023   1011 ;
0023   1012 ;--
```

```
                                  0023  ;*******************************************************************************
                                  0023  ;///////////////////////////////////////////////////////////////////////////////
                                  0023
                                  0023          INITIALIZE                                  ;INIT CPU
                                  0025          ERRLOOP                                     ;ERROR LOOP
                                  0027          FETCH           0                           ;START DCS PROGRAM
                                  002C          BRSTCLK         4                           ;END DCS PROGRAM
                                  0030          CMPREG          WHREG,1$,,L,                ;COMPARE RESULTS
                                  0039          IFERROR         ,<<ADK><CAK><MDR>>,         ;POSSIBLE FAILING ARRAYS
                                  0041          CMPREGMSK       CSTRP,2$,,3$,,W,            ;TEST CS ADDRESS
                                  004D          IFERROR         ,,<DPM>                     ;BRANCH INCORRECT
                                  0053          SKIP                                        ;GO TO NEXT TEST
                                  005A
                                  005A  BEGIN_TEST_DATA
                                  005A
                                  005A  ;-----------------------------------------------------------------------
                                  005A
                        00000000  005A  1$:     .LONG   ^X00000000                          ;RESULTS OF TEST
                            1802  005E  2$:     .WORD   ^X1802                              ;BRANCH ADDRESS
                            3FFF  0060  3$:     .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                                  0062
                                  0062  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,C800,5BE4,0300,4670,1802  000C  ;% 01   MDR_R[TEMP0]                               ;1'S TO MDR
U 02, 7700,C800,0364,0300,4E70,1803  0017  ;% 02   WCTRL/MDR_0                                ;0'S TO MDR
U 03, 6700,4812,5924,AB00,0470,1800  0022  ;% 03   RDM_WB,WB_M[MDR],BUT/FRO.FLTZ,NEXT/1800 ;MDR TO RDM
U 04, 7300,4800,0364,0300,0470,1805  002D  ;% 04   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
                                  0038
                                  0038  RTEMP_DATA
                                  0001
                        FFFFFFFF  0001          .LONG   ^XFFFFFFFF                          ;1'S FOR MDR
                                  0005
                                  0005  END_CPU_DATA
                                  0062
                                  0062  END_TEST_DATA
                                  0062
                                  0062  ;-----------------------------------------------------------------------
                                  0062
                                  0062  ENDTEST
```

```
0015          .SBTTL  "TEST   008     VA REGISTER TEST
0015  1067 ;*******************************************************************************
0015  1068 ;++
0015  1069 ;
0015  1070 ; FUNCTIONAL DESCRIPTION
0015  1071 ;
0015  1072 ;       This test examines the data integrity of the VA register.
0015  1073 ;
0015  1074 ; TEST ALGORITHM
0015  1075 ;
0015  1076 ;       1. Initialize the CPU.
0015  1077 ;       2. Prepare a loop of 6 iterations.
0015  1078 ;       3. Load a data pattern for the current pass into the RDM D register.
0015  1079 ;       4. Execute the micro-instructions:
0015  1080 ;               1. Load the VA with the pattern from the RDM D register.
0015  1081 ;               2. Read the VA back to the RDM H register.
0015  1082 ;       5. Compare the H register with the correct pattern from the RDM.
0015  1083 ;       6. Signal an error if the two are different.
0015  1084 ;       7. Looped 6 times? If not go to STEP 3.
0015  1085 ;       8. Test completed.
0015  1086 ;
0015  1087 ; DATA PATTERNS
0015  1088 ;
0015  1089 ;       Each of the following six longword hex patterns is used to perform
0015  1090 ;       the test and are listed in the order of occurance.
0015  1091 ;
0015  1092 ;               Loop Count                      VA DATA
0015  1093 ;
0015  1094 ;                   1                       A A A A A A A A
0015  1095 ;                   2                       5 5 5 5 5 5 5 5
0015  1096 ;                   3                       3 3 3 3 3 3 3 3
0015  1097 ;                   4                       0 F 0 F 0 F 0 F
0015  1098 ;                   5                       0 0 F F 0 0 F F
0015  1099 ;                   6                       0 0 0 0 F F F F
0015  1100 ;
0015  1101 ; LOGIC DESCRIPTION
0015  1102 ;
0015  1103 ;       The VA register is a longword that is bit sliced on four ADD gate
0015  1104 ;       arrays. The enable to the VA register comes from the ADK gate array.
0015  1105 ;       Outputs from the ADD gate arrays can be passed to the MBUS through
0015  1106 ;       an output mulitplexer on the MDR gate arrays.
0015  1107 ;
0015  1108 ; ASSUMPTIONS
0015  1109 ;
0015  1110 ;       This test assumes that the DPM and WBUS are functional.
0015  1111 ;
0015  1112 ; ERROR DESCRIPTION
0015  1113 ;
0015  1114 ;       Defects in the ADD gate array will most likely cause stuck bits in
0015  1115 ;       the VA register. Defects in the ADK gate array can cause the entire
0015  1116 ;       VA register to be invalid, while a defective MDR gate array can
0015  1117 ;       cause the VA register to be off by a single hex digit.
0015  1118 ;
0015  1119 ;       On error this test will call out the ADD, ADK, and MDR gate arrays on
0015  1120 ;       the MIC module. Other data that will be typed out include:
```

```
                     0015  1121 ;
                     0015  1122 ;       Expected Data:
                     0015  1123 ;       Received Data:
                     0015  1124 ;       Loop Count:
                     0015  1125 ;--
                     0015  1126 ;*******************************************************************************
                     0015  1127 ;///////////////////////////////////////////////////////////////////////////////
                     0015  1128
                     0015  1129         INITIALIZE                                  ; INITIALIZE THE CPU
                     0017  1130         LOOP            I,1,6                       ; LOOP FOR SIX TEST PATTERNS
                     001E  1131         SA01PAT         I,1$,,                      ; SELECTS A TEST PATTERN
                     0026  1132         LOADREG         WDREG,1$,,LONG              ; LOAD RDM WITH TEST DATA
                     002E  1133         ERRLOOP                                     ; ERROR LOOP USING SAME DATA
                     0030  1134         FETCH           0                           ; WRITE AND THEN READ THE VA
                     0035  1135         BRSTCLK         3                           ; REGISTER
                     0039  1136         CMPREG          WHREG,1$,,LONG              ; VA REGISTER = EXPECTED DATA?
                     0042  1137         IFERROR         ,<<ADD><ADK><MDR>>,         ; IF NOT SIGNAL AN ERROR
                     004A  1138         ENDLOOP         I                           ; OTHERWISE CONTINUE THE LOOP
                     004D  1139         SKIP                                        ; TEST COMPLETED
                     0054  1140
                     0054  1141 BEGIN_TEST_DATA
                     0054
                     0054       ;-------------------------------------------------------------------------
                     0054  1142
                00000000  0054  1143 1$:     .LONG   0                           ; TEST DATA
                     0058  1144
                     0058  1145 BEGIN_CPU_DATA
                     0002  1146
                     0002  1147 DCS_DATA
                     0001  1148
U 00, 7700,C800,0364,0300,0470,1801  0001  1149 ;% 00   NOP
U 01, 5700,C800,0364,0300,4A70,1802  000C  1153 ;% 01   VA_RDM                       ; WRITE THE VA
U 02, 6700,081B,0024,03D8,0470,1803  0017  1157 ;% 02   RDM_VA                      ; READ THE VA BACK
U 03, 7300,C800,0364,0300,0470,1804  0022  1161 ;% 03   STOP.CLOCK                  ; STOP
                     002D  1165
                     002D  1166 END_CPU_DATA
                     0058  1167
                     0058  1168 END_TEST_DATA
                     0058
                     0058       ;-------------------------------------------------------------------------
                     0058  1169
                     0058  1170 ENDTEST
```

```
0015          .SBTTL  "TEST   009     PC REGISTER TEST
0015  1172 ;*******************************************************************************
0015  1173 ;++
0015  1174 ;
0015  1175 ; FUNCTIONAL DESCRIPTION
0015  1176 ;
0015  1177 ;       This test checks for the data integrity of the PC register by loading
0015  1178 ;       and reading six longword patterns.
0015  1179 ;
0015  1180 ; TEST ALGORITHM
0015  1181 ;
0015  1182 ;       1. Initialize the CPU.
0015  1183 ;       2. Prepare a loop of 6 iterations.
0015  1184 ;       3. Load a data pattern for the current pass into the RDM D register.
0015  1185 ;       4. Execute the micro-instructions:
0015  1186 ;               1. Load the PC with the pattern from the RDM D register.
0015  1187 ;               2. Read the PC back to the RDM H register.
0015  1188 ;       5. Compare the H register with the correct pattern from the RDM.
0015  1189 ;       6. Signal an error if the two are different.
0015  1190 ;       7. Looped 6 times? If not go to STEP 3.
0015  1191 ;       8. Test completed.
0015  1192 ;
0015  1193 ; DATA PATTERNS
0015  1194 ;
0015  1195 ;       Each of the following six longword hex patterns is used to perform
0015  1196 ;       the test and are listed in the order of occurance.
0015  1197 ;
0015  1198 ;               Loop Count                      PC DATA
0015  1199 ;
0015  1200 ;                   1                   A A A A A A A A
0015  1201 ;                   2                   5 5 5 5 5 5 5 5
0015  1202 ;                   3                   3 3 3 3 3 3 3 3
0015  1203 ;                   4                   0 F 0 F 0 F 0 F
0015  1204 ;                   5                   0 0 F F 0 0 F F
0015  1205 ;                   6                   0 0 0 0 F F F F
0015  1206 ;
0015  1207 ; LOGIC DESCRIPTION
0015  1208 ;
0015  1209 ;       The PC register is a longword that is bit sliced on four ADD gate
0015  1210 ;       arrays. The enable to the PC register comes from the ADK gate array.
0015  1211 ;       Outputs from the ADD gate arrays can be passed to the MBUS through
0015  1212 ;       an output multiplexer on the MDR gate arrays.
0015  1213 ;
0015  1214 ; ASSUMPTIONS
0015  1215 ;
0015  1216 ;       This test assumes that the DPM, WBUS and MBUS are functional.
0015  1217 ;
0015  1218 ; ERROR DESCRIPTION
0015  1219 ;
0015  1220 ;       Defects in the ADD gate array will most likely cause stuck bits in
0015  1221 ;       the PC register. Defects in the ADK gate array can cause the entire
0015  1222 ;       PC register to be invalid, while a defective MDR gate array can cause
0015  1223 ;       the PC register to be off by a single hex digit.
0015  1224 ;
0015  1225 ;       On error this test will call out the ADD, ADK and MDR gate arrays on
```

```
                        0015  1226 ;       the MIC module and will type out the following information:
                        0015  1227 ;
                        0015  1228 ;       Expected Data:
                        0015  1229 ;       Received Data:
                        0015  1230 ;       Loop Count:
                        0015  1231 ;--
                        0015  1232 ;*******************************************************************************
                        0015  1233 ;///////////////////////////////////////////////////////////////////////////////
                        0015  1234
                        0015  1235         INITIALIZE                                      ; INITIALIZE THE CPU
                        0017  1236         LOOP            I,1,6                           ; SET UP FOR SIX ITERATIONS
                        001E  1237         SA01PAT         I,1$,,                          ; SELECTS STANDARD PATTERN
                        0026  1238         LOADREG         WDREG,1$,,LONG                  ; LOAD TEST DATA INTO WDREG
                        002E  1239         ERRLOOP                                         ; ERROR LOOP USING SAME DATA
                        0030  1240         FETCH           0                               ; FETCH MICRO-INSTRUCTIONS
                        0035  1241         BRSTCLK         3                               ; EXECUTE MICRO-INSTRUCTIONS
                        0039  1242         CMPREG          WHREG,1$,,LONG                  ; DOES WHREG = CORRECT PATTERN?
                        0042  1243         IFERROR         ,<<ADD><ADK><MDR>>,             ; IF NOT SIGNAL AN ERROR
                        004A  1244         ENDLOOP         I                               ; OTHERWISE CONTINUE LOOP
                        004D  1245         SKIP                                            ; TEST COMPLETED
                        0054  1246
                        0054  1247 BEGIN_TEST_DATA
                        0054
                        0054       ;-------------------------------------------------------------------------
                        0054  1248
                00000000  0054  1249 1$:     .LONG   0                                     ; TEST DATA LOCATION
                        0058  1250
                        0058  1251 BEGIN_CPU_DATA
                        0002  1252
                        0002  1253 DCS_DATA
                        0001  1254
U 00, 7700,C800,0364,0300,0470,1801     0001  1255 ;% 00    NOP                             ; PLACES CPU INTO NATIVE MODE
U 01, 5700,4800,0364,0300,4870,1802     000C  1259 ;% 01    PC_RDM                          ; WRITES TEST DATA INTO PC
U 02, 6700,481A,0024,03D8,0470,1803     0017  1263 ;% 02    RDM_PC                          ; READS THE PC BACK TO THE RDM
U 03, 7300,C800,0364,0300,0470,1804     0022  1267 ;% 03    STOP.CLOCK                      ; HALT
                        002D  1271
                        002D  1272 END_CPU_DATA
                        0058  1273
                        0058  1274 END_TEST_DATA
                        0058
                        0058       ;-------------------------------------------------------------------------
                        0058  1275
                        0058  1276 ENDTEST
```

```
0016        .SBTTL  "TEST   00A     TEST WDR REGISTER
0016  1278 ;*******************************************************************************
0016  1279 ;++
0016  1280 ;
0016  1281 ; FUNCTIONAL DESCRIPTION:
0016  1282 ;
0016  1283 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE WDR REGISTER
0016  1284 ;       BY LOADING AND READING 6 UNROTATED DATA PATTERNS THROUGH
0016  1285 ;       THE REGISTER.
0016  1286 ;
0016  1287 ;
0016  1288 ; TEST ALGORITHM:
0016  1289 ;
0016  1290 ;       1.  SET UP A LOOP OF SIX ITERATIONS
0016  1291 ;       3.  GENERATE A TEST PATTERN
0016  1292 ;       4.  LOAD PATTERN INTO WD REGISTER (RDM)
0016  1293 ;       5.  EXECUTE DCS PROGRAM
0016  1294 ;               1.  LOAD TEST PATTERN INTO WDR REGISTER
0016  1295 ;               2.  READ TEST PATTERN BACK TO WH REGISTER (RDM)
0016  1296 ;       6.  COMPARE DATA (EXPECTED AND RECEIVED)
0016  1297 ;       7.  IF NO ERROR GO TO STEP 3 FOR REMAINING TEST PATTERNS
0016  1298 ;
0016  1299 ;
0016  1300 ; TEST PATTERNS:
0016  1301 ;
0016  1302 ;       ITERATION       TEST PATTERN
0016  1303 ;           1             AAAAAAAA
0016  1304 ;           2             55555555
0016  1305 ;           3             33333333
0016  1306 ;           4             0F0F0F0F
0016  1307 ;           5             00FF00FF
0016  1308 ;           6             0000FFFF
0016  1309 ;
0016  1310 ;
0016  1311 ; LOGIC DESCRIPTION:
0016  1312 ;
0016  1313 ;       THIS TEST IS DESIGNED TO PRIMARILY CHECK THE WDR REGISTER ON THE
0016  1314 ;       MIC MODULE.  IN DOING SO OTHER GATE ARRAYS, USED FOR CONTROL, WILL
0016  1315 ;       COME INTO PLAY.  ADK CONTROLS THE WDR SELECT AND PRK CONTROLS THE
0016  1316 ;       MMUX.  IF ERRORS OCCUR, AND REPLACING MDR DOES NOT FIX THE PROBLEM,
0016  1317 ;       THEN THESE OTHER ARRAYS SHOULD BE TAKEN INTO CONSIDERATION.
0016  1318 ;
0016  1319 ;
0016  1320 ; ASSUMPTIONS:
0016  1321 ;
0016  1322 ;       THIS TEST ASSUMES THE DPM, WBUS, AND MBUS ARE OPERATIONAL.
0016  1323 ;
0016  1324 ;
0016  1325 ; ERROR DESCRIPTION:
0016  1326 ;
0016  1327 ;       EXPECTED WDR DATA
0016  1328 ;       RECEIVED WDR DATA
0016  1329 ;       LOOP COUNT
0016  1330 ;       ANALYSIS OF WHICH MDR BIT SLICE FAILED
0016  1331 ;
```

```
                     0016  1332 ;--
                     0016  1333 ;*******************************************************************************
                     0016  1334 ;///////////////////////////////////////////////////////////////////////////////
                     0016  1335
                     0016  1336         INITIALIZE                                          ;INIT CPU
                     0018  1337         LOOP            I,1,6                               ;CREATE TEST LOOP
                     001F  1338         SA01PAT         I,1$,,                              ;CREATE TEST PATTERN
                     0027  1339         LOADREG         WDREG,1$,,L                         ;TEST PATTERN TO WD REG
                     002F  1340         ERRLOOP                                             ;ERROR LOOP
                     0031  1341         FETCH           0                                   ;START DCS PROGRAM
                     0036  1342         BRSTCLK         3                                   ;END DCS PROGRAM
                     003A  1343         CMPREG          WHREG,1$,,L,                        ;COMPARE RESULTS
                     0043  1344         IFERROR         ,<<MDR><ADK><PRK>>,
                     004B  1345         ENDLOOP         I                                   ;END TEST LOOP
                     004E  1346         SKIP                                                ;GO TO NEXT TEST
                     0055  1347
                     0055  1348 BEGIN_TEST_DATA
                     0055
                     0055          ;-------------------------------------------------------------------------
                     0055  1349
                00000000  0055  1350 1$:      .LONG   ^X00000000                          ;TEST PATTERN STORAGE
                     0059  1351
                     0059  1352 BEGIN_CPU_DATA
                     0002  1353
                     0002  1354 DCS_DATA
                     0001  1355
U 00, 7700,C800,0364,0300,0470,1801  0001  1356 ;% 00    NOP
U 01, 5700,C800,0364,0300,5470,1802  000C  1360 ;% 01    WB_RDM,WCTRL/WDR_WB.UR              ;LOAD WDR FROM RDM
U 02, 6700,4813,5924,0300,4C70,1803  0017  1364 ;% 02    RDM_WB,WB_M[WDR],WCTRL/MBUS_WDR     ;READ WDR TO RDM
U 03, 7300,C800,0364,0300,0470,1804  0022  1368 ;% 03    NOP,STOP.CLOCK                      ;STOP CPU CLOCK
                     002D  1372
                     002D  1373 END_CPU_DATA
                     0059  1374
                     0059  1375 END_TEST_DATA
                     0059
                     0059          ;-------------------------------------------------------------------------
                     0059  1376
                     0059  1377 ENDTEST
```

```
0015       .SBTTL  "TEST   00B     TEST WDR ROTATED
0015  1379 ;*******************************************************************************
0015  1380 ;++
0015  1381 ;
0015  1382 ; FUNCTIONAL DESCRIPTION:
0015  1383 ;
0015  1384 ;       THIS TEST CHECKS THE ABILITY OF THE DBUS ROTATOR TO ROTATE AND
0015  1385 ;       LOAD TEST PATTERNS INTO THE WRITE DATA REGISTER.
0015  1386 ;
0015  1387 ;
0015  1388 ; TEST ALGORITHM:
0015  1389 ;
0015  1390 ;       1.  LOAD DCS PROGRAM WITH TEST PATTERN (SUBTEST #2)
0015  1391 ;       2.  CREATE A TEST LOOP OF 4 ITERATIONS
0015  1392 ;       3.  SELECT AN ADDRESS FOR THE VA REGISTER AND LOAD INTO
0015  1393 ;           WD REGISTER (RDM)
0015  1394 ;       4.  EXECUTE DCS PROGRAM
0015  1395 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
0015  1396 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
0015  1397 ;               2.  TURN MEMORY MANAGEMENT OFF
0015  1398 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
0015  1399 ;               4.  TURN CMI OFF
0015  1400 ;               5.  LOAD TEST PATTERN INTO LONGLIT REGISTER
0015  1401 ;               6.  LOAD VA REGISTER WITH ADDRESS IN RDM WD REGISTER (CONTROLS
0015  1402 ;                   ROTATE OF DBUS)
0015  1403 ;               7.  WRITE TEST PATTERN IN LONGLIT REGISTER TO WDR REGISTER
0015  1404 ;               8.  READ TEST PATTERN BACK FROM WDR TO RDM
0015  1405 ;       5.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0015  1406 ;       6.  IF NO ERROR GO TO STEP 3 UNTIL TEST LOOP IS EXHAUSTED
0015  1407 ;
0015  1408 ;
0015  1409 ; TEST PATTERNS:
0015  1410 ;
0015  1411 ;       SUBTEST #1  000000FF
0015  1412 ;       SUBTEST #2  FFFFFF00
0015  1413 ;
0015  1414 ;       ITERATION       # OF BYTES ROTATED
0015  1415 ;
0015  1416 ;           1                   0
0015  1417 ;           2                   1
0015  1418 ;           3                   2
0015  1419 ;           4                   3
0015  1420 ;
0015  1421 ;
0015  1422 ; LOGIC DESCRIPTION:
0015  1423 ;
0015  1424 ;       THIS TEST ATTEMPTS TO CHECK THE DBUS ROTATOR LOCATED WITHIN THE MDR
0015  1425 ;       GATE ARRAY.  TWO TEST PATTERNS ARE ROTATED THROUGH ALL BYTE POSITIONS
0015  1426 ;       AND LOADED INTO THE WRITE DATA REGISTER.  THE "MDR" SHOULD BE THE
0015  1427 ;       MOST LIKELY GATE ARRAY TO FAIL.  SECOND CHOICE SHOULD BE "CAK" WHICH
0015  1428 ;       CONTROLS THE ROTATOR.  THE LAST POSSIBILITY SHOULD BE "ADD" WHICH
0015  1429 ;       HANDLES THE ADDRESSING.
0015  1430 ;
0015  1431 ;
0015  1432 ; ASSUMPTIONS:
```

```
                    1433 ;
                    1434 ;       THIS TEST ASSUMES THE WBUS, MBUS, AND VA REGISTER ARE OPERATIONAL
                    1435 ;
                    1436 ;
                    1437 ; ERROR DESCRIPTION:
                    1438 ;
                    1439 ;       EXPECTED WDR DATA
                    1440 ;       RECEIVED WDR DATA
                    1441 ;       LOOP COUNT
                    1442 ;
                    1443 ;--
                    1444 ;*******************************************************************************
                    1445
                    1446         SUBTEST                                        ;SUBTEST #1
                    0018
                         ;////////////////////////////////////////////////////////////////////////////////
                    1447 ;+
                    1448 ;SUBTEST #1
                    1449 ;
                    1450 ;       TEST WRITE DATA REGISTER USING TEST PATTERN 000000FF
                    1451 ;-
                    1452         INITIALIZE                                     ;INIT CPU
                    1453         LOADDCS         5$,,05,1                       ;MODIFY DCS ADD 05
                    1454         LOOP            I,1,4                          ;CREATE A TEST LOOP
                    1455         LOADREG         WDREG,1$,I,L                   ;VA ADDRESS TO WD REGISTER
                    1456         ERRLOOP                                        ;ERROR LOOP
                    1457         FETCH           0                              ;START DCS PROGRAM
                    1458         BRSTCLK         9                              ;END DCS PROGRAM
                    1459         CMPREG          WHREG,2$,I,L                   ;COMPARE RESULTS
                    1460         IFERROR         ,<<MDR><CAK><ADD>>,
                    1461         ENDLOOP         I                              ;END TEST LOOP
                    1462
                    1463         SUBTEST                                        ;SUBTEST #2
                    0052
                         ;////////////////////////////////////////////////////////////////////////////////
                    1464 ;+
                    1465 ;SUBTEST #2
                    1466 ;
                    1467 ;       TEST WRITE DATA REGISTER USING TEST PATTERN FFFFFF00
                    1468 ;-
                    1469         INITIALIZE                                     ;INIT CPU
                    1470         LOADDCS         3$,,05,1                       ;TEST PATTERN TO LONGLIT REG
                    1471         LOOP            I,1,4                          ;CREATE A TEST LOOP
                    1472         LOADREG         WDREG,1$,I,L                   ;VA ADDRESS TO WD REGISTER
                    1473         ERRLOOP                                        ;ERROR LOOP
                    1474         FETCH           0                              ;START DCS PROGRAM
                    1475         BRSTCLK         9                              ;END DCS PROGRAM
                    1476         CMPREG          WHREG,4$,I,L                   ;COMPARE RESULTS
                    1477         IFERROR         ,<<MDR><CAK><ADD>>,
                    1478         ENDLOOP         I                              ;END TEST LOOP
                    1479         SKIP                                           ;GO TO NEXT TEST
                    1480
                    1481 BEGIN_TEST_DATA
                    0090
                         ;-------------------------------------------------------------------------------
```

```
                                  0090
                        00000000  0090  1$:     .LONG   ^X00000000                  ;VA ADDRESS TABLE
                        00000001  0094          .LONG   ^X00000001
                        00000002  0098          .LONG   ^X00000002
                        00000003  009C          .LONG   ^X00000003
                                  00A0
                        000000FF  00A0  2$:     .LONG   ^X000000FF                  ;RESULTS FOR SUBTEST #1
                        0000FF00  00A4          .LONG   ^X0000FF00
                        00FF0000  00A8          .LONG   ^X00FF0000
                        FF000000  00AC          .LONG   ^XFF000000
                                  00B0  3$:
U 05, 7700,F800,0000,007F,8470,1806  00B0  ;% 05   LONLIT_[0FFFFFF00]                  ;TEST PATTERN FOR SUBTEST #2
                                  00BB
                        FFFFFF00  00BB  4$:     .LONG   ^XFFFFFF00                  ;RESULTS FOR SUBTEST #2
                        FFFF00FF  00BF          .LONG   ^XFFFF00FF
                        FF00FFFF  00C3          .LONG   ^XFF00FFFF
                        00FFFFFF  00C7          .LONG   ^X00FFFFFF
                                  00CB  5$:
U 05, 7700,3800,7FFF,FF80,0470,1806  00CB  ;% 05   LONLIT_[000000FF]                   ;TEST PATTERN FOR SUBTEST #1
                                  00D6
                                  00D6  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  ;% 01   MEMSCAR_R[TEMP0]                    ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  ;% 02   MEMSCR_R[TEMP1]                     ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  ;% 03   MEMSCAR_R[TEMP2]                    ;REF CACHE ONLY ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805  002D  ;% 04   MEMSCR_R[TEMP3]                     ;TURN CMI OFF
U 05, 7700,3800,7FFF,FF80,0470,1806  0038  ;% 05   LONLIT_[000000FF]                   ;TEST PATTERN TO LONGLIT REG
U 06, 5700,C800,0364,0300,4A70,1807  0043  ;% 06   VA_RDM                              ;ADDRESS TO VIRTUAL ADD REG
U 07, 7700,4800,5BE4,03D4,5C70,1808  004E  ;% 07   WB_R[LONLIT],WCTRL/WDR_WB           ;TEST PATTERN TO WDR
U 08, 6700,4813,5924,0300,4C70,1809  0059  ;% 08   RDM_WB,WCTRL/MBUS_WDR,WB_M[WDR]     ;READ WDR TO RDM
U 09, 7300,4800,0364,0300,0470,180A  0064  ;% 09   NOP,STOP.CLOCK                      ;STOP CPU CLOCK
                                  006F
                                  006F  RTEMP_DATA
                                  0001
                        00000000  0001          .LONG   ^X00000000                  ;PHY/VIR REG ADD
                        00000000  0005          .LONG   ^X00000000                  ;SET MME OFF
                        0E000000  0009          .LONG   ^X0E000000                  ;REF CACHE ONLY ADD
                        01000000  000D          .LONG   ^X01000000                  ;TURN CMI OFF
                                  0011
                                  0011  END_CPU_DATA
                                  00D6
                                  00D6  END_TEST_DATA
                                  00D6
                                  00D6      ;-----------------------------------------------------------------------
                                  00D6
                                  00D6  ENDTEST
```

```
                     0031        .SBTTL  "TEST  00C     TEST ASYNCHRONOUS SYSTEM TRAP LEVEL REGISTER
                     0031  1566 ;*******************************************************************************
                     0031  1567 ;++
                     0031  1568 ;
                     0031  1569 ; FUNCTIONAL DESCRIPTION:
                     0031  1570 ;
                     0031  1571 ;       THIS TEST CHECKS THE DATA INTEGRITY OF THE ASYCHRONOUS SYSTEM TRAP
                     0031  1572 ;       LEVEL REGISTER (ASTLVL) BY WRITING AND READING 3 TEST PATTERNS
                     0031  1573 ;       THROUGH THE REGISTER.
                     0031  1574 ;
                     0031  1575 ;
                     0031  1576 ; TEST ALGORITHM:
                     0031  1577 ;
                     0031  1578 ;       1.  CREATE A LOOP OF 3 ITERATIONS
                     0031  1579 ;       2.  SELECT A TEST PATTERN AND LOAD INTO D REGISTER (RDM)
                     0031  1580 ;       3.  EXECUTE DCS PROGRAM
                     0031  1581 ;               1.  WRITE TEST PATTERN INTO ASTLVL
                     0031  1582 ;               2.  READ TEST PATTERN BACK TO H REGISTER (RDM)
                     0031  1583 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                     0031  1584 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                     0031  1585 ;
                     0031  1586 ;
                     0031  1587 ; TEST PATTERNS:
                     0031  1588 ;
                     0031  1589 ;       ITERATION       TEST PATTERN
                     0031  1590 ;           1                5
                     0031  1591 ;           2                2
                     0031  1592 ;           3                1
                     0031  1593 ;
                     0031  1594 ;
                     0031  1595 ; LOGIC DESCRIPTION:
                     0031  1596 ;
                     0031  1597 ;       THIS TEST CHECKS THE ASYCHRONOUS SYSTEM TRAP LEVEL REGISTER WHICH
                     0031  1598 ;       IS IN THE INT GATE ARRAY ON THE UBI MODULE.  THE REGISTER IS WRITTEN
                     0031  1599 ;       AND READ VIA THE WBUS<26:24>.  ALL CONTROL LOGIC FOR THE REGISTER
                     0031  1600 ;       IS CONTAINED IN THE INT CHIP.
                     0031  1601 ;
                     0031  1602 ;
                     0031  1603 ; ASSUMPTIONS:
                     0031  1604 ;
                     0031  1605 ;       THIS TEST ASSUMES THE WBUS IS OPERATIONAL.
                     0031  1606 ;
                     0031  1607 ;
                     0031  1608 ; ERROR DESCRIPTION:
                     0031  1609 ;
                     0031  1610 ;       EXPECTED ASTLVL DATA
                     0031  1611 ;       RECEIVED ASTLVL DATA
                     0031  1612 ;       LOOP COUNT
                     0031  1613 ;
                     0031  1614 ;--
                     0031  1615 ;*******************************************************************************
                     0031  1616 ;///////////////////////////////////////////////////////////////////////////////
                     0031  1617
                     0031  1618         INITIALIZE                                      ;INIT CPU
                     0033  1619         LOOP            I,1,3                           ;CREATE TEST LOOP
```

```
                                  003A          LOADREG         WDREG,1$,I,L            ;TEST PATTERN TO D REGISTER
                                  0042          ERRLOOP                                 ;ERROR LOOP
                                  0044          FETCH           0                       ;START DCS PROGRAM
                                  0049          BRSTCLK         3                       ;END DCS PROGRAM
                                  004D          CMPREGMSK       WHREG,1$,I,2$,,L,       ;COMPARE RESULTS
                                  0059          IFERROR         ,<<INT>>,<<UBI>>        ;POSSIBLE FAILING ARRAY
                                  0060          ENDLOOP         I                       ;END TEST LOOP
                                  0063          SKIP
                                  006A
                                  006A  BEGIN_TEST_DATA
                                  006A
                                  006A      ;-----------------------------------------------------------------------
                                  006A
                        05000000  006A  1$:     .LONG   ^X05000000                      ;TEST PATTERN
                        02000000  006E          .LONG   ^X02000000                      ;  "      "
                        01000000  0072          .LONG   ^X01000000                      ;  "      "
                                  0076
                        07000000  0076  2$:     .LONG   ^X07000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                                  007A
                                  007A  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,C800,0364,0300,7070,1802  000C  ;% 01   WB_RDM,WCTRL/ASTLVL_WB                  ;WRITE TEST PATTERN TO ASTLVL
U 02, 6700,C800,0364,0300,7470,1803  0017  ;% 02   RDM_WB,WCTRL/ASTLVL                     ;READ TEST PATTERN TO RDM
U 03, 7300,C800,0364,0300,0470,1804  0022  ;% 03   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
                                  002D
                                  002D  END_CPU_DATA
                                  007A
                                  007A  END_TEST_DATA
                                  007A
                                  007A      ;-----------------------------------------------------------------------
                                  007A
                                  007A  ENDTEST
```

```
0019        .SBTTL  "TEST   00D     PC BACKUP LATCH TEST
0019  1664 ;*******************************************************************************
0019  1665 ;++
0019  1666 ;
0019  1667 ; FUNCTIONAL DESCRIPTION
0019  1668 ;
0019  1669 ;       This test examines the data integrity of the PC BACKUP latch by
0019  1670 ;       loading and reading six longword patterns. The patterns to be used
0019  1671 ;       must be two less than the expected result pattern and have to be
0019  1672 ;       loaded into the PC. This is because the PC BACKUP latch can only be
0019  1673 ;       written with PC+2 when an IRD1 is executed. The PC BACKUP latch can
0019  1674 ;       however, be read directly onto the MBUS for examination.
0019  1675 ;
0019  1676 ; TEST ALGORITHM
0019  1677 ;
0019  1678 ;       1. Initialize the CPU.
0019  1679 ;       2. Prepare a loop of six iterations.
0019  1680 ;       3. Load a data pattern for the current pass into the RDM D register.
0019  1681 ;       4. Execute micro-instructions to:
0019  1682 ;               1. Load the PC with a pattern from the RDM D register.
0019  1683 ;               2. Load PC BACKUP with PC+2 by executing a BUT/IRD1TST.
0019  1684 ;               3. Read PC BACKUP back to the RDM H register.
0019  1685 ;       5. Compare the H register with the correct pattern from SA01PAT.
0019  1686 ;       6. If the two are different signal an error.
0019  1687 ;       7. Looped 6 times? If not go to STEP 3.
0019  1688 ;       8. Test completed.
0019  1689 ;
0019  1690 ; DATA PATTERNS
0019  1691 ;
0019  1692 ;       Each of the following six longword hex patterns is used to perform
0019  1693 ;       the test and are listed in the order of occurance:
0019  1694 ;
0019  1695 ;       Loop Count          PC  DATA          EXPECTED PC BACKUP DATA
0019  1696 ;
0019  1697 ;           1           A A A A A A A 8         A A A A A A A A
0019  1698 ;           2           5 5 5 5 5 5 5 3         5 5 5 5 5 5 5 5
0019  1699 ;           3           3 3 3 3 3 3 3 1         3 3 3 3 3 3 3 3
0019  1700 ;           4           0 F 0 F 0 F 0 D         0 F 0 F 0 F 0 F
0019  1701 ;           5           0 0 F F 0 0 F D         0 0 F F 0 0 F F
0019  1702 ;           6           0 0 0 0 F F F D         0 0 0 0 F F F F
0019  1703 ;
0019  1704 ; LOGIC DESCRIPTION
0019  1705 ;
0019  1706 ;       The PC BACKUP latch is a longword that is bit sliced on four ADD
0019  1707 ;       gate arrays. The enable to the PC BACKUP latch comes from a 74S00
0019  1708 ;       on the MIC module. (Printset: MIC-04 E26) The MA mux select needed to
0019  1709 ;       read the PC BACKUP latch from the ADD chips is 1 hex. The select
0019  1710 ;       lines for the MA mux come from the PRK gate array on the MIC module.
0019  1711 ;
0019  1712 ;
0019  1713 ; ASSUMPTIONS
0019  1714 ;
0019  1715 ;       This test assumes that the DPM, WBUS and MBUS are functional.
0019  1716 ;
0019  1717 ; ERROR DESCRIPTION
```

```
                                0019  1718 ;
                                0019  1719 ;       Defects in the ADD gate array will most likely cause stuck bits in the
                                0019  1720 ;       PC BACKUP latch. Defects in the ASRC decode logic on the MIC module
                                0019  1721 ;       can cause an improper increment of the PC to be placed into the PC
                                0019  1722 ;       BACKUP latch. A defective ADK gate array can cause the PC BACKUP
                                0019  1723 ;       latch to be the +2 increment of the VA or PC registers.
                                0019  1724 ;
                                0019  1725 ;       On error this test will call out the ADD, ADK and PRK gate arrays on
                                0019  1726 ;       the MIC module. The following information will also typed out:
                                0019  1727 ;
                                0019  1728 ;       Expected Data:
                                0019  1729 ;       Received Data:
                                0019  1730 ;       Loop Count:
                                0019  1731 ;--
                                0019  1732 ;*******************************************************************************
                                0019  1733 ;///////////////////////////////////////////////////////////////////////////////
                                0019  1734
                                0019  1735         INITIALIZE                                      ; INITIALIZE THE CPU
                                001B  1736         EXPUTRAP                                        ; EXPECT U-TRAP ON BUT/IRD1TST
                                001D  1737         LOOP            I,1,6                           ; LOOP FOR SIX ITERATIONS
                                0024  1738         LOADREG         WDREG,1$,I,LONG                 ; LOAD ADDRESS INTO RDM WDREG
                                002C  1739         SA01PAT         I,2$                            ; EXPECTED DATA PATTERN
                                0034  1740         ERRLOOP                                         ; ERROR LOOP USING SAME DATA
                                0036  1741         FETCH           0                               ; EXECUTE PC LOAD WITH TRAP
                                003B  1742         BRSTCLK         5,INH                           ;
                                003F  1743         FETCH           3                               ; EXECUTE PCBACK READ
                                0044  1744         BRSTCLK         5,INH                           ;
                                0048  1745         CMPREG          WHREG,2$,,LONG                  ; DOES WHREG = CORRECT PATTERN?
                                0051  1746         IFERROR         ,<<ADD><ADK><PRK>>,             ; IF NOT SIGNAL AN ERROR
                                0059  1747         ENDLOOP         I                               ; OTHERWISE CONTINUE LOOP
                                005C  1748         SKIP                                            ; TEST COMPLETED
                                0063  1749
                                0063  1750 BEGIN_TEST_DATA
                                0063
                                0063       ;-------------------------------------------------------------------------
                                0063  1751
                    AAAAAAA8    0063  1752 1$:      .LONG    ^X AAAAAAA8
                    55555553    0067  1753         .LONG    ^X 55555553
                    33333331    006B  1754         .LONG    ^X 33333331
                    0F0F0F0D    006F  1755         .LONG    ^X 0F0F0F0D
                    00FF00FD    0073  1756         .LONG    ^X 00FF00FD
                    0000FFFD    0077  1757         .LONG    ^X 0000FFFD
                                007B  1758
                    00000000    007B  1759 2$:      .LONG    0
                                007F  1760
                                007F  1761 BEGIN_CPU_DATA
                                0002  1762
                                0002  1763 DCS_DATA
                                0001  1764
U 00, 7700,C800,0364,0300,0470,1801 0001  1765 ;% 00    NOP
U 01, 5700,4800,0364,0300,4870,1802 000C  1769 ;% 01    PC_RDM
U 02, 7700,4800,0364,1700,0470,1803 0017  1773 ;% 02    BUT/IRD1TST
U 03, 7700,C800,0364,0300,0470,1804 0022  1777 ;% 03    NOP
U 04, 6700,4819,0024,03D8,0470,1805 002D  1781 ;% 04    RDM_PCBACK
U 05, 7300,4800,0364,0300,0470,1806 0038  1785 ;% 05    NOP,STOP.CLOCK
```

```
                0043  1789
                0043  1790 END_CPU_DATA
                007F  1791
                007F  1792 END_TEST_DATA
                007F
                007F        ;-----------------------------------------------------------------------------
                007F  1793
                007F  1794 ENDTEST
```

```
0011          .SBTTL  "TEST   00E     PC_PC+1 TEST
0011   1796 ;*******************************************************************************
0011   1797 ;++
0011   1798 ;
0011   1799 ; FUNCTIONAL DESCRIPTION
0011   1800 ;
0011   1801 ;       This test will examine the PC_PC+1 address path by first loading the
0011   1802 ;       PC register with a zero and then incrementing it by one. It then reads
0011   1803 ;       back the new value of the PC and compares it with the expected value
0011   1804 ;       stored in the RDM. If the two are not equal an error has occurred.
0011   1805 ;
0011   1806 ; TEST ALGORITHM
0011   1807 ;
0011   1808 ;       1. Initialize the CPU.
0011   1809 ;       2. Load the RDM D register with zero.
0011   1810 ;       3. Execute the micro-instructions:
0011   1811 ;               1. PC_RDM       ; PC_0
0011   1812 ;               2. PC_PC+1      ; Increment the PC
0011   1813 ;               3. RDM_PC       ; Read the PC back to the RDM H register
0011   1814 ;       4. Compare the RDM H register with a constant 1.
0011   1815 ;       5. Signal an error if the H register does not equal one.
0011   1816 ;       6. Test completed.
0011   1817 ;
0011   1818 ; DATA PATTERNS
0011   1819 ;
0011   1820 ;       These are the data patterns to be used in this test:
0011   1821 ;
0011   1822 ;               PC INPUT                RESULT
0011   1823 ;
0011   1824 ;           0 0 0 0 0 0 0 0         0 0 0 0 0 0 0 1
0011   1825 ;
0011   1826 ; LOGIC DESCRIPTION
0011   1827 ;
0011   1828 ;       This test examines the PC_PC+1 address path for functionality only.
0011   1829 ;       This test is implemented by using the micro-instruction PC_PC+1 MB_XB
0011   1830 ;       which is a macro defined for the following instruction:
0011   1831 ;
0011   1832 ;               MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,DTYPE/BYTE
0011   1833 ;
0011   1834 ;       The MSRC micro-order XB.PC_PC+I is used by this test to increment the
0011   1835 ;       PC. Although this micro-instruction also sources an XB onto the MBUS,
0011   1836 ;       this operation is treated as a side effect and is ignored. The PC
0011   1837 ;       increment is normally instruction dependent but for this test it is
0011   1838 ;       determined by the DTYPE micro-field by setting the ISTRM bit in the
0011   1839 ;       micro-word. This is accomplished by using the ISTRM/ISIZE_DSIZE micro-
0011   1840 ;       order. Once this has been done the PC can be incremented by a constant
0011   1841 ;       1, 2 or 4 depending on the value of DTYPE. To obtain an increment of 1
0011   1842 ;       the micro-order DTYPE/BYTE must be used.
0011   1843 ;
0011   1844 ; ASSUMPTIONS
0011   1845 ;
0011   1846 ;       This test assumes that the DPM, WBUS and MBUS are functional, and that
0011   1847 ;       the PC has been successfully tested.
0011   1848 ;
0011   1849 ; ERROR DESCRIPTION
```

```
0011  ;
0011  ;       Below is a table of possible error symptoms and their most likely
0011  ;       causes:
0011  ;
0011  ;               Symptom                         Probable Cause
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Bad carries, and/or   | A defective adder in the ADD slice for|
0011  ;       | stuck bits within a   | that byte. More than one defective ADD|
0011  ;       | particular byte.      | slice will cause groups of bad bytes. |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Bad sums across byte  | A defective 74LS182 chip on the MIC   |
0011  ;       | boundaries due to an  | module. (MIC-03)                      |
0011  ;       | incorrect carry.      |                                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Byte besides the least| The CHIP ID signal to that ADD gate   |
0011  ;       | significant byte of   | array has been accidentally shorted to|
0011  ;       | the PC is being       | ground.                               |
0011  ;       | incremented.          |                                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Random bad sums from  | Defects in the ASRC decode logic. Bad |
0011  ;       | the adder or improper | ASRC selections can cause whatever is |
0011  ;       | PC increments.        | currently on the WBUS to be added to  |
0011  ;       |                       | the PC, or an increment of 0,2 or 4.  |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Sum of the PC is the  | A defective ADK gate array can cause a|
0011  ;       | +1 increment of VA,   | register other than the PC to be      |
0011  ;       | VA SAVE or PC BACKUP. | incremented.                          |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | One particular bad sum| A defective PRK gate array will cause |
0011  ;       | coming from the adder.| the wrong register to be read from the|
0011  ;       |                       | ADD gate array.                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;
0011  ;       On error this test will call out the ADK and ADD gate arrays on the
0011  ;       MIC module and the following information:
0011  ;
0011  ;       Expected Data:
0011  ;       Received Data:
0011  ;
0011  ;--
0011  ;*****************************************************************************
0011  ;/////////////////////////////////////////////////////////////////////////////
0011
0011          INITIALIZE                                  ; INITIALIZE THE CPU
0013          EXPUTRAP                                    ; POSSIBLE FAILURE MODE
0015          LOADREG         WDREG,1$,,LONG              ; RDM WDREG_0
001D          ERRLOOP                                     ; ERROR LOOP
001F          FETCH           0                           ; EXECUTE PC_PC+1
0024          BRSTCLK         5                           ;    "      "
0028          FETCH           3                           ; REEXECUTE INCASE OF U-TRAP
002D          BRSTCLK         5                           ; END DCS PROGRAM
0031          CMPREG          WHREG,2$,,LONG              ; DOES WHREG = CORRECT VALUE?
003A          IFERROR         ,<<ADK><ADD><PRK>>,         ; IF NOT SIGNAL AN ERROR.
0042          SKIP                                        ; TEST COMPLETED
0049
```

```
                                0049  1905 BEGIN_TEST_DATA
                                0049
                                0049       ;-------------------------------------------------------------------------
                                0049  1906
                     00000000   0049  1907 1$:     .LONG   ^X 00000000     ; INPUT DATA PATTERN
                     00000001   004D  1908 2$:     .LONG   ^X 00000001     ; RESULT DATA PATTERN
                                0051  1909
                                0051  1910 BEGIN_CPU_DATA
                                0002  1911
                                0002  1912 DCS_DATA
                                0001  1913
U 00, 7700,C800,0364,0300,0470,1801   0001  1914 ;% 00   NOP
U 01, 5700,4800,0364,0300,4870,1802   000C  1918 ;% 01   PC_RDM
U 02, 7701,C817,0364,0002,0470,1803   0017  1922 ;% 02   PC_PC+1 MB_XB
U 03, 7700,C800,0364,0300,0470,1804   0022  1926 ;% 03   NOP
U 04, 6700,481A,0024,03D8,0470,1805   002D  1930 ;% 04   RDM_PC
U 05, 7300,4800,0364,0300,0470,1806   0038  1934 ;% 05   STOP.CLOCK
                                0043  1938
                                0043  1939 CACHE_DATA
                                0001  1940
                     00000000   0001  1941         .LONG   0
                                0005  1942
                                0005  1943 END_CPU_DATA
                                0051  1944
                                0051  1945 END_TEST_DATA
                                0051
                                0051       ;-------------------------------------------------------------------------
                                0051  1946
                                0051  1947 ENDTEST
```

```
0011          .SBTTL  "TEST   00F     PC_PC+2 TEST
0011  1949  ;*******************************************************************************
0011  1950  ;++
0011  1951  ;
0011  1952  ; FUNCTIONAL DESCRIPTION
0011  1953  ;
0011  1954  ;       This test will examine the PC_PC+2 address path. This test is
0011  1955  ;       implemented by loading the PC with zero and then incrementing it by
0011  1956  ;       two. It then reads back the new value of the PC and compares it with
0011  1957  ;       the expected value stored in the RDM. If the two are not equal an
0011  1958  ;       error has occurred.
0011  1959  ;
0011  1960  ; TEST ALGORITHM
0011  1961  ;
0011  1962  ;       1. Initialize the CPU.
0011  1963  ;       2. Load the RDM D register with zero.
0011  1964  ;       3. Execute the following micro-instructions:
0011  1965  ;               1. PC_RDM       ; PC_0
0011  1966  ;               2. PC_PC+2      ; Increment the PC
0011  1967  ;               3. RDM_PC       ; Read the PC back to the RDM H register
0011  1968  ;       4. Compare the RDM H register with a constant 1.
0011  1969  ;       5. Signal an error if the H register does not equal one.
0011  1970  ;       6. Test completed.
0011  1971  ;
0011  1972  ; DATA PATTERNS
0011  1973  ;
0011  1974  ;       There are two longword patterns used in this test:
0011  1975  ;
0011  1976  ;               PC INPUT                RESULT
0011  1977  ;
0011  1978  ;           0 0 0 0 0 0 0 0         0 0 0 0 0 0 0 2
0011  1979  ;
0011  1980  ; LOGIC DESCRIPTION
0011  1981  ;
0011  1982  ;       This test examines the PC_PC+2 address path for functionality.
0011  1983  ;       This test is implemented by using the macro PC_PC+2 MB_XB which has
0011  1984  ;       been defined for the following micro-instruction:
0011  1985  ;
0011  1986  ;               MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,DTYPE/WORD
0011  1987  ;
0011  1988  ;       The MSRC micro-order XB.PC_PC+I is used by this test to increment the
0011  1989  ;       PC. Although this micro-instruction also sources an XB onto the MBUS,
0011  1990  ;       this operation is treated as a side effect and is ignored. The PC
0011  1991  ;       increment is normally instruction dependent but for this test it is
0011  1992  ;       determined by the DTYPE micro-field by setting the ISTRM bit in the
0011  1993  ;       micro-word. This is accomplished by using the ISTRM/ISIZE_DSIZE micro-
0011  1994  ;       order. Once this has been done the PC can be incremented by a constant
0011  1995  ;       1,2 or 4 depending on the value of DTYPE. To obtain an increment of 2
0011  1996  ;       the DTYPE/WORD micro-order is necessary.
0011  1997  ;
0011  1998  ; ASSUMPTIONS
0011  1999  ;
0011  2000  ;       This test assumes that the DPM, WBUS and MBUS are functional and that
0011  2001  ;       the PC has been successfully tested.
0011  2002  ;
```

```
0011  ; ERROR DESCRIPTION
0011  ;
0011  ;       Below is a table of possible error symptoms and their most likel
0011  ;       causes:
0011  ;
0011  ;               Symptom                         Probable Cause
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Bad carries, and/or   | A defective adder in the ADD slice for|
0011  ;       | stuck bits within a   | that byte. More than one defective ADD|
0011  ;       | particular byte.      | slice will cause groups of bad bytes. |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Bad sums across byte  | A defective 74LS182 chip on the MIC   |
0011  ;       | boundaries due to an  | module. (MIC-03)                      |
0011  ;       | incorrect carry.      |                                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Byte other than the   | The CHIP ID signal to that ADD gate   |
0011  ;       | least significant byte| array has been accidentally shorted to|
0011  ;       | of the PC is being    | ground.                               |
0011  ;       | incremented.          |                                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Random bad sums from  | Defects in the ASRC decode logic. Bad |
0011  ;       | the adder or improper | ASRC selections can cause whatever is |
0011  ;       | PC increments.        | currently on the WBUS to be added to  |
0011  ;       |                       | the PC, or an increment of 0,1 or 4.  |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Sum of the PC is the  | A defective ADK gate array can cause a|
0011  ;       | +2 increment of VA,   | register other than the PC to be      |
0011  ;       | VA SAVE or PC BACKUP  | incremented.                          |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | One particular bad sum| A defective PRK gate array will cause |
0011  ;       | coming from the adder.| the wrong register to be read from the|
0011  ;       |                       | ADD gate array.                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;
0011  ;       On error this test will call out the ADD, ADK and PRK gate arrays on
0011  ;       the MIC module. The following information will also be typed out:
0011  ;
0011  ;       Expected Data:
0011  ;       Received Data:
0011  ;
0011  ;--
0011  ;*******************************************************************************
0011  ;///////////////////////////////////////////////////////////////////////////////
0011
0011          INITIALIZE                                      ; INITIALIZE THE CPU
0013          EXPUTRAP                                        ; POSSIBLE FAILURE MODE
0015          LOADREG         WDREG,1$,,LONG                  ; RDM WDREG_0
001D          ERRLOOP                                         ; ERROR LOOP
001F          FETCH           0                               ; EXECUTE PC_PC+2 MB_XB
0024          BRSTCLK         5                               ;        "        "
0028          FETCH           3                               ; REEXECUTE INCASE OF U-TRAP
002D          BRSTCLK         5                               ; END DCS PROGRAM
0031          CMPREG          WHREG,2$,,LONG                  ; DOES WHREG = CORRECT VALUE?
003A          IFERROR         ,<<ADD><ADK><PRK>>,             ; IF NOT, SIGNAL AN ERROR
0042          SKIP                                            ; TEST COMPLETED
```

```
                                0049  2058
                                0049  2059 BEGIN_TEST_DATA
                                0049
                                0049       ;-------------------------------------------------------------------------------
                                0049  2060
                     00000000   0049  2061 1$:     .LONG   ^X 00000000     ; INPUT DATA
                     00000002   004D  2062 2$:     .LONG   ^X 00000002     ; RESULT DATA
                                0051  2063
                                0051  2064 BEGIN_CPU_DATA
                                0002  2065
                                0002  2066 DCS_DATA
                                0001  2067
U 00, 7700,C800,0364,0300,0470,1801   0001  2068 ;* 00   NOP
U 01, 5700,4800,0364,0300,4870,1802   000C  2072 ;* 01   PC_RDM
U 02, 7701,8817,0364,0102,0470,1803   0017  2076 ;* 02   PC_PC+2 MB_XB
U 03, 7700,C800,0364,0300,0470,1804   0022  2080 ;* 03   NOP
U 04, 6700,481A,0024,03D8,0470,1805   002D  2084 ;* 04   RDM_PC
U 05, 7300,4800,0364,0300,0470,1806   0038  2088 ;* 05   STOP.CLOCK
                                0043  2092
                                0043  2093 CACHE_DATA
                                0001  2094
                     00000000   0001  2095         .LONG   0
                                0005  2096
                                0005  2097 END_CPU_DATA
                                0051  2098
                                0051  2099 END_TEST_DATA
                                0051
                                0051       ;-------------------------------------------------------------------------------
                                0051  2100
                                0051  2101 ENDTEST
```

```
0011          .SBTTL  "TEST   010     PC_PC+4 TEST
0011  2103 ;*******************************************************************************
0011  2104 ;++
0011  2105 ;
0011  2106 ; FUNCTIONAL DESCRIPTION
0011  2107 ;
0011  2108 ;       This test will examine the PC_PC+4 address path by first loading the
0011  2109 ;       PC register with a zero and then incrementing it by four. It then
0011  2110 ;       reads back the new value of the PC and compares it with the expected
0011  2111 ;       value stored in the RDM. If the two are not equal an error has occur-
0011  2112 ;       red.
0011  2113 ;
0011  2114 ; TEST ALGORITHM
0011  2115 ;
0011  2116 ;       1. Initialize the CPU.
0011  2117 ;       2. Load the RDM D register with zero.
0011  2118 ;       3. Execute the micro-instructions:
0011  2119 ;               1. PC_RDM       ; PC_0
0011  2120 ;               2. PC_PC+4      ; Increment the PC
0011  2121 ;               3. RDM_PC       ; Read the PC back to the RDM H register
0011  2122 ;       4. Compare the RDM H register with a constant 4.
0011  2123 ;       5. Signal an error if the H register does not equal four.
0011  2124 ;       6. Test completed.
0011  2125 ;
0011  2126 ; DATA PATTERNS
0011  2127 ;
0011  2128 ;       These are the data patterns to be used in this test:
0011  2129 ;
0011  2130 ;               PC INPUT                RESULT
0011  2131 ;
0011  2132 ;           0 0 0 0 0 0 0 0         0 0 0 0 0 0 0 4
0011  2133 ;
0011  2134 ; LOGIC DESCRIPTION
0011  2135 ;
0011  2136 ;       This test examines the PC_PC+4 address path for functionality.
0011  2137 ;       This test is implemented by using the micro-instruction:
0011  2138 ;
0011  2139 ;               MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,DTYPE/LONG
0011  2140 ;
0011  2141 ;       The MSRC micro-order XB.PC_PC+I is used by this test to increment the
0011  2142 ;       PC. Although this micro-instruction also sources an XB onto the MBUS,
0011  2143 ;       this operation is treated as a side effect and is ignored. The PC
0011  2144 ;       increment is normally instruction dependent but for this test it is
0011  2145 ;       determined by the DTYPE micro-field by setting the ISTRM bit in the
0011  2146 ;       micro-word. This is accomplished by using the ISTRM/ISIZE_DSIZE micro-
0011  2147 ;       order. Once this has been done the PC can be incremented by a constant
0011  2148 ;       1, 2 or 4 depending on the value of DTYPE. To obtain an increment of 4
0011  2149 ;       it is necessary to use the micro-order DTYPE/LONG.
0011  2150 ;
0011  2151 ; ASSUMPTIONS
0011  2152 ;
0011  2153 ;       This test assumes that the DPM, WBUS and MBUS are functional, and that
0011  2154 ;       the PC has been successfully tested.
0011  2155 ;
0011  2156 ; ERROR DESCRIPTION
```

```
0011  ;
0011  ;       Below is a table of possible error symptoms and their most likely
0011  ;       causes:
0011  ;
0011  ;               Symptom                         Probable Cause
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Bad carries, and/or   | A defective adder in the ADD slice for|
0011  ;       | stuck bits within a   | that byte. More than one defective ADD|
0011  ;       | particular byte.      | slice will cause groups of bad bytes. |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Bad sums across byte  | A defective 74LS182 chip on the MIC   |
0011  ;       | boundaries due to an  | module. (MIC-03)                      |
0011  ;       | incorrect carry.      |                                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Byte besides the least| The CHIP ID signal to that ADD gate   |
0011  ;       | significant byte of   | array has been accidentally shorted to|
0011  ;       | the PC is being       | ground.                               |
0011  ;       | incremented.          |                                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Random bad sums from  | Defects in the ASRC decode logic. Bad |
0011  ;       | the adder or improper | ASRC selections can cause whatever is |
0011  ;       | PC increments.        | currently on the WBUS to be added to  |
0011  ;       |                       | the PC, or an increment of 0,1 or 2.  |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | Sum of the PC is the  | A defective ADK gate array can cause a|
0011  ;       | +4 increment of VA,   | register other than the PC to be      |
0011  ;       | VA SAVE or PC BACKUP. | incremented.                          |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;       | One particular bad sum| A defective PRK gate array will cause |
0011  ;       | coming from the adder.| the wrong register to be read from the|
0011  ;       |                       | ADD gate array.                       |
0011  ;       +-----------------------+-----------------------------------------+
0011  ;
0011  ;       On error this test will call out the ADD, ADK and PRK gate arrays on
0011  ;       the MIC module and the following information:
0011  ;
0011  ;       Expected Data:
0011  ;       Received Data:
0011  ;
0011  ;--
0011  ;*******************************************************************************
0011  ;///////////////////////////////////////////////////////////////////////////////
0011
0011          INITIALIZE                                  ; INITIALIZE THE CPU
0013          EXPUTRAP                                    ; POSSIBLE FAILURE MODE
0015          LOADREG         WDREG,1$,,LONG              ; RDM WDREG_0
001D          ERRLOOP                                     ; ERROR LOOP
001F          FETCH           0                           ; EXECUTE PC_PC+4 MB_XB
0024          BRSTCLK         5                           ;        "        "
0028          FETCH           3                           ; REEXECUTE INCASE OF U-TRAP
002D          BRSTCLK         5                           ; END DCS PROGRAM
0031          CMPREG          WHREG,2$,,LONG              ; DOES WHREG = CORRECT VALUE?
003A          IFERROR         ,<<ADD><ADK><PRK>>,         ; IF NOT, SIGNAL AN ERROR
0042          SKIP                                        ; TEST COMPLETED
0049
```

```
                                0049  2212 BEGIN_TEST_DATA
                                0049
                                0049       ;-----------------------------------------------------------------------
                                0049  2213
                     00000000   0049  2214 1$:      .LONG    ^X 00000000      ; INPUT DATA
                     00000004   004D  2215 2$:      .LONG    ^X 00000004      ; RESULT DATA
                                0051  2216
                                0051  2217 BEGIN_CPU_DATA
                                0002  2218
                                0002  2219 DCS_DATA
                                0001  2220
U 00. 7700,C800,0364,0300,0470,1801   0001  2221 ;* 00    NOP
U 01. 5700,4800,0364,0300,4870,1802   000C  2225 ;* 01    PC_RDM
U 02. 7701,8817,0364,0202,0470,1803   0017  2229 ;* 02    PC_PC+4 MB_XB
U 03. 7700,C800,0364,0300,0470,1804   0022  2233 ;* 03    NOP
U 04. 6700,481A,0024,03D8,0470,1805   002D  2237 ;* 04    RDM_PC
U 05. 7300,4800,0364,0300,0470,1806   0038  2241 ;* 05    STOP.CLOCK
                                0043  2245
                                0043  2246 CACHE_DATA
                                0001  2247
                     00000000   0001  2248          .LONG    0
                                0005  2249
                                0005  2250 END_CPU_DATA
                                0051  2251
                                0051  2252 END_TEST_DATA
                                0051
                                0051       ;-----------------------------------------------------------------------
                                0051  2253
                                0051  2254 ENDTEST
```

```
0011        .SBTTL  "TEST   011     VA_VA+4 TEST
0011  2256 ;*******************************************************************************
0011  2257 ;++
0011  2258 ;
0011  2259 ; FUNCTIONAL DESCRIPTION
0011  2260 ;
0011  2261 ;       This test will analyze the VA_VA+4 address path by loading a zero into
0011  2262 ;       the VA, incrementing it by four using the WCTRL/VA_VA+4 micro-
0011  2263 ;       instruction, and then comparing the result in the VA with a constant
0011  2264 ;       four stored in the RDM memory. If the two are not equal an error has
0011  2265 ;       occurred.
0011  2266 ;
0011  2267 ; TEST ALGORITHM
0011  2268 ;
0011  2269 ;       1. Initialize the CPU.
0011  2270 ;       2. Load a zero into the RDM D register.
0011  2271 ;       3. Execute the micro-instructions:
0011  2272 ;               1. Load the VA with zero.
0011  2273 ;               2. Increment the VA by 4 and store the result back to the VA.
0011  2274 ;               3. Read the VA back to the RDM H register.
0011  2275 ;       4. Signal an error if the RDM H register is not equal to four.
0011  2276 ;       5. Test completed.
0011  2277 ;
0011  2278 ; DATA PATTERNS
0011  2279 ;
0011  2280 ;                       INPUT                           RESULT
0011  2281 ;
0011  2282 ;                   0 0 0 0 0 0 0 0                 0 0 0 0 0 0 0 4
0011  2283 ;
0011  2284 ; LOGIC DESCRIPTION
0011  2285 ;
0011  2286 ;       This test examines the VA_VA+4 data path for open or shorted lines.
0011  2287 ;
0011  2288 ; ASSUMPTIONS
0011  2289 ;
0011  2290 ;       This test assumes that the DPM, WBUS and MBUS are functional and that
0011  2291 ;       the VA register has been successfully tested prior to this test.
0011  2292 ;
0011  2293 ; ERROR DESCRIPTION
0011  2294 ;
0011  2295 ;
0011  2296 ;       Below is a table of possible error symptoms and their most likely
0011  2297 ;       causes:
0011  2298 ;
0011  2299 ;               Symptom                         Probable Cause
0011  2300 ;       +-----------------------+-----------------------------------------+
0011  2301 ;       | Bad carries, and/or   | A defective adder in the ADD slice for  |
0011  2302 ;       | stuck bits within a   | that byte. More than one defective ADD  |
0011  2303 ;       | particular byte.      | slice will cause groups of bad bytes.   |
0011  2304 ;       +-----------------------+-----------------------------------------+
0011  2305 ;       | Bad sums across byte  | A defective 74LS182 chip on the MIC     |
0011  2306 ;       | boundaries due to a   | module. (MIC-03)                        |
0011  2307 ;       | loss of carry.        |                                         |
0011  2308 ;       +-----------------------+-----------------------------------------+
0011  2309 ;       | Byte besides the least| The CHIP ID signal to that ADD gate     |
```

```
                                  0011  ;        | significant byte of    | array has been accidentally shorted to|
                                  0011  ;        | the VA is being        | ground.                               |
                                  0011  ;        | incremented.           |                                       |
                                  0011  ;        +------------------------+---------------------------------------+
                                  0011  ;        | Random bad sums from   | Defects in the ASRC decode logic. Bad |
                                  0011  ;        | the adder or improper  | ASRC selections can cause whatever is |
                                  0011  ;        | VA increments.         | currently on the WBUS to be added to  |
                                  0011  ;        |                        | the VA, or an increment of 0,1 or 2.  |
                                  0011  ;        +------------------------+---------------------------------------+
                                  0011  ;        | Sum of the VA is the   | A defective ADK gate array can cause a|
                                  0011  ;        | +4 increment of PC,    | register other than the VA to be      |
                                  0011  ;        | VA SAVE or PC BACKUP.  | incremented.                          |
                                  0011  ;        +------------------------+---------------------------------------+
                                  0011  ;        | One particular bad sum| A defective PRK gate array will cause |
                                  0011  ;        | coming from the adder.| the wrong register to be read from the|
                                  0011  ;        |                        | ADD gate array.                       |
                                  0011  ;        +------------------------+---------------------------------------+
                                  0011  ;
                                  0011  ;        On error this test will call out the ADD, ADK and PRK gate arrays on
                                  0011  ;        the MIC module and the following information:
                                  0011  ;
                                  0011  ;        Expected Data:
                                  0011  ;        Received Data:
                                  0011  ;
                                  0011  ;--
                                  0011  ;*******************************************************************************
                                  0011  ;///////////////////////////////////////////////////////////////////////////////
                                  0011
                                  0011          INITIALIZE                                  ; INITIALIZE THE CPU
                                  0013          LOADREG         WDREG,1$,,LONG              ; RDM WDREG_0
                                  001B          ERRLOOP                                     ; ERROR LOOP
                                  001D          FETCH           0                           ; EXECUTE VA_VA+4
                                  0022          BRSTCLK         4                           ;        "
                                  0026          CMPREG          WHREG,2$,,LONG              ; DOES RDM WHREG = 4?
                                  002F          IFERROR         ,<<ADD><ADK><PRK>>,         ; IF NOT SIGNAL AN ERROR
                                  0037          SKIP                                        ; TEST COMPLETED
                                  003E
                                  003E  BEGIN_TEST_DATA
                                  003E
                                  003E  ;-----------------------------------------------------------------------
                                  003E
                        00000000  003E  1$:     .LONG   ^X 00000000     ; INPUT PATTERN
                        00000004  0042  2$:     .LONG   ^X 00000004     ; RESULT PATTERN
                                  0046
                                  0046  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,C800,0364,0300,4A70,1802  000C  ;% 01   VA_RDM
U 02, 7700,C800,0364,0300,4470,1803  0017  ;% 02   VA_VA+4
U 03, 6700,881B,0024,03D8,0470,1804  0022  ;% 03   RDM_VA
U 04, 7300,4800,0364,0300,0470,1805  002D  ;% 04   NOP,STOP.CLOCK
                                  0038
                                  0038  END_CPU_DATA
```

```
0046  2378
0046  2379 END_TEST_DATA
0046
0046       ;-----------------------------------------------------------------------------
0046  2380
0046  2381 ENDTEST
```

```
0013        .SBTTL  "TEST   012     ADD ADDER TEST
0013  2383 ;*******************************************************************************
0013  2384 ;++
0013  2385 ;
0013  2386 ; FUNCTIONAL DESCRIPTION
0013  2387 ;
0013  2388 ;       This test will check the functionality of the two input adder on the
0013  2389 ;       ADD gate array. This adder is used to increment the PC and VA regis-
0013  2390 ;       ters. This test is implemented by using a 32 bit floating zero pattern
0013  2391 ;       to test the sum and carry logic for each bit position in the adder.
0013  2392 ;       The PC_PC+1 micro-instruction is used to carry out this test.
0013  2393 ;
0013  2394 ; TEST ALGORITHM
0013  2395 ;
0013  2396 ;       1. Initialize the CPU.
0013  2397 ;       2. Prepare a loop of 33 iterations.
0013  2398 ;       3. Load the RDM D register with a data pattern.
0013  2399 ;       4. Execute the micro instruction:
0013  2400 ;               1. PC_WB
0013  2401 ;       6. Execute the micro instructions:
0013  2402 ;               2. PC_PC+1
0013  2403 ;               3. RDM_PC
0013  2404 ;       7. Compare the result with the known correct sum in RDM memory.
0013  2405 ;       8. Signal an error if the comparsion shows that the two are not equal.
0013  2406 ;       9. Looped 33 times? If not go to STEP 3.
0013  2407 ;      10. Test completed.
0013  2408 ;
0013  2409 ; TEST DATA
0013  2410 ;
0013  2411 ;       Two data tables are used to implement this test. They each contain
0013  2412 ;       33 data patterns for testing the adder circuitry. They are listed as
0013  2413 ;       follows:
0013  2414 ;
0013  2415 ;                       LOOP COUNT          PC                      SUM  RESULT
0013  2416 ;
0013  2417 ;   ADD <07:00>
0013  2418 ;
0013  2419 ;           1           1       F F F F F F F E         F F F F F F F F
0013  2420 ;           2           2       F F F F F F F D         F F F F F F F E
0013  2421 ;           3           3       F F F F F F F B         F F F F F F F C
0013  2422 ;           4           4       F F F F F F F 7         F F F F F F F 8
0013  2423 ;           5           5       F F F F F F E F         F F F F F F F 0
0013  2424 ;           6           6       F F F F F F D F         F F F F F F E 0
0013  2425 ;           7           7       F F F F F F B F         F F F F F F C 0
0013  2426 ;           8           8       F F F F F F 7 F         F F F F F F 8 0
0013  2427 ;
0013  2428 ;   ADD <15:08>
0013  2429 ;
0013  2430 ;           9           9       F F F F F E F F         F F F F F F 0 0
0013  2431 ;          10           A       F F F F F D F F         F F F F F E 0 0
0013  2432 ;          11           B       F F F F F B F F         F F F F F C 0 0
0013  2433 ;          12           C       F F F F F 7 F F         F F F F F 8 0 0
0013  2434 ;          13           D       F F F F E F F F         F F F F F 0 0 0
0013  2435 ;          14           E       F F F F D F F F         F F F F E 0 0 0
0013  2436 ;          15           F       F F F F B F F F         F F F F C 0 0 0
```

```
0013  ;          16          10      F F F F 7 F F F        F F F F 8 0 0 0
0013  ;
0013  ;    ADD <23:16>
0013  ;
0013  ;          17          11      F F F E F F F F        F F F F 0 0 0 0
0013  ;          18          12      F F F D F F F F        F F F E 0 0 0 0
0013  ;          19          13      F F F B F F F F        F F F C 0 0 0 0
0013  ;          20          14      F F F 7 F F F F        F F F 8 0 0 0 0
0013  ;          21          15      F F E F F F F F        F F F 0 0 0 0 0
0013  ;          22          16      F F D F F F F F        F F E 0 0 0 0 0
0013  ;          23          17      F F B F F F F F        F F C 0 0 0 0 0
0013  ;          24          18      F F 7 F F F F F        F F 8 0 0 0 0 0
0013  ;
0013  ;    ADD <31:24>
0013  ;
0013  ;          25          19      F E F F F F F F        F F 0 0 0 0 0 0
0013  ;          26          1A      F D F F F F F F        F E 0 0 0 0 0 0
0013  ;          27          1B      F B F F F F F F        F C 0 0 0 0 0 0
0013  ;          28          1C      F 7 F F F F F F        F 8 0 0 0 0 0 0
0013  ;          29          1D      E F F F F F F F        F 0 0 0 0 0 0 0
0013  ;          30          1E      D F F F F F F F        E 0 0 0 0 0 0 0
0013  ;          31          1F      B F F F F F F F        C 0 0 0 0 0 0 0
0013  ;          32          20      7 F F F F F F F        8 0 0 0 0 0 0 0
0013  ;
0013  ;    ADD <31:00>
0013  ;
0013  ;          33          21      F F F F F F F F        0 0 0 0 0 0 0 0
0013  ;
0013  ; LOGIC DESCRIPTION
0013  ;
0013  ;       The ADD adder is an 8 bit fast adder that is bit sliced across four
0013  ;       ADD gate arrays. The carry propogate and carry generate signals for
0013  ;       the MSB of each gate array are connected to an external 74LS182 carry
0013  ;       look ahead generator on the MIC module. (MIC-03) The carries from the
0013  ;       LS182 are then sent back to the carry input on the next higher order
0013  ;       ADD gate array. There are two input multiplexers that control the
0013  ;       inputs to the adder. The control lines to these multiplexers are call-
0013  ;       ed ASRC and BSRC. ASRC is generated by discrete logic on the MIC mod-
0013  ;       ule (MIC-04; E14,E15,E16), BSRC is generated by the ADK gate array.
0013  ;       The outputs from the adder are fed internally to the VA and PC reg-
0013  ;       isters and the VA SAVE and PC BACKUP latches.
0013  ;
0013  ; ASSUMPTIONS
0013  ;
0013  ;       This test assumes the WBUS, DPM, MBUS, PC and PC_PC+1 data paths are
0013  ;       functional.
0013  ;
0013  ; ERROR DESCRIPTION
0013  ;
0013  ;       Below is a table of possible error symptoms and their most likely
0013  ;       causes:
0013  ;
0013  ;               Symptom                    Probable Cause
0013  ;       +-----------------------+-----------------------------------+
0013  ;       | Bad carries, and/or   | A defective adder in the ADD slice for|
```

```
0013  2492 ;       | stuck bits within a    | that byte. More than one defective ADD |
0013  2493 ;       | particular byte.       | slice will cause groups of bad bytes.  |
0013  2494 ;       +------------------------+----------------------------------------+
0013  2495 ;       | Bad sums across byte   | A defective 74LS182 chip on the MIC    |
0013  2496 ;       | boundaries due to an   | module.                                |
0013  2497 ;       | incorrect carry.       |                                        |
0013  2498 ;       +------------------------+----------------------------------------+
0013  2499 ;       | Byte other that the    | The CHIP ID signal to that ADD gate    |
0013  2500 ;       | least significant byte | array has been accidentally shorted to |
0013  2501 ;       | of the PC is being     | ground.                                |
0013  2502 ;       | incremented.           |                                        |
0013  2503 ;       +------------------------+----------------------------------------+
0013  2504 ;       | Random bad sums from   | Defects in the ASRC decode logic. Bad  |
0013  2505 ;       | the adder or improper  | ASRC selections can cause whatever is  |
0013  2506 ;       | PC increments.         | currently on the WBUS to be added to   |
0013  2507 ;       |                        | the PC, or an increment of 0,2 or 4.   |
0013  2508 ;       +------------------------+----------------------------------------+
0013  2509 ;       | Sum of PC is the +1    | A defective ADK gate array can cause a |
0013  2510 ;       | increment of VA,       | register other that the PC to be       |
0013  2511 ;       | VA SAVE or PC BACKUP.  | incremented.                           |
0013  2512 ;       +------------------------+----------------------------------------+
0013  2513 ;       | One particular bad sum | A defective PRK gate array will cause  |
0013  2514 ;       | coming from the adder. | the wrong register to be read from the |
0013  2515 ;       |                        | ADD gate array.                        |
0013  2516 ;       +------------------------+----------------------------------------+
0013  2517 ;
0013  2518 ;       On error this test will call out the ADD, ADK and PRK gate arrays on
0013  2519 ;       the MIC module.  The following information is also typed out:
0013  2520 ;
0013  2521 ;       Expected Data:
0013  2522 ;       Received Data:
0013  2523 ;       Loop Count:
0013  2524 ;
0013  2525 ;--
0013  2526 ;*******************************************************************************
0013  2527 ;///////////////////////////////////////////////////////////////////////////////
0013  2528
0013  2529         INITIALIZE                                      ; INITIALIZE THE CPU
0015  2530         LOADDCS         3$,,0,7
001C  2531         FETCH           0                               ; ZERO THE VA
0021  2532         BRSTCLK         2                               ; ...
0025  2533
0025  2534         LOOP            I,1,1024                        ; INVALIDATE ALL OF CACHE
002C  2535         FETCH           3                               ; ...
0031  2536         BRSTCLK         6                               ; ...
0035  2537         ENDLOOP         I                               ; CONTINUE LOOPING
0038  2538
0038  2539         LOADDCS         4$,,0,14                        ; LOAD THE TEST MICRO-CODE
003F  2540         EXPUTRAP                                        ; EXPECT A U-TRAP ON ERROR
0041  2541         LOOP            I,1,33                          ; SET UP FOR 33 ITERATIONS
0048  2542         LOADREG         WDREG,1$,I,LONG                 ; RDM WDREG_TEST PATTERN
0050  2543         ERRLOOP                                         ; ERROR LOOP
0052  2544         FETCH           0                               ; EXECUTE PC_PC+1 MB_XB
0057  2545         BRSTCLK         5                               ; ...
005B  2546         FETCH           6                               ; PICK UP THE MEMSCRS
```

```
                    0060  2547          BRSTCLK         <^X0D>                          ; ...
                    0064  2548          CMPREG          WHREG,2$,I,LONG                 ; RDM WHREG = TEST PATTERN?
                    006D  2549          IFERROR         6,<<ADD><ADK><PRK>>,            ; IF NOT SIGNAL AN ERROR
                    0075  2550          ENDLOOP         I                               ; CONTINUE LOOP
                    0078  2551          SKIP                                            ; TEST COMPLETED
                    007F  2552
                    007F  2553 BEGIN_TEST_DATA
                    007F
                    007F          ;-------------------------------------------------------------------------
                    007F  2554
FFFFFFFE            007F  2555 1$:      .LONG   ^X FFFFFFFE     ; INPUT DATA PATTERNS
FFFFFFFD            0083  2556          .LONG   ^X FFFFFFFD
FFFFFFFB            0087  2557          .LONG   ^X FFFFFFFB
FFFFFFF7            008B  2558          .LONG   ^X FFFFFFF7
FFFFFFEF            008F  2559          .LONG   ^X FFFFFFEF
FFFFFFDF            0093  2560          .LONG   ^X FFFFFFDF
FFFFFFBF            0097  2561          .LONG   ^X FFFFFFBF
FFFFFF7F            009B  2562          .LONG   ^X FFFFFF7F
FFFFFEFF            009F  2563          .LONG   ^X FFFFFEFF
FFFFFDFF            00A3  2564          .LONG   ^X FFFFFDFF
FFFFFBFF            00A7  2565          .LONG   ^X FFFFFBFF
FFFFF7FF            00AB  2566          .LONG   ^X FFFFF7FF
FFFFEFFF            00AF  2567          .LONG   ^X FFFFEFFF
FFFFDFFF            00B3  2568          .LONG   ^X FFFFDFFF
FFFFBFFF            00B7  2569          .LONG   ^X FFFFBFFF
FFFF7FFF            00BB  2570          .LONG   ^X FFFF7FFF
FFFEFFFF            00BF  2571          .LONG   ^X FFFEFFFF
FFFDFFFF            00C3  2572          .LONG   ^X FFFDFFFF
FFFBFFFF            00C7  2573          .LONG   ^X FFFBFFFF
FFF7FFFF            00CB  2574          .LONG   ^X FFF7FFFF
FFEFFFFF            00CF  2575          .LONG   ^X FFEFFFFF
FFDFFFFF            00D3  2576          .LONG   ^X FFDFFFFF
FFBFFFFF            00D7  2577          .LONG   ^X FFBFFFFF
FF7FFFFF            00DB  2578          .LONG   ^X FF7FFFFF
FEFFFFFF            00DF  2579          .LONG   ^X FEFFFFFF
FDFFFFFF            00E3  2580          .LONG   ^X FDFFFFFF
FBFFFFFF            00E7  2581          .LONG   ^X FBFFFFFF
F7FFFFFF            00EB  2582          .LONG   ^X F7FFFFFF
EFFFFFFF            00EF  2583          .LONG   ^X EFFFFFFF
DFFFFFFF            00F3  2584          .LONG   ^X DFFFFFFF
BFFFFFFF            00F7  2585          .LONG   ^X BFFFFFFF
7FFFFFFF            00FB  2586          .LONG   ^X 7FFFFFFF
FFFFFFFF            00FF  2587          .LONG   ^X FFFFFFFF
                    0103  2588
FFFFFFFF            0103  2589 2$:      .LONG   ^X FFFFFFFF     ; RESULT DATA PATTERNS
FFFFFFFE            0107  2590          .LONG   ^X FFFFFFFE
FFFFFFFC            010B  2591          .LONG   ^X FFFFFFFC
FFFFFFF8            010F  2592          .LONG   ^X FFFFFFF8
FFFFFFF0            0113  2593          .LONG   ^X FFFFFFF0
FFFFFFE0            0117  2594          .LONG   ^X FFFFFFE0
FFFFFFC0            011B  2595          .LONG   ^X FFFFFFC0
FFFFFF80            011F  2596          .LONG   ^X FFFFFF80
FFFFFF00            0123  2597          .LONG   ^X FFFFFF00
FFFFFE00            0127  2598          .LONG   ^X FFFFFE00
FFFFFC00            012B  2599          .LONG   ^X FFFFFC00
```

```
                    FFFFF800  012F  2600          .LONG   ^X FFFFF800
                    FFFFF000  0133  2601          .LONG   ^X FFFFF000
                    FFFFE000  0137  2602          .LONG   ^X FFFFE000
                    FFFFC000  013B  2603          .LONG   ^X FFFFC000
                    FFFF8000  013F  2604          .LONG   ^X FFFF8000
                    FFFF0000  0143  2605          .LONG   ^X FFFF0000
                    FFFE0000  0147  2606          .LONG   ^X FFFE0000
                    FFFC0000  014B  2607          .LONG   ^X FFFC0000
                    FFF80000  014F  2608          .LONG   ^X FFF80000
                    FFF00000  0153  2609          .LONG   ^X FFF00000
                    FFE00000  0157  2610          .LONG   ^X FFE00000
                    FFC00000  015B  2611          .LONG   ^X FFC00000
                    FF800000  015F  2612          .LONG   ^X FF800000
                    FF000000  0163  2613          .LONG   ^X FF000000
                    FE000000  0167  2614          .LONG   ^X FE000000
                    FC000000  016B  2615          .LONG   ^X FC000000
                    F8000000  016F  2616          .LONG   ^X F8000000
                    F0000000  0173  2617          .LONG   ^X F0000000
                    E0000000  0177  2618          .LONG   ^X E0000000
                    C0000000  017B  2619          .LONG   ^X C0000000
                    80000000  017F  2620          .LONG   ^X 80000000
                    00000000  0183  2621          .LONG   ^X 00000000
                              0187  2622
                              0187  2623 3$:
U 00, 7700,C800,0364,0300,0470,1801   0187  2624 ;* 00    NOP                     ; GET CPU UP TO SPEED
U 01, 7700,8800,5BE6,03D8,4A70,1802   0192  2628 ;* 01    D_VA_0                  ; ZERO THE VA
U 02, 7300,4800,0364,0300,0470,1803   019D  2632 ;* 02    STOP.CLOCK              ; STOP
U 03, 7700,C800,0364,0300,0470,1804   01A8  2636 ;* 03    NOP                     ; GET CPU UP TO SPEED
U 04, 7700,881B,0024,03D8,5A70,1805   01B3  2640 ;* 04    CLRCH.VA_VA             ; INVALIDATE LOCATION AT VA ADDRESS
U 05, 7700,C800,0364,0300,4470,1806   01BE  2644 ;* 05    VA_VA+4                 ; INCREMENT THE VA FOR THE NEXT LOCATION
U 06, 7300,C800,0364,0300,0470,1807   01C9  2648 ;* 06    STOP.CLOCK              ; HALT
                              01D4  2652
                              01D4  2653 4$:
U 00, 7700,C800,0364,0300,0470,1801   01D4  2654 ;* 00    NOP                     ; GET CPU UP TO SPEED
U 01, 5700,C800,0364,0300,4A70,1802   01DF  2658 ;* 01    VA_RDM                  ; LOAD THE VA WITH THE TEST ADDRESS
U 02, 5700,C800,0364,0300,4870,1803   01EA  2662 ;* 02    PC_RDM                  ; LOAD THE PC WITH THE TEST ADDRESS
U 03, 7701,4817,0364,0002,0470,1804   01F5  2666 ;* 03    PC_PC+1 MB_XB           ; INCREMENT THE PC
U 04, 6700,481A,0024,03D8,0470,1805   0200  2670 ;* 04    RDM_PC                  ; READ THE PC BACK TO THE RDM
U 05, 7300,4800,0364,0300,0470,1806   020B  2674 ;* 05    STOP.CLOCK              ; HALT
U 06, 7700,C800,0364,0300,0470,1807   0216  2678 ;* 06    NOP                     ; GET CPU UP TO SPEED
U 07, 7700,C800,5BE4,030C,6870,1808   0221  2682 ;* 07    MEMSCAR_R[TEMP3]        ; SAVE MACHINE CHECK REGISTER
U 08, 7700,0840,0364,0300,6470,1809   022C  2686 ;* 08    R[TEMP0]_MEMSCR
U 09, 7700,0800,5BE4,0310,6870,180A   0237  2690 ;* 09    MEMSCAR_R[TEMP4]        ; SAVE BUS ERROR REGISTER
U 0A, 7700,C840,0364,0304,6470,180B   0242  2694 ;* 0A    R[TEMP1]_MEMSCR
U 0B, 7700,4800,5BE4,0314,6870,180C   024D  2698 ;* 0B    MEMSCAR_R[TEMP5]        ; SAVE CACHE ERROR REGISTER
U 0C, 7700,C840,0364,0308,6470,180D   0258  2702 ;* 0C    R[TEMP2]_MEMSCR
U 0D, 7300,C800,0364,0300,0470,180E   0263  2706 ;* 0D    STOP.CLOCK
                              026E  2710
                              026E  2711 BEGIN_CPU_DATA
                              0002  2712
                              0002  2713 RTEMP_DATA
                              0001  2714
                    00000000  0001  2715          .LONG   0               ; MACHINE CHECK REGISTER CONTENTS
                    00000000  0005  2716          .LONG   0               ; BUS ERROR REGISTER CONTENTS
                    00000000  0009  2717          .LONG   0               ; CACHE ERROR REGISTER CONTENTS
```

```
08000000  000D  2718          .LONG   ^X08000000      ; MACHINE CHECK REGISTER ADDRESS
09000000  0011  2719          .LONG   ^X09000000      ; BUS ERROR REGISTER ADDRESS
04000000  0015  2720          .LONG   ^X04000000      ; CACHE ERROR REGISTER CONTENTS
          0019  2721
          0019  2722 END_CPU_DATA'
          026E  2723
          026E  2724 END_TEST_DATA
          026E
          026E        ;-------------------------------------------------------------------------
          026E  2725
          026E  2726 ENDTEST
```

```
0014       .SBTTL  "TEST   013     PC_PC+WBUS TEST
0014  2728 ;*******************************************************************************
0014  2729 ;++
0014  2730 ;
0014  2731 ; FUNCTIONAL DESCRIPTION
0014  2732 ;
0014  2733 ;       This test will examine the PC_PC+WBUS address path by summing the
0014  2734 ;       contents of the PC with the WBUS. Each sum will be compared with the
0014  2735 ;       known correct value stored in the RDM memory and if the two are not
0014  2736 ;       equal an error is signalled.
0014  2737 ;
0014  2738 ; TEST ALGORITHM
0014  2739 ;
0014  2740 ;       1. Initialize the CPU.
0014  2741 ;       2. Zero the PC.
0014  2742 ;       3. Prepare a loop of two iterations.
0014  2743 ;       4. Source an input data pattern from the RDM onto the WBUS.
0014  2744 ;       5. Execute the micro-instructions:
0014  2745 ;               1. Sum the PC with the input data pattern off the WBUS.
0014  2746 ;               2. Read the PC back to the RDM.
0014  2747 ;       6. Compare the new contents of the PC with the known correct value
0014  2748 ;          stored in the RDM.
0014  2749 ;       7. Signal an error if the comparison shows that the two are not equal.
0014  2750 ;       8. Looped twice? If not go to STEP 4.
0014  2751 ;       9. Test completed.
0014  2752 ;
0014  2753 ; TEST DATA
0014  2754 ;
0014  2755 ;       The following four longword patterns are used for this test:
0014  2756 ;
0014  2757 ;           LOOP COUNT              WBUS  INPUT              PC RESULT
0014  2758 ;
0014  2759 ;               1               A A A A A A A A         A A A A A A A A
0014  2760 ;               2               5 5 5 5 5 5 5 5         F F F F F F F F
0014  2761 ;
0014  2762 ; LOGIC DESCRIPTION
0014  2763 ;
0014  2764 ;       The PC_PC+WBUS micro-instruction requires that ASRC be greater than or
0014  2765 ;       equal to 4 hex and that BSRC be equal to 1 hex for correct operation.
0014  2766 ;       Alternating bit patterns are used to insure that there are no stuck
0014  2767 ;       bits in the address path nor bad carries from the ADD adder.
0014  2768 ;
0014  2769 ;       ASRC is generated by discrete logic on the MIC module (MIC-03; E14,
0014  2770 ;       E15,E16). The ADK gate array generates BSRC which is used to select
0014  2771 ;       the PC for the addition. The PRK gate array selects the PC register
0014  2772 ;       to be read out of the ADD gate array, and through the MDR gate array
0014  2773 ;       to the MBUS.
0014  2774 ;
0014  2775 ; ASSUMPTIONS
0014  2776 ;
0014  2777 ;       This test assumes that the DPM, WBUS, and MBUS are functional.
0014  2778 ;
0014  2779 ; ERROR DESCRIPTION
0014  2780 ;
0014  2781 ;       Below is a table of possible error symptoms and their most likely
```

```
0014  ;       causes:
0014  ;
0014  ;               Symptom                         Probable Cause
0014  ;       +-----------------------+-----------------------------------------+
0014  ;       | Bad carries, and/or   | A defective adder in the ADD slice for |
0014  ;       | stuck bits within a   | that byte. More than one defective ADD |
0014  ;       | particular byte.      | slice will cause groups of bad bytes.  |
0014  ;       +-----------------------+-----------------------------------------+
0014  ;       | Bad sums across byte  | A defective 74LS182 chip on the MIC    |
0014  ;       | boundaries due to an  | module.                                |
0014  ;       | incorrect carry.      |                                         |
0014  ;       +-----------------------+-----------------------------------------+
0014  ;       | Byte other that the   | The CHIP ID signal to that ADD gate    |
0014  ;       | least significant byte| array has been accidentally shorted to|
0014  ;       | of the PC is being    | ground.                                |
0014  ;       | incremented.          |                                         |
0014  ;       +-----------------------+-----------------------------------------+
0014  ;       | Random bad sums from  | Defects in the ASRC decode logic. Bad  |
0014  ;       | the adder or improper | ASRC selections can cause whatever is  |
0014  ;       | PC increments.        | currently on the WBUS to be added to   |
0014  ;       |                       | the PC, or an increment of 0,2 or 4.   |
0014  ;       +-----------------------+-----------------------------------------+
0014  ;       | Sum of PC is the +1   | A defective ADK gate array can cause a |
0014  ;       | increment of VA,      | register other that the PC to be       |
0014  ;       | VA SAVE or PC BACKUP. | incremented.                           |
0014  ;       +-----------------------+-----------------------------------------+
0014  ;       | One particular bad sum| A defective PRK gate array will cause  |
0014  ;       | coming from the adder.| the wrong register to be read from the|
0014  ;       |                       | ADD gate array.                        |
0014  ;       +-----------------------+-----------------------------------------+
0014  ;
0014  ;       On error this test will call out the ADD, ADK and PRK gate arrays on
0014  ;       the MIC module. The following information will also be typed out:
0014  ;
0014  ;       Expected Data:
0014  ;       Received Data:
0014  ;       Loop Count:
0014  ;--
0014  ;*******************************************************************************
0014  ;///////////////////////////////////////////////////////////////////////////////
0014
0014          INITIALIZE                                      ; INITIALIZE THE CPU
0016          LOADDCS         3$,,0,4                         ; U-CODE FOR PC_0
001D          FETCH           0                               ; EXECUTE PC_0
0022          BRSTCLK         3                               ;       "
0026          LOADDCS         4$,,0,4                         ; LOAD TEST U-CODE
002D          LOOP            I,1,2                           ; LOOP FOR 2 ITERATIONS
0034          LOADREG         WDREG,1$,I,LONG                 ; RDM WDREG_TEST PATTERN
003C          ERRLOOP                                         ; ERROR LOOP
003E          FETCH           0                               ; EXECUTE PC_PC+RDM
0043          BRSTCLK         3                               ;       "
0047          CMPREG          WHREG,2$,I,LONG                 ; RDM WHREG = TEST PATTERN?
0050          IFERROR         ,<<ADD><ADK><PRK>>,             ; IF NOT SIGNAL AN ERROR
0058          ENDLOOP         I                               ; CONTINUE LOOP
005B          SKIP                                            ; TEST COMPLETED
```

```
                                0062  2837
                                0062  2838 BEGIN_TEST_DATA
                                0062
                                0062        ;-------------------------------------------------------------------------
                                0062  2839
                     AAAAAAAA   0062  2840 1$:      .LONG   ^X AAAAAAAA      ; INPUT DATA PATTERNS
                     55555555   0066  2841         .LONG   ^X 55555555
                                006A  2842
                     AAAAAAAA   006A  2843 2$:      .LONG   ^X AAAAAAAA      ; RESULT DATA PATTERNS
                     FFFFFFFF   006E  2844         .LONG   ^X FFFFFFFF
                                0072  2845
                                0072  2846 3$:
U 00, 7700,C800,0364,0300,0470,1801 0072  2847 ;* 00   NOP
U 01, 7700,7800,7FFF,FFFF,8470,1802 007D  2851 ;* 01   LONLIT_[0]
U 02, 7700,C800,5BE4,03D4,4870,1803 0088  2855 ;* 02   PC_R[LONLIT]
U 03, 7300,C800,0364,0300,0470,1804 0093  2859 ;* 03   STOP.CLOCK
                                009E  2863
                                009E  2864 4$:
U 00, 7700,C800,0364,0300,0470,1801 009E  2865 ;* 00   NOP
U 01, 5700,C800,0364,0300,5870,1802 00A9  2869 ;* 01   PC_PC+RDM
U 02, 6700,481A,0024,03D8,0470,1803 00B4  2873 ;* 02   RDM_PC
U 03, 7300,C800,0364,0300,0470,1804 00BF  2877 ;* 03   STOP.CLOCK
                                00CA  2881
                                00CA  2882 END_TEST_DATA
                                00CA
                                00CA        ;-------------------------------------------------------------------------
                                00CA  2883
                                00CA  2884 ENDTEST
```

```
                        .SBTTL  "TEST   014     TEST MBUS_CACHE_WBUS DATA PATH
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST CHECKS THE DATA PATH MBUS_CACHE_WBUS BY LOADING AND
                ;       READING 6 DATA PATTERNS THROUGH THE CACHE.
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  CREATE A TEST LOOP OF 6 ITERATIONS
                ;       2.  GENERATE A TEST PATTERN
                ;       3.  LOAD TEST PATTERN INTO WD REGISTER (RDM)
                ;       4.  EXECUTE DCS PROGRAM
                ;               a.  WRITE 0 TO VIRTUAL ADDRESS REGISTER
                ;               b.  WRITE TEST PATTERN FROM RDM TO ADDRESS 0 OF CACHE
                ;               c.  WRITE TEST PATTERN TO WDR FOR AN EXTRA CYCLE SINCE CPU
                ;                   DOES NOT STALL WITH CMI DISABLED
                ;               d.  READ TEST PATTERN BACK TO RDM
                ;       5.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       ITERATION       TEST PATTERN
                ;           1             AAAAAAAA
                ;           2             55555555
                ;           3             33333333
                ;           4             0F0F0F0F
                ;           5             00FF00FF
                ;           6             0000FFFF
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       THIS TEST CHECKS THE MDR GATE ARRAY DATA PATH TO AND FROM CACHE.
                ;       TEST PATTERNS ARE LOADED FROM THE RDM VIA THE WBUS, TO THE MDR
                ;       GATE ARRAY, THROUGH THE WDR, AND TO THE CACHE.  THE PATTERNS ARE
                ;       THEN READ BACK TO THE RDM VIA THE MDR REGISTER, MMUX, MBUS, DPM,
                ;       AND WBUS.  OTHER GATE ARRAYS THAT AFFECT THIS TEST ARE: CAK
                ;       (CONTROLS CACHE) AND ADK (DBUS SELECT).
                ;
                ;
                ; ASSUMPTIONS
                ;
                ;       THIS TEST ASSUMES THE VA, CACHE, WBUS, MBUS AND DPM ARE OPERATIONAL.
                ;
                ;
                ; ERROR DESCRIPTION:
                ;
                ;       EXPECTED CACHE DATA
                ;       RECEIVED CACHE DATA
```

```
                                0023  2940 ;       LOOP COUNT
                                0023  2941 ;
                                0023  2942 ;--
                                0023  2943 ;*******************************************************************************
                                0023  2944 ;///////////////////////////////////////////////////////////////////////////////
                                0023  2945
                                0023  2946         INITIALIZE                                      ;INIT CPU
                                0025  2947         EXPUTRAP                                        ;IGNORE U-TRAPS
                                0027  2948         LOOP            I,1,6                           ;CREATE TEST LOOP
                                002E  2949         SA01PAT         I,1$,,                          ;GENERATE A TEST PATTERN
                                0036  2950         LOADREG         WDREG,1$,,L                     ;TEST PATTERN TO D REGISTER
                                003E  2951         ERRLOOP                                         ;ERROR LOOP
                                0040  2952         FETCH           0                               ;START DCS PROGRAM
                                0045  2953         BRSTCLK         5                               ;END DCS PROGRAM
                                0049  2954         FETCH           6                               ;START DCS PROGRAM
                                004E  2955         BRSTCLK         8                               ;END DCS PROGRAM
                                0052  2956         CMPREG          WHREG,1$,,L,                    ;COMPARE RESULTS
                                005B  2957         IFERROR         ,<<MDR><CAK><ADK>>,             ;POSSIBLE FAILING ARRAY
                                0063  2958         ENDLOOP         I                               ;END TEST LOOP
                                0066  2959         SKIP                                            ;GO TO NEXT TEST
                                006D  2960
                                006D  2961 BEGIN_TEST_DATA
                                006D
                                006D        ;-------------------------------------------------------------------------
                                006D  2962
                    00000000    006D  2963 1$:      .LONG   ^X00000000                              ;TEST PATTERN STORAGE
                                0071  2964
                                0071  2965 BEGIN_CPU_DATA
                                0002  2966
                                0002  2967 DCS_DATA
                                0001  2968
U 00, 7700,C800,0364,0300,0470,1801 0001  2969 ;% 00    NOP
U 01, 7700,C800,5BE4,03D8,4A70,1802 000C  2973 ;% 01    VA_R[ZERO]                                  ;0 TO VA REGISTER
U 02, 5700,4800,0364,0300,5480,1803 0017  2977 ;% 02    WB_RDM,WCTRL/WDR_WB.UR,WRITE.PHY            ;WRITE TEST PATTERN TO CACHE
U 03, 5700,C800,0364,0300,5470,1804 0022  2981 ;% 03    WB_RDM,WCTRL/WDR_WB.UR                      ;HOLD DATA FOR EXTRA CYCLE
U 04, 7700,C800,0364,0300,0400,1805 002D  2985 ;% 04    READ.PHY                                    ;READ TEST PATTERN TO MDR
U 05, 7300,4800,0364,0300,0470,1806 0038  2989 ;% 05    NOP,STOP.CLOCK                              ;STOP CPU CLOCK
U 06, 7700,C800,0364,0300,0470,1807 0043  2993 ;% 06    NOP
U 07, 6700,8812,5924,0300,0470,1808 004E  2997 ;% 07    RDM_WB,WB_M[MDR]                            ;MOVE TEST PATTERN TO RDM
U 08, 7300,4800,0364,0300,0470,1809 0059  3001 ;% 08    NOP,STOP.CLOCK                              ;STOP CPU CLOCK
                                0064  3005
                                0064  3006 END_CPU_DATA
                                0071  3007
                                0071  3008 END_TEST_DATA
                                0071
                                0071        ;-------------------------------------------------------------------------
                                0071  3009
                                0071  3010 ENDTEST
```

```
0020          .SBTTL  "TEST   015     CACHE HIT COMPARATOR TEST 1
0020  3012 ;*******************************************************************************
0020  3013 ;++
0020  3014 ;
0020  3015 ;       CACHE HIT COMPARATOR TEST 1
0020  3016 ;
0020  3017 ; FUNCTIONAL DESCRIPTION
0020  3018 ;       THIS IS A TEST OF THE CACHE MEMORY HIT/MISS COMPARATOR. ALONG WITH
0020  3019 ;       THE OTHER CACHE HIT COMPARATOR TESTS, THIS TEST INSURES THAT
0020  3020 ;       HITS AND MISSES ARE GENERATED CORRECTLY.
0020  3021 ;
0020  3022 ; TEST ALGORITHM
0020  3023 ;       1.  SET UP A LOOP TO SELECT TEST PATTERNS
0020  3024 ;       2.  LOAD U-CODE INTO DCS
0020  3025 ;       3.  PUT TEST PATTERN IN WD REGISTER
0020  3026 ;       4.  EXECUTE DCS PROGRAM
0020  3027 ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
0020  3028 ;               b.  WRITE LONLIT TO VA REGISTER
0020  3029 ;               c.  WRITE 0'S TO CACHE LOCATION 00AAA000
0020  3030 ;               d.  WRITE VA REGISTER WITH TEST PATTERN
0020  3031 ;               e.  PERFORM CACHE READ ON THIS LOCATION
0020  3032 ;               f.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
0020  3033 ;       5.  TEST CACHE ERROR REGISTER FOR A MISS
0020  3034 ;       6.  LOAD NEW U-CODE
0020  3035 ;       7.  EXECUTE DCS PROGRAM
0020  3036 ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
0020  3037 ;               b.  WRITE LONLIT TO VA REGISTER
0020  3038 ;               c.  PERFORM READ ON THIS LOCATION
0020  3039 ;               d.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
0020  3040 ;       8.  TEST CACHE ERROR REGISTER FOR A HIT
0020  3041 ;       9.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0020  3042 ;
0020  3043 ;
0020  3044 ; TEST PATTERNS
0020  3045 ;
0020  3046 ;               1.      002AA000
0020  3047 ;               2.      00EAA000
0020  3048 ;               3.      008AA000
0020  3049 ;               4.      00BAA000
0020  3050 ;               5.      00A2A000
0020  3051 ;               6.      00AEA000
0020  3052 ;               7.      00A8A000
0020  3053 ;               8.      00ABA000
0020  3054 ;               9.      00AA2000
0020  3055 ;               A.      00AAE000
0020  3056 ;               B.      00AA8000
0020  3057 ;               C.      00AAB000
0020  3058 ;
0020  3059 ; LOGIC DESCRIPTION
0020  3060 ;       TESTED HERE ARE THE PATH PA<23:12> AND THE PATH OF
0020  3061 ;       THE TAG MEMORY DATA TO THE CACHE HIT/MISS COMPARATOR.
0020  3062 ;       AND THE DC102'S WHICH ARE THE COMPARATOR.
0020  3063 ;
0020  3064 ; ASSUMPTIONS
0020  3065 ;       THIS TEST ASSUMES THE DPM IS OPERATIONAL.
```

```
                     0020  3066 ;
                     0020  3067 ; ERROR DESCRIPTION
                     0020  3068 ;       EXPECTED CACHE HIT/MISS BIT
                     0020  3069 ;       GOT CACHE HIT/MISS BIT
                     0020  3070 ;       INDEX VALUE, INDICATES WHICH TEST PATTERN WAS BEING USED
                     0020  3071 ;
                     0020  3072 ;--
                     0020  3073 ;*******************************************************************************
                     0020  3074 ;///////////////////////////////////////////////////////////////////////////////
                     0020  3075
                     0020  3076         INITIALIZE                              ;INIT THE CPU
                     0022  3077         EXPUTRAP                                ;IGNORE U-TRAPS
                     0024  3078         LOOP            I,1,12                  ;CREATE A TEST LOOP OF 12
                     002B  3079         LOADDCS         10$,,0,<^X0C>           ;LOAD TEST U-CODE INTO DCS
                     0032  3080         LOADREG         WDREG,1$,I,L            ;TEST PATTERN TO WD REGISTER
                     003A  3081         ERRLOOP                                 ;ERROR LOOP POINT
                     003C  3082         FETCH           0                       ;START DCS PROGRAM
                     0041  3083         BRSTCLK         7                       ;END DCS PROGRAM
                     0045  3084         FETCH           8                       ;START DCS PROGRAM
                     004A  3085         BRSTCLK         <^X0B>                  ;END DCS PROGRAM
                     004E  3086         CMPREGMSK       WHREG,2$,,4$,,L,        ;TEST FOR CACHE MISS
                     005A  3087         IFERROR         ,<<CAK>>,               ;FAILING ARRAY
                     0060  3088         LOADDCS         20$,,0,9                ;LOAD NEW TEST U-CODE
                     0067  3089         ERRLOOP                                 ;ERROR LOOP POINT
                     0069  3090         FETCH           0                       ;START DCS PROGRAM
                     006E  3091         BRSTCLK         4                       ;END DCS PROGRAM
                     0072  3092         FETCH           5                       ;START DCS PROGRAM
                     0077  3093         BRSTCLK         8                       ;END DCS PROGRAM
                     007B  3094         CMPREGMSK       WHREG,3$,,4$,,L,        ;TEST FOR CACHE HIT
                     0087  3095         IFERROR         ,<<CAK>>,               ;FAILING ARRAY
                     008D  3096         ENDLOOP         I                       ;END TEST LOOP
                     0090  3097         SKIP                                    ;GO TO NEXT TEST
                     0097  3098
                     0097  3099 BEGIN_TEST_DATA
                     0097
                     0097          ;-----------------------------------------------------------------------
                     0097  3100
002AA000             0097  3101 1$:      .LONG    ^X002AA000                     ;TEST PATTERN TABLE
00EAA000             009B  3102          .LONG    ^X00EAA000
008AA000             009F  3103          .LONG    ^X008AA000
00BAA000             00A3  3104          .LONG    ^X00BAA000
00A2A000             00A7  3105          .LONG    ^X00A2A000
00AEA000             00AB  3106          .LONG    ^X00AEA000
00A8A000             00AF  3107          .LONG    ^X00A8A000
00ABA000             00B3  3108          .LONG    ^X00ABA000
00AA2000             00B7  3109          .LONG    ^X00AA2000
00AAE000             00BB  3110          .LONG    ^X00AAE000
00AA8000             00BF  3111          .LONG    ^X00AA8000
00AAB000             00C3  3112          .LONG    ^X00AAB000
                     00C7  3113
00000000             00C7  3114 2$:      .LONG    ^X00000000                     ;COMPARE DATA FOR CACHE MISS
                     00CB  3115
01000000             00CB  3116 3$:      .LONG    ^X01000000                     ;COMPARE DATA FOR CACHE HIT
                     00CF  3117
01000000             00CF  3118 4$:      .LONG    ^X01000000                     ;MASK FOR CMPREGMSK PSEUDO-OP
```

```
                                  00D3
                                  00D3  10$:                                                  ;TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801  00D3  ;% 00   NOP
U 01, 7700,7800,7FAA,AFFF,8470,1802  00DE  ;% 01   LONLIT_[00AAA000]                          ;ADDRESS FOR CACHE WRITE
U 02, 7700,4800,5BE4,03D4,4A70,1803  00E9  ;% 02   VA_R[LONLIT]                               ;MOVE ADDRESS TO VA REGISTER
U 03, 7701,8800,5BE4,02D8,5590,1804  00F4  ;% 03   WRITE.LONG R[ZERO]                         ;WRITE 0 TO CACHE
U 04, 7700,4800,5BE4,03D8,5470,1805  00FF  ;% 04   WDR_UNROT(R[ZERO])                         ;HOLD DATA FOR AN EXTRA CYCLE
U 05, 5700,4800,0364,0300,4A70,1806  010A  ;% 05   VA_RDM                                     ;LOAD VA WITH TEST PATTERN
U 06, 7700,4800,0364,0300,0510,1807  0115  ;% 06   READ.LONG                                  ;TRY TO READ TEST LOCATION
U 07, 7300,C800,0364,0300,0470,1808  0120  ;% 07   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
U 08, 7700,4800,0364,0300,0470,1809  012B  ;% 08   NOP
U 09, 7700,4800,5BE4,0300,6870,180A  0136  ;% 09   MEMSCAR_R[TEMP0]                           ;LOAD ADD OF CACHE ERROR REG
U 0A, 6700,C800,0364,0300,6470,180B  0141  ;% 0A   RDM_WB,WCTRL/MEMSCR                        ;READ CACHE ERROR REG TO RDM
U 0B, 7300,4800,0364,0300,0470,180C  014C  ;% 0B   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
                                  0157
                                  0157  20$:                                                  ;MORE TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801  0157  ;% 00   NOP
U 01, 7700,7800,7FAA,AFFF,8470,1802  0162  ;% 01   LONLIT_[00AAA000]                          ;ADDRESS FOR CACHE READ
U 02, 7700,4800,5BE4,03D4,4A70,1803  016D  ;% 02   VA_R[LONLIT]                               ;MOVE ADDRESS TO VA REGISTER
U 03, 7700,4800,0364,0300,0510,1804  0178  ;% 03   READ.LONG                                  ;READ CACHE LOCATION 00AAA000
U 04, 7300,4800,0364,0300,0470,1805  0183  ;% 04   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
U 05, 7700,4800,0364,0300,0470,1806  018E  ;% 05   NOP
U 06, 7700,C800,5BE4,0300,6870,1807  0199  ;% 06   MEMSCAR_R[TEMP0]                           ;LOAD ADD OF CACHE ERROR REG
U 07, 6700,C800,0364,0300,6470,1808  01A4  ;% 07   RDM_WB,WCTRL/MEMSCR                        ;READ CACHE ERROR REG TO RDM
U 08, 7300,4800,0364,0300,0470,1809  01AF  ;% 08   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
                                  01BA
                                  01BA  BEGIN_CPU_DATA
                                  0002
                                  0002  RTEMP_DATA
                                  0001
                        04000000  0001          .LONG   ^X04000000                        ;CACHE ERROR REGISTER ADDRESS
                                  0005
                                  0005  END_CPU_DATA
                                  01BA
                                  01BA  END_TEST_DATA
                                  01BA
                                  01BA      ;-----------------------------------------------------------------------
                                  01BA
                                  01BA  ENDTEST
```

```
                        .SBTTL  "TEST   016     CACHE HIT COMPARATOR TEST 2
                ;*****************************************************************************
                ;++
                ;
                ;       CACHE HIT COMPARATOR TEST 2
                ;
                ; FUNCTIONAL DESCRIPTION
                ;       THIS IS A TEST OF THE CACHE MEMORY HIT/MISS COMPARATOR. ALONG WITH
                ;       THE OTHER CACHE HIT COMPARATOR TESTS, THIS TEST INSURES THAT
                ;       HITS AND MISSES ARE GENERATED CORRECTLY.
                ;
                ;
                ; TEST ALGORITHM
                ;       1.  SET UP A LOOP TO SELECT TEST PATTERNS
                ;       2.  LOAD U-CODE INTO DCS
                ;       3.  PUT TEST PATTERN IN WD REGISTER
                ;       4.  EXECUTE DCS PROGRAM
                ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
                ;               b.  WRITE LONLIT TO VA REGISTER
                ;               c.  WRITE 0'S TO CACHE LOCATION 00AAA000
                ;               d.  WRITE VA REGISTER WITH TEST PATTERN
                ;               e.  PERFORM CACHE READ ON THIS LOCATION
                ;               f.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
                ;       5.  TEST CACHE ERROR REGISTER FOR A MISS
                ;       6.  LOAD NEW U-CODE
                ;       7.  EXECUTE DCS PROGRAM
                ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
                ;               b.  WRITE LONLIT TO VA REGISTER
                ;               c.  PERFORM READ ON THIS LOCATION
                ;               d.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
                ;       8.  TEST CACHE ERROR REGISTER FOR A HIT
                ;       9.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                ;
                ;
                ; TEST PATTERNS
                ;
                ;               1.      002AA000
                ;               2.      00EAA000
                ;               3.      008AA000
                ;               4.      00BAA000
                ;               5.      00A2A000
                ;               6.      00AEA000
                ;               7.      00A8A000
                ;               8.      00ABA000
                ;               9.      00AA2000
                ;               A.      00AAE000
                ;               B.      00AA8000
                ;               C.      00AAB000
                ;
                ; LOGIC DESCRIPTION
                ;       TESTED HERE ARE THE PATH PA<23:12> AND THE PATH OF
                ;       THE TAG MEMORY DATA TO THE CACHE HIT/MISS COMPARATOR.
                ;       AND THE DC102'S WHICH ARE THE COMPARATOR.
                ;
                ; ASSUMPTIONS
```

```
                0020  3274 ;       THIS TEST ASSUMES THE DPM IS OPERATIONAL.
                0020  3275 ;
                0020  3276 ;
                0020  3277 ; ERROR DESCRIPTION
                0020  3278 ;       EXPECTED CACHE HIT/MISS BIT
                0020  3279 ;       GOT CACHE HIT/MISS BIT
                0020  3280 ;       INDEX VALUE, INDICATES WHICH TEST PATTERN WAS BEING USED
                0020  3281 ;
                0020  3282 ;--
                0020  3283 ;*******************************************************************************
                0020  3284 ;///////////////////////////////////////////////////////////////////////////////
                0020  3285
                0020  3286         INITIALIZE                                      ;INIT THE CPU
                0022  3287         EXPUTRAP                                        ;IGNORE U-TRAPS
                0024  3288         LOOP            I,1,12                          ;CREATE A TEST LOOP OF 12
                002B  3289         LOADDCS         10$,,0,<^X0C>                   ;LOAD TEST U-CODE INTO DCS
                0032  3290         LOADREG         WDREG,1$,I,L                    ;LOAD TEST PATTERN INTO WD REG
                003A  3291         ERRLOOP                                         ;ERROR LOOP POINT
                003C  3292         FETCH           0                               ;START DCS PROGRAM
                0041  3293         BRSTCLK         7                               ;END DCS PROGRAM
                0045  3294         FETCH           8                               ;START DCS PROGRAM
                004A  3295         BRSTCLK         <^X0B>                          ;END DCS PROGRAM
                004E  3296         CMPREGMSK       WHREG,2$,,4$,,L,                ;TEST FOR CACHE MISS
                005A  3297         IFERROR         ,<<CAK>>,                       ;FAILING ARRAY
                0060  3298         LOADDCS         20$,,0,8                        ;LOAD NEW TEST U-CODE
                0067  3299         ERRLOOP                                         ;ERROR LOOP POINT
                0069  3300         FETCH           0                               ;START DCS PROGRAM
                006E  3301         BRSTCLK         3                               ;END DCS PROGRAM
                0072  3302         FETCH           4                               ;START DCS PROGRAM
                0077  3303         BRSTCLK         7                               ;END DCS PROGRAM
                007B  3304         CMPREGMSK       WHREG,3$,,4$,,L,                ;TEST FOR CACHE HIT
                0087  3305         IFERROR         ,<<CAK>>                        ;FAILING ARRAY
                008D  3306         ENDLOOP         I                               ;END TEST LOOP
                0090  3307         SKIP                                            ;GO TO NEXT TEST
                0097  3308
                0097  3309 BEGIN_TEST_DATA
                0097
                0097         ;-------------------------------------------------------------------------
                0097  3310
002AA000        0097  3311 1$:      .LONG   ^X002AA000                              ;TEST PATTERN TABLE
00EAA000        009B  3312          .LONG   ^X00EAA000
008AA000        009F  3313          .LONG   ^X008AA000
00BAA000        00A3  3314          .LONG   ^X00BAA000
00A2A000        00A7  3315          .LONG   ^X00A2A000
00AEA000        00AB  3316          .LONG   ^X00AEA000
00A8A000        00AF  3317          .LONG   ^X00A8A000
00ABA000        00B3  3318          .LONG   ^X00ABA000
00AA2000        00B7  3319          .LONG   ^X00AA2000
00AAE000        00BB  3320          .LONG   ^X00AAE000
00AA8000        00BF  3321          .LONG   ^X00AA8000
00AAB000        00C3  3322          .LONG   ^X00AAB000
                00C7  3323
00000000        00C7  3324 2$:      .LONG   ^X00000000                              ;COMPARE DATA FOR CACHE MISS
                00CB  3325
01000000        00CB  3326 3$:      .LONG   ^X01000000                              ;COMPARE DATA FOR CACHE HIT
```

```
                                00CF  3327
                     01000000   00CF  3328 4$:     .LONG   ^X01000000                      ;MASK FOR CMPREGMSK PSEUDO-OP
                                00D3  3329
                                00D3  3330 10$:                                          ;TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801  00D3  3331 ;% 00   NOP
U 01, 5700,C800,0364,0300,4A70,1802  00DE  3335 ;% 01   VA_RDM                           ;TEST PATTERN TO VA REGISTER
U 02, 7701,0800,5BE4,02D8,5590,1803  00E9  3339 ;% 02   WRITE.LONG R[ZERO]               ;WRITE 0 TO CACHE
U 03, 7700,C800,5BE4,03D8,5470,1804  00F4  3343 ;% 03   WDR_UNROT(R[ZERO])              ;HOLD DATA FOR EXTRA CYCLE
U 04, 7700,F800,7FAA,AFFF,8470,1805  00FF  3347 ;% 04   LONLIT_[00AAA000]              ;ADDRESS TO LONGLIT REG
U 05, 7700,4800,5BE4,03D4,4A70,1806  010A  3351 ;% 05   VA_R[LONLIT]                   ;MOVE ADDRESS TO VA REGISTER
U 06, 7700,4800,0364,0300,0510,1807  0115  3355 ;% 06   READ.LONG                      ;PERFORM READ
U 07, 7300,C800,0364,0300,0470,1808  0120  3359 ;% 07   NOP,STOP.CLOCK                 ;STOP CPU CLOCK
U 08, 7700,4800,0364,0300,0470,1809  012B  3363 ;% 08   NOP
U 09, 7700,4800,5BE4,0300,6870,180A  0136  3367 ;% 09   MEMSCAR_R[TEMP0]               ;ADD OF CACHE ERROR REGISTER
U 0A, 6700,C800,0364,0300,6470,180B  0141  3371 ;% 0A   RDM_WB,WCTRL/MEMSCR           ;READ CACHE ERROR REG TO RDM
U 0B, 7300,4800,0364,0300,0470,180C  014C  3375 ;% 0B   NOP,STOP.CLOCK                 ;STOP CPU CLOCK
                                0157  3379
                                0157  3380 20$:                                          ;MORE TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801  0157  3381 ;% 00   NOP
U 01, 5700,C800,0364,0300,4A70,1802  0162  3385 ;% 01   VA_RDM                           ;TEST PATTERN TO VA REGISTER
U 02, 7700,C800,0364,0300,0510,1803  016D  3389 ;% 02   READ.LONG                      ;READ TEST ADDRESS OF CACHE
U 03, 7300,C800,0364,0300,0470,1804  0178  3393 ;% 03   NOP,STOP.CLOCK                 ;STOP CPU CLOCK
U 04, 7700,4800,0364,0300,0470,1805  0183  3397 ;% 04   NOP
U 05, 7700,4800,5BE4,0300,6870,1806  018E  3401 ;% 05   MEMSCAR_R[TEMP0]               ;ADD OF CACHE ERROR REGISTER
U 06, 6700,C800,0364,0300,6470,1807  0199  3405 ;% 06   RDM_WB,WCTRL/MEMSCR           ;READ CACHE ERROR REG TO RDM
U 07, 7300,C800,0364,0300,0470,1808  01A4  3409 ;% 07   NOP,STOP.CLOCK                 ;STOP CPU CLOCK
                                01AF  3413
                                01AF  3414 BEGIN_CPU_DATA
                                0002  3415
                                0002  3416 RTEMP_DATA
                                0001  3417
                     04000000   0001  3418         .LONG   ^X04000000                 ;CACHE ERROR REGISTER ADDRESS
                                0005  3419
                                0005  3420 END_CPU_DATA
                                01AF  3421
                                01AF  3422 END_TEST_DATA
                                01AF
                                01AF        ;-----------------------------------------------------------------------
                                01AF  3423
                                01AF  3424 ENDTEST
```

```
0020          .SBTTL  "TEST   017     CACHE HIT COMPARATOR TEST 3
0020  3426 ;*******************************************************************************
0020  3427 ;++
0020  3428 ;
0020  3429 ;       CACHE HIT COMPARATOR TEST 3
0020  3430 ;
0020  3431 ; FUNCTIONAL DESCRIPTION
0020  3432 ;       THIS IS A TEST OF THE CACHE MEMORY HIT/MISS COMPARATOR. ALONG WITH
0020  3433 ;       THE OTHER CACHE HIT COMPARATOR TESTS, THIS TEST INSURES THAT
0020  3434 ;       HITS AND MISSES ARE GENERATED CORRECTLY.
0020  3435 ;
0020  3436 ;
0020  3437 ; TEST ALGORITHM
0020  3438 ;       1.  SET UP A LOOP TO SELECT TEST PATTERNS
0020  3439 ;       2.  LOAD U-CODE INTO DCS
0020  3440 ;       3.  PUT TEST PATTERN IN WD REGISTER
0020  3441 ;       4.  EXECUTE DCS PROGRAM
0020  3442 ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
0020  3443 ;               b.  WRITE LONLIT TO VA REGISTER
0020  3444 ;               c.  WRITE 0'S TO CACHE LOCATION 00AAA000
0020  3445 ;               d.  WRITE VA REGISTER WITH TEST PATTERN
0020  3446 ;               e.  PERFORM CACHE READ ON THIS LOCATION
0020  3447 ;               f.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
0020  3448 ;       5.  TEST CACHE ERROR REGISTER FOR A MISS
0020  3449 ;       6.  LOAD NEW U-CODE
0020  3450 ;       7.  EXECUTE DCS PROGRAM
0020  3451 ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
0020  3452 ;               b.  WRITE LONLIT TO VA REGISTER
0020  3453 ;               c.  PERFORM READ ON THIS LOCATION
0020  3454 ;               d.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
0020  3455 ;       8.  TEST CACHE ERROR REGISTER FOR A HIT
0020  3456 ;       9.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0020  3457 ;
0020  3458 ;
0020  3459 ; TEST PATTERNS
0020  3460 ;
0020  3461 ;               1.      00D55000
0020  3462 ;               2.      00155000
0020  3463 ;               3.      00755000
0020  3464 ;               4.      00455000
0020  3465 ;               5.      005D5000
0020  3466 ;               6.      00515000
0020  3467 ;               7.      00575000
0020  3468 ;               8.      00545000
0020  3469 ;               9.      0055D000
0020  3470 ;               A.      00551000
0020  3471 ;               B.      00557000
0020  3472 ;               C.      00554000
0020  3473 ;
0020  3474 ; LOGIC DESCRIPTION
0020  3475 ;       TESTED HERE ARE THE PATH PA<23:12> AND THE PATH OF
0020  3476 ;       THE TAG MEMORY DATA TO THE CACHE HIT/MISS COMPARATOR.
0020  3477 ;       AND THE DC102'S WHICH ARE THE COMPARATOR.
0020  3478 ;
0020  3479 ;
```

```
                     0020  3480 ; ASSUMPTIONS
                     0020  3481 ;       THIS TEST ASSUMES THE DPM IS OPERATIONAL.
                     0020  3482 ;
                     0020  3483 ;
                     0020  3484 ; ERROR DESCRIPTION
                     0020  3485 ;       EXPECTED CACHE HIT/MISS BIT
                     0020  3486 ;       GOT CACHE HIT/MISS BIT
                     0020  3487 ;       INDEX VALUE, INDICATES WHICH TEST PATTERN WAS BEING USED
                     0020  3488 ;
                     0020  3489 ;--
                     0020  3490 ;*******************************************************************************
                     0020  3491 ;///////////////////////////////////////////////////////////////////////////////
                     0020  3492
                     0020  3493         INITIALIZE                                      ;INIT THE CPU
                     0022  3494         EXPUTRAP                                        ;IGNORE U-TRAPS
                     0024  3495         LOOP            I,1,12                          ;CREATE A TEST LOOP
                     002B  3496         LOADDCS         10$,,0,<^X0C>                   ;LOAD U-CODE INTO DCS
                     0032  3497         LOADREG         WDREG,1$,I,L                    ;LOAD TEST PATTERN INTO WD REG
                     003A  3498         ERRLOOP                                         ;ERROR LOOP POINT
                     003C  3499         FETCH           0                               ;START DCS PROGRAM
                     0041  3500         BRSTCLK         7                               ;END DCS PROGRAM
                     0045  3501         FETCH           8                               ;START DCS PROGRAM
                     004A  3502         BRSTCLK         <^X0B>                          ;END DCS PROGRAM
                     004E  3503         CMPREGMSK       WHREG,2$,,4$,,L,                ;TEST FOR CACHE MISS
                     005A  3504         IFERROR         ,<<CAK>>                        ;FAILING ARRAY
                     0060  3505         LOADDCS         20$,,0,9                        ;LOAD NEW TEST U-CODE
                     0067  3506         ERRLOOP                                         ;ERROR LOOP POINT
                     0069  3507         FETCH           0                               ;START DCS PROGRAM
                     006E  3508         BRSTCLK         4                               ;END DCS PROGRAM
                     0072  3509         FETCH           5                               ;START DCS PROGRAM
                     0077  3510         BRSTCLK         8                               ;END DCS PROGRAM
                     007B  3511         CMPREGMSK       WHREG,3$,,4$,,L,                ;TEST FOR CACHE HIT
                     0087  3512         IFERROR         ,<<CAK>>                        ;FAILING ARRAY
                     008D  3513         ENDLOOP         I                               ;END TEST LOOP
                     0090  3514         SKIP                                            ;GO TO NEXT TEST
                     0097  3515
                     0097  3516 BEGIN_TEST_DATA
                     0097
                     0097         ;-------------------------------------------------------------------------
                     0097  3517
00D55000             0097  3518 1$:      .LONG   ^X00D55000                              ;TEST PATTERN TABLE
00155000             009B  3519          .LONG   ^X00155000
00755000             009F  3520          .LONG   ^X00755000
00455000             00A3  3521          .LONG   ^X00455000
005D5000             00A7  3522          .LONG   ^X005D5000
00515000             00AB  3523          .LONG   ^X00515000
00575000             00AF  3524          .LONG   ^X00575000
00545000             00B3  3525          .LONG   ^X00545000
0055D000             00B7  3526          .LONG   ^X0055D000
00551000             00BB  3527          .LONG   ^X00551000
00557000             00BF  3528          .LONG   ^X00557000
00554000             00C3  3529          .LONG   ^X00554000
                     00C7  3530
00000000             00C7  3531 2$:      .LONG   ^X00000000                              ;COMPARE DATA FOR CACHE MISS
                     00CB  3532
```

```
                          01000000  00CB  3533 3$:     .LONG   ^X01000000                      ;COMPARE DATA FOR CACHE HIT
                                    00CF  3534
                          01000000  00CF  3535 4$:     .LONG   ^X01000000                      ;MASK FOR CMPREGMSK PSEUDO-OP
                                    00D3  3536
                                    00D3  3537 10$:                                          ;TEST U-CODE
U 00. 7700,C800,0364,0300,0470,1801 00D3  3538 ;% 00   NOP
U 01. 7700,7800,7FD5,57FF,8470,1802 00DE  3542 ;% 01   LONLIT_[00555000]                     ;ADDRESS TO WRITE CACHE
U 02. 7700,4800,5BE4,03D4,4A70,1803 00E9  3546 ;% 02   VA_R[LONLIT]                          ;MOVE ADDRESS TO VA REGISTER
U 03. 7701,8800,5BE4,02D8,5590,1804 00F4  3550 ;% 03   WRITE.LONG R[ZERO]                    ;WRITE 0 TO CACHE
U 04. 7700,4800,5BE4,03D8,5470,1805 00FF  3554 ;% 04   WDR_UNROT(R[ZERO])                   ;HOLD DATA FOR EXTRA CYCLE
U 05. 5700,4800,0364,0300,4A70,1806 010A  3558 ;% 05   VA_RDM                               ;WRITE TEST PATTERN TO VA REG
U 06. 7700,4800,0364,0300,0510,1807 0115  3562 ;% 06   READ.LONG                            ;READ TEST ADDRESS
U 07. 7300,C800,0364,0300,0470,1808 0120  3566 ;% 07   NOP,STOP.CLOCK                       ;STOP CPU CLOCK
U 08. 7700,4800,0364,0300,0470,1809 012B  3570 ;% 08   NOP
U 09. 7700,4800,5BE4,0300,6870,180A 0136  3574 ;% 09   MEMSCAR_R[TEMP0]                     ;LOAD ADD OF CACHE ERROR REG
U 0A. 6700,C800,0364,0300,6470,180B 0141  3578 ;% 0A   RDM_WB,WCTRL/MEMSCR                  ;CACHE ERROR REGISTER TO RDM
U 0B. 7300,4800,0364,0300,0470,180C 014C  3582 ;% 0B   NOP,STOP.CLOCK                       ;STOP CPU CLOCK
                                    0157  3586
                                    0157  3587 20$:                                          ;MORE TEST U-CODE
U 00. 7700,C800,0364,0300,0470,1801 0157  3588 ;% 00   NOP
U 01. 7700,7800,7FD5,57FF,8470,1802 0162  3592 ;% 01   LONLIT_[00555000]                     ;ADDRESS TO READ CACHE
U 02. 7700,4800,5BE4,03D4,4A70,1803 016D  3596 ;% 02   VA_R[LONLIT]                          ;MOVE ADDRESS TO VA REGISTER
U 03. 7700,4800,0364,0300,0510,1804 0178  3600 ;% 03   READ.LONG                            ;READ CACHE LOCATION 00555000
U 04. 7300,4800,0364,0300,0470,1805 0183  3604 ;% 04   NOP,STOP.CLOCK                       ;STOP CPU CLOCK
U 05. 7700,4800,0364,0300,0470,1806 018E  3608 ;% 05   NOP
U 06. 7700,C800,5BE4,0300,6870,1807 0199  3612 ;% 06   MEMSCAR_R[TEMP0]                     ;LOAD ADD OF CACHE ERROR REG
U 07. 6700,C800,0364,0300,6470,1808 01A4  3616 ;% 07   RDM_WB,WCTRL/MEMSCR                  ;CACHE ERROR REG TO RDM
U 08. 7300,4800,0364,0300,0470,1809 01AF  3620 ;% 08   NOP,STOP.CLOCK                       ;STOP CPU CLOCK
                                    01BA  3624
                                    01BA  3625 BEGIN_CPU_DATA
                                    0002  3626
                                    0002  3627 RTEMP_DATA
                                    0001  3628
                          04000000  0001  3629         .LONG   ^X04000000                  ;CACHE ERROR REGISTER ADDRESS
                                    0005  3630
                                    0005  3631 END_CPU_DATA
                                    01BA  3632
                                    01BA  3633 END_TEST_DATA
                                    01BA
                                    01BA        ;-----------------------------------------------------------------------------
                                    01BA  3634
                                    01BA  3635 ENDTEST
```

```
0020        .SBTTL  "TEST   018     CACHE HIT COMPARATOR TEST 4
0020  3637 ;*******************************************************************************
0020  3638 ;++
0020  3639 ;
0020  3640 ;       CACHE HIT COMPARATOR TEST 4
0020  3641 ;
0020  3642 ; FUNCTIONAL DESCRIPTION
0020  3643 ;       THIS IS A TEST OF THE CACHE MEMORY HIT/MISS COMPARATOR. ALONG WITH
0020  3644 ;       THE OTHER CACHE HIT COMPARATOR TESTS, THIS TEST INSURES THAT
0020  3645 ;       HITS AND MISSES ARE GENERATED CORRECTLY.
0020  3646 ;
0020  3647 ;
0020  3648 ; TEST ALGORITHM
0020  3649 ;       1.  SET UP A LOOP TO SELECT TEST PATTERNS
0020  3650 ;       2.  LOAD U-CODE INTO DCS
0020  3651 ;       3.  PUT TEST PATTERN IN WD REGISTER
0020  3652 ;       4.  EXECUTE DCS PROGRAM
0020  3653 ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
0020  3654 ;               b.  WRITE LONLIT TO VA REGISTER
0020  3655 ;               c.  WRITE 0'S TO CACHE LOCATION 00AAA000
0020  3656 ;               d.  WRITE VA REGISTER WITH TEST PATTERN
0020  3657 ;               e.  PERFORM CACHE READ ON THIS LOCATION
0020  3658 ;               f.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
0020  3659 ;       5.  TEST CACHE ERROR REGISTER FOR A MISS
0020  3660 ;       6.  LOAD NEW U-CODE
0020  3661 ;       7.  EXECUTE DCS PROGRAM
0020  3662 ;               a.  PUT 00AAA000 INTO LONGLIT REGISTER
0020  3663 ;               b.  WRITE LONLIT TO VA REGISTER
0020  3664 ;               c.  PERFORM READ ON THIS LOCATION
0020  3665 ;               d.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM
0020  3666 ;       8.  TEST CACHE ERROR REGISTER FOR A HIT
0020  3667 ;       9.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0020  3668 ;
0020  3669 ;
0020  3670 ; TEST PATTERNS
0020  3671 ;
0020  3672 ;               1.      00D55000
0020  3673 ;               2.      00155000
0020  3674 ;               3.      00755000
0020  3675 ;               4.      00455000
0020  3676 ;               5.      005D5000
0020  3677 ;               6.      00515000
0020  3678 ;               7.      00575000
0020  3679 ;               8.      00545000
0020  3680 ;               9.      0055D000
0020  3681 ;               A.      00551000
0020  3682 ;               B.      00557000
0020  3683 ;               C.      00554000
0020  3684 ;
0020  3685 ; LOGIC DESCRIPTION
0020  3686 ;       TESTED HERE ARE THE PATH PA<23:12> AND THE PATH OF
0020  3687 ;       THE TAG MEMORY DATA TO THE CACHE HIT/MISS COMPARATOR.
0020  3688 ;       AND THE DC102'S WHICH ARE THE COMPARATOR.
0020  3689 ;
0020  3690 ;
```

```
                        0020  3691 ; ASSUMPTIONS
                        0020  3692 ;       THIS TEST ASSUMES THE DPM IS OPERATIONAL.
                        0020  3693 ;
                        0020  3694 ;
                        0020  3695 ; ERROR DESCRIPTION
                        0020  3696 ;       EXPECTED CACHE HIT/MISS BIT
                        0020  3697 ;       GOT CACHE HIT/MISS BIT
                        0020  3698 ;       INDEX VALUE, INDICATES WHICH TEST PATTERN WAS BEING USED
                        0020  3699 ;
                        0020  3700 ;--
                        0020  3701 ;*******************************************************************************
                        0020  3702 ;///////////////////////////////////////////////////////////////////////////////
                        0020  3703
                        0020  3704         INITIALIZE                                  ;INIT THE CPU
                        0022  3705         EXPUTRAP                                    ;IGNORE U-TRAPS
                        0024  3706         LOOP            I,1,12                      ;CREATE A TEST LOOP
                        002B  3707         LOADDCS         10$,,0,<^X0C>               ;LOAD TEST U-CODE INTO DCS
                        0032  3708         LOADREG         WDREG,1$,I,L                ;LOAD TEST PATTERN INTO WD REG
                        003A  3709         ERRLOOP                                     ;ERROR LOOP POINT
                        003C  3710         FETCH           0                           ;START DCS PROGRAM
                        0041  3711         BRSTCLK         7                           ;END DCS PROGRAM
                        0045  3712         FETCH           8                           ;START DCS PROGRAM
                        004A  3713         BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                        004E  3714         CMPREGMSK       WHREG,2$,,4$,,L,            ;TEST FOR CACHE MISS
                        005A  3715         IFERROR         ,<<CAK>>,                   ;FAILING ARRAY
                        0060  3716         LOADDCS         20$,,0,8                    ;LOAD NEW TEST U-CODE
                        0067  3717         ERRLOOP                                     ;ERROR LOOP POINT
                        0069  3718         FETCH           0                           ;START DCS PROGRAM
                        006E  3719         BRSTCLK         3                           ;END DCS PROGRAM
                        0072  3720         FETCH           4                           ;START DCS PROGRAM
                        0077  3721         BRSTCLK         7                           ;END DCS PROGRAM
                        007B  3722         CMPREGMSK       WHREG,3$,,4$,,L,            ;TEST FOR CACHE HIT
                        0087  3723         IFERROR         ,<<CAK>>                    ;FAILING ARRAY
                        008D  3724         ENDLOOP         I                           ;END TEST LOOP
                        0090  3725         SKIP                                        ;GO TO NEXT TEST
                        0097  3726
                        0097  3727 BEGIN_TEST_DATA
                        0097
                        0097         ;-------------------------------------------------------------------------
                        0097  3728
00D55000                0097  3729 1$:      .LONG    ^X00D55000                      ;TEST PATTERN TABLE
00155000                009B  3730         .LONG    ^X00155000
00755000                009F  3731         .LONG    ^X00755000
00455000                00A3  3732         .LONG    ^X00455000
005D5000                00A7  3733         .LONG    ^X005D5000
00515000                00AB  3734         .LONG    ^X00515000
00575000                00AF  3735         .LONG    ^X00575000
00545000                00B3  3736         .LONG    ^X00545000
0055D000                00B7  3737         .LONG    ^X0055D000
00551000                00BB  3738         .LONG    ^X00551000
00557000                00BF  3739         .LONG    ^X00557000
00554000                00C3  3740         .LONG    ^X00554000
                        00C7  3741
00000000                00C7  3742 2$:      .LONG    ^X00000000                      ;COMPARE DATA FOR CACHE MISS
                        00CB  3743
```

```
                          01000000  00CB  3744 3$:     .LONG   ^X01000000                    ;COMPARE DATA FOR CACHE HIT
                                    00CF  3745
                          01000000  00CF  3746 4$:     .LONG   ^X01000000                    ;MASK FOR CMPREGMSK PSEUDO OP
                                    00D3  3747
                                    00D3  3748 10$:                                        ;TEST U-CODE
U 00. 7700,C800,0364,0300,0470,1801 00D3  3749 ;% 00   NOP
U 01. 5700,C800,0364,0300,4A70,1802 00DE  3753 ;% 01   VA_RDM                               ;MOVE TEST PATTERN TO VA REG
U 02. 7701,0800,5BE4,02D8,5590,1803 00E9  3757 ;% 02   WRITE.LONG R[ZERO]                   ;WRITE 0 TO CACHE
U 03. 7700,C800,5BE4,03D8,5470,1804 00F4  3761 ;% 03   WDR_UNROT(R[ZERO])                  ;HOLD DATA FOR EXTRA CYCLE
U 04. 7700,F800,7FD5,57FF,8470,1805 00FF  3765 ;% 04   LONLIT_[00555000]                   ;READ ADDRESS TO LONGLIT REG
U 05. 7700,4800,5BE4,03D4,4A70,1806 010A  3769 ;% 05   VA_R[LONLIT]                        ;MOVE ADDRESS TO VA REGISTER
U 06. 7700,4800,0364,0300,0510,1807 0115  3773 ;% 06   READ.LONG                          ;PERFORM CACHE READ
U 07. 7300,C800,0364,0300,0470,1808 0120  3777 ;% 07   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
U 08. 7700,4800,0364,0300,0470,1809 012B  3781 ;% 08   NOP
U 09. 7700,4800,5BE4,0300,6870,180A 0136  3785 ;% 09   MEMSCAR_R[TEMP0]                   ;LOAD ADD OF CACHE ERROR REG
U 0A. 6700,C800,0364,0300,6470,180B 0141  3789 ;% 0A   RDM_WB,WCTRL/MEMSCR                ;CACHE ERROR REG TO RDM
U 0B. 7300,4800,0364,0300,0470,180C 014C  3793 ;% 0B   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
                                    0157  3797
                                    0157  3798 20$:                                        ;MORE TEST U-CODE
U 00. 7700,C800,0364,0300,0470,1801 0157  3799 ;% 00   NOP
U 01. 5700,C800,0364,0300,4A70,1802 0162  3803 ;% 01   VA_RDM                               ;TEST ADDRESS TO VA REGISTER
U 02. 7700,C800,0364,0300,0510,1803 016D  3807 ;% 02   READ.LONG                          ;PERFORM CACHE READ
U 03. 7300,C800,0364,0300,0470,1804 0178  3811 ;% 03   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
U 04. 7700,4800,0364,0300,0470,1805 0183  3815 ;% 04   NOP
U 05. 7700,4800,5BE4,0300,6870,1806 018E  3819 ;% 05   MEMSCAR_R[TEMP0]                   ;LOAD ADD OF CACHE ERROR REG
U 06. 6700,C800,0364,0300,6470,1807 0199  3823 ;% 06   RDM_WB,WCTRL/MEMSCR                ;CACHE ERROR REG TO RDM
U 07. 7300,C800,0364,0300,0470,1808 01A4  3827 ;% 07   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
                                    01AF  3831
                                    01AF  3832 BEGIN_CPU_DATA
                                    0002  3833
                                    0002  3834 RTEMP_DATA
                                    0001  3835
                          04000000  0001  3836         .LONG   ^X04000000                ;CACHE ERROR REGISTER
                                    0005  3837
                                    0005  3838 END_CPU_DATA
                                    01AF  3839
                                    01AF  3840 END_TEST_DATA
                                    01AF
                                    01AF        ;-----------------------------------------------------------------------------
                                    01AF  3841
                                    01AF  3842 ENDTEST
```

```
002B          .SBTTL  "TEST   019     CACHE DATA MEMORY DUAL ADDRESSING TEST
002B  3844 ;*******************************************************************************
002B  3845 ;++
002B  3846 ;
002B  3847 ;       CACHE DATA MEMORY DUAL ADDRESSING TEST
002B  3848 ;
002B  3849 ; FUNCTIONAL DESCRIPTION
002B  3850 ;       THIS IS A DUAL ADDRESSING TEST OF THE CACHE DATA MEMORY.
002B  3851 ;       TESTED HERE IS THE CACHE INDEX, PA<11:02>, IN THE
002B  3852 ;       DATA MEMORY.
002B  3853 ;
002B  3854 ; TEST ALGORITHM
002B  3855 ;       1.  CREATE A TEST LOOP OF 11 (I)
002B  3856 ;       2.  LOAD U-CODE TO VALIDATE A CACHE LOCATION
002B  3857 ;       3.  LOAD A TEST PATTERN INTO THE WD REGISTER
002B  3858 ;       4.  EXECUTE DCS PROGRAM
002B  3859 ;               a.  MOVE TEST PATTERN TO RTEMP0
002B  3860 ;               b.  LOAD TEST PATTERN INTO VA REGISTER
002B  3861 ;               c.  WRITE RTEMP0 INTO CACHE AS SPECIFIED BY VA
002B  3862 ;       5.  LOAD U-CODE TO WRITE ANOTHER CACHE LOCATION AND READ LOCATION
002B  3863 ;           WRITTEN ABOVE
002B  3864 ;       6.  CREATE A TEST LOOP OF 11 (J)
002B  3865 ;       7.  IF I AND J POINT TO SAME TEST PATTERN GO TO STEP 11
002B  3866 ;       8.  LOAD A TEST PATTERN INTO THE WD REGISTER
002B  3867 ;       9.  EXECUTE DCS PROGRAM
002B  3868 ;               a.  MOVE TEST PATTERN TO RTEMP1
002B  3869 ;               b.  LOAD TEST PATTERN INTO VA REGISTER
002B  3870 ;               c.  WRITE RTEMP1 INTO CACHE AS SPECIFIED BY VA
002B  3871 ;               d.  LOAD VA WITH RTEMP0
002B  3872 ;               e.  PERFORM CACHE READ
002B  3873 ;               f.  MOVE DATA READ TO RDM WH REGISTER
002B  3874 ;      10.  TEST WH REGISTER FOR TEST PATTERN POINTED TO BY I
002B  3875 ;      11.  IF NO ERROR GO TO STEP 7 FOR REMAINING TEST PATTERNS
002B  3876 ;      12.  GO TO STEP 2 UNTIL LOOP I IS EXHAUSTED
002B  3877 ;
002B  3878 ; TEST PATTERNS
002B  3879 ;       EVERY COMBINATION OF TWO ADDRESSES FROM THE FOLLOWING TABLE
002B  3880 ;       ARE USED FOR TEST ADDRESS (I) AND TEST ADDRESS (J), EXCEPT FOR
002B  3881 ;       THOSE IN WHICH TEST ADDRESS (I) AND TEST ADDRESS (J) ARE EQUAL.
002B  3882 ;               1.      00000000
002B  3883 ;               2.      00000800
002B  3884 ;               3.      00000400
002B  3885 ;               4.      00000200
002B  3886 ;               5.      00000100
002B  3887 ;               6.      00000080
002B  3888 ;               7.      00000040
002B  3889 ;               8.      00000020
002B  3890 ;               9.      00000010
002B  3891 ;               A.      00000008
002B  3892 ;               B.      00000004
002B  3893 ;
002B  3894 ; LOGIC DESCRIPTION
002B  3895 ;       TESTED HERE IS THE ADDRESS PATH FROM VA<11:02> TO
002B  3896 ;       THE CACHE DATA MEMORY RAM CHIPS.
002B  3897 ;
```

```
                     002B  3898 ; ASSUMPTIONS
                     002B  3899 ;
                     002B  3900 ; ERROR DESCRIPTION
                     002B  3901 ;       EXPECTED CACHE DATA
                     002B  3902 ;       GOT CACHE DATA
                     002B  3903 ;       INDEX VALUE, INDICATING WHICH OF THE ABOVE TEST PATTERNS
                     002B  3904 ;               WAS BEING USED AS TEST ADDRESS 1 WHEN THE
                     002B  3905 ;               FAULT WAS DETECTED.
                     002B  3906 ;       INDEX VALUE, INDICATING WHICH OF THE ABOVE TEST PATTERNS
                     002B  3907 ;               WAS BEING USED AS TEST ADDRESS 2 WHEN THE
                     002B  3908 ;               FAULT WAS DETECTED.
                     002B  3909 ;
                     002B  3910 ;--
                     002B  3911 ;*******************************************************************************
                     002B  3912 ;///////////////////////////////////////////////////////////////////////////////
                     002B  3913
                     002B  3914         INITIALIZE                                  ;INIT THE CPU
                     002D  3915         EXPUTRAP                                    ;IGNORE U-TRAPS
                     002F  3916         LOOP            I,1,11                      ;LOOP TO SELECT TEST ADDRESS
                     0036  3917         LOADDCS         10$,,0,6                    ;LOAD TEST U-CODE
                     003D  3918         LOADREG         WDREG,2$,I,L                ;LOAD ADDRESS INTO WD REG
                     0045  3919         FETCH           0                           ;START DCS PROGRAM
                     004A  3920         BRSTCLK         5                           ;END DCS PROGRAM
                     004E  3921         LOADDCS         20$,,0,<^X0B>               ;LOAD NEW TEST U-CODE
                     0055  3922         LOOP            J,1,11                      ;LOOP TO SELECT ALTERNATE ADD
                     005C  3923         SKIP            1$,ONEQUAL,I,J              ;SKIP IF I = J
                     0063  3924         LOADREG         WDREG,2$,J,L                ;LOAD ADDRESS INTO WD REGISTER
                     006B  3925         ERRLOOP                                     ;ERROR LOOP POINT
                     006D  3926         FETCH           0                           ;START DCS PROGRAM
                     0072  3927         BRSTCLK         7                           ;END DCS PROGRAM
                     0076  3928         FETCH           8                           ;START DCS PROGRAM
                     007B  3929         BRSTCLK         <^X0A>                      ;END DCS PROGRAM
                     007F  3930         CMPREG          WHREG,2$,I,L,               ;TEST FOR CORRECT DATA
                     0088  3931         IFERROR         ,<<CAK>>                    ;FAILING ARRAY
                     008E  3932 1$:     ENDLOOP         J                           ;END TEST LOOP J
                     0091  3933         ENDLOOP         I                           ;END TEST LOOP I
                     0094  3934         SKIP                                        ;GO TO NEXT TEST
                     009B  3935
                     009B  3936 BEGIN_TEST_DATA
                     009B
                     009B               ;-------------------------------------------------------------------------
                     009B  3937
00000000             009B  3938 2$:     .LONG   ^X00000000                          ;TEST PATTERN TABLE
00000800             009F  3939         .LONG   ^X00000800
00000400             00A3  3940         .LONG   ^X00000400
00000200             00A7  3941         .LONG   ^X00000200
00000100             00AB  3942         .LONG   ^X00000100
00000080             00AF  3943         .LONG   ^X00000080
00000040             00B3  3944         .LONG   ^X00000040
00000020             00B7  3945         .LONG   ^X00000020
00000010             00BB  3946         .LONG   ^X00000010
00000008             00BF  3947         .LONG   ^X00000008
00000004             00C3  3948         .LONG   ^X00000004
                     00C7  3949
                     00C7  3950 10$:                                                ;TEST U-CODE
```

```
U 00. 7700.C800.0364.0300.0470.1801   00C7  3951 ;% 00   NOP
U 01. 5700.8840.0364.0300.0470.1802   00D2  3955 ;% 01   R[TEMP0]_RDM                    ;TEST ADDRESS TO RTEMP0
U 02. 7700.4800.5BE4.0300.4A70.1803   00DD  3959 ;% 02   VA_R[TEMP0]                    ;MOVE ADDRESS TO VA
U 03. 7701.8800.5BE4.0200.5590.1804   00E8  3963 ;% 03   WRITE.LONG R[TEMP0]           ;WRITE CACHE
U 04. 7700.4800.5BE4.0300.5470.1805   00F3  3967 ;% 04   WDR_UNROT(R[TEMP0])           ;HOLD DATA FOR EXTRA CYCLE
U 05. 7300.4800.0364.0300.0470.1806   00FE  3971 ;% 05   NOP,STOP.CLOCK                 ;STOP CPU CLOCK
                                      0109  3975
                                      0109  3976 20$:                                 ;MORE TEST U-CODE
U 00. 7700.C800.0364.0300.0470.1801   0109  3977 ;% 00   NOP
U 01. 5700.C840.0364.0304.0470.1802   0114  3981 ;% 01   R[TEMP1]_RDM                    ;ADDRESS TO RTEMP1
U 02. 7700.0800.5BE4.0304.4A70.1803   011F  3985 ;% 02   VA_R[TEMP1]                    ;MOVE ADDRESS TO VA
U 03. 7701.C800.5BE4.0204.5590.1804   012A  3989 ;% 03   WRITE.LONG R[TEMP1]           ;WRITE CACHE
U 04. 7700.0800.5BE4.0304.5470.1805   0135  3993 ;% 04   WDR_UNROT(R[TEMP1])           ;HOLD DATA FOR EXTRA CYCLE
U 05. 7700.4800.5BE4.0300.4A70.1806   0140  3997 ;% 05   VA_R[TEMP0]                    ;PUT TEST ADDRESS IN VA
U 06. 7700.4800.0364.0300.0510.1807   014B  4001 ;% 06   READ.LONG                      ;READ CACHE
U 07. 7300.C800.0364.0300.0470.1808   0156  4005 ;% 07   NOP,STOP.CLOCK                 ;STOP CPU CLOCK
U 08. 7700.4800.0364.0300.0470.1809   0161  4009 ;% 08   NOP
U 09. 6700.0812.5924.0300.0470.180A   016C  4013 ;% 09   RDM_WB,WB_M[MDR]               ;MOVE DATA TO RDM
U 0A. 7300.C800.0364.0300.0470.180B   0177  4017 ;% 0A   NOP,STOP.CLOCK                 ;STOP CUP CLOCK
                                      0182  4021
                                      0182  4022 END_TEST_DATA
                                      0182
                                      0182       ;-----------------------------------------------------------------------
                                      0182  4023
                                      0182  4024 ENDTEST
```

```
                        002A        .SBTTL  "TEST   01A     CACHE TAG MEMORY DUAL ADDRESSING TEST
                        002A  4026 ;*******************************************************************************
                        002A  4027 ;++
                        002A  4028 ;
                        002A  4029 ;       CACHE TAG MEMORY DUAL ADDRESSING TEST
                        002A  4030 ;
                        002A  4031 ; FUNCTIONAL DESCRIPTION
                        002A  4032 ;       THIS IS A DUAL ADDRESSING TEST OF THE CACHE TAG MEMORY.
                        002A  4033 ;       TESTED HERE IS THE CACHE INDEX, PA<11:02>, IN THE
                        002A  4034 ;       TAG MEMORY.
                        002A  4035 ;
                        002A  4036 ; TEST ALGORITHM
                        002A  4037 ;       1.  CREATE A TEST LOOP OF 11 (I)
                        002A  4038 ;       2.  SELECT A TEST PATTERN ACCORDING TO (I) AND LOAD INTO DCS
                        002A  4039 ;           ADDRESS 01
                        002A  4040 ;       3.  CREATE A TEST LOOP OF 11 (J)
                        002A  4041 ;       4.  IF I AND J POINT A SAME TEST PATTERN SKIP TO STEP 8
                        002A  4042 ;       5.  LOAD A TEST PATTERN ACCORDING TO (J) INTO WD REGISTER
                        002A  4043 ;       6.  EXECUTE DCS PROGRAM
                        002A  4044 ;               a.  PUT TEST PATTERN AT (I) INTO LONLIT
                        002A  4045 ;               b.  MOVE TEST PATTERN TO VA AND RTEMP0
                        002A  4046 ;               c.  WRITE CACHE AS SPECIFIED BY VA WITH RTEMP0
                        002A  4047 ;               d.  PUT TEST PATTERN AT (J) INTO RTEMP2
                        002A  4048 ;               e.  MOVE TAG FIELD IN RTEMP3 TO Q REGISTER
                        002A  4049 ;               f.  OR Q WITH RTEMP2 AND SAVE IN VA AND RTEMP2
                        002A  4050 ;               g.  WRITE CACHE AS SPECIFIED BY VA WITH RTEMP2
                        002A  4051 ;               h.  MOVE TEST PATTERN (I) TO VA
                        002A  4052 ;               i.  READ CACHE AS SPECIFIED BY VA
                        002A  4053 ;               j.  STORE DATA READ IN RTEMP1
                        002A  4054 ;               k.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM WH REGISTER
                        002A  4055 ;       7.  TEST WH REGISTER FOR CACHE HIT
                        002A  4056 ;       8.  IF NO ERROR GO TO STEP 4 FOR REMAINING TEST PATTERNS
                        002A  4057 ;       9.  GO TO STEP 2 UNTIL LOOP I IS EXHAUSTED
                        002A  4058 ;
                        002A  4059 ; TEST PATTERNS
                        002A  4060 ;       EVERY COMBINATION OF TWO ADDRESSES FROM THE FOLLOWING TABLE
                        002A  4061 ;       ARE USED FOR TEST ADDRESS (I) AND TEST ADDRESS (J), EXCEPT FOR
                        002A  4062 ;       THOSE IN WHICH TEST ADDRESS (I) AND TEST ADDRESS (J) ARE EQUAL.
                        002A  4063 ;               1.      00000000
                        002A  4064 ;               2.      00000800
                        002A  4065 ;               3.      00000400
                        002A  4066 ;               4.      00000200
                        002A  4067 ;               5.      00000100
                        002A  4068 ;               6.      00000080
                        002A  4069 ;               7.      00000040
                        002A  4070 ;               8.      00000020
                        002A  4071 ;               9.      00000010
                        002A  4072 ;               A.      00000008
                        002A  4073 ;               B.      00000004
                        002A  4074 ;
                        002A  4075 ; LOGIC DESCRIPTION
                        002A  4076 ;       TESTED HERE IS THE ADDRESS PATH FROM VA<11:02> TO
                        002A  4077 ;       THE CACHE TAG MEMORY RAM CHIPS.
                        002A  4078 ;
                        002A  4079 ; ASSUMPTIONS
```

```
                     002A  4080 ;       THIS TEST ASSUMES THE DPM IS OPERATIONAL
                     002A  4081 ;
                     002A  4082 ; ERROR DESCRIPTION
                     002A  4083 ;       EXPECTED CACHE HIT/MISS BIT
                     002A  4084 ;       GOT CACHE HIT/MISS BIT
                     002A  4085 ;       INDEX VALUE, INDICATING WHICH OF THE ABOVE TEST PATTERNS
                     002A  4086 ;              WAS BEING USED AS TEST ADDRESS 1 WHEN THE
                     002A  4087 ;              FAULT WAS DETECTED.
                     002A  4088 ;       INDEX VALUE, INDICATING WHICH OF THE ABOVE TEST PATTERNS
                     002A  4089 ;              WAS BEING USED AS TEST ADDRESS 2 WHEN THE
                     002A  4090 ;              FAULT WAS DETECTED.
                     002A  4091 ;       RTEMP0; ADDRESS BEING READ (I)
                     002A  4092 ;       RTEMP1; DATA READ
                     002A  4093 ;       RTEMP2; 2ND ADDRESS BEING WRITTEN (J)
                     002A  4094 ;
                     002A  4095 ;--
                     002A  4096 ;*******************************************************************************
                     002A  4097 ;///////////////////////////////////////////////////////////////////////////////
                     002A  4098
                     002A  4099         INITIALIZE                                  ;INIT THE CPU
                     002C  4100         EXPUTRAP                                    ;IGNORE U-TRAPS
                     002E  4101         LOOP            I,1,11                      ;LOOP TO SELECT TEST PATTERNS
                     0035  4102         LDFIELD         2$,I,L,3$,31,62,LONLIT      ;TEST PATTERN TO LONLIT
                     0046  4103         LOADDCS         3$,,1,1                     ;LOAD TEST PATTERN TO DCS
                     004D  4104         LOOP            J,1,11                      ;LOOP TO SELECT 2ND TEST PATT.
                     0054  4105         SKIP            1$,ONEQUAL,I,J              ;SKIP IF TEST PATTERNS THE SAME
                     005B  4106         LOADREG         WDREG,2$,J,L                ;2ND PATTERN TO WD REG
                     0063  4107         ERRLOOP                                     ;ERROR LOOP POINT
                     0065  4108         FETCH           0                           ;START DCS PROGRAM
                     006A  4109         BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                     006E  4110         FETCH           <^X0D>                      ;START DCS PROGRAM
                     0073  4111         BRSTCLK         <^X11>                      ;END DCS PROGRAM
                     0077  4112         CMPREGMSK       WHREG,4$,,4$,,L,            ;TEST FOR CACHE HIT
                     0083  4113         IFERROR         3,<<CAK>>,                  ;FAILING ARRAY
                     0089  4114 1$:     ENDLOOP         J                           ;END LOOP J
                     008C  4115         ENDLOOP         I                           ;END LOOP I
                     008F  4116         SKIP                                        ;GO TO NEXT TEST
                     0096  4117
                     0096  4118 BEGIN_TEST_DATA
                     0096
                     0096       ;-------------------------------------------------------------------------
                     0096  4119
 00000000            0096  4120 2$:     .LONG   ^X00000000                          ;TEST PATTERN TABLE
 00000800            009A  4121         .LONG   ^X00000800
 00000400            009E  4122         .LONG   ^X00000400
 00000200            00A2  4123         .LONG   ^X00000200
 00000100            00A6  4124         .LONG   ^X00000100
 00000080            00AA  4125         .LONG   ^X00000080
 00000040            00AE  4126         .LONG   ^X00000040
 00000020            00B2  4127         .LONG   ^X00000020
 00000010            00B6  4128         .LONG   ^X00000010
 00000008            00BA  4129         .LONG   ^X00000008
 00000004            00BE  4130         .LONG   ^X00000004
                     00C2  4131
                     00C2  4132 3$:
```

```
U 01. 7700.7800.7FFF.FFFF.8470.1802  00C2  4133 ;* 01   LONLIT_[0]                                  ;U-WORD TO HOLD TEST PATTERNS
                                     00CD  4137
                          01000000   00CD  4138 4$:     .LONG   ^X01000000                         ;COMPARE DATA FOR CACHE HIT
                                     00D1  4139
                                     00D1  4140 BEGIN_CPU_DATA
                                     0002  4141
                                     0002  4142 DCS_DATA
                                     0001  4143
U 00. 7700.C800.0364.0300.0470.1801  0001  4144 ;* 00   NOP
U 01. 7700.7800.7FFF.FFFF.8470.1802  000C  4148 ;* 01   LONLIT_[0]                                  ;TEST ADDRESS TO LONLIT
U 02. 7700.0840.5BE4.03D4.4A70.1803  0017  4152 ;* 02   VA_R[LONLIT],SPW/RLONG                      ;ADD TO VA & RTEMP0
U 03. 7701.8800.5BE4.02D4.5590.1804  0022  4156 ;* 03   WRITE.LONG R[LONLIT]                        ;WRITE CACHE
U 04. 7700.4800.5BE4.03D4.5470.1805  002D  4160 ;* 04   WDR_UNROT(R[LONLIT])                       ;HOLD DATA FOR EXTRA CYCLE
U 05. 5700.4840.0364.0308.0470.1806  0038  4164 ;* 05   R[TEMP2]_RDM                               ;2ND TEST ADDRESS TO RTEMP2
U 06. 7700.8800.5BE5.030C.0470.1807  0043  4168 ;* 06   Q_R[TEMP3]                                 ;MOVE TAG TO Q REG
U 07. 7700.C840.03A4.0308.4A70.1808  004E  4172 ;* 07   VA_Q.OR.R[TEMP2],SPW/RLONG                 ;OR TAG WITH INDEX
U 08. 7701.4800.5BE4.0208.5590.1809  0059  4176 ;* 08   WRITE.LONG R[TEMP2]                        ;WRITE 2ND CACHE LOCATION
U 09. 7700.0800.5BE4.0308.5470.180A  0064  4180 ;* 09   WDR_UNROT(R[TEMP2])                       ;HOLD DATA FOR EXTRA CYCLE
U 0A. 7700.C800.5BE4.03D4.4A70.180B  006F  4184 ;* 0A   VA_R[LONLIT]                              ;MOVE 1ST ADDRESS TO VA
U 0B. 7700.C800.0364.0300.0510.180C  007A  4188 ;* 0B   READ.LONG                                 ;READ CACHE
U 0C. 7300.C800.0364.0300.0470.180D  0085  4192 ;* 0C   NOP,STOP.CLOCK                            ;STOP CPU CLOCK
U 0D. 7700.C800.0364.0300.0470.180E  0090  4196 ;* 0D   NOP
U 0E. 7700.0852.5924.0304.0470.180F  009B  4200 ;* 0E   R[TEMP1]_WB,WB_M[MDR]                     ;MOVE DATA TO RTEMP1
U 0F. 7700.8800.5BE4.0310.6870.1810  00A6  4204 ;* 0F   MEMSCAR_R[TEMP4]                          ;LOAD CACHE ERROR REG ADDRESS
U 10. 6700.4800.0364.0300.6470.1811  00B1  4208 ;* 10   RDM_WB,WCTRL/MEMSCR                       ;CACHE ERROR REG TO RDM
U 11. 7300.4800.0364.0300.0470.1812  00BC  4212 ;* 11   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                     00C7  4216
                                     00C7  4217 RTEMP_DATA
                                     0001  4218
                          00000000   0001  4219         .LONG   ^X00000000                       ;RTEMP0
                          00000000   0005  4220         .LONG   ^X00000000                       ;RTEMP1
                          00000000   0009  4221         .LONG   ^X00000000                       ;RTEMP2
                          000FF000   000D  4222         .LONG   ^X000FF000                       ;TAG FOR 2ND ADDRESS
                          04000000   0011  4223         .LONG   ^X04000000                       ;CACHE ERROR REG ADDRESS
                                     0015  4224
                                     0015  4225 END_CPU_DATA
                                     00D1  4226
                                     00D1  4227 END_TEST_DATA
                                     00D1
                                     00D1      ;-----------------------------------------------------------------------------
                                     00D1  4228
                                     00D1  4229 ENDTEST
```

```
0021          .SBTTL  "TEST   01B     CACHE DATA MEMORY MARCH TEST
0021   4231 ;*******************************************************************************
0021   4232 ;++
0021   4233 ;
0021   4234 ;       CACHE DATA MEMORY MARCH TEST
0021   4235 ;
0021   4236 ; FUNCTIONAL DESCRIPTION
0021   4237 ;       THIS IS A MARCH TEST OF THE CACHE MEMORY'S DATA STORE.
0021   4238 ;
0021   4239 ; TEST ALGORITHM
0021   4240 ;       1.  LOAD U-CODE TO WRITE 0'S IN ALL CACHE LOCATIONS
0021   4241 ;       2.  LOAD INDEX I INTO WD REGISTER TO GENERATE CACHE ADDRESS
0021   4242 ;       3.  EXECUTE DCS PROGRAM
0021   4243 ;               a.  MOVE INDEX INTO RTEMP0
0021   4244 ;               b.  SET PLATCH TO 2
0021   4245 ;               c.  ROTATE INDEX LEFT 2 AND LOAD INTO VA AND RTEMP0
0021   4246 ;               d.  WRITE CACHE ACCORDING TO VA WITH 0'S
0021   4247 ;       4.  GO TO STEP 3 UNTIL ALL LOCATIONS ARE ZEROED.
0021   4248 ;       5.  LOAD U-CODE TO READ AND WRITE CACHE
0021   4249 ;       6.  CREATE A LOOP OF 4 TO SELECT TEST PATTERNS (I)
0021   4250 ;       7.  SELECT A TEST PATTERN AND LOAD INTO LONLIT FIELD
0021   4251 ;       8.  LOAD LONLIT INTO DCS U-CODE
0021   4252 ;       9.  CREATE A LOOP OF 1024 TO SELECT CACHE LOCATIONS (J)
0021   4253 ;      10.  LOAD INDEX J INTO WD REGISTER TO GENERATE CACHE ADDRESS
0021   4254 ;      11.  EXECUTE DCS PROGRAM
0021   4255 ;               a.  MOVE INDEX INTO RTEMP0
0021   4256 ;               b.  SET PLATCH TO 2
0021   4257 ;               c.  ROTATE INDEX LEFT 2 AND LOAD INTO VA AND RTEMP0
0021   4258 ;               d.  READ CACHE ACCORDING TO VA
0021   4259 ;               e.  MOVE DATA READ TO RDM WH REG
0021   4260 ;               f.  WRITE CACHE ACCORDING TO VA WITH NEXT TEST PATTERN
0021   4261 ;      12.  TEST WH REGISTER FOR CORRECT DATA
0021   4262 ;      13.  IF NO ERROR GO TO STEP 9 FOR REMAING CACHE LOCATIONS.
0021   4263 ;      14.  GO TO STEP 6 FOR REMAINING TEST PATTERNS
0021   4264 ;      15.  REPEAT STEPS 6 THROUGH 14 WITH INDEX J DECREMENTING FROM
0021   4265 ;           1024 TO 1.
0021   4266 ;
0021   4267 ;
0021   4268 ; TEST PATTERNS
0021   4269 ;       EACH OF THE FOLLOWING DATA PATTERNS IS USED IN THIS TEST:
0021   4270 ;               0.      00000000
0021   4271 ;               1.      FFFFFFFF
0021   4272 ;               2.      07070707
0021   4273 ;               3.      F0F0F0F0
0021   4274 ;               4.      00000000
0021   4275 ;
0021   4276 ; LOGIC DESCRIPTION
0021   4277 ;       TEST HERE ARE THE 1x1024 RAM CHIPS USED IN THE CACHE
0021   4278 ;       DATA MEMORY.
0021   4279 ;
0021   4280 ; ASSUMPTIONS
0021   4281 ;       LOGIC OUTSIDE OF THESE RAM CHIPS IS ASSUMED TO BE FUNCTIONAL.
0021   4282 ;
0021   4283 ; ERROR DESCRIPTION
0021   4284 ;       EXPECTED CACHE DATA
```

```
                     ;       GOT CACHE DATA
                     ;       INDEX I
                     ;       INDEX J
                     ;       RTEMP0; CACHE ADDRESS BEING READ
                     ;
                     ;--
                     ;*****************************************************************************
                     ;/////////////////////////////////////////////////////////////////////////////

                             INITIALIZE                                      ;INIT THE CPU
                             EXPUTRAP                                        ;IGNORE U-TRAPS
                             LOADDCS         1$,,0,7                         ;LOAD TEST U-CODE INTO DCS
                             LOOP            I,1,1024                        ;CREATE A TEST LOOP OF 1024
                             SAVEINDEX       I,10$                           ;STORE VALUE OF INDEX IN 10$
                             LOADREG         WDREG,10$,,L                    ;LOAD LOOP COUNT INTO WD REG
                             FETCH           0                               ;START DCS PROGRAM
                             BRSTCLK         6                               ;END DCS PROGRAM
                             ENDLOOP         I                               ;END TEST LOOP
                             LOADDCS         30$,,0,<^X0E>                   ;LOAD TEST U-CODE INTO DCS
                             LOOP            I,1,4                           ;CREATE A TEST LOOP OF 4
                             LDFIELD         20$,I,L,31$,31,62,LONLIT        ;LOAD A TEST PATTERN
                             LOADDCS         31$,,<^X0A>,1                   ;LOAD TEST PATTERN INTO DCS
                             LOOP            J,1,1024                        ;CREATE A TEST LOOP OF 1024
                             SAVEINDEX       J,10$                           ;SAVE LOOP COUNT IN 10$
                             LOADREG         WDREG,10$,,L                    ;LOAD LOOP COUNT INTO WD REG
                             ERRLOOP                                         ;ERROR LOOP POINT
                             FETCH           0                               ;START DCS PROGRAM
                             BRSTCLK         5                               ;END DCS PROGRAM
                             FETCH           6                               ;START DCS PROGRAM
                             BRSTCLK         8                               ;END DCS PROGRAM
                             CMPREG          WHREG,15$,I,L,                  ;TEST FOR CORRECT TEST PATTERN
                             IFERROR         1,<<CAK>>                       ;FAILING ARRAY
                             FETCH           9                               ;START DCS PROGRAM
                             BRSTCLK         <^X0D>                          ;END DCS PROGRAM
                             ENDLOOP         J                               ;END TEST LOOP J
                             ENDLOOP         I                               ;END TEST LOOP I
                             LOOP            I,1,4                           ;CREATE A TEST LOOP OF 4
                             LDFIELD         20$,I,L,31$,31,62,LONLIT        ;LOAD A TEST PATTERN
                             LOADDCS         31$,,<^X0A>,1                   ;LOAD TEST PATTERN INTO DCS
                             LOOP            J,1024,1                        ;CREATE A DECREMENTING LOOP
                             SAVEINDEX       J,10$                           ;SAVE LOOP COUNT IN 10$
                             LOADREG         WDREG,10$,,L                    ;LOAD LOOP COUNT INTO WD REG
                             ERRLOOP                                         ;ERROR LOOP POINT
                             FETCH           0                               ;START DCS PROGRAM
                             BRSTCLK         5                               ;END DCS PROGRAM
                             FETCH           6                               ;START DCS PROGRAM
                             BRSTCLK         8                               ;END DCS PROGRAM
                             CMPREG          WHREG,15$,I,L,                  ;TEST FOR CORRECT TEST PATTERN
                             IFERROR         1,<<CAK>>                       ;FAILING ARRAY
                             FETCH           9                               ;START DCS PROGRAM
                             BRSTCLK         <^X0D>                          ;END DCS PROGRAM
                             ENDLOOP         J                               ;END TEST LOOP J
                             ENDLOOP         I                               ;END TEST LOOP I
                             SKIP                                            ;GO TO NEXT TEST

                     ;
```

```
                                0124  4340 BEGIN_TEST_DATA
                                0124
                                0124       ;-------------------------------------------------------------------------
                                0124  4341
                                0124  4342 1$:                                                    ;TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801  0124  4343 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802  012F  4347 ;% 01   R[TEMP0]_RDM                              ;LOAD LOOP COUNT INTO RTEMP0
U 02, 7700,D800,EF64,0301,0470,1803  013A  4351 ;% 02   PL_[00000002]                            ;SET UP SHIFT COUNT OF 2
U 03, 7700,8840,8B70,0300,4A70,1804  0145  4355 ;% 03   VA_R[TEMP0].RL.P,SPW/RLONG               ;SHIFT LOOP COUNT LEFT 2 &
                                0150  4359                                                      ;LOAD INTO VA
U 04, 7701,0800,5BE4,02D8,5590,1805  0150  4360 ;% 04   WRITE.LONG R[ZERO]                       ;WRITE 0 TO SELECTED ADDRESS
U 05, 7700,4800,5BE4,03D8,5470,1806  015B  4364 ;% 05   WDR_UNROT(R[ZERO])                       ;HOLD DATA FOR EXTRA CYCLE
U 06, 7300,C800,0364,0300,0470,1807  0166  4368 ;% 06   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
                                0171  4372
                    00000000    0171  4373 10$:    .LONG   ^X00000000                          ;TEMPORARY STORAGE
                                0175  4374
                    00000000    0175  4375 15$:    .LONG   ^X00000000                          ;TEST PATTERN TABLE
                    FFFFFFFF    0179  4376 20$:    .LONG   ^XFFFFFFFF
                    07070707    017D  4377         .LONG   ^X07070707
                    F0F0F0F0    0181  4378         .LONG   ^XF0F0F0F0
                    00000000    0185  4379         .LONG   ^X00000000
                                0189  4380
                                0189  4381 30$:                                                    ;MORE TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801  0189  4382 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802  0194  4386 ;% 01   R[TEMP0]_RDM                              ;LOOP COUNT TO RTEMP0
U 02, 7700,D800,EF64,0301,0470,1803  019F  4390 ;% 02   PL_[00000002]                            ;LOAD SHIFT COUNT OF 2
U 03, 7700,8840,8B70,0300,4A70,1804  01AA  4394 ;% 03   VA_R[TEMP0].RL.P,SPW/RLONG               ;SHIFT LOOP COUNT LEFT BY 2 &
                                01B5  4398                                                      ;LOAD INTO VA
U 04, 7700,C800,0364,0300,0510,1805  01B5  4399 ;% 04   READ.LONG                               ;READ THE SELECTED LOCATION
U 05, 7300,4800,0364,0300,0470,1806  01C0  4403 ;% 05   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
U 06, 7700,C800,0364,0300,0470,1807  01CB  4407 ;% 06   NOP
U 07, 6700,8812,5924,0300,0470,1808  01D6  4411 ;% 07   RDM_WB,WB_M[MDR]                        ;MOVE DATA TO RDM
U 08, 7300,4800,0364,0300,0470,1809  01E1  4415 ;% 08   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
U 09, 7700,4800,0364,0300,0470,180A  01EC  4419 ;% 09   NOP
                                01F7  4423 31$:
U 0A, 7700,7800,7FFF,FFFF,8470,180B  01F7  4424 ;% 0A   LONLIT_[0]                               ;LOAD NEW TEST PATTERN
U 0B, 7701,0800,5BE4,02D4,5590,180C  0202  4428 ;% 0B   WRITE.LONG R[LONLIT]                     ;WRITE NEW TEST PATTERN
U 0C, 7700,C800,5BE4,03D4,5470,180D  020D  4432 ;% 0C   WDR_UNROT(R[LONLIT])                     ;HOLD DATA FOR EXTRA CYCLE
U 0D, 7300,C800,0364,0300,0470,180E  0218  4436 ;% 0D   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
                                0223  4440
                                0223  4441 END_TEST_DATA
                                0223
                                0223       ;-------------------------------------------------------------------------
                                0223  4442
                                0223  4443 ENDTEST
```

```
0020                    .SBTTL  "TEST   01C     CACHE TAG MEMORY MARCH TEST
0020  4445 ;*******************************************************************************
0020  4446 ;++
0020  4447 ;
0020  4448 ;       CACHE TAG MEMORY MARCH TEST
0020  4449 ;
0020  4450 ; FUNCTIONAL DESCRIPTION
0020  4451 ;       THIS IS A MARCH TEST OF THE CACHE MEMORY'S TAG STORE.
0020  4452 ;
0020  4453 ; TEST ALGORITHM
0020  4454 ;       1.  LOAD U-CODE INTO DCS TO WRITE CACHE
0020  4455 ;       2.  CREATE A TEST LOOP OF 1024 TO WRITE ALL OF CACHE
0020  4456 ;       3.  SAVE THE LOOP COUNT IN 10$
0020  4457 ;       4.  MOVE THE LOOP COUNT INTO THE WD REGISTER
0020  4458 ;       5.  EXECUTE DCS PROGRAM
0020  4459 ;               a.  MOVE LOOP COUNT INTO RTEMP1
0020  4460 ;               b.  LOAD 2 INTO THE PLATCH
0020  4461 ;               c.  ROTATE RTEMP1 LEFT BY 2 AND STORE IN Q REGISTER
0020  4462 ;               d.  LOAD TAG FIELD INTO LONLIT (00AAA000)
0020  4463 ;               e.  OR Q WITH LONLIT AND STORE IN RTEMP0 AND VA REGISTER
0020  4464 ;               f.  WRITE RTEMP0 TO CACHE LOCATION SPECIFIED BY VA
0020  4465 ;       6.  GO TO STEP 3 FOR THE REMAINING CACHE LOCATIONS
0020  4466 ;       7.  LOAD U-CODE TO READ THEN WRITE CACHE
0020  4467 ;       8.  CREATE A TEST LOOP TO SELECT TEST PATTERNS (I)
0020  4468 ;       9.  LOAD TEST PATTERN FOR READ INTO LONLIT
0020  4469 ;      10.  LOAD TEST PATTERN FOR WRITE INTO LONLIT
0020  4470 ;      11.  CREATE A TEST LOOP OF 1024 TO ACCESS ALL CACHE LOCATIONS (J)
0020  4471 ;      12.  LOAD INDEX J INTO 10$
0020  4472 ;      13.  MOVE INDEX TO THE WD REGISTER
0020  4473 ;      14.  EXECUTE DCS PROGRAM
0020  4474 ;               a.  MOVE LOOP COUNT INTO RTEMP2
0020  4475 ;               b.  LOAD 2 INTO THE PLATCH
0020  4476 ;               c.  ROTATE RTEMP2 LEFT BY 2 AND STORE IN THE Q REGISTER
0020  4477 ;               d.  LOAD TAG FIELD INTO LONLIT
0020  4478 ;               e.  OR Q WITH LONLIT AND SAVE IN RTEMP0 AND VA REGISTER
0020  4479 ;               f.  PERFORM READ OF CACHE AS SPECIFIED BY VA
0020  4480 ;               g.  MOVE READ DATA TO RTEMP1
0020  4481 ;               h.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM WH REGISTER
0020  4482 ;               i.  LOAD WRITE TEST PATTERN TAG INTO LONLIT
0020  4483 ;               j.  OR Q WITH LONLIT AND SAVE IN Q REGISTER
0020  4484 ;               k.  LOAD Q INTO VA REGISTER
0020  4485 ;               l.  MOVE VA REGISTER TO RTEMP4
0020  4486 ;               m.  WRITE CONTENTS OF RTEMP4 TO CACHE LOCATION SPECIFIED BY VA
0020  4487 ;      15.  TEST CACHE ERROR REGISTER FOR A CACHE HIT
0020  4488 ;      16.  IF NO ERROR GO TO STEP 12 FOR REMAINING CACHE LOCATIONS
0020  4489 ;      17.  GO TO STEP 9 FOR ALL TEST PATTERNS
0020  4490 ;      18.  REPEAT STEPS 9 THROUGH 17 WITH LOOP J DECREMENTING FROM 1024 TO 1
0020  4491 ;
0020  4492 ;
0020  4493 ; TEST PATTERNS
0020  4494 ;       EACH OF THE FOLLOWING DATA PATTERNS IS USED IN THIS TEST:
0020  4495 ;               0.      00AAA000
0020  4496 ;               1.      00555000
0020  4497 ;               2.      00ABF000
0020  4498 ;               3.      007C0000
```

```
0020  4499 ;                4.      00AAA000
0020  4500 ;
0020  4501 ; LOGIC DESCRIPTION
0020  4502 ;       TEST HERE ARE THE 1x1024 RAM CHIPS USED IN THE CACHE
0020  4503 ;       DATA MEMORY.
0020  4504 ;
0020  4505 ; ASSUMPTIONS
0020  4506 ;       LOGIC OUTSIDE OF THESE RAM CHIPS IS ASSUMED TO BE FUNCTIONAL.
0020  4507 ;
0020  4508 ; ERROR DESCRIPTION
0020  4509 ;       EXPECTED CACHE HIT/MISS BIT
0020  4510 ;       GOT CACHE HIT/MISS BIT
0020  4511 ;       INDEX I
0020  4512 ;       INDEX J
0020  4513 ;       RTEMP0; ADDRESS BEING READ
0020  4514 ;       RTEMP1; DATA READ
0020  4515 ;
0020  4516 ;--
0020  4517 ;*******************************************************************************
0020  4518 ;///////////////////////////////////////////////////////////////////////////////
0020  4519
0020  4520         INITIALIZE                                      ;INIT THE CPU
0022  4521         EXPUTRAP                                        ;IGNORE U-TRAPS
0024  4522         LOADDCS         1$,,0,9                         ;LOAD TEST U-CODE INTO DCS
002B  4523         LOOP            I,1,1024                        ;CREATE A TEST LOOP OF 1024
0032  4524         SAVEINDEX       I,10$                           ;STORE LOOP COUNT IN 10$
0037  4525         LOADREG         WDREG,10$,,L                    ;PUT LOOP COUNT IN WD REG
003F  4526         FETCH           0                               ;START DCS PROGRAM
0044  4527         BRSTCLK         8                               ;END DCS PROGRAM
0048  4528         ENDLOOP         I                               ;END TEST LOOP
004B  4529         LOADDCS         20$,,0,<^X15>                   ;LOAD NEW U-CODE INTO DCS
0052  4530         LOOP            I,1,4                           ;LOOP TO SELECT TEST PATTERNS
0059  4531         LDFIELD         25$,I,L,21$,31,62,LONLIT        ;READ TEST PATTERN TO LONLIT
006A  4532         LOADDCS         21$,,4,1                        ;LOAD U-WORD INTO DCS
0071  4533         LDFIELD         30$,I,L,22$,31,62,LONLIT        ;WRITE TEST PATTERN TO LONLIT
0082  4534         LOADDCS         22$,,<^X0A>,1                   ;LOAD U-WORD INTO DCS
0089  4535         LOOP            J,1,1024                        ;LOOP TO INDEX CACHE
0090  4536         SAVEINDEX       J,10$                           ;STORE LOOP COUNT IN 10$
0095  4537         LOADREG         WDREG,10$,,L                    ;PUT LOOP COUNT IN WD REG
009D  4538         ERRLOOP                                         ;ERROR LOOP POINT
009F  4539         FETCH           0                               ;START DCS PROGRAM
00A4  4540         BRSTCLK         7                               ;END DCS PROGRAM
00A8  4541         FETCH           8                               ;START DCS PROGRAM
00AD  4542         BRSTCLK         <^X0C>                          ;END DCS PROGRAM
00B1  4543         CMPREGMSK       WHREG,40$,,40$,,L,              ;TEST FOR CACHE HIT
00BD  4544         IFERROR         2,<<CAK>>,                      ;FAILING ARRAY
00C3  4545         FETCH           <^X0D>                          ;START DCS PROGRAM
00C8  4546         BRSTCLK         <^X14>                          ;END DCS PROGRAM
00CC  4547         ENDLOOP         J                               ;END LOOP J
00CF  4548         ENDLOOP         I                               ;END LOOP I
00D2  4549         LOOP            I,1,4                           ;LOOP TO SELECT TEST PATTERNS
00D9  4550         LDFIELD         25$,I,L,21$,31,62,LONLIT        ;READ TEST PATTERN TO LONLIT
00EA  4551         LOADDCS         21$,,4,1                        ;LOAD U-WORD INTO DCS
00F1  4552         LDFIELD         30$,I,L,22$,31,62,LONLIT        ;WRITE TEST PATTERN TO LONLIT
0102  4553         LOADDCS         22$,,<^X0A>,1                   ;LOAD U-WORD INTO DCS
```

```
                        0109  4554          LOOP            J,1024,1                ;LOOP TO INDEX CACHE
                        0110  4555          SAVEINDEX       J,10$                   ;STORE LOOP COUNT IN 10$
                        0115  4556          LOADREG         WDREG,10$,,L            ;LOAD LOOP COUNT INTO WD REG
                        011D  4557          ERRLOOP                                 ;ERROR LOOP POINT
                        011F  4558          FETCH           0                       ;START DCS PROGRAM
                        0124  4559          BRSTCLK         7                       ;END DCS PROGRAM
                        0128  4560          FETCH           8                       ;START DCS PROGRAM
                        012D  4561          BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                        0131  4562          CMPREGMSK       WHREG,40$,,40$,,L,      ;TEST FOR CACHE HIT
                        013D  4563          IFERROR         2,<<CAK>>               ;FAILING ARRAY
                        0143  4564          FETCH           <^X0D>                  ;START DCS PROGRAM
                        0148  4565          BRSTCLK         <^X14>                  ;END DCS PROGRAM
                        014C  4566          ENDLOOP         J                       ;END LOOP J
                        014F  4567          ENDLOOP         I                       ;END LOOP I
                        0152  4568          SKIP                                    ;GO TO NEXT TEST
                        0159  4569
                        0159  4570 BEGIN_TEST_DATA
                        0159
                        0159          ;-------------------------------------------------------------------------
                        0159  4571
                        0159  4572 1$:                                             ;TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801   0159  4573 ;% 00    NOP
U 01, 5700,C840,0364,0304,0470,1802   0164  4577 ;% 01    R[TEMP1]_RDM                ;PUT LOOP COUNT IN RTEMP1
U 02, 7700,D800,EF64,0301,0470,1803   016F  4581 ;% 02    PL_[00000002]               ;SHIFT COUNT TO FORM INDEX
U 03, 7700,C800,8B71,0304,0470,1804   017A  4585 ;% 03    Q_R[TEMP1].RL.P             ;SHIFT LOOP COUNT
U 04, 7700,F800,7FAA,AFFF,8470,1805   0185  4589 ;% 04    LONLIT_[00AAA000]           ;TAG DATA
U 05, 7700,0840,03A4,03D4,4A70,1806   0190  4593 ;% 05    VA_Q.OR.R[LONLIT],SPW/RLONG ;MERGE INDEX AND TAG
U 06, 7701,8800,5BE4,0200,5590,1807   019B  4597 ;% 06    WRITE.LONG R[TEMP0]         ;WRITE CACHE
U 07, 7700,C800,5BE4,0300,5470,1808   01A6  4601 ;% 07    WDR_UNROT(R[TEMP0])         ;HOLD DATA FOR EXTRA CYCLE
U 08, 7300,4800,0364,0300,0470,1809   01B1  4605 ;% 08    NOP,STOP.CLOCK              ;STOP CPU CLOCK
                                      01BC  4609
                           00000000   01BC  4610 10$:     .LONG   ^X00000000          ;TEMPORARY STORAGE
                                      01C0  4611
                                      01C0  4612 20$:                               ;MORE TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801   01C0  4613 ;% 00    NOP
U 01, 5700,C840,0364,0308,0470,1802   01CB  4617 ;% 01    R[TEMP2]_RDM                ;LOOP COUNT TO RTEMP2
U 02, 7700,D800,EF64,0301,0470,1803   01D6  4621 ;% 02    PL_[00000002]               ;SHIFT COUNT TO FORM INDEX
U 03, 7700,C800,8B71,0308,0470,1804   01E1  4625 ;% 03    Q_R[TEMP2].RL.P             ;SHIFT LOOP COUNT TO INDEX
                                      01EC  4629 21$:
U 04, 7700,F800,7FFF,FFFF,8470,1805   01EC  4630 ;% 04    LONLIT_[0]                  ;READ TAG FIELD
U 05, 7700,0840,03A4,03D4,4A70,1806   01F7  4634 ;% 05    VA_Q.OR.R[LONLIT],SPW/RLONG ;MERGE LOOP COUNT & TAG
U 06, 7700,4800,0364,0300,0510,1807   0202  4638 ;% 06    READ.LONG                   ;READ CACHE
U 07, 7300,C800,0364,0300,0470,1808   020D  4642 ;% 07    NOP,STOP.CLOCK              ;STOP CPU CLOCK
U 08, 7700,4800,0364,0300,0470,1809   0218  4646 ;% 08    NOP
U 09, 7700,0852,5924,0304,0470,180A   0223  4650 ;% 09    RSRC/TEMP1,SPW/RLONG,WB_M[MDR] ;STORE DATA IN RTEMP1
U 0A, 7700,C800,5BE4,030C,6870,180B   022E  4654 ;% 0A    MEMSCAR_R[TEMP3]            ;LOAD CACHE ERROR REG ADDRESS
U 0B, 6700,4800,0364,0300,6470,180C   0239  4658 ;% 0B    RDM_WB,WCTRL/MEMSCR         ;CACHE ERROR REG TO RDM
U 0C, 7300,C800,0364,0300,0470,180D   0244  4662 ;% 0C    NOP,STOP.CLOCK              ;STOP CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E   024F  4666 ;% 0D    NOP
                                      025A  4670 22$:
U 0E, 7700,F800,7FFF,FFFF,8470,180F   025A  4671 ;% 0E    LONLIT_[0]                  ;WRITE TAG FILED
U 0F, 7700,8800,03A5,03D4,0470,1810   0265  4675 ;% 0F    Q_R[LONLIT].OR.Q            ;MERGE LOOP COUNT & TAG
U 10, 7700,4800,03A4,03D8,4A70,1811   0270  4679 ;% 10    VA_Q                        ;PUT ADDRESS IN VA REGISTER
U 11, 7700,085B,5924,0310,0470,1812   027B  4683 ;% 11    RSRC/TEMP4,SPW/RLONG,WB_M[VA] ;STORE ADDRESS IN RTEMP4
U 12, 7701,C800,5BE4,0210,5590,1813   0286  4687 ;% 12    WRITE.LONG R[TEMP4]         ;WRITE CACHE
```

```
U 13, 7700,0800,5BE4,0310,5470,1814  0291  4691 ;% 13   WDR_UNROT(R[TEMP4])                    ;HOLD DATA FOR EXTRA CYCLE
U 14, 7300,C800,0364,0300,0470,1815  029C  4695 ;% 14   NOP,STOP.CLOCK                         ;STOP CPU CLOCK
                                     02A7  4699
                           00AAA000  02A7  4700 25$:    .LONG   ^X00AAA000                    ;TEST PATTERN TABLE
                           00555000  02AB  4701 30$:    .LONG   ^X00555000
                           00ABF000  02AF  4702        .LONG   ^X00ABF000
                           007C0000  02B3  4703        .LONG   ^X007C0000
                           00AAA000  02B7  4704        .LONG   ^X00AAA000
                                     02BB  4705
                           01000000  02BB  4706 40$:    .LONG   ^X01000000                    ;COMPARE DATA FOR CACHE HIT
                                     02BF  4707
                                     02BF  4708 BEGIN_CPU_DATA
                                     0002  4709
                                     0002  4710 RTEMP_DATA
                                     0001  4711
                           00000000  0001  4712        .LONG   ^X00000000                    ;RTEMP0
                           00000000  0005  4713        .LONG   ^X00000000                    ;RTEMP1
                           00000000  0009  4714        .LONG   ^X00000000                    ;RTEMP2
                           04000000  000D  4715        .LONG   ^X04000000                    ;RTEMP3/CACHE ERROR REG ADD
                                     0011  4716
                                     0011  4717 END_CPU_DATA
                                     02BF  4718
                                     02BF  4719 END_TEST_DATA
                                     02BF
                                     02BF        ;-------------------------------------------------------------------------
                                     02BF  4720
                                     02BF  4721 ENDTEST
```

```
                        .SBTTL  "TEST   01D     CACHE VALID BIT MARCH TEST
001F  4723 ;*******************************************************************************
001F  4724 ;++
001F  4725 ;
001F  4726 ;       CACHE VALID BIT MARCH TEST
001F  4727 ;
001F  4728 ; FUNCTIONAL DESCRIPTION
001F  4729 ;       THIS IS A MARCH TEST OF THE CACHE MEMORY VALID BIT RAM.
001F  4730 ;
001F  4731 ; TEST ALGORITHM
001F  4732 ;       1.  LOAD U-CODE TO WRITE ALL CACHE LOCATIONS
001F  4733 ;       2.  CREATE A TEST LOOP OF 1024 (I)
001F  4734 ;       3.  SAVE THE LOOP COUNT IN 10$
001F  4735 ;       4.  LOAD THE LOOP COUNT INTO THE WD REGISTER
001F  4736 ;       5.  EXECUTE DCS PROGRAM
001F  4737 ;               a.  MOVE LOOP COUNT INTO RTEMP0
001F  4738 ;               b.  LOAD 2 INTO PLATCH
001F  4739 ;               c.  ROTATE RTEMP0 LEFT BY 2 AND SAVE IN VA AND RTEMP0
001F  4740 ;               d.  WRITE CACHE AS SPECIFIED BY VA WITH RTEMP0
001F  4741 ;       6.  GO TO STEP 3 FOR REMAINING CACHE LOCATIONS
001F  4742 ;       7.  LOAD U-CODE TO READ THEN INVALIDATE CACHE
001F  4743 ;       8.  CREATE A TEST LOOP OF 1024 (I)
001F  4744 ;       9.  SAVE LOOP COUNT IN 10$
001F  4745 ;      10.  LOAD THE LOOP COUNT INTO THE WD REGISTER
001F  4746 ;      11.  EXECUTE DCS PROGRAM
001F  4747 ;               a.  MOVE LOOP COUNT INTO RTEMP0
001F  4748 ;               b.  LOAD 2 INTO PLATCH
001F  4749 ;               c.  ROTATE RTEMP0 LEFT BY 2 AND SAVE IN VA AND RTEMP0
001F  4750 ;               d.  READ CACHE AS SPECIFIED BY VA REGISTER
001F  4751 ;               e.  SAVE DATA READ IN RTEMP1
001F  4752 ;               f.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM WH REGISTER
001F  4753 ;               g.  INVALIDATE LOCATION JUST READ
001F  4754 ;      12.  TEST WH REGISTER FOR A CACHE HIT
001F  4755 ;      13.  IF NO ERROR GO TO STEP 9 FOR REMAINING CACHE LOCATIONS
001F  4756 ;      14.  LOAD U-CODE TO READ THEN VALIDATE CACHE
001F  4757 ;      15.  CREATE A TEST LOOP OF 1024 (I)
001F  4758 ;      16.  SAVE LOOP COUNT IN 10$
001F  4759 ;      17.  LOAD THE LOOP COUNT INTO THE WD REGISTER
001F  4760 ;      18.  EXECUTE DCS PROGRAM
001F  4761 ;               a.  MOVE LOOP COUNT INTO RTEMP0
001F  4762 ;               b.  LOAD 2 INTO PLATCH
001F  4763 ;               c.  ROTATE RTEMP0 LEFT BY 2 AND SAVE IN VA AND RTEMP0
001F  4764 ;               d.  READ CACHE AS SPECIFIED BY VA REGISTER
001F  4765 ;               e.  SAVE DATA READ IN RTEMP1
001F  4766 ;               f.  MOVE CONTENTS OF CACHE ERROR REGISTER TO RDM WH REGISTER
001F  4767 ;               g.  VALIDATE LOCATION JUST READ WITH CONTENTS OF RTEMP0
001F  4768 ;      19.  TEST WH REGISTER FOR A CACHE MISS
001F  4769 ;      20.  IF NO ERROR GO TO STEP 16 FOR REMAINING CACHE LOCATIONS
001F  4770 ;      21.  REPEAT STEPS 7 THROUGH 20 WITH A DECREMENTING TEST LOOP
001F  4771 ;           (1024 TO 1)
001F  4772 ;
001F  4773 ; LOGIC DESCRIPTION
001F  4774 ;       THIS IS A TEST OF THE 1x1024 RAM CHIP WHICH IS USED TO STORE
001F  4775 ;       THE CACHE VALID BIT.
001F  4776 ;
```

```
                        ; ASSUMPTIONS
                        ;       ALL LOGIC USED IN THIS TEST WHICH IS EXTERNAL TO THIS RAM CHIP
                        ;       IS ASSUMED TO BE FUNCTIONAL.
                        ;
                        ; ERROR DESCRIPTION
                        ;       EXPECTED CACHE HIT/MISS BIT
                        ;       GOT CACHE HIT/MISS BIT
                        ;       INDEX VALUE I
                        ;       RTEMP0; ADDRESS BEING READ
                        ;       RTEMP1; DATA READ
                        ;
                        ;--
                        ;*****************************************************************************
                        ;/////////////////////////////////////////////////////////////////////////////

                                INITIALIZE                                      ;INIT THE CPU
                                EXPUTRAP                                        ;IGNORE U-TRAPS
                                LOADDCS         1$,,0,7                         ;LOAD TEST U-CODE INTO DCS
                                LOOP            I,1,1024                        ;LOOP TO VALIDATE CACHE
                                SAVEINDEX       I,10$                           ;SAVE LOOP COUNT IN 10$
                                LOADREG         WDREG,10$,,L                    ;LOOP COUNT TO WD REG
                                FETCH           0                               ;START DCS PROGRAM
                                BRSTCLK         6                               ;END DCS PROGRAM
                                ENDLOOP         I                               ;END CACHE VALIDATE LOOP
                                LOADDCS         2$,,0,<^X0E>                    ;LOAD NEW TEST U-CODE
                                LOOP            I,1,1024                        ;LOOP TO READ THEN INVALIDATE
                                SAVEINDEX       I,10$                           ;SAVE LOOP COUNT IN 10$
                                LOADREG         WDREG,10$,,L                    ;LOAD LOOP COUNT INTO WD REG
                                ERRLOOP                                         ;ERROR LOOP POINT
                                FETCH           0                               ;START DCS PROGRAM
                                BRSTCLK         5                               ;END DCS PROGRAM
                                FETCH           6                               ;START DCS PROGRAM
                                BRSTCLK         <^X0A>                          ;END DCS PROGRAM
                                CMPREGMSK       WHREG,20$,,20$,,L,              ;TEST FOR CACHE HIT
                                IFERROR         2,<<CAK>>                       ;FAILING ARRAY
                                FETCH           <^X0B>                          ;START DCS PROGRAM
                                BRSTCLK         <^X0D>                          ;END DCS PROGRAM
                                ENDLOOP         I                               ;END LOOP I
                                LOADDCS         3$,,0,<^X0F>                    ;LOAD NEW TEST U-CODE
                                LOOP            I,1,1024                        ;LOOP TO READ THEN VALIDATE
                                SAVEINDEX       I,10$                           ;SAVE LOOP COUNT IN 10$
                                LOADREG         WDREG,10$,,L                    ;LOAD LOOP COUNT INTO WD REG
                                ERRLOOP                                         ;ERROR LOOP POINT
                                FETCH           0                               ;START DCS PROGRAM
                                BRSTCLK         5                               ;END DCS PROGRAM
                                FETCH           6                               ;START DCS PROGRAM
                                BRSTCLK         <^X0A>                          ;END DCS PROGRAM
                                CMPREGMSK       WHREG,21$,,20$,,L,              ;TEST FOR CACHE MISS
                                IFERROR         2,<<CAK>>                       ;FAILING ARRAY
                                FETCH           <^X0B>                          ;START DCS PROGRAM
                                BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                                ENDLOOP         I                               ;END LOOP I
                                LOADDCS         2$,,0,<^X0E>                    ;LOAD NEW TEST U-CODE
                                LOOP            I,1024,1                        ;LOOP TO READ THEN INVALIDATE
                                SAVEINDEX       I,10$                           ;SAVE LOOP COUNT IN 10$
```

```
                00F7  4832          LOADREG         WDREG,10$,,L                ;LOAD LOOP COUNT INTO WD REG
                00FF  4833          ERRLOOP                                     ;ERROR LOOP POINT
                0101  4834          FETCH           0                           ;START DCS PROGRAM
                0106  4835          BRSTCLK         5                           ;END DCS PROGRAM
                010A  4836          FETCH           6                           ;START DCS PROGRAM
                010F  4837          BRSTCLK         <^X0A>                      ;END DCS PROGRAM
                0113  4838          CMPREGMSK       WHREG,20$,,20$,,L,          ;TEST FOR CACHE HIT
                011F  4839          IFERROR         2,<<CAK>>,                  ;FAILING ARRAY
                0125  4840          FETCH           <^X0B>                      ;START DCS PROGRAM
                012A  4841          BRSTCLK         <^X0D>                      ;END DCS PROGRAM
                012E  4842          ENDLOOP         I                           ;END LOOP I
                0131  4843          LOADDCS         3$,,0,<^X0F>                ;LOAD NEW TEST U-CODE
                0138  4844          LOOP            I,1024,1                    ;LOOP TO READ THEN VALIDATE
                013F  4845          SAVEINDEX       I,10$                       ;SAVE LOOP COUNT IN 10$
                0144  4846          LOADREG         WDREG,10$,,L                ;LOAD LOOP COUNT INTO WD REG
                014C  4847          ERRLOOP                                     ;ERROR LOOP POINT
                014E  4848          FETCH           0                           ;START DCS PROGRAM
                0153  4849          BRSTCLK         5                           ;END DCS PROGRAM
                0157  4850          FETCH           6                           ;START DCS PROGRAM
                015C  4851          BRSTCLK         <^X0A>                      ;END DCS PROGRAM
                0160  4852          CMPREGMSK       WHREG,21$,,20$,,L,          ;TEST FOR CACHE MISS
                016C  4853          IFERROR         2,<<CAK>>,                  ;FAILING ARRAY
                0172  4854          FETCH           <^X0B>                      ;START DCS PROGRAM
                0177  4855          BRSTCLK         <^X0E>                      ;END DCS PROGRAM
                017B  4856          ENDLOOP         I                           ;END LOOP I
                017E  4857          SKIP                                        ;GO TO NEXT TEST
                0185  4858
                0185  4859 BEGIN_TEST_DATA
                0185
                0185      ;-------------------------------------------------------------------------
                0185  4860
                0185  4861 1$:                                                  ;TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801   0185  4862 ;* 00    NOP
U 01, 5700,8840,0364,0300,0470,1802   0190  4866 ;* 01    R[TEMP0]_RDM                ;LOOP COUNT TO RTEMP0
U 02, 7700,D800,EF64,0301,0470,1803   019B  4870 ;* 02    PL_[00000002]               ;SHIFT COUNT
U 03, 7700,8840,8B70,0300,4A70,1804   01A6  4874 ;* 03    VA_R[TEMP0].RL.P,SPW/RLONG  ;SHIFT LOOP COUNT TO ADDRESS
U 04, 7701,0800,5BE4,0200,5590,1805   01B1  4878 ;* 04    WRITE.LONG R[TEMP0]         ;WRITE CACHE
U 05, 7700,4800,5BE4,0300,5470,1806   01BC  4882 ;* 05    WDR_UNROT(R[TEMP0])         ;HOLD DATA FOR EXTRA CYCLE
U 06, 7300,C800,0364,0300,0470,1807   01C7  4886 ;* 06    NOP,STOP.CLOCK              ;STOP CPU CLOCK
                01D2  4890
                01D2  4891 2$:                                                  ;MORE TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801   01D2  4892 ;* 00    NOP
U 01, 5700,8840,0364,0300,0470,1802   01DD  4896 ;* 01    R[TEMP0]_RDM                ;LOOP COUNT TO RTEMP0
U 02, 7700,D800,EF64,0301,0470,1803   01E8  4900 ;* 02    PL_[00000002]               ;SHIFT COUNT
U 03, 7700,8840,8B70,0300,4A70,1804   01F3  4904 ;* 03    VA_R[TEMP0].RL.P,SPW/RLONG  ;SHIFT LOOP COUNT TO ADDRESS
U 04, 7700,C800,0364,0300,0510,1805   01FE  4908 ;* 04    READ.LONG                   ;READ CACHE
U 05, 7300,4800,0364,0300,0470,1806   0209  4912 ;* 05    NOP,STOP.CLOCK              ;STOP CPU CLOCK
U 06, 7700,C800,0364,0300,0470,1807   0214  4916 ;* 06    NOP
U 07, 7700,8852,5924,0304,0470,1808   021F  4920 ;* 07    RSRC/TEMP1,SPW/RLONG,WB_M[MDR]  ;SAVE DATA IN RTEMP1
U 08, 7700,0800,5BE4,0308,6870,1809   022A  4924 ;* 08    MEMSCAR_R[TEMP2]            ;LOAD CACHE ERROR REG ADDRESS
U 09, 6700,4800,0364,0300,6470,180A   0235  4928 ;* 09    RDM_WB,WCTRL/MEMSCR         ;READ CACHE ERROR REG TO RDM
U 0A, 7300,C800,0364,0300,0470,180B   0240  4932 ;* 0A    NOP,STOP.CLOCK              ;STOP CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C   024B  4936 ;* 0B    NOP
U 0C, 7700,081B,0024,03D8,5A70,180D   0256  4940 ;* 0C    CLRCH.VA_VA                 ;INVALIDATE CACHE LOCATION
U 0D, 7300,C800,0364,0300,0470,180E   0261  4944 ;* 0D    NOP,STOP.CLOCK              ;STOP CPU CLOCK
```

```
                                  026C  4948
                                  026C  4949 3$:                                                  ;MORE TEST U-CODE
U 00, 7700,C800,0364,0300,0470,1801  026C  4950 ;% 00   NOP
U 01, 5700,8840,0364,0300,0470,1802  0277  4954 ;% 01   R[TEMP0]_RDM                              ;LOOP COUNT TO RTEMP0
U 02, 7700,D800,EF64,0301,0470,1803  0282  4958 ;% 02   PL_[00000002]                             ;SHIFT COUNT
U 03, 7700,8840,8B70,0300,4A70,1804  028D  4962 ;% 03   VA_R[TEMP0].RL.P,SPW/RLONG                ;SHIFT LOOP COUNT TO ADDRESS
U 04, 7700,C800,0364,0300,0510,1805  0298  4966 ;% 04   READ.LONG                                 ;READ CACHE
U 05, 7300,4800,0364,0300,0470,1806  02A3  4970 ;% 05   NOP,STOP.CLOCK                            ;STOP CPU CLOCK
U 06, 7700,C800,0364,0300,0470,1807  02AE  4974 ;% 06   NOP
U 07, 7700,8852,5924,0304,0470,1808  02B9  4978 ;% 07   RSRC/TEMP1,SPW/RLONG,WB_M[MDR]            ;SAVE DATA IN RTEMP1
U 08, 7700,0800,5BE4,0308,6870,1809  02C4  4982 ;% 08   MEMSCAR_R[TEMP2]                          ;LOAD CACHE ERROR REG ADDRESS
U 09, 6700,4800,0364,0300,6470,180A  02CF  4986 ;% 09   RDM_WB,WCTRL/MEMSCR                       ;READ CACHE ERROR REG TO RDM
U 0A, 7300,C800,0364,0300,0470,180B  02DA  4990 ;% 0A   NOP,STOP.CLOCK                            ;STOP CPU CLOCK
U 0B, 7700,4800,0364,0300,0470,180C  02E5  4994 ;% 0B   NOP
U 0C, 7701,8800,5BE4,0200,5590,180D  02F0  4998 ;% 0C   WRITE.LONG R[TEMP0]                       ;VALIDATE CACHE LOCATION
U 0D, 7700,C800,5BE4,0300,5470,180E  02FB  5002 ;% 0D   WDR_UNROT(R[TEMP0])                       ;HOLD DATA FOR EXTRA CYCLE
U 0E, 7300,4800,0364,0300,0470,180F  0306  5006 ;% 0E   NOP,STOP.CLOCK                            ;STOP CPU CLOCK
                                  0311  5010
                        00000000  0311  5011 10$:    .LONG   ^X00000000                        ;TEMPORARY STORAGE
                                  0315  5012
                        01000000  0315  5013 20$:    .LONG   ^X01000000                        ;COMPARE DATA FOR CACHE HIT
                        00000000  0319  5014 21$:    .LONG   ^X00000000                        ;COMPARE DATA FOR CACHE MISS
                                  031D  5015
                                  031D  5016 BEGIN_CPU_DATA
                                  0002  5017
                                  0002  5018 RTEMP_DATA
                                  0001  5019
                        00000000  0001  5020         .LONG   ^X00000000                        ;RTEMP0
                        00000000  0005  5021         .LONG   ^X00000000                        ;RTEMP1
                        04000000  0009  5022         .LONG   ^X04000000                        ;RTEMP2/CACHE ERROR REG ADDRESS
                                  000D  5023
                                  000D  5024 END_CPU_DATA
                                  031D  5025
                                  031D  5026 END_TEST_DATA
                                  031D
                                  031D       ;-----------------------------------------------------------------------
                                  031D  5027
                                  031D  5028 ENDTEST
```

```
                        .SBTTL  "TEST   01E     TEST FORCE PARITY ERROR MICRO FUNCTION
                ;*******************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST CHECKS THE "MISC/FORCE.TB" AND THE "MISC/FORCE.CACHE"
                ;       MICRO ORDER TO INSURE THESE FUNCTIONS CAN BE SET BY THE MICROWORD.
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  CREATE A TEST LOOP OF 3 TO SELECT U-ORDERS
                ;       2.  LOAD THE U-ORDER FOR THIS LOOP
                ;       3.  EXECUTE DCS PROGRAM
                ;               a.  STROBE THE VBUS WHILE EXECUTING THE SELECTED FUNCTION
                ;       4.  TEST THE VBUS BITS <01:00> FOR THE CORRECT STATE
                ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING U-ORDERS
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       ITERATION       U-ORDER         VBUS DATA
                ;           1           NOP                1,1
                ;           2           FORCE.CACHE        0,1
                ;           3           FORCE.TB           1,0
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       THIS TEST SIMPLY VERIFIES THAT THE FORCE PARITY ERROR FUNCTION
                ;       CAN BE DECODED BY THE UBI MODULE AND THE APPROPRIATE BITS SET ON
                ;       THE VBUS.  IF YOU HAVE A FAILURE IT IS MOST LIKELY THE DECODER
                ;       ON THE UBI MODULE OR THE VBUS SHIFT REGISTER ON UBI.
                ;
                ;
                ; ASSUMPTIONS:
                ;
                ;       THIS TEST ASSUMES THE VBUS FUNCTIONS ON THE RDM MODULE ARE OPERATIONAL
                ;
                ;
                ; ERROR DESCRIPTION:
                ;
                ;       EXPECTED VBUS DATA
                ;       RECEIVED VBUS DATA
                ;       LOOP COUNT I
                ;
                ;               BITS <07:00> = VBUS BIT POSITION
                ;               BIT <08> = VBUS STATE
                ;
                ;--
                ;*******************************************************************************
                ;///////////////////////////////////////////////////////////////////////////////
                
                        INITIALIZE                                      ;INIT THE CPU
```

```
                    002D  5084          LOOP            I,1,3                   ;TEST LOOP OF 3
                    0034  5085          LOADDCS         1$,I,1,1                ;MODIFY DCS ADDRESS 01
                    003B  5086          ERRLOOP                                 ;ERROR LOOP POINT
                    003D  5087          FETCH           0                       ;START DCS PROGRAM
                    0042  5088          BRSTCLK         2                       ;END DCS PROGRAM
                    0046  5089          CMPVBUS         2$,I                    ;TEST VBUS FOR CORRECT PATTERN
                    004B  5090          IFERROR         ,,<<UBI>>               ;BITS INCORRECT AT UBI MODULE
                    0051  5091          ENDLOOP         I                       ;END TEST LOOP
                    0054  5092          SKIP                                    ;GO TO NEXT TEST
                    005B  5093
                    005B  5094 BEGIN_TEST_DATA
                    005B
                    005B          ;-------------------------------------------------------------------------
                    005B  5095
                    005B  5096 1$:                                              ;DCS U-CODE FOR:
U 01, 7600,C800,0364,0300,0470,1802 005B  5097 ;% 01    NOP,STROBE.V.BUS                ;LOOP 1
U 01, 7600,CF80,0364,0300,0470,1802 0066  5101 ;% 01    MISC/FORCE.CACHE,STROBE.V.BUS   ;LOOP 2
U 01, 7600,4F00,0364,0300,0470,1802 0071  5105 ;% 01    MISC/FORCE.TB,STROBE.V.BUS      ;LOOP 3
                    007C  5109
                 02 007C  5110 2$:      .BYTE           ^X2                     ;VBUS RESULTS
                    007D  5111          VBUSDATA        <^X00>,1
                    007E  5112          VBUSDATA        <^X01>,1
                    007F  5113
                 02 007F  5114          .BYTE           ^X2
                    0080  5115          VBUSDATA        <^X00>,1
                    0081  5116          VBUSDATA        <^X01>,0
                    0082  5117
                 02 0082  5118          .BYTE           ^X2
                    0083  5119          VBUSDATA        <^X00>,0
                    0084  5120          VBUSDATA        <^X01>,1
                    0085  5121
                    0085  5122 BEGIN_CPU_DATA
                    0002  5123
                    0002  5124 DCS_DATA
                    0001  5125
U 00, 7700,C800,0364,0300,0470,1801 0001  5126 ;% 00    NOP
U 01, 7600,C800,0364,0300,0470,1802 000C  5130 ;% 01    NOP,STROBE.V.BUS                ;STROBE THE VBUS
U 02, 7300,4800,0364,0300,0470,1803 0017  5134 ;% 02    NOP,STOP.CLOCK
                    0022  5138
                    0022  5139 END_CPU_DATA
                    0085  5140
                    0085  5141 END_TEST_DATA
                    0085
                    0085          ;-------------------------------------------------------------------------
                    0085  5142
                    0085  5143 ENDTEST
```

```
001D        .SBTTL  "TEST   01F     CACHE DATA PARITY TEST 1
001D  5145 ;*******************************************************************************
001D  5146 ;++
001D  5147 ;
001D  5148 ; FUNCTIONAL DESCRIPTION:
001D  5149 ;
001D  5150 ;       THIS TEST CHECKS THE CACHE DATA PARITY ERROR LOGIC
001D  5151 ;
001D  5152 ;
001D  5153 ; TEST ALGORITHM:
001D  5154 ;
001D  5155 ;       1.  EXECUTE DCS PROGRAM
001D  5156 ;               a.  LOAD VA WITH ZERO
001D  5157 ;               b.  READ CACHE LOCATION ZERO WHILE FORCING A PARITY ERROR
001D  5158 ;       2.  TEST CS ADDRESS FOR MACHINE CHECK VECTOR
001D  5159 ;       3.  IF NO ERROR EXECUTE DCS PROGRAM
001D  5160 ;               a.  READ CACHE ERROR REGISTER TO RDM
001D  5161 ;       4.  TEST CACHE ERROR REGISTER FOR DATA ERROR AND HIT
001D  5162 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
001D  5163 ;               a.  PERFORM A SECOND READ WHILE FORCING A PARITY ERROR
001D  5164 ;               b.  READ CACHE ERROR REGISTER TO RDM
001D  5165 ;               c.  CLEAR CACHE ERROR REGISTER
001D  5166 ;       6.  TEST CACHE ERROR REGISTER FOR DATA ERROR, LOST ERROR AND HIT
001D  5167 ;       7.  IF NO ERROR EXECUTE DCS PROGRAM
001D  5168 ;               a.  READ BUS ERROR REGISTER TO RDM
001D  5169 ;       8.  TEST BUS ERROR REGISTER FOR UNCORRECTABLE DATA AND LOST ERROR
001D  5170 ;       9.  IF NO ERROR EXECUTE DCS PROGRAM
001D  5171 ;               a.  READ THE MACHINE CHECK REGISTER TO THE RDM
001D  5172 ;               b.  CLEAT THE MACHINE CHECK REGISTER
001D  5173 ;      10.  TEST MACHINE CHECK REGISTER FOR BUS ERROR
001D  5174 ;
001D  5175 ;
001D  5176 ; TEST PATTERNS:
001D  5177 ;
001D  5178 ;       NONE
001D  5179 ;
001D  5180 ;
001D  5181 ; LOGIC DESCRIPTION:
001D  5182 ;
001D  5183 ;       THIS TEST CHECKS THAT THE CACHE DATA PARITY CHECKERS CAN GENERATE
001D  5184 ;       A PARITY ERROR AND THAT THE CPU MACHINE CHECKS AS A RESULT.  THE
001D  5185 ;       CAK GATE ARRAY IS RESPONSIBLE FOR REPORTING THE ERROR TO THE CML
001D  5186 ;       GATE ARRAY WHICH INTURN NOTIFIES THE UTR ARRAY.  STATUS AND CONTROL
001D  5187 ;       REGISTERS HOUSED IN CAK AND UTR ARE ALSO VERIFIED.
001D  5188 ;
001D  5189 ;
001D  5190 ; ASSUMPTIONS:
001D  5191 ;
001D  5192 ;       THIS TEST ASSUMES THE CACHE DATA AND TAG MEMORIES ARE OPERATIONAL.
001D  5193 ;
001D  5194 ;
001D  5195 ; ERROR DESCRIPTION:
001D  5196 ;
001D  5197 ;       FIRST IFERROR:
001D  5198 ;               EXPECTED CS ADDRESS
```

```
                        001D  5199 ;                RECEIVED CS ADDRESS
                        001D  5200 ;
                        001D  5201 ;        SECOND IFERROR:
                        001D  5202 ;                EXPECTED CACHE ERROR REGISTER CONTENTS
                        001D  5203 ;                RECEIVED CACHE ERROR REGISTER CONTENTS
                        001D  5204 ;
                        001D  5205 ;        THIRD IFERROR:
                        001D  5206 ;                EXPECTED CACHE ERROR REGISTER CONTENTS
                        001D  5207 ;                RECEIVED CACHE ERROR REGISTER CONTENTS
                        001D  5208 ;
                        001D  5209 ;        FORTH IFERROR:
                        001D  5210 ;                EXPECTED BUS ERROR REGISTER CONTENTS
                        001D  5211 ;                RECEIVED BUS ERROR REGISTER CONTENTS
                        001D  5212 ;
                        001D  5213 ;        FIFTH IFERROR:
                        001D  5214 ;                EXPECTED MACHINE CHECK REGISTER CONTENTS
                        001D  5215 ;                RECEIVED MACHINE CHECK REGISTER CONTENTS
                        001D  5216 ;
                        001D  5217 ;--
                        001D  5218 ;*****************************************************************************
                        001D  5219
                        001D  5220
                        001D  5221          INITIALIZE                              ;INIT THE CPU
                        001F  5222          EXPUTRAP                                ;EXPECT A MICRO TRAP
                        0021  5223          ERRLOOP                                 ;ERROR LOOP POINT
                        0023  5224          FETCH           0                       ;START DCS PROGRAM
                        0028  5225          BRSTCLK         4                       ;END DCS PROGRAM
                        002C  5226          CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST CS ADD FOR MACHINE CHECK
                        0038  5227          IFERROR         ,<<CAK><CML><UTR>>,     ;DID NOT MICRO TRAP
                        0040  5228          FETCH           5                       ;START DCS PROGRAM
                        0045  5229          BRSTCLK         8                       ;END DCS PROGRAM
                        0049  5230          CMPREGMSK       WHREG,3$,,4$,,L,        ;TEST CACHE ERROR REGISTER
                        0055  5231          IFERROR         ,<CAK>,                 ;REGISTER NOT SET CORRECTLY
                        005B  5232          FETCH           9                       ;START DCS PROGRAM
                        0060  5233          BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                        0064  5234          FETCH           <^X0D>                  ;START DCS PROGRAM
                        0069  5235          BRSTCLK         <^X10>                  ;END DCS PROGRAM
                        006D  5236          CMPREGMSK       WHREG,7$,,4$,,L,        ;TEST CACHE ERROR REGISTER
                        0079  5237          IFERROR         ,<CAK>,                 ;REGISTER NOT SET CORRECTLY
                        007F  5238          FETCH           <^X11>                  ;START DCS PROGRAM
                        0084  5239          BRSTCLK         <^X14>                  ;END DCS PROGRAM
                        0088  5240          CMPREGMSK       WHREG,5$,,4$,,L,        ;TEST BUS ERROR REGISTER
                        0094  5241          IFERROR         ,<UTR>,                 ;REGISTER NOT SET CORRECTLY
                        009A  5242          FETCH           <^X15>                  ;START DCS PROGRAM
                        009F  5243          BRSTCLK         <^X19>                  ;END DCS PROGRAM
                        00A3  5244          CMPREGMSK       WHREG,6$,,4$,,L,        ;TEST MACHINE CHECK ERROR REG
                        00AF  5245          IFERROR         ,<UTR>,                 ;REGISTER NOT SET CORRECTLY
                        00B5  5246          SKIP                                    ;GO TO NEXT TEST
                        00BC  5247
                        00BC  5248
                        00BC  5249 BEGIN_TEST_DATA
                        00BC
                        00BC        ;-------------------------------------------------------------------------
                        00BC  5250
                0028    00BC  5251 1$:      .WORD   ^X0028                          ;MACHINE CHECK VECTOR
```

```
                        3FFF  00BE  5252 2$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                    05000000  00C0  5253 3$:     .LONG   ^X05000000                      ;CONTENTS OF CACHE ERROR REG
                    0F000000  00C4  5254 4$:     .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                    06000000  00C8  5255 5$:     .LONG   ^X06000000                      ;CONTENTS OF BUS ERROR REGISTER
                    08000000  00CC  5256 6$:     .LONG   ^X08000000                      ;CONTENTS OF MACHINE CHECK REG
                    07000000  00D0  5257 7$:     .LONG   ^X07000000                      ;CONTENTS OF CACHE ERROR REG
                              00D4  5258
                              00D4  5259
                              00D4  5260 BEGIN_CPU_DATA
                              0002  5261
                              0002  5262 DCS_DATA
                              0001  5263
U 00. 7700,C800,0364,0300,0470,1801   0001  5264 ;% 00   NOP
U 01. 7700,C800,5BE4,03D8,4A70,1802   000C  5268 ;% 01   VA_R[ZERO]                       ;LOAD VA WITH ZERO
U 02. 7700,CF80,0364,0300,0400,1803   0017  5272 ;% 02   READ.PHY,MISC/FORCE.CACHE        ;READ CACHE WITH PARITY ERROR
U 03. 7700,C800,0364,0300,0470,1804   0022  5276 ;% 03   NOP                             ;...
U 04. 7300,4800,0364,0300,0470,1805   002D  5280 ;% 04   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 05. 7700,4800,0364,0300,0470,1806   0038  5284 ;% 05   NOP
U 06. 7700,C800,5BE4,0300,6870,1807   0043  5288 ;% 06   MEMSCAR_R[TEMP0]                ;READ CACHE ERROR REG TO RDM
U 07. 6700,C800,0364,0300,6470,1808   004E  5292 ;% 07   RDM_WB,WCTRL/MEMSCR             ;...
U 08. 7300,4800,0364,0300,0470,1809   0059  5296 ;% 08   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 09. 7700,4800,0364,0300,0470,180A   0064  5300 ;% 09   NOP
U 0A. 7700,4F80,0364,0300,0400,180B   006F  5304 ;% 0A   READ.PHY,MISC/FORCE.CACHE        ;FORCE A SECOND PARITY ERROR
U 0B. 7700,4800,0364,0300,0470,180C   007A  5308 ;% 0B   NOP                             ;...
U 0C. 7300,C800,0364,0300,0470,180D   0085  5312 ;% 0C   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 0D. 7700,C800,0364,0300,0470,180E   0090  5316 ;% 0D   NOP
U 0E. 6700,4800,0364,0300,6470,180F   009B  5320 ;% 0E   RDM_WB,WCTRL/MEMSCR             ;READ CACHE ERROR REG TO RDM
U 0F. 7700,4800,5BE4,03D8,6070,1810   00A6  5324 ;% 0F   MEMSCR_R[ZERO]                  ;CLEAR CACHE ERROR REGISTER
U 10. 7300,4800,0364,0300,0470,1811   00B1  5328 ;% 10   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 11. 7700,4800,0364,0300,0470,1812   00BC  5332 ;% 11   NOP
U 12. 7700,8800,5BE4,0304,6870,1813   00C7  5336 ;% 12   MEMSCAR_R[TEMP1]                ;READ BUS ERROR REG TO RDM
U 13. 6700,4800,0364,0300,6470,1814   00D2  5340 ;% 13   RDM_WB,WCTRL/MEMSCR             ;...
U 14. 7300,C800,0364,0300,0470,1815   00DD  5344 ;% 14   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 15. 7700,C800,0364,0300,0470,1816   00E8  5348 ;% 15   NOP
U 16. 7700,0800,5BE4,0308,6870,1817   00F3  5352 ;% 16   MEMSCAR_R[TEMP2]                ;READ THE MACHINE CHECK REG
U 17. 6700,4800,0364,0300,6470,1818   00FE  5356 ;% 17   RDM_WB,WCTRL/MEMSCR             ;...
U 18. 7700,4800,5BE4,03D8,6070,1819   0109  5360 ;% 18   MEMSCR_R[ZERO]                  ;CLEAR MACHINE CHECK REGISTER
U 19. 7300,C800,0364,0300,0470,181A   0114  5364 ;% 19   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
                              011F  5368
                              011F  5369 RTEMP_DATA
                              0001  5370
                        04000000  0001  5371          .LONG   ^X04000000                ;CACHE ERROR REG ADDRESS
                        09000000  0005  5372          .LONG   ^X09000000                ;BUS ERROR REGISTER ADDRESS
                        08000000  0009  5373          .LONG   ^X08000000                ;MACHINE CHECK REGISTER ADDRESS
                              000D  5374
                              000D  5375 CACHE_DATA
                              0001  5376
                        00000000  0001  5377          .LONG   ^X00000000                ;VALIDATE CACHE ADDRESS ZERO
                              0005  5378
                              0005  5379 END_CPU_DATA
                              00D4  5380
                              00D4  5381
                              00D4  5382 END_TEST_DATA
                              00D4
                              00D4      ;-------------------------------------------------------------------------
```

```
                00D4  5383 ENDTEST
```

```
                        001D        .SBTTL  "TEST   020     CACHE DATA PARITY TEST 2
                        001D  5385 ;*******************************************************************************
                        001D  5386 ;++
                        001D  5387 ;
                        001D  5388 ; FUNCTIONAL DESCRIPTION:
                        001D  5389 ;
                        001D  5390 ;       THIS TEST VERIFIES THAT THE INDIVIDUAL PARITY CHECKERS CAN GENERATE
                        001D  5391 ;       A PARITY ERROR.
                        001D  5392 ;
                        001D  5393 ;
                        001D  5394 ; TEST ALGORITHM:
                        001D  5395 ;
                        001D  5396 ;       1.  CREATE A TEST LOOP OF 4
                        001D  5397 ;       2.  SELECT A TEST PATTERN AND LOAD INTO THE RDM WDREG
                        001D  5398 ;       3.  EXECUTE DCS PROGRAM
                        001D  5399 ;               a.  LOAD VA REGISTER WITH ZERO
                        001D  5400 ;               b.  WRITE THE TEST PATTERN TO CACHE LOCATION 0
                        001D  5401 ;               c.  READ LOCATION ZERO WHILE FORCING A PARITY ERROR
                        001D  5402 ;       4.  TEST CS ADDRESS FOR MACHINE CHECK VECTOR
                        001D  5403 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                        001D  5404 ;
                        001D  5405 ;
                        001D  5406 ; TEST PATTERNS:
                        001D  5407 ;
                        001D  5408 ;       ITERATION       TEST PATTERN
                        001D  5409 ;           1             00010101
                        001D  5410 ;           2             01000101
                        001D  5411 ;           3             01010001
                        001D  5412 ;           4             01010100
                        001D  5413 ;
                        001D  5414 ;
                        001D  5415 ; LOGIC DESCRIPTION:
                        001D  5416 ;
                        001D  5417 ;       THIS TEST VERIFIES THE ABILITY OF THE INDIVIDUAL CACHE DATA PARITY
                        001D  5418 ;       ERROR CHECKERS TO GENERATE A PARITY ERROR.  THE CHECKERS ARE DIVIDED
                        001D  5419 ;       UP ON BYTE BOUNDARIES, SO THE TEST PATTERNS SHOULD LEAD YOU TO THE
                        001D  5420 ;       CHECKER AT FAULT.  NO NEW GATE ARRAY FUNCTIONS ARE INVOLVED IN THIS
                        001D  5421 ;       TEST.
                        001D  5422 ;
                        001D  5423 ;
                        001D  5424 ; ASSUMPTIONS:
                        001D  5425 ;
                        001D  5426 ;       THIS TEST ASSUMES THE CAK, CML AND UTR GATE ARRAYS CAN REPORT A
                        001D  5427 ;       CACHE PARITY ERROR.
                        001D  5428 ;
                        001D  5429 ;
                        001D  5430 ; ERROR DESCRIPTION:
                        001D  5431 ;
                        001D  5432 ;       EXPECTED CS ADDRESS
                        001D  5433 ;       RECEIVED CS ADDRESS
                        001D  5434 ;
                        001D  5435 ;--
                        001D  5436 ;*******************************************************************************
                        001D  5437
                        001D  5438
```

```
                    001D  5439          INITIALIZE                                ;INIT THE CPU
                    001F  5440          EXPUTRAP                                  ;EXPECT A MACHINE CHECK
                    0021  5441          LOOP            I,1,4                     ;CREATE A TEST LOOP OF 4
                    0028  5442          LOADREG         WDREG,1$,I,L              ;LOAD TEST PATTERN INTO WDREG
                    0030  5443          ERRLOOP                                   ;ERROR LOOP POINT
                    0032  5444          FETCH           0                         ;START DCS PROGRAM
                    0037  5445          BRSTCLK         6                         ;END DCS PROGRAM
                    003B  5446          CMPREGMSK       CSTRP,2$,,3$,,W,          ;TEST FOR MACHINE CHECK
                    0047  5447          IFERROR                                   ;NO PARITY ERROR OCCURED
                    004C  5448          ENDLOOP         I                         ;END TEST LOOP
                    004F  5449          SKIP                                      ;GO TO NEXT TEST
                    0056  5450
                    0056  5451
                    0056  5452 BEGIN_TEST_DATA
                    0056
                    0056          ;-------------------------------------------------------------------------
                    0056  5453
 00010101           0056  5454 1$:       .LONG    ^X00010101                      ;TEST PATTERN TABLE
 01000101           005A  5455           .LONG    ^X01000101
 01010001           005E  5456           .LONG    ^X01010001
 01010100           0062  5457           .LONG    ^X01010100
     0028           0066  5458 2$:       .WORD    ^X0028                          ;MACHINE CHECK VECTOR
     3FFF           0068  5459 3$:       .WORD    ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                    006A  5460
                    006A  5461
                    006A  5462 BEGIN_CPU_DATA
                    0002  5463
                    0002  5464 DCS_DATA
                    0001  5465
U 00, 7700,C800,0364,0300,0470,1801   0001  5466 ;% 00    NOP
U 01, 7700,C800,5BE4,03D8,4A70,1802   000C  5470 ;% 01    VA_R[ZERO]                ;ZERO THE VA REGISTER
U 02, 5700,4800,0364,0300,5480,1803   0017  5474 ;% 02    WRITE.PHY,WCTRL/WDR_WB.UR,WB_RDM  ;WRITE TEST PATTERN TO CACHE
U 03, 5700,C800,0364,0300,5470,1804   0022  5478 ;% 03    WCTRL/WDR_WB.UR,WB_RDM    ;...
U 04, 7700,CF80,0364,0300,0400,1805   002D  5482 ;% 04    READ.PHY,MISC/FORCE.CACHE ;READ WHILE FORCING PARITY ERR
U 05, 7700,4800,0364,0300,0470,1806   0038  5486 ;% 05    NOP                       ;...
U 06, 7300,C800,0364,0300,0470,1807   0043  5490 ;% 06    NOP,STOP.CLOCK            ;STOP THE CPU CLOCK
                    004E  5494
                    004E  5495 END_CPU_DATA
                    006A  5496
                    006A  5497
                    006A  5498 END_TEST_DATA
                    006A
                    006A          ;-------------------------------------------------------------------------
                    006A  5499 ENDTEST
```

```
                        001D        .SBTTL  "TEST   021     CACHE DATA PARITY TEST 3
                        001D  5501 ;*******************************************************************************
                        001D  5502 ;++
                        001D  5503 ;
                        001D  5504 ; FUNCTIONAL DESCRIPTION:
                        001D  5505 ;
                        001D  5506 ;       THIS TEST IS DESIGNED TO THROUGHLY EXERCISE THE ENTERNAL GATES OF
                        001D  5507 ;       THE CACHE DATA PARITY GENERATOR/CHECKERS.
                        001D  5508 ;
                        001D  5509 ;
                        001D  5510 ; TEST ALGORITHM:
                        001D  5511 ;
                        001D  5512 ;       1.  CREATE A TEST LOOP OF 20
                        001D  5513 ;       2.  SAVE THE LOOP COUNT IN 1$
                        001D  5514 ;       3.  LOAD THE LOOP COUNT INTO THE RDM WDREG
                        001D  5515 ;       4.  EXECUTE DCS PROGRAM
                        001D  5516 ;               a.  PUT THE LOOP COUNT INTO RTEMP0
                        001D  5517 ;               b.  LOAD 2 INTO THE PLATCH
                        001D  5518 ;               c.  LOAD THE VA WITH THE LOOP COUNT SHIFTED LEFT BY 2
                        001D  5519 ;               d.  READ THE CACHE ADDRESS POINTED TO BY VA
                        001D  5520 ;               e.  MOVE THE READ DATA TO THE RDM WHREG
                        001D  5521 ;       5.  TEST THE CS ADDRESS FOR SUCCESSFUL COMPLETION
                        001D  5522 ;       6.  IF NO ERROR TEST THE READ DATA
                        001D  5523 ;       7.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST LOOPS
                        001D  5524 ;
                        001D  5525 ;
                        001D  5526 ; TEST PATTERNS:
                        001D  5527 ;
                        001D  5528 ;       ITERATION       TEST PATTERNS
                        001D  5529 ;           1             6F6F6F6F
                        001D  5530 ;           2             91919191
                        001D  5531 ;           3             36363636
                        001D  5532 ;           4             44444444
                        001D  5533 ;           5             A8A8A8A8
                        001D  5534 ;           6             1D1D1D1D
                        001D  5535 ;           7             E0E0E0E0
                        001D  5536 ;           8             D9D9D9D9
                        001D  5537 ;           9             6D6D6D6D
                        001D  5538 ;          10             90909090
                        001D  5539 ;          11             26262626
                        001D  5540 ;          12             57575757
                        001D  5541 ;          13             E1E1E1E1
                        001D  5542 ;          14             D8D8D8D8
                        001D  5543 ;          15             92929292
                        001D  5544 ;          16             54545454
                        001D  5545 ;          17             A9A9A9A9
                        001D  5546 ;          18             1C1C1C1C
                        001D  5547 ;          19             E3E3E3E3
                        001D  5548 ;          20             07070707
                        001D  5549 ;
                        001D  5550 ;
                        001D  5551 ; LOGIC DESCRIPTION:
                        001D  5552 ;
                        001D  5553 ;       THIS TEST ATTEMPTS TO EXERCISE ALL INTERNAL NODES OF THE 74S280
                        001D  5554 ;       PARITY GENERATOR/CHECKERS.  ANY ERRORS SHOULD BE CONFINED TO ONE OF
```

```
                     001D  5555 ;       THESE DEVICES.
                     001D  5556 ;
                     001D  5557 ;
                     001D  5558 ; ASSUMPTIONS:
                     001D  5559 ;
                     001D  5560 ;       THIS TEST ASSUMES THE CACHE DATA AND TAG MEMORIES HAVE BEEN TESTED.
                     001D  5561 ;
                     001D  5562 ;
                     001D  5563 ; ERROR DESCRIPTION:
                     001D  5564 ;
                     001D  5565 ;       FIRST IFERROR:
                     001D  5566 ;               EXPECTED CS ADDRESS
                     001D  5567 ;               RECEIVED CS ADDRESS
                     001D  5568 ;               LOOP COUNT I
                     001D  5569 ;
                     001D  5570 ;       SECOND IFERROR:
                     001D  5571 ;               EXPECTED CACHE DATA
                     001D  5572 ;               RECEIVED CACHE DATA
                     001D  5573 ;               LOOP COUNT I
                     001D  5574 ;
                     001D  5575 ;--
                     001D  5576 ;*****************************************************************************
                     001D  5577
                     001D  5578
                     001D  5579         INITIALIZE                              ;INIT THE CPU
                     001F  5580         EXPUTRAP                                ;POSSIBLE FAILURE MODE
                     0021  5581         LOOP            I,1,20                  ;CREATE A TEST LOOP OF 20
                     0028  5582         SAVEINDEX       I,1$                    ;SAVE LOOP COUNT IN 1$
                     002D  5583         LOADREG         WDREG,1$,,L             ;LOAD WDREG WITH LOOP COUNT
                     0035  5584         ERRLOOP                                 ;ERROR LOOP POINT
                     0037  5585         FETCH           0                       ;START DCS PROGRAM
                     003C  5586         BRSTCLK         6                       ;DCS PROGRAM
                     0040  5587         CMPREGMSK       CSTRP,2$,,3$,,W,        ;TEST FOR MACHINE CHECK
                     004C  5588         IFERROR                                 ;DATA PARITY ERROR
                     0051  5589         CMPREG          WHREG,4$,I,L,           ;TEST FOR CORRECT DATA
                     005A  5590         IFERROR                                 ;DATA ERROR
                     005F  5591         ENDLOOP         I                       ;END TEST LOOP
                     0062  5592         SKIP                                    ;GO TO NEXT TEST
                     0069  5593
                     0069  5594
                     0069  5595 BEGIN_TEST_DATA
                     0069
                     0069               ;-------------------------------------------------------------------------
                     0069  5596
00000000             0069  5597 1$:      .LONG   ^X00000000                      ;LOOP COUNT STORAGE
    1806             006D  5598 2$:      .WORD   ^X1806                          ;LAST ADDRESS IN DCS
    3FFF             006F  5599 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
6F6F6F6F             0071  5600 4$:      .LONG   ^X6F6F6F6F                      ;TEST PATTERN TABLE
91919191             0075  5601          .LONG   ^X91919191
36363636             0079  5602          .LONG   ^X36363636
44444444             007D  5603          .LONG   ^X44444444
A8A8A8A8             0081  5604          .LONG   ^XA8A8A8A8
1D1D1D1D             0085  5605          .LONG   ^X1D1D1D1D
E0E0E0E0             0089  5606          .LONG   ^XE0E0E0E0
D9D9D9D9             008D  5607          .LONG   ^XD9D9D9D9
```

```
                    6D6D6D6D  0091  5608          .LONG   ^X6D6D6D6D
                    90909090  0095  5609          .LONG   ^X90909090
                    26262626  0099  5610          .LONG   ^X26262626
                    57575757  009D  5611          .LONG   ^X57575757
                    E1E1E1E1  00A1  5612          .LONG   ^XE1E1E1E1
                    D8D8D8D8  00A5  5613          .LONG   ^XD8D8D8D8
                    92929292  00A9  5614          .LONG   ^X92929292
                    54545454  00AD  5615          .LONG   ^X54545454
                    A9A9A9A9  00B1  5616          .LONG   ^XA9A9A9A9
                    1C1C1C1C  00B5  5617          .LONG   ^X1C1C1C1C
                    E3E3E3E3  00B9  5618          .LONG   ^XE3E3E3E3
                    07070707  00BD  5619          .LONG   ^X07070707
                              00C1  5620
                              00C1  5621
                              00C1  5622 BEGIN_CPU_DATA
                              0002  5623
                              0002  5624 DCS_DATA
                              0001  5625
U 00, 7700,C800,0364,0300,0470,1801   0001  5626 ;% 00    NOP
U 01, 5700,8840,0364,0300,0470,1802   000C  5630 ;% 01    R[TEMP0]_RDM                    ;LOOP COUNT TO RTEMP0
U 02, 7700,D800,EF64,0301,0470,1803   0017  5634 ;% 02    PL_[2]                          ;POSITION LATCH GETS 2
U 03, 7700,C800,8B70,0300,4A70,1804   0022  5638 ;% 03    VA_R[TEMP0].RL.P                ;VA GETS LOOP COUNT SHIFTED 2
U 04, 7700,C800,0364,0300,0400,1805   002D  5642 ;% 04    READ.PHY                        ;READ CACHE DATA
U 05, 6700,0812,5924,0300,0470,1806   0038  5646 ;% 05    RDM_WB,WB_M[MDR]                ;TO RDM
U 06, 7300,C800,0364,0300,0470,1807   0043  5650 ;% 06    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                              004E  5654
                              004E  5655 CACHE_DATA
                              0001  5656
                    6F6F6F6F  0001  5657          .LONG   ^X6F6F6F6F
                    91919191  0005  5658          .LONG   ^X91919191
                    36363636  0009  5659          .LONG   ^X36363636
                    44444444  000D  5660          .LONG   ^X44444444
                    A8A8A8A8  0011  5661          .LONG   ^XA8A8A8A8
                    1D1D1D1D  0015  5662          .LONG   ^X1D1D1D1D
                    E0E0E0E0  0019  5663          .LONG   ^XE0E0E0E0
                    D9D9D9D9  001D  5664          .LONG   ^XD9D9D9D9
                    6D6D6D6D  0021  5665          .LONG   ^X6D6D6D6D
                    90909090  0025  5666          .LONG   ^X90909090
                    26262626  0029  5667          .LONG   ^X26262626
                    57575757  002D  5668          .LONG   ^X57575757
                    E1E1E1E1  0031  5669          .LONG   ^XE1E1E1E1
                    D8D8D8D8  0035  5670          .LONG   ^XD8D8D8D8
                    92929292  0039  5671          .LONG   ^X92929292
                    54545454  003D  5672          .LONG   ^X54545454
                    A9A9A9A9  0041  5673          .LONG   ^XA9A9A9A9
                    1C1C1C1C  0045  5674          .LONG   ^X1C1C1C1C
                    E3E3E3E3  0049  5675          .LONG   ^XE3E3E3E3
                    07070707  004D  5676          .LONG   ^X07070707
                              0051  5677
                              0051  5678 END_CPU_DATA
                              00C1  5679
                              00C1  5680
                              00C1  5681 END_TEST_DATA
                              00C1
                              00C1       ;-----------------------------------------------------------------------------
```

```
ZZ-ECKAC-8.0    V08.00                  "TEST 021 CACHE 17-FEB-1986      Fiche 1  Frame H15        Sequence 189
ECKAC                    VAX-11/750 MIC MICRO DIAGNOSTICS          17-FEB-1986 13:38:57  VAX/VMS Macro V04-00        Page 12
V08.00                   "TEST 021 CACHE DATA PARITY TEST 3        17-FEB-1986 13:38:39  [VAX750.MIC]ECKAC1.MAR;1          (1

                    00C1  5682 ENDTEST
```

```
                        .SBTTL  "TEST   022     CACHE TAG PARITY TEST 1
001C  5684 ;*****************************************************************************
001C  5685 ;++
001C  5686 ;
001C  5687 ; FUNCTIONAL DESCRIPTION:
001C  5688 ;
001C  5689 ;       THIS TEST VERIFIES THAT THE CACHE TAG PARITY GENERATOR/CHECKERS CAN
001C  5690 ;       CREATE AN ERROR CONDITION.
001C  5691 ;
001C  5692 ;
001C  5693 ; TEST ALGORITHM:
001C  5694 ;
001C  5695 ;       1.  CREATE A TEST LOOP OF 2
001C  5696 ;       2.  LOAD A TEST ADDRESS INTO THE RDM WDREG
001C  5697 ;       3.  EXECUTE DCS PROGRAM
001C  5698 ;               a.  PUT TEST ADDRESS INTO VA REGISTER
001C  5699 ;               b.  WRITE ZEROS TO THE TEST ADDRESS WHILE FORCING A PARITY ERR
001C  5700 ;               c.  READ THE TEST ADDRESS
001C  5701 ;       4.  TEST THE CS ADDRESS FOR A MACHINE CHECK
001C  5702 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
001C  5703 ;               a.  READ THE CACHE ERROR REGISTER TO THE RDM
001C  5704 ;               b.  CLEAR THE CACHE ERROR REGISTER
001C  5705 ;               c.  CLEAR THE MACHINE CHECK REGISTER
001C  5706 ;       6.  TEST THE CACHE ERROR REGISTER FOR TAG, DATA AND LOST ERRORS
001C  5707 ;       7.  IF NO ERROR GO TO STEP 2 FOR RAMAINING TEST PATTERN
001C  5708 ;
001C  5709 ;
001C  5710 ; TEST PATTERNS:
001C  5711 ;
001C  5712 ;       ITERATION       TEST PATTERN (TAG ADDRESS)
001C  5713 ;           1             00000000
001C  5714 ;           2             00001000
001C  5715 ;
001C  5716 ;
001C  5717 ; LOGIC DESCRIPTION:
001C  5718 ;
001C  5719 ;       THIS TEST INSURES THAT THE CAHE TAG PARITY GENERATOR/CHECKERS CAN
001C  5720 ;       CAN CREATE AND DETECT A PARITY ERROR.  TWO TEST PATTERNS ARE USED
001C  5721 ;       TO ALLOW EACH CHECKER TO DRIVE THE ERROR SIGNAL.  THE CAK GATE
001C  5722 ;       ARRAY IS RESPONSIBLE FOR DETECTING THE ERROR SIGNAL AND REPORTING
001C  5723 ;       IT TO THE CML ARRAY WHICH INTURN NOTIFIES THE UTR ARRAY.
001C  5724 ;
001C  5725 ;
001C  5726 ; ASSUMPTIONS:
001C  5727 ;
001C  5728 ;       THIS TEST ASSUMES THE CACHE TAG AND DATA MEMORIES HAVE BEEN TESTED
001C  5729 ;
001C  5730 ;
001C  5731 ; ERROR DESCRIPTION:
001C  5732 ;
001C  5733 ;       FIRST IFERROR:
001C  5734 ;               EXPECTED CS ADDRESS
001C  5735 ;               RECEIVED CS ADDRESS
001C  5736 ;               LOOP COUNT I
001C  5737 ;
```

```
                                001C  5738 ;       SECOND IFERROR:
                                001C  5739 ;               EXPECTED CACHE ERROR REGISTER CONTENTS
                                001C  5740 ;               RECEIVED CACHE ERROR REGISTER CONTENTS
                                001C  5741 ;               LOOP COUNT I
                                001C  5742 ;
                                001C  5743 ;--
                                001C  5744 ;*****************************************************************************
                                001C  5745
                                001C  5746
                                001C  5747         INITIALIZE                                  ;INIT THE CPU
                                001E  5748         EXPUTRAP                                    ;EXPECT A MACHINE CHECK
                                0020  5749         LOOP            I,1,2                       ;CREATE A TEST LOOP OF 2
                                0027  5750         LOADREG         WDREG,1$,I,L                ;LOAD TEST ADDRESS INTO WDREG
                                002F  5751         ERRLOOP                                     ;ERROR LOOP POINT
                                0031  5752         FETCH           0                           ;START DCS PROGRAM
                                0036  5753         BRSTCLK         6                           ;END DCS PROGRAM
                                003A  5754         CMPREGMSK       CSTRP,2$,,3$,,W,            ;TEST FOR MACHINE CHECK
                                0046  5755         IFERROR         ,<<CAK><CML><UTR>>,         ;DID NOT VECTOR CORRECTLY
                                004E  5756         FETCH           7                           ;START DCS PROGRAM
                                0053  5757         BRSTCLK         <^X0D>                      ;END DCS PROGRAM
                                0057  5758         CMPREGMSK       WHREG,4$,,5$,,L,            ;TEST CACHE ERROR REGISTER
                                0063  5759         IFERROR         ,<CAK>,                     ;REGISTER INCORRECT
                                0069  5760         ENDLOOP         I                           ;END TEST LOOP
                                006C  5761         SKIP                                        ;GO TO NEXT TEST
                                0073  5762
                                0073  5763
                                0073  5764 BEGIN_TEST_DATA
                                0073
                                0073       ;-------------------------------------------------------------------------
                                0073  5765
                    00000000    0073  5766 1$:      .LONG    ^X00000000                         ;TEST ADDRESSES
                    00001000    0077  5767         .LONG    ^X00001000                         ;...
                        0028    007B  5768 2$:      .WORD    ^X0028                             ;MACHINE CHECK VECTOR
                        3FFF    007D  5769 3$:      .WORD    ^X3FFF                             ;MASK FOR CMPREGMSK PSEUDO OP
                    0D000000    007F  5770 4$:      .LONG    ^X0D000000                         ;CONTENTS OF CACHE ERROR REG
                    0F000000    0083  5771 5$:      .LONG    ^X0F000000                         ;MASK FOR CMPREGMSK PSEUDO OP
                                0087  5772
                                0087  5773
                                0087  5774 BEGIN_CPU_DATA
                                0002  5775
                                0002  5776 DCS_DATA
                                0001  5777
U 00, 7700,C800,0364,0300,0470,1801 0001  5778 ;% 00   NOP
U 01, 5700,C800,0364,0300,4A70,1802 000C  5782 ;% 01   VA_RDM                                   ;TEST ADDRESS TO VA REGISTER
U 02, 7700,4F80,5BE4,03D8,5480,1803 0017  5786 ;% 02   WRITE.PHY,WCTRL/WDR_WB.UR,WB_R[ZERO],MISC/FORCE.CACHE
                                0022  5790                                                    ;WRITE CACHE WITH PARITY ERROR
U 03, 7700,CF80,5BE4,03D8,5470,1804 0022  5791 ;% 03   WCTRL/WDR_WB.UR,WB_R[ZERO],MISC/FORCE.CACHE
U 04, 7700,C800,0364,0300,0400,1805 002D  5795 ;% 04   READ.PHY                                 ;READ THE PARITY ERROR
U 05, 7700,4800,0364,0300,0470,1806 0038  5799 ;% 05   NOP                                     ;...
U 06, 7300,C800,0364,0300,0470,1807 0043  5803 ;% 06   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
U 07, 7700,C800,0364,0300,0470,1808 004E  5807 ;% 07   NOP
U 08, 7700,4800,5BE4,0300,6870,1809 0059  5811 ;% 08   MEMSCAR_R[TEMP0]                         ;ADDRESS OF CACHE ERROR REG
U 09, 6700,4800,0364,0300,6470,180A 0064  5815 ;% 09   RDM_WB,WCTRL/MEMSCR                     ;READ CACHE ERROR REG TO RDM
U 0A, 7700,0800,5B70,0300,6070,180B 006F  5819 ;% 0A   MEMSCR_0                               ;CLEAR CACHE ERROR REGISTER
U 0B, 7700,0800,5BE4,0304,6870,180C 007A  5823 ;% 0B   MEMSCAR_R[TEMP1]                       ;ADDRESS OF MACHINE CHECK REG
```

```
U 0C. 7700,0800,5B70,0300,6070,180D  0085  5827 ;% 0C   MEMSCR_0                                  ;CLEAR MACHINE CHECK REGISTER
U 0D. 7300,C800,0364,0300,0470,180E  0090  5831 ;% 0D   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
                                     009B  5835
                                     009B  5836 RTEMP_DATA
                                     0001  5837
                           04000000  0001  5838         .LONG   ^X04000000                    ;ADDRESS OF CACHE ERROR REG
                           08000000  0005  5839         .LONG   ^X08000000                    ;ADDRESS OF MACHINE CHECK REG
                                     0009  5840
                                     0009  5841 END_CPU_DATA
                                     0087  5842
                                     0087  5843
                                     0087  5844 END_TEST_DATA
                                     0087
                                     0087        ;-------------------------------------------------------------------------
                                     0087  5845 ENDTEST
```

```
                        001C        .SBTTL  "TEST   023     CACHE TAG PARITY TEST 2
                        001C  5847 ;*******************************************************************************
                        001C  5848 ;++
                        001C  5849 ;
                        001C  5850 ; FUNCTIONAL DESCRIPTION:
                        001C  5851 ;
                        001C  5852 ;       THIS TEST WILL VERIFY THE INTEGRITY OF THE CACHE TAG PARITY
                        001C  5853 ;       CHECKERS.
                        001C  5854 ;
                        001C  5855 ;
                        001C  5856 ; TEST ALGORITHM:
                        001C  5857 ;
                        001C  5858 ;       1.  CREATE A TEST LOOP OF 8
                        001C  5859 ;       2.  LOAD A TEST ADDRESS INTO THE RDM WDREG
                        001C  5860 ;       3.  EXECUTE DCS PROGRAM
                        001C  5861 ;               a.  LOAD TEST ADDRESS INTO VA AND RTEMP0
                        001C  5862 ;               b.  WRITE THE TEST ADDRESS TO CACHE
                        001C  5863 ;               c.  READ THE TEST LOCATION BACK TO THE RDM
                        001C  5864 ;       4.  TEST THE CS ADDRESS FOR NOT A MACHINE CHECK
                        001C  5865 ;       5.  IF NO ERROR TEST FOR CORRECT READ DATA
                        001C  5866 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
                        001C  5867 ;
                        001C  5868 ;
                        001C  5869 ; TEST PATTERNS:
                        001C  5870 ;
                        001C  5871 ;       ITERATION       TEST PATTERN (TAG ADDRESS)
                        001C  5872 ;           1               0066F000
                        001C  5873 ;           2               00690000
                        001C  5874 ;           3               00125000
                        001C  5875 ;           4               002B6000
                        001C  5876 ;           5               00548000
                        001C  5877 ;           6               008FC000
                        001C  5878 ;           7               00D00000
                        001C  5879 ;           8               00ED8000
                        001C  5880 ;
                        001C  5881 ;
                        001C  5882 ; LOGIC DESCRIPTION:
                        001C  5883 ;
                        001C  5884 ;       THIS TEST WRITES AND READS 8 CACHE LOCATIONS TO VERIFY THE INTEGRITY
                        001C  5885 ;       OF THE 74S280 CACHE TAG PARITY GENERATOR/CHECKERS.  ANY ERRORS SHOULD
                        001C  5886 ;       BE CONFINED TO THESE GATES.
                        001C  5887 ;
                        001C  5888 ;
                        001C  5889 ; ASSUMPTIONS:
                        001C  5890 ;
                        001C  5891 ;       THIS TEST ASSUMES THE CACHE TAG AND DATA MEMORIES ARE OPERATIONAL.
                        001C  5892 ;
                        001C  5893 ;
                        001C  5894 ; ERROR DESCRIPTION:
                        001C  5895 ;
                        001C  5896 ;       FIRST IFERROR:
                        001C  5897 ;               EXPECTED CS ADDRESS
                        001C  5898 ;               RECEIVED CS ADDRESS
                        001C  5899 ;               LOOP COUNT I
                        001C  5900 ;
```

```
                     001C  5901 ;       SECOND IFERROR:
                     001C  5902 ;                EXPECTED CACHE DATA
                     001C  5903 ;                RECEIVED CACHE DATA
                     001C  5904 ;                LOOP COUNT I
                     001C  5905 ;
                     001C  5906 ;--
                     001C  5907 ;*******************************************************************************
                     001C  5908
                     001C  5909
                     001C  5910
                     001C  5911          INITIALIZE                          ;INIT THE CPU
                     001E  5912          EXPUTRAP                            ;POSSIBLE FAILURE MODE
                     0020  5913          LOOP            I,1,8               ;CREATE A TEST LOOP OF 8
                     0027  5914          LOADREG         WDREG,1$,I,L        ;LOAD TEST ADDRESS INTO WDREG
                     002F  5915          ERRLOOP                             ;ERROR LOOP POINT
                     0031  5916          FETCH           0                   ;START DCS PROGRAM
                     0036  5917          BRSTCLK         6                   ;END DCS PROGRAM
                     003A  5918          CMPREGMSK       CSTRP,2$,,3$,,W,    ;TEST FOR CORRECT CS ADDRESS
                     0046  5919          IFERROR                             ;TAG PARITY ERROR
                     004B  5920          CMPREG          WHREG,1$,I,L,       ;TEST FOR CORRECT DATA
                     0054  5921          IFERROR                             ;INCORRECT CACHE DATA
                     0059  5922          ENDLOOP         I                   ;END TEST LOOP
                     005C  5923          SKIP                                ;GO TO NEXT TEST
                     0063  5924
                     0063  5925
                     0063  5926 BEGIN_TEST_DATA
                     0063  5927          ;-------------------------------------------------------------------------
                     0063  5928 1$:       .LONG    ^X0066F000                 ;TABLE OF CACHE TAGS
                     0067  5929          .LONG    ^X00690000
                     006B  5930          .LONG    ^X00125000
                     006F  5931          .LONG    ^X002B6000
                     0073  5932          .LONG    ^X00548000
                     0077  5933          .LONG    ^X008FC000
                     007B  5934          .LONG    ^X00D00000
                     007F  5935          .LONG    ^X00ED8000
                     0083  5936 2$:       .WORD    ^X1806                     ;FINAL DCS ADDRESS
                     0085  5937 3$:       .WORD    ^X3FFF                     ;MASK FOR CMPREGMSK PSEUDO OP
                     0087  5938
                     0087  5939
                     0087  5940 BEGIN_CPU_DATA
                     0002  5941
                     0002  5942 DCS_DATA
                     0001  5943
U 00, 7700,C800,0364,0300,0470,1801  0001  5944 ;% 00    NOP
U 01, 5700,8840,0364,0300,4A70,1802  000C  5948 ;% 01    VA_RDM,RSRC/TEMP0,SPW/RLONG          ;WRITE TEST ADDRESS INTO VA REG
U 02, 7700,4800,5BE4,0300,5480,1803  0017  5952 ;% 02    WRITE.PHY,WCTRL/WDR_WB.UR,WB_R[TEMP0]  ;WRITE TEST ADDRESS TO CACHE
U 03, 7700,C800,5BE4,0300,5470,1804  0022  5956 ;% 03    WCTRL/WDR_WB.UR,WB_R[TEMP0]          ;...
U 04, 7700,C800,0364,0300,0400,1805  002D  5960 ;% 04    READ.PHY                             ;READ TEST ADDRESS
U 05, 6700,0812,5924,0300,0470,1806  0038  5964 ;% 05    RDM_WB,WB_M[MDR]                     ;...
U 06, 7300,C800,0364,0300,0470,1807  0043  5968 ;% 06    NOP,STOP.CLOCK                       ;STOP THE CPU CLOCK
                     004E  5972
                     004E  5973 END_CPU_DATA
                     0087  5974
```

```
0087  5975
0087  5976 END_TEST_DATA
0087
0087        ;-------------------------------------------------------------------------------
0087  5977 ENDTEST
```

```
001C          .SBTTL  "TEST   024     CACHE TAG PARITY TEST 3
001C  5979 ;*******************************************************************************
001C  5980 ;++
001C  5981 ;
001C  5982 ; FUNCTIONAL DESCRIPTION:
001C  5983 ;
001C  5984 ;       THIS TEST WILL VERIFY THE INTEGRITY OF THE CACHE TAG PARITY CHECKERS
001C  5985 ;       WHILE FORCING A PARITY ERROR.
001C  5986 ;
001C  5987 ;
001C  5988 ; TEST ALGORITHM:
001C  5989 ;
001C  5990 ;       1.  CREATE A TEST LOOP OF 8
001C  5991 ;       2.  LOAD A TEST ADDRESS INTO THE WDREG
001C  5992 ;       3.  EXECUTE DCS PROGRAM
001C  5993 ;               a.  WRITE TEST ADDRESS INTO VA REGISTER
001C  5994 ;               b.  WRITE CACHE WITH PARITY ERROR
001C  5995 ;               c.  READ CACHE LOCATION BACK
001C  5996 ;       4.  TEST CS ADDRESS FOR A MACHINE CHECK
001C  5997 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
001C  5998 ;               a.  READ CACHE ERROR REGISTER TO RDM
001C  5999 ;               b.  CLEAR CACHE ERROR REGISTER
001C  6000 ;       6.  TEST CACHE ERROR REGISTER FOR TAG, DATA AND HIT
001C  6001 ;       7.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
001C  6002 ;
001C  6003 ;
001C  6004 ; TEST PATTERNS:
001C  6005 ;
001C  6006 ;       ITERATION       TEST PATTERN (TAG ADDRESS)
001C  6007 ;           1             00B6F000
001C  6008 ;           2             00490000
001C  6009 ;           3             00925000
001C  6010 ;           4             002B6000
001C  6011 ;           5             00548000
001C  6012 ;           6             008FC000
001C  6013 ;           7             00D00000
001C  6014 ;           8             006D8000
001C  6015 ;
001C  6016 ;
001C  6017 ; LOGIC DESCRIPTION:
001C  6018 ;
001C  6019 ;       THIS TEST WILL VERITFY THE INTEGRITY OF THE 74S280 CACHE TAG PARITY
001C  6020 ;       CHECKERS WHILE CREATING A PARITY ERROR.  ANY ERRORS SHOULD BE CONFINED
001C  6021 ;       TO THESE GATES.
001C  6022 ;
001C  6023 ;
001C  6024 ; ASSUMPTIONS:
001C  6025 ;
001C  6026 ;       THIS TEST ASSUMES THE CACHE DATA AND TAG MEMORIES HAVE BEEN TESTED
001C  6027 ;
001C  6028 ;
001C  6029 ; ERROR DESCRIPTION:
001C  6030 ;
001C  6031 ;       FIRST IFERROR:
001C  6032 ;               EXPECTED CS ADDRESS
```

```
                                001C  ;                RECEIVED CS ADDRESS
                                001C  ;                LOOP COUNT I
                                001C  ;
                                001C  ;        SECOND IFERROR:
                                001C  ;                EXPECTED CONTENTS OF CACHE ERROR REGISTER
                                001C  ;                RECEIVED CONTENTS OF CACHE ERROR REGISTER
                                001C  ;                LOOP COUNT I
                                001C  ;
                                001C  ;--
                                001C  ;*******************************************************************************
                                001C
                                001C
                                001C
                                001C          INITIALIZE                                ;INIT THE CPU
                                001E          EXPUTRAP                                  ;POSSIBLE FAILURE MODE
                                0020          LOOP            I,1,8                     ;CREATE A TEST LOOP OF 8
                                0027          LOADREG         WDREG,1$,I,L              ;LOAD TEST ADDRESS INTO WDREG
                                002F          ERRLOOP                                   ;ERROR LOOP POINT
                                0031          FETCH           0                         ;START DCS PROGRAM
                                0036          BRSTCLK         6                         ;END DCS PROGRAM
                                003A          CMPREGMSK       CSTRP,2$,,3$,,W,          ;TEST FOR CORRECT CS ADDRESS
                                0046          IFERROR                                   ;NO TAG PARITY ERROR
                                004B          FETCH           7                         ;START DCS PROGRAM
                                0050          BRSTCLK         <^X0B>                    ;END DCS PROGRAM
                                0054          CMPREGMSK       WHREG,4$,,5$,,L,          ;TEST CACHE ERROR REGISTER
                                0060          IFERROR                                   ;NOT A TAG PARITY ERROR
                                0065          ENDLOOP         I                         ;END TEST LOOP
                                0068          SKIP                                      ;GO TO NEXT TEST
                                006F
                                006F
                                006F  BEGIN_TEST_DATA
                                006F
                                006F      ;-----------------------------------------------------------------------
                                006F
                        00B6F000  006F  1$:     .LONG   ^X00B6F000                        ;TABLE OF CACHE TAGS
                        00490000  0073          .LONG   ^X00490000
                        00925000  0077          .LONG   ^X00925000
                        002B6000  007B          .LONG   ^X002B6000
                        00548000  007F          .LONG   ^X00548000
                        008FC000  0083          .LONG   ^X008FC000
                        00D00000  0087          .LONG   ^X00D00000
                        006D8000  008B          .LONG   ^X006D8000
                            0028  008F  2$:     .WORD   ^X0028                            ;MACHINE CHECK VECTOR
                            3FFF  0091  3$:     .WORD   ^X3FFF                            ;MASK FOR CMPREGMSK PSEUDO OP
                        0D000000  0093  4$:     .LONG   ^X0D000000                        ;CONTENTS OF CACHE ERROR REG
                        0F000000  0097  5$:     .LONG   ^X0F000000                        ;MASK FOR CMPREGMSK PSEUDO OP
                                009B
                                009B
                                009B  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 5700,C840,0364,0304,4A70,1802  000C  ;% 01   VA_RDM,RSRC/TEMP1,SPW/RLONG              ;WRITE TEST ADDRESS INTO VA REG
U 02, 7700,0F80,5BE4,0304,5480,1803  0017  ;% 02   WRITE.PHY,WCTRL/WDR_WB.UR,WB_R[TEMP1],MISC/FORCE.CACHE
```

```
                                  0022  6095                                                  ;WRITE TEST ADDRESS TO CACHE
U 03, 7700,8F80,5BE4,0304,5470,1804  0022  6096 ;% 03   WCTRL/WDR_WB.UR,WB_R[TEMP1],MISC/FORCE.CACHE
U 04, 7700,C800,0364,0300,0400,1805  002D  6100 ;% 04   READ.PHY                                 ;READ TEST ADDRESS
U 05, 6700,0812,5924,0300,0470,1806  0038  6104 ;% 05   RDM_WB,WB_M[MDR]                         ;...
U 06, 7300,C800,0364,0300,0470,1807  0043  6108 ;% 06   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
U 07, 7700,C800,0364,0300,0470,1808  004E  6112 ;% 07   NOP
U 08, 7700,4800,5BE4,0300,6870,1809  0059  6116 ;% 08   MEMSCAR_R[TEMP0]                         ;ADDRESS OF CACHE ERROR REG
U 09, 6700,4800,0364,0300,6470,180A  0064  6120 ;% 09   RDM_WB,WCTRL/MEMSCR                      ;READ CACHE ERROR REG TO RDM
U 0A, 7700,0800,5B70,0300,6070,180B  006F  6124 ;% 0A   MEMSCR_0                                 ;CLEAR CACHE ERROR REG
U 0B, 7300,4800,0364,0300,0470,180C  007A  6128 ;% 0B   NOP,STOP.CLOCK                           ;STOP THE CPU CLOCK
                                  0085  6132
                                  0085  6133 RTEMP_DATA
                                  0001  6134
                         04000000 0001  6135         .LONG   ^X04000000                        ;ADDRESS OF CACHE ERROR REG
                                  0005  6136
                                  0005  6137 END_CPU_DATA
                                  009B  6138
                                  009B  6139
                                  009B  6140 END_TEST_DATA
                                  009B
                                  009B       ;-----------------------------------------------------------------------
                                  009B  6141 ENDTEST
```

```
0020          .SBTTL  "TEST   025     TEST MBUS_TB_WBUS DATA PATH
0020   6143 ;*****************************************************************************
0020   6144 ;++
0020   6145 ;
0020   6146 ; FUNCTIONAL DESCRIPTION:
0020   6147 ;
0020   6148 ;       THIS TEST WILL CHECK THE MBUS_TB_WBUS DATA PATH BY LOADING AND
0020   6149 ;       READING 6 TEST PATTERNS THROUGH THE TRANSLATION BUFFER.
0020   6150 ;
0020   6151 ;
0020   6152 ; TEST ALGORITHM:
0020   6153 ;
0020   6154 ;       1.  CREATE A TEST LOOP OF 6 ITERATIONS
0020   6155 ;       2.  GENERATE A TEST PATTERN
0020   6156 ;       3.  LOAD TEST PATTERN INTO WD REGISTER (RDM)
0020   6157 ;       4.  EXECUTE DCS PROGRAM
0020   6158 ;               a.  SET TB DISABLE REGISTER TO USE GROUP 0
0020   6159 ;               b.  WRITE 0 INTO VA REGISTER
0020   6160 ;               c.  WRITE TEST PATTERN FROM WD REGISTER (RDM) INTO TB
0020   6161 ;               d.  READ TEST PATTERN BACK TO WH REGISTER (RDM)
0020   6162 ;       5.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0020   6163 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0020   6164 ;
0020   6165 ;
0020   6166 ; TEST PATTERNS:
0020   6167 ;
0020   6168 ;       ITERATION       TEST PATTERN
0020   6169 ;           1             00AAAAA8
0020   6170 ;           2             00555550
0020   6171 ;           3             00333330
0020   6172 ;           4             000F0F08
0020   6173 ;           5             00FF00F8
0020   6174 ;           6             0000FFF8
0020   6175 ;
0020   6176 ;
0020   6177 ; LOGIC DESCRIPTION:
0020   6178 ;
0020   6179 ;       THIS TEST WILL CHECK THE MDR GATE ARRAY DATA PATH TO AND FROM THE
0020   6180 ;       TRANSLATION BUFFER.  TEST PATTERNS WILL BE LOADED FROM THE RDM
0020   6181 ;       TO THE TB VIA THE WBUS AND AMUX WITHIN THE MDR GATE ARRAY.  THE
0020   6182 ;       PATTERNS ARE THEN READ BACK TO THE RDM THROUGH THE MMUX OF MDR,
0020   6183 ;       MBUS, DPM, AND FINALLY BACK TO WBUS AND RDM.  IF REPLACING MDR
0020   6184 ;       DOES NOT FIX ANY PROBLEMS, THEN YOU MIGHT TRY REPLACING ADK AND
0020   6185 ;       PRK WHICH CONTROL THE AMUX AND MMUX RESPECTIVELY.  IF YOUR STILL
0020   6186 ;       HAVING TROUBLE THEN YOU PROBABLY HAVE A TB FAILURE WHICH WILL
0020   6187 ;       REQUIRE SOME SCOPING IN THAT AREA.  NOTE THAT ONLY BITS <23:03>
0020   6188 ;       ARE USED BY THE TB.
0020   6189 ;
0020   6190 ;
0020   6191 ; ASSUMPTIONS:
0020   6192 ;
0020   6193 ;       THIS TEST ASSUMES THE VA, WBUS, AND MBUS ARE OPERATIONAL.
0020   6194 ;
0020   6195 ;
0020   6196 ; ERROR DESCRIPTION:
```

```
                        0020  6197 ;
                        0020  6198 ;       EXPECTED TB DATA
                        0020  6199 ;       RECEIVED TB DATA
                        0020  6200 ;       LOOP COUNT
                        0020  6201 ;
                        0020  6202 ;--
                        0020  6203 ;*******************************************************************************
                        0020  6204 ;///////////////////////////////////////////////////////////////////////////////
                        0020  6205
                        0020  6206         INITIALIZE                                      ;INIT CPU
                        0022  6207         LOOP            I,1,6                           ;CREATE TEST LOOP
                        0029  6208         SA01PAT         I,1$,,                          ;GENERATE TEST PATTERN
                        0031  6209         LOADREG         WDREG,1$,,L                     ;LOAD TEST PATTERN INTO D REG
                        0039  6210         ERRLOOP                                         ;ERROR LOOP
                        003B  6211         FETCH           0                               ;START DCS PROGRAM
                        0040  6212         BRSTCLK         6                               ;END DCS PROGRAM
                        0044  6213         CMPREGMSK       WHREG,1$,,2$,,L,                ;COMPARE RESULTS
                        0050  6214         IFERROR         ,<<MDR><ADK><PRK>>,             ;POSSIBLE FAILING ARRAYS
                        0058  6215         ENDLOOP         I                               ;END TEST LOOP
                        005B  6216         SKIP
                        0062  6217
                        0062  6218 BEGIN_TEST_DATA
                        0062
                        0062       ;-------------------------------------------------------------------------
                        0062  6219
00000000                0062  6220 1$:     .LONG   ^X00000000                              ;TEST PATTERN STORAGE LOCATION
00FFFFF8                0066  6221 2$:     .LONG   ^X00FFFFF8                              ;MASK FOR CMPREGMSK PSEUDO OP
                        006A  6222
                        006A  6223 BEGIN_CPU_DATA
                        0002  6224
                        0002  6225 DCS_DATA
                        0001  6226
U 00, 7700,C800,0364,0300,0470,1801     0001  6227 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802     000C  6231 ;% 01   MEMSCAR_R[TEMP0]                 ;ADD OF TB DISABLE TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803     0017  6235 ;% 02   MEMSCR_R[TEMP1]                  ;DISABLE GROUP 1
U 03, 7700,C800,5BE4,03D8,4A70,1804     0022  6239 ;% 03   VA_R[ZERO]                       ;MOVE 0 TO VA REG
U 04, 5700,C81B,0364,0300,5070,1805     002D  6243 ;% 04   WB_RDM,WCTRL/TB_WB,MSRC/VA       ;WRITE TEST PATTERN TO TB
U 05, 6700,481F,5924,0300,0470,1806     0038  6247 ;% 05   RDM_WB,WB_M[TB]                  ;READ TEST PATTERN TO RDM
U 06, 7300,C800,0364,0300,0470,1807     0043  6251 ;% 06   NOP,STOP.CLOCK                   ;STOP CPU CLOCK
                        004E  6255
                        004E  6256 RTEMP_DATA
                        0001  6257
                03000000        0001  6258         .LONG   ^X03000000                      ;TB DISABLE S/C ADDRESS
                0A000000        0005  6259         .LONG   ^X0A000000                      ;DISABLE GROUP 1
                        0009  6260
                        0009  6261 END_CPU_DATA
                        006A  6262
                        006A  6263 END_TEST_DATA
                        006A
                        006A       ;-------------------------------------------------------------------------
                        006A  6264
                        006A  6265 ENDTEST
```

```
0035          .SBTTL  "TEST   026     TB DATA PATH TO AND FROM GROUP 0 TAG MEMORY TEST
0035  6267  ;*******************************************************************************
0035  6268  ;++
0035  6269  ;
0035  6270  ;       TB DATA PATH TO AND FROM GROUP 0 TAG MEMORY TEST
0035  6271  ;
0035  6272  ; FUNCTIONAL DESCRIPTION
0035  6273  ;       THIS IS A TEST OF THE PATH TAKEN BY DATA WHEN WRITING INTO
0035  6274  ;       AND READING OUT OF THE TB GROUP 0 TAG MEMORY. THE DATA WRITTEN
0035  6275  ;       INTO THIS MEMORY IS PART OF THE VA. THE DATA READ OUT OF THE
0035  6276  ;       MEMORY GOES INTO THE GROUP 0 HIT COMPARATOR LOGIC.
0035  6277  ;
0035  6278  ; TEST ALGORITHM
0035  6279  ;       1.      TURN OFF CACHE AND CMI.
0035  6280  ;       2.      INVALIDATE BOTH GROUPS IN THE TB AT INDEX LOCATION 0.
0035  6281  ;       3.      SET UP TO FORCE MISSES TO GROUP 1 AND REPLACEMENT
0035  6282  ;               OF GROUP 0.
0035  6283  ;       4.      WRITE THE TEST PATTERN INTO THE VA.
0035  6284  ;       5.      WRITE PTE INTO TB GROUP 0 ALONG WITH VA<TAG>=TEST PATTERN.
0035  6285  ;       6.      READ PTE OUT OF TB AND CHECK FOR TB HIT, CHECK
0035  6286  ;               PTE AND NO ERRORS.
0035  6287  ;       7.      REPEAT FOR 6 DATA PATTERNS.
0035  6288  ;
0035  6289  ; TEST PATTERNS
0035  6290  ;       THE TEST PATTERNS USED IN THIS TEST ARE TAG FIELDS IN
0035  6291  ;       THE VIRTUAL ADDRESS WHEN WRITING INTO THE TB. SIX PATTERNS ARE
0035  6292  ;       USED FOR THE VA (NOTE THAT THE INDEX=0 FOR ALL PATTERNS):
0035  6293  ;               1.      2AAA0000
0035  6294  ;               2.      55550000
0035  6295  ;               3.      33330000
0035  6296  ;               4.      0F0F0000
0035  6297  ;               5.      00FF0000
0035  6298  ;               6.      00000000
0035  6299  ;
0035  6300  ; LOGIC DESCRIPTION
0035  6301  ;       THE PATH FOLLOWED BY THE DATA INTO THE TB GROUP 0 TAG MEMORY
0035  6302  ;       IS FROM THE VA, THROUGH THE MA, BY THE TB TAG PARITY
0035  6303  ;       GENERATER AND INTO THE TAG MEMORY. THE PATH FOLLOWED BY
0035  6304  ;       THE DATA COMING OUT OF THE MEMORY IS BY THE PARITY CHECKERS
0035  6305  ;       AN INTO THE GROUP 0 HIT COMPARATOR. THIS HIT SIGNAL IS THEN
0035  6306  ;       USED TO CHECK THE RESULT.
0035  6307  ;
0035  6308  ; ASSUMPTIONS
0035  6309  ;       THE VA REGISTER AND MA REGISTER ARE ASSUMED TO FUNCTION PROPERLY.
0035  6310  ;
0035  6311  ; ERROR DESCRIPTION
0035  6312  ;       EXPECTED PTE
0035  6313  ;       GOT PTE
0035  6314  ;       LOOP COUNT WHICH IDENTIFIES WHICH OF THE ABOVE 6 PATTERNS FAILED
0035  6315  ;
0035  6316  ;--
0035  6317  ;*******************************************************************************
0035  6318  ;///////////////////////////////////////////////////////////////////////////////
0035  6319
0035  6320          INITIALIZE
```

```
                     0037  6321          LOADDCS         1$,,0,<^X0D>                    ; LOAD MICROCODE.
                     003E  6322          LOOP            I,1,6                           ; SET UP A LOOP THRU 6 DATA
                     0045  6323                                                          ; PATTERNS.
                     0045  6324          LOADREG         WDREG,2$,I,LONG                 ; GET TEST DATA PATTERN.
                     004D  6325          ERRLOOP
                     004F  6326          FETCH           0                               ; EXECUTE THE UCODE. WHICH
                     0054  6327          BRSTCLK         <^X0C>                          ; WILL MAKE THE TEST PATTERN
                     0058  6328                                                          ; A HIT IN TB GROUP 0.
                     0058  6329          CMPREGMSK       WHREG,3$,,5$,,LONG              ; SEE IF THE TEST PATTERN
                     0064  6330                                                          ; WAS A TB HIT.
                     0064  6331          IFERROR         ,,<MIC>                         ; UNABLE TO MAKE TEST PATTERN
                     006A  6332                                                          ; A HIT IN TB GROUP 0'S TAG
                     006A  6333                                                          ; MEMORY.
                     006A  6334          ENDLOOP         I                               ; LOOP THRU 6 DATA PATTERNS.
                     006D  6335          SKIP                                            ; GO TO NEXT TEST
                     0074  6336
                     0074  6337 BEGIN_TEST_DATA
                     0074
                     0074        ;-------------------------------------------------------------------------
                     0074  6338
                     0074  6339 1$:
U 00, 7700,C800,0364,0300,0470,1801  0074  6340 ;% 00    NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  007F  6344 ;% 01    D_R[ZERO]                       ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803  008A  6348 ;% 02    VA_Q_D_D+ZLIT8[0]               ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804  0095  6352 ;% 03    VA_D_D+ZLIT8[0] INVALIDATE TB   ; 200.............
U 04, 7700,F800,7E7F,FFFF,8470,1805  00A0  6356 ;% 04    LONLIT_[03000000]               ; SET THE FORCE MISS TO GROUP
U 05, 7700,4800,5BE4,03D4,6870,1806  00AB  6360 ;% 05    MEMSCAR_R[LONLIT]               ; 1 AND FORCE REPLACE GROUP 0
U 06, 7700,7800,7AFF,FFFF,8470,1807  00B6  6364 ;% 06    LONLIT_[0A000000]               ; MAINTENANCE BITS.
U 07, 7700,4800,5BE4,03D4,6070,1808  00C1  6368 ;% 07    MEMSCR_R[LONLIT]
U 08, 5700,4800,0364,0300,4A70,1809  00CC  6372 ;% 08    VA_RDM                          ; LOAD THE TEST PATTERN INTO VA.
U 09, 7700,F800,7FAA,190B,8470,180A  00D7  6376 ;% 09    LONLIT_[00ABCDE8]               ; PTE TO BE WRITTEN.
U 0A, 7700,481B,5BE4,03D4,5070,180B  00E2  6380 ;% 0A    TB_R[LONLIT]                    ; MAKE VA A HIT IN GROUP 0.
U 0B, 6701,081F,5924,0200,05F0,180C  00ED  6384 ;% 0B    RDM_TB                          ; READ PTE OUT OF TB.
U 0C, 7300,C800,0364,0300,0470,180D  00F8  6388 ;% 0C    STOP.CLOCK
                                     0103  6392
                          2AAA0000   0103  6393 2$:      .LONG   ^X2AAA0000              ; TABLE OF TEST PATTERNS.
                          55550000   0107  6394          .LONG   ^X55550000
                          33330000   010B  6395          .LONG   ^X33330000
                          0F0F0000   010F  6396          .LONG   ^X0F0F0000
                          00FF0000   0113  6397          .LONG   ^X00FF0000
                          00000000   0117  6398          .LONG   ^X00000000
                                     011B  6399
                          00ABCDE8   011B  6400 3$:      .LONG   ^X00ABCDE8              ; EXPECTED PTE TO BE READ OUT
                                     011F  6401                                          ; OF GROUP 0.
                          00FFFFF8   011F  6402 5$:      .LONG   ^X00FFFFF8
                                     0123  6403
                                     0123  6404 END_TEST_DATA
                                     0123
                                     0123        ;-------------------------------------------------------------------------
                                     0123  6405
                                     0123  6406 ENDTEST
```

```
0035          .SBTTL  "TEST   027     TB DATA PATH TO AND FROM GROUP 1 TAG MEMORY TEST
0035   6408 ;*******************************************************************************
0035   6409 ;++
0035   6410 ;
0035   6411 ;       TB DATA PATH TO AND FROM GROUP 1 TAG MEMORY TEST
0035   6412 ;
0035   6413 ; FUNCTIONAL DESCRIPTION
0035   6414 ;       THIS IS A TEST OF THE PATH TAKEN BY DATA WHEN WRITING INTO
0035   6415 ;       AND READING OUT OF THE TB GROUP 1 TAG MEMORY. THE DATA WRITTEN
0035   6416 ;       INTO THIS MEMORY IS PART OF THE VA. THE DATA READ OUT OF THE
0035   6417 ;       MEMORY GOES INTO THE GROUP 1 HIT COMPARATOR LOGIC.
0035   6418 ;
0035   6419 ; TEST ALGORITHM
0035   6420 ;       1.      TURN OFF CACHE AND CMI.
0035   6421 ;       2.      INVALIDATE BOTH GROUPS IN THE TB AT INDEX LOCATION 0.
0035   6422 ;       3.      SET UP TO FORCE MISSES TO GROUP 0 AND REPLACEMENT
0035   6423 ;               OF GROUP 1.
0035   6424 ;       4.      WRITE THE TEST PATTERN INTO THE VA.
0035   6425 ;       5.      WRITE PTE INTO TB GROUP 0 ALONG WITH VA<TAG>=TEST PATTERN.
0035   6426 ;       6.      READ PTE OUT OF TB AND CHECK FOR TB HIT, CHECK
0035   6427 ;               PTE AND NO ERRORS.
0035   6428 ;       7.      REPEAT FOR 6 DATA PATTERNS.
0035   6429 ;
0035   6430 ; TEST PATTERNS
0035   6431 ;       THE TEST PATTERNS USED IN THIS TEST ARE TAG FIELDS IN
0035   6432 ;       THE VIRTUAL ADDRESS WHEN WRITING INTO THE TB. SIX PATTERNS ARE
0035   6433 ;       USED FOR THE VA (NOTE THAT THE INDEX=0 FOR ALL PATTERNS):
0035   6434 ;               1.      2AAA0000
0035   6435 ;               2.      55550000
0035   6436 ;               3.      33330000
0035   6437 ;               4.      0F0F0000
0035   6438 ;               5.      00FF0000
0035   6439 ;               6.      00000000
0035   6440 ;
0035   6441 ; LOGIC DESCRIPTION
0035   6442 ;       THE PATH FOLLOWED BY THE DATA INTO THE TB GROUP 1 TAG MEMORY
0035   6443 ;       IS FROM THE VA, THROUGH THE MA, BY THE TB TAG PARITY
0035   6444 ;       GENERATER AND INTO THE TAG MEMORY. THE PATH FOLLOWED BY
0035   6445 ;       THE DATA COMING OUT OF THE MEMORY IS BY THE PARITY CHECKERS
0035   6446 ;       AN INTO THE GROUP 1 HIT COMPARATOR. THIS HIT SIGNAL IS THEN
0035   6447 ;       USED TO CHECK THE RESULT.
0035   6448 ;
0035   6449 ; ASSUMPTIONS
0035   6450 ;       THE VA REGISTER AND MA REGISTER ARE ASSUMED TO FUNCTION PROPERLY.
0035   6451 ;
0035   6452 ; ERROR DESCRIPTION
0035   6453 ;       EXPECTED PTE
0035   6454 ;       GOT PTE
0035   6455 ;       LOOP COUNT WHICH IDENTIFIES WHICH OF THE ABOVE 6 PATTERNS FAILED
0035   6456 ;
0035   6457 ;--
0035   6458 ;*******************************************************************************
0035   6459 ;///////////////////////////////////////////////////////////////////////////////
0035   6460
0035   6461         INITIALIZE
```

```
                        0037  6462          LOADDCS         1$,,0,<^X0D>                    ; LOAD MICROCODE.
                        003E  6463          LOOP            I,1,6                           ; SET UP A LOOP THRU 6 DATA
                        0045  6464                                                          ; PATTERNS.
                        0045  6465          LOADREG         WDREG,2$,I,LONG                 ; GET TEST DATA PATTERN.
                        004D  6466          ERRLOOP
                        004F  6467          FETCH           0                               ; EXECUTE THE UCODE. WHICH
                        0054  6468          BRSTCLK         <^X0C>                          ; WILL MAKE THE TEST PATTERN
                        0058  6469                                                          ; A HIT IN TB GROUP 1.
                        0058  6470          CMPREGMSK       WHREG,3$,,5$,,LONG              ; SEE IF THE TEST PATTERN
                        0064  6471                                                          ; WAS A TB HIT.
                        0064  6472          IFERROR         ,,<MIC>                         ; UNABLE TO MAKE TEST PATTERN
                        006A  6473                                                          ; A HIT IN TB GROUP 1'S TAG
                        006A  6474                                                          ; MEMORY.
                        006A  6475          ENDLOOP         I                               ; LOOP THRU 6 DATA PATTERNS.
                        006D  6476          SKIP                                            ; GO TO NEXT TEST
                        0074  6477
                        0074  6478 BEGIN_TEST_DATA
                        0074
                        0074       ;-------------------------------------------------------------------------
                        0074  6479
                        0074  6480 1$:
U 00. 7700,C800,0364,0300,0470,1801  0074  6481 ;% 00    NOP
U 01. 7700,8800,5BE6,03D8,0470,1802  007F  6485 ;% 01    D_R[ZERO]                       ; zero the D register
U 02. 7700,D800,DB13,0300,4A70,1803  008A  6489 ;% 02    VA_Q_D_D+ZLIT8[0]               ; invalidate the TB at 0 &
U 03. 7700,1800,DB12,0300,5360,1804  0095  6493 ;% 03    VA_D_D+ZLIT8[0] INVALIDATE TB   ; 200.............
U 04. 7700,F800,7E7F,FFFF,8470,1805  00A0  6497 ;% 04    LONLIT_[03000000]               ; SET THE FORCE MISS TO GROUP
U 05. 7700,4800,5BE4,03D4,6870,1806  00AB  6501 ;% 05    MEMSCAR_R[LONLIT]               ; 0 AND FORCE REPLACE GROUP 1
U 06. 7700,3800,797F,FFFF,8470,1807  00B6  6505 ;% 06    LONLIT_[0D000000]               ; MAINTENANCE BITS.
U 07. 7700,4800,5BE4,03D4,6070,1808  00C1  6509 ;% 07    MEMSCR_R[LONLIT]
U 08. 5700,4800,0364,0300,4A70,1809  00CC  6513 ;% 08    VA_RDM                          ; LOAD THE TEST PATTERN INTO VA.
U 09. 7700,F800,7FAA,190B,8470,180A  00D7  6517 ;% 09    LONLIT_[00ABCDE8]               ; PTE TO BE WRITTEN.
U 0A. 7700,481B,5BE4,03D4,5070,180B  00E2  6521 ;% 0A    TB_R[LONLIT]                    ; MAKE VA A HIT IN GROUP 1.
U 0B. 6701,081F,5924,0200,05F0,180C  00ED  6525 ;% 0B    RDM_TB                          ; READ PTE OUT OF TB.
U 0C. 7300,C800,0364,0300,0470,180D  00F8  6529 ;% 0C    STOP.CLOCK
                                     0103  6533
                        2AAA0000     0103  6534 2$:      .LONG   ^X2AAA0000              ; TABLE OF TEST PATTERNS.
                        55550000     0107  6535         .LONG   ^X55550000
                        33330000     010B  6536         .LONG   ^X33330000
                        0F0F0000     010F  6537         .LONG   ^X0F0F0000
                        00FF0000     0113  6538         .LONG   ^X00FF0000
                        00000000     0117  6539         .LONG   ^X00000000
                                     011B  6540
                        00ABCDE8     011B  6541 3$:      .LONG   ^X00ABCDE8              ; EXPECTED PTE TO BE READ OUT
                                     011F  6542                                         ; OF GROUP 1.
                        00FFFFF8     011F  6543 5$:      .LONG   ^X00FFFFF8
                                     0123  6544
                                     0123  6545 END_TEST_DATA
                                     0123
                                     0123       ;-------------------------------------------------------------------------
                                     0123  6546
                                     0123  6547 ENDTEST
```

```
                        .SBTTL  "TEST   028     TB GROUP 0 FORCE MISS TEST
001F  6549 ;*******************************************************************************
001F  6550 ;++
001F  6551 ;
001F  6552 ;       TB GROUP 0 FORCE MISS TEST
001F  6553 ;
001F  6554 ; FUNCTIONAL DESCRIPTION:
001F  6555 ;       THIS IS A TEST OF THE TRANSLATION BUFFER GROUP 0 FORCE MISS BIT.
001F  6556 ;
001F  6557 ;
001F  6558 ; TEST ALGORITHM:
001F  6559 ;       1.      INVALIDATE BOTH TB GROUPS AT INDEX LOCATION 0.
001F  6560 ;       2.      SET THE TB GROUP 1 FORCE MISS BIT AND THE TB GROUP 0
001F  6561 ;               FORCE REPLACE BIT.
001F  6562 ;       3.      WRITE PATTERN INTO VA.
001F  6563 ;       4.      WRITE PTE INTO GROUP 0 WITH VA.
001F  6564 ;       5.      CLEAR TB GROUP 1 FORCE MISS BIT AND SET TB GROUP 0
001F  6565 ;               FORCE MISS BIT.
001F  6566 ;       6.      READ PTE OUT OF TB AT VA AND CHECK FOR MISS.
001F  6567 ;
001F  6568 ;
001F  6569 ; TEST PATTERNS:
001F  6570 ;       NONE
001F  6571 ;
001F  6572 ;
001F  6573 ; LOGIC DESCRIPTION:
001F  6574 ;       THIS TESTS THE FORCE MISS BIT INPUT TO THE TB GROUP 0
001F  6575 ;       HIT COMPARATOR.
001F  6576 ;
001F  6577 ;
001F  6578 ; ASSUMPTIONS:
001F  6579 ;       THIS TEST ASSUMES THE WB AND DPM ARE OPERATIONAL.
001F  6580 ;
001F  6581 ;
001F  6582 ; ERROR DESCRIPTION:
001F  6583 ;       EXPECTED PTE
001F  6584 ;       GOT PTE
001F  6585 ;
001F  6586 ;--
001F  6587 ;*******************************************************************************
001F  6588 ;///////////////////////////////////////////////////////////////////////////////
001F  6589
001F  6590         INITIALIZE
0021  6591         LOADDCS         1$,,0,<^X12>            ; LOAD MICRO CODE INTO DCS.
0028  6592         ERRLOOP                                 ; ERROR LOOP
002A  6593         FETCH           0                       ; EXECUTE THE MICRO CODE.
002F  6594         BRSTCLK         <^X11>
0033  6595         CMPREGMSK       WHREG,3$,,2$,,L.        ; MAKE SURE THE VALID BIT IN
003F  6596                                                 ; THE PTE JUST READ BACK IS OFF.
003F  6597                                                 ; THIS INDICATES THAT THE
003F  6598                                                 ; REFERENCE WAS A TB MISS.
003F  6599         IFERROR         ,,<MIC>                 ; A REFERENCE WAS MADE
0045  6600                                                 ; TO AN ADDRESS WHICH WAS A HIT
0045  6601                                                 ; IN TB GROUP 0 WITH THE FORCE
0045  6602                                                 ; MISS TO GROUP 0 ASSERTED. THIS
```

```
                                  0045  6603                                                  ; REFERENCE RESULTED IN A TB HIT.
                                  0045  6604          SKIP
                                  004C  6605
                                  004C  6606 BEGIN_TEST_DATA
                                  004C
                                  004C        ;-----------------------------------------------------------------------
                                  004C  6607
                                  004C  6608 1$:
U 00, 7700,C800,0364,0300,0470,1801  004C  6609 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  0057  6613 ;% 01   D_R[ZERO]                                 ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803  0062  6617 ;% 02   VA_Q_D_D+ZLIT8[0]                         ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804  006D  6621 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB             ; 200.............
U 04, 7700,F800,7E7F,FFFF,8470,1805  0078  6625 ;% 04   LONLIT_[03000000]                         ; SET UP FOR FORCE REPLACE GROUP
U 05, 7700,4800,5BE4,03D4,6870,1806  0083  6629 ;% 05   MEMSCAR_R[LONLIT]                         ; 0 FORCE MISS GROUP 1 IN THE TB.
U 06, 7700,7800,7AFF,FFFF,8470,1807  008E  6633 ;% 06   LONLIT_[0A000000]
U 07, 7700,4800,5BE4,03D4,6070,1808  0099  6637 ;% 07   MEMSCR_R[LONLIT]
U 08, 7700,B800,6AAA,FFFF,8470,1809  00A4  6641 ;% 08   LONLIT_[2AAA0000]                         ; MAKE VA=2AAA0000 AND
U 09, 7700,4800,5BE4,03D4,4A70,180A  00AF  6645 ;% 09   VA_R[LONLIT]                              ; PTE=00234568 A HIT IN TB
U 0A, 7700,3800,7FEE,5D4B,8470,180B  00BA  6649 ;% 0A   LONLIT_[00234568]                         ; GROUP 0.
U 0B, 7700,C81B,5BE4,03D4,5070,180C  00C5  6653 ;% 0B   TB_R[LONLIT]
U 0C, 7700,7800,7E7F,FFFF,8470,180D  00D0  6657 ;% 0C   LONLIT_[03000000]                         ; SET FORCE MISS GROUP 0.
U 0D, 7700,C800,5BE4,03D4,6870,180E  00DB  6661 ;% 0D   MEMSCAR_R[LONLIT]
U 0E, 7700,B800,797F,FFFF,8470,180F  00E6  6665 ;% 0E   LONLIT_[0D000000]
U 0F, 7700,4800,5BE4,03D4,6070,1810  00F1  6669 ;% 0F   MEMSCR_R[LONLIT]
U 10, 6701,081F,5924,0200,05F0,1811  00FC  6673 ;% 10   RDM_TB                                    ; READ THE VA BACK OUT OF GROUP
U 11, 7300,4800,0364,0300,0470,1812  0107  6677 ;% 11   STOP.CLOCK                                ; 0 IT SHOULD NOW BE A MISS.
                                  0112  6681
                        00000100  0112  6682 2$:     .LONG   ^X00000100
                        00000000  0116  6683 3$:     .LONG   ^X00000000
                                  011A  6684
                                  011A  6685 END_TEST_DATA
                                  011A
                                  011A        ;-----------------------------------------------------------------------
                                  011A  6686
                                  011A  6687 ENDTEST
```

```
001F        .SBTTL  "TEST   029     TB GROUP 1 FORCE MISS TEST
001F  6689 ;*******************************************************************************
001F  6690 ;++
001F  6691 ;
001F  6692 ;       TB GROUP 1 FORCE MISS TEST
001F  6693 ;
001F  6694 ; FUNCTIONAL DESCRIPTION:
001F  6695 ;       THIS IS A TEST OF THE TRANSLATION BUFFER GROUP 1 FORCE MISS BIT.
001F  6696 ;
001F  6697 ;
001F  6698 ; TEST ALGORITHM:
001F  6699 ;       1.      INVALIDATE BOTH TB GROUPS AT INDEX LOCATION 0.
001F  6700 ;       2.      SET THE TB GROUP 0 FORCE MISS BIT AND THE TB GROUP 1
001F  6701 ;               FORCE REPLACE BIT.
001F  6702 ;       3.      WRITE PATTERN INTO VA.
001F  6703 ;       4.      WRITE PTE INTO GROUP 1 WITH VA.
001F  6704 ;       5.      CLEAR TB GROUP 0 FORCE MISS BIT AND SET TB GROUP 1
001F  6705 ;               FORCE MISS BIT.
001F  6706 ;       6.      READ PTE OUT OF TB AT VA AND CHECK FOR MISS.
001F  6707 ;
001F  6708 ;
001F  6709 ; TEST PATTERNS:
001F  6710 ;       NONE
001F  6711 ;
001F  6712 ;
001F  6713 ; LOGIC DESCRIPTION:
001F  6714 ;       THIS TESTS THE FORCE MISS BIT INPUT TO THE TB GROUP 1
001F  6715 ;       HIT COMPARATOR.
001F  6716 ;
001F  6717 ;
001F  6718 ; ASSUMPTIONS:
001F  6719 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
001F  6720 ;
001F  6721 ;
001F  6722 ; ERROR DESCRIPTION:
001F  6723 ;       EXPECTED PTE
001F  6724 ;       GOT PTE
001F  6725 ;
001F  6726 ;--
001F  6727 ;*******************************************************************************
001F  6728 ;///////////////////////////////////////////////////////////////////////////////
001F  6729
001F  6730         INITIALIZE
0021  6731         LOADDCS         1$,,0,<^X12>            ; LOAD MICRO CODE INTO DCS.
0028  6732         ERRLOOP                                 ; ERROR LOOP
002A  6733         FETCH           0                       ; EXECUTE THE MICRO CODE.
002F  6734         BRSTCLK         <^X11>
0033  6735         CMPREGMSK       WHREG,3$,,2$,,L,        ; MAKE SURE THE VALID BIT IN
003F  6736                                                 ; THE PTE JUST READ BACK IS OFF.
003F  6737                                                 ; THIS INDICATES THAT THE
003F  6738                                                 ; REFERENCE WAS A TB MISS.
003F  6739         IFERROR         ,,<MIC>                 ; A REFERENCE WAS MADE
0045  6740                                                 ; TO AN ADDRESS WHICH WAS A HIT
0045  6741                                                 ; IN TB GROUP 1 WITH THE FORCE
0045  6742                                                 ; MISS TO GROUP 1 ASSERTED. THIS
```

```
                                  0045  6743                                                  ; REFERENCE RESULTED IN A TB HIT.
                                  0045  6744          SKIP
                                  004C  6745
                                  004C  6746 BEGIN_TEST_DATA
                                  004C
                                  004C        ;-----------------------------------------------------------------------
                                  004C  6747
                                  004C  6748 1$:
U 00, 7700,C800,0364,0300,0470,1801  004C  6749 ;% 00    NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  0057  6753 ;% 01    D_R[ZERO]                                ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803  0062  6757 ;% 02    VA_Q_D_D+ZLIT8[0]                        ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804  006D  6761 ;% 03    VA_D_D+ZLIT8[0] INVALIDATE TB            ; 200.............
U 04, 7700,F800,7E7F,FFFF,8470,1805  0078  6765 ;% 04    LONLIT_[03000000]                        ; SET UP FOR FORCE REPLACE GROUP
U 05, 7700,4800,5BE4,03D4,6870,1806  0083  6769 ;% 05    MEMSCAR_R[LONLIT]                        ; 1 FORCE MISS GROUP 0 IN THE TB.
U 06, 7700,3800,797F,FFFF,8470,1807  008E  6773 ;% 06    LONLIT_[0D000000]
U 07, 7700,4800,5BE4,03D4,6070,1808  0099  6777 ;% 07    MEMSCR_R[LONLIT]
U 08, 7700,B800,6AAA,FFFF,8470,1809  00A4  6781 ;% 08    LONLIT_[2AAA0000]                        ; MAKE VA=2AAA0000 AND
U 09, 7700,4800,5BE4,03D4,4A70,180A  00AF  6785 ;% 09    VA_R[LONLIT]                             ; PTE=00234568 A HIT IN TB
U 0A, 7700,3800,7FEE,5D4B,8470,180B  00BA  6789 ;% 0A    LONLIT_[00234568]                        ; GROUP 1.
U 0B, 7700,C81B,5BE4,03D4,5070,180C  00C5  6793 ;% 0B    TB_R[LONLIT]
U 0C, 7700,7800,7E7F,FFFF,8470,180D  00D0  6797 ;% 0C    LONLIT_[03000000]                        ; SET FORCE MISS GROUP 1.
U 0D, 7700,C800,5BE4,03D4,6870,180E  00DB  6801 ;% 0D    MEMSCAR_R[LONLIT]
U 0E, 7700,F800,7AFF,FFFF,8470,180F  00E6  6805 ;% 0E    LONLIT_[0A000000]
U 0F, 7700,4800,5BE4,03D4,6070,1810  00F1  6809 ;% 0F    MEMSCR_R[LONLIT]
U 10, 6701,081F,5924,0200,05F0,1811  00FC  6813 ;% 10    RDM_TB                                   ; READ THE VA BACK OUT OF GROUP
U 11, 7300,4800,0364,0300,0470,1812  0107  6817 ;% 11    STOP.CLOCK                               ; 1 IT SHOULD NOW BE A MISS.
                                  0112  6821
                        00000100  0112  6822 2$:      .LONG    ^X00000100
                        00000000  0116  6823 3$:      .LONG    ^X00000000
                                  011A  6824
                                  011A  6825 END_TEST_DATA
                                  011A
                                  011A        ;-----------------------------------------------------------------------
                                  011A  6826
                                  011A  6827 ENDTEST
```

```
0030        .SBTTL  "TEST   02A     TB GROUP 0 VALID BIT AND TB MISS UTRAP TEST
0030  6829 ;*******************************************************************************
0030  6830 ;++
0030  6831 ;
0030  6832 ;       TB GROUP 0 VALID BIT AND TB MISS UTRAP TEST
0030  6833 ;
0030  6834 ; FUNCTIONAL DESCRIPTION:
0030  6835 ;       THIS IS A TEST OF THE TB MISS UTRAP LOGIC. A PTE IS WRITTEN
0030  6836 ;       INTO TB GROUP 0 WITH THE VALID BIT CLEAR. A REFERENCE IS THEN
0030  6837 ;       MADE WHICH SHOULD CALL THIS PTE OUT OF TB GROUP 0 RESULTING IN
0030  6838 ;       A TB MISS UTRAP.
0030  6839 ;
0030  6840 ; TEST ALGORITHM:
0030  6841 ;       1.      INVALIDATE BOTH TB GROUPS AT INDEX LOCATION 0.
0030  6842 ;       2.      SET THE TB GROUP 1 FORCE MISS BIT AND SET THE TB
0030  6843 ;               GROUP 0 FORCE REPLACEMENT BIT.
0030  6844 ;       3.      LOAD THE VA.
0030  6845 ;       4.      WRITE A PTE INTO TB GROUP 0 WHICH HAS THE VALID BIT
0030  6846 ;               CLEAR.
0030  6847 ;       5.      PERFORM A BUS READ BUS FUNCTION WHICH SHOULD RESULT
0030  6848 ;               IN A TB MISS UTRAP.
0030  6849 ;       6.      TEST THE VBUS TO SEE IF MIC IS GENERATING THE UTRAP SIGNAL
0030  6850 ;       7.      CHECK THE UTRAP VECTOR IN THE TRACE ADDRESS REGISTER.
0030  6851 ;
0030  6852 ;
0030  6853 ; TEST PATTERNS:
0030  6854 ;       NONE
0030  6855 ;
0030  6856 ;
0030  6857 ; LOGIC DESCRIPTION:
0030  6858 ;       THIS TESTS LOGIC IN THE UTR GATE ARRAY WHICH GENERATES THE
0030  6859 ;       UTRAP CONDITION AND LOGIC IN THE MICRO SEQUENCER WHICH CAUSES
0030  6860 ;       THE ACTUAL UTRAP TO OCCUR.
0030  6861 ;
0030  6862 ;
0030  6863 ; ASSUMPTIONS:
0030  6864 ;       THIS TEST ASSUMES THE DATA PATH SECTION OF DPM AND THE WBUS ARE
0030  6865 ;       OPERATIONAL.
0030  6866 ;
0030  6867 ;
0030  6868 ; ERROR DESCRIPTION:
0030  6869 ;
0030  6870 ;       FIRST IFERROR:
0030  6871 ;       EXPECTED VBUS BITS 08,09,0A,0B,0D,1B
0030  6872 ;       RECEIVED VBUS BITS
0030  6873 ;               (BITS: <0B:08> = MICRO VECTOR LINES FROM MIC TO DPM)
0030  6874 ;               (BIT 0D = UTRAP SIGNAL FROM MIC TO DPM)
0030  6875 ;               (BIT 1B = UVCTR BRANCH ON DPM)
0030  6876 ;
0030  6877 ;       SECOND IFERROR:
0030  6878 ;       EXPECTED MICRO PC (THE UTRAP VECTOR)
0030  6879 ;       GOT MICRO PC
0030  6880 ;
0030  6881 ;--
0030  6882 ;*******************************************************************************
```

```
                        0030  6883 ;////////////////////////////////////////////////////////////////////////////////
                        0030  6884
                        0030  6885         INITIALIZE
                        0032  6886         EXPUTRAP                                        ; SET UP FOR UCODE TRAP.
                        0034  6887         LOADDCS         1$,,0,<^X1A>                    ; LOAD TEST UCODE.
                        003B  6888         ERRLOOP                                         ; ERROR LOOP
                        003D  6889         FETCH           0                               ; EXECUTE SERIES OF MICRO
                        0042  6890         BRSTCLK         <^X19>                          ; INSTRUCTIONS WHICH WILL
                        0046  6891                                                         ; WRITE AN INVALID PTE INTO
                        0046  6892                                                         ; GROUP 0 AND THEN DO A BUS
                        0046  6893                                                         ; READ FUNCTION WHICH SHOULD
                        0046  6894                                                         ; RESULT IN A TB MISS UTRAP.
                        0046  6895         CMPVBUS         4$                              ; TEST VBUS SIGNALS
                        004B  6896         IFERROR         ,<<SAC><MSQ><UTR>>,<<DPM>>      ; INCORRECT VBUS DATA
                        0054  6897         CMPREGMSK       CSTRP,2$,,3$,,W,                ; SEE IF TRAP OCCURRED.
                        0060  6898         IFERROR         ,<<SAC>>,<<DPM>>                ; TB MISS UTRAP DIDN'T OCCUR.
                        0067  6899         SKIP                                            ; GO TO NEXT TEST
                        006E  6900
                        006E  6901 BEGIN_TEST_DATA
                        006E
                        006E       ;-------------------------------------------------------------------------
                        006E  6902
                        006E  6903 1$:
U 00, 7700,C800,0364,0300,0470,1801  006E  6904 ;% 00    NOP
U 01, 7700,3800,78FF,FFFF,8470,1802  0079  6908 ;% 01    LONLIT_[0E000000]                  ; TURN OFF THE CMI.
U 02, 7700,4800,5BE4,03D4,6870,1803  0084  6912 ;% 02    MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804  008F  6916 ;% 03    LONLIT_[01000000]
U 04, 7700,C800,5BE4,03D4,6070,1805  009A  6920 ;% 04    MEMSCR_R[LONLIT]
U 05, 7700,F800,7CFF,FFFF,8470,1806  00A5  6924 ;% 05    LONLIT_[06000000]                  ; TURN OFF THE CACHE.
U 06, 7700,C800,5BE4,03D4,6870,1807  00B0  6928 ;% 06    MEMSCAR_R[LONLIT]
U 07, 7700,3800,7F7F,FFFF,8470,1808  00BB  6932 ;% 07    LONLIT_[01000000]
U 08, 7700,C800,5BE4,03D4,6070,1809  00C6  6936 ;% 08    MEMSCR_R[LONLIT]
U 09, 7700,F800,7FFF,FFFF,8470,180A  00D1  6940 ;% 09    LONLIT_[00000000]                  ; SET MME.
U 0A, 7700,C800,5BE4,03D4,6870,180B  00DC  6944 ;% 0A    MEMSCAR_R[LONLIT]
U 0B, 7700,B800,7F7F,FFFF,8470,180C  00E7  6948 ;% 0B    LONLIT_[01000000]
U 0C, 7700,4800,5BE4,03D4,6070,180D  00F2  6952 ;% 0C    MEMSCR_R[LONLIT]
U 0D, 7700,8800,5BE6,03D8,0470,180E  00FD  6956 ;% 0D    D_R[ZERO]                          ; zero the D register
U 0E, 7700,D800,DB13,0300,4A70,180F  0108  6960 ;% 0E    VA_Q_D_D+ZLIT8[0]                  ; invalidate the TB at 0 &
U 0F, 7700,1800,DB12,0300,5360,1810  0113  6964 ;% 0F    VA_D_D+ZLIT8[0] INVALIDATE TB      ; 200.............
U 10, 7700,F800,7E7F,FFFF,8470,1811  011E  6968 ;% 10    LONLIT_[03000000]                  ; SET FORCE REPLACE GROUP 0
U 11, 7700,4800,5BE4,03D4,6870,1812  0129  6972 ;% 11    MEMSCAR_R[LONLIT]                  ; FORCE MISS TO GROUP 1.
U 12, 7700,7800,7AFF,FFFF,8470,1813  0134  6976 ;% 12    LONLIT_[0A000000]
U 13, 7700,C800,5BE4,03D4,6070,1814  013F  6980 ;% 13    MEMSCR_R[LONLIT]
U 14, 7700,7800,5555,7FFF,8470,1815  014A  6984 ;% 14    LONLIT_[55550000]                  ; MAKE VA=55550000 AND
U 15, 7700,C800,5BE4,03D4,4A70,1816  0155  6988 ;% 15    VA_R[LONLIT]                       ; PTE=00123450 (NOTE VALID BIT
U 16, 7700,B800,7FF6,E5D7,8470,1817  0160  6992 ;% 16    LONLIT_[00123450]                  ; OFF) A HIT IN GROUP 0.
U 17, 7700,C81B,5BE4,03D4,5070,1818  016B  6996 ;% 17    TB_R[LONLIT]
U 18, 7601,8800,0364,0200,0500,1819  0176  7000 ;% 18    BUS/READ,SIZE[LONG],STROBE.V.BUS   ; THIS READ SHOULD RESULT IN
U 19, 7300,C800,0364,0300,0470,181A  0181  7004 ;% 19    STOP.CLOCK                         ; A TB MISS UTRAP.
                                     018C  7008
                              002A   018C  7009 2$:     .WORD   ^X002A                     ; EXPECTED MICRO PC
                              3FFF   018E  7010 3$:     .WORD   ^X3FFF                     ; MASK FOR CMPREGMSK PSEUDO OP
                                07   0190  7011 4$:     .BYTE   ^X7
                                     0191  7012         VBUSDATA        <^X08>,1           ; MICRO VECTOR LINES
                                     0192  7013         VBUSDATA        <^X09>,0           ; ...
```

```
0193  7014          VBUSDATA        <^X0A>,1                ; ...
0194  7015          VBUSDATA        <^X0B>,0                ; ...
0195  7016          VBUSDATA        <^X0D>,0                ; UTRAP L ON VBUS
0196  7017          VBUSDATA        <^X1B>,1                ; UVCTR BRANCH SIGNAL
0197  7018          VBUSDATA        <^X1C>,1                ; DISABLE HI NEXT
0198  7019
0198  7020 END_TEST_DATA
0198
0198          ;-------------------------------------------------------------------------
0198  7021
0198  7022 ENDTEST
```

```
0030        .SBTTL  "TEST   02B     TB GROUP 1 VALID BIT AND TB MISS UTRAP TEST
0030  7024 ;*******************************************************************************
0030  7025 ;++
0030  7026 ;
0030  7027 ;       TB GROUP 1 VALID BIT AND TB MISS UTRAP TEST
0030  7028 ;
0030  7029 ; FUNCTIONAL DESCRIPTION:
0030  7030 ;       THIS IS A TEST OF THE TB MISS UTRAP LOGIC. A PTE IS WRITTEN
0030  7031 ;       INTO TB GROUP 1 WITH THE VALID BIT CLEAR. A REFERENCE IS THEN
0030  7032 ;       MADE WHICH SHOULD CALL THIS PTE OUT OF TB GROUP 1 RESULTING IN
0030  7033 ;       A TB MISS UTRAP.
0030  7034 ;
0030  7035 ; TEST ALGORITHM:
0030  7036 ;       1.      INVALIDATE BOTH TB GROUPS AT INDEX LOCATION 0.
0030  7037 ;       2.      SET THE TB GROUP 0 FORCE MISS BIT AND SET THE TB
0030  7038 ;               GROUP 1 FORCE REPLACEMENT BIT.
0030  7039 ;       3.      LOAD THE VA.
0030  7040 ;       4.      WRITE A PTE INTO TB GROUP 1 WHICH HAS THE VALID BIT
0030  7041 ;               CLEAR.
0030  7042 ;       5.      PERFORM A BUS READ BUS FUNCTION WHICH SHOULD RESULT
0030  7043 ;               IN A TB MISS UTRAP.
0030  7044 ;       6.      CHECK THE UTRAP VECTOR IN THE TRACE ADDRESS REGISTER.
0030  7045 ;
0030  7046 ; TEST PATTERNS:
0030  7047 ;       NONE
0030  7048 ;
0030  7049 ;
0030  7050 ; LOGIC DESCRIPTION:
0030  7051 ;       THIS TESTS LOGIC IN THE UTR GATE ARRAY WHICH GENERATES THE
0030  7052 ;       UTRAP CONDITION AND LOGIC IN THE MICRO SEQUENCER WHICH CAUSES
0030  7053 ;       THE ACTUAL UTRAP TO OCCUR.
0030  7054 ;
0030  7055 ;
0030  7056 ; ASSUMPTIONS:
0030  7057 ;       THIS TEST ASSUMES THE DPM AND WBUS ARE OPERATIONAL
0030  7058 ;
0030  7059 ;
0030  7060 ; ERROR DESCRIPTION:
0030  7061 ;       EXPECTED MICRO PC (THE UTRAP VECTOR)
0030  7062 ;       GOT MICRO PC
0030  7063 ;
0030  7064 ;--
0030  7065 ;*******************************************************************************
0030  7066 ;///////////////////////////////////////////////////////////////////////////////
0030  7067
0030  7068         INITIALIZE
0032  7069         EXPUTRAP                                        ; SET UP FOR UCODE TRAP.
0034  7070         LOADDCS         1$,,0,<^X1A>                    ; LOAD TEST UCODE.
003B  7071         ERRLOOP                                         ; ERROR LOOP
003D  7072         FETCH           0                               ; EXECUTE SERIES OF MICRO
0042  7073         BRSTCLK         <^X19>                          ; INSTRUCTIONS WHICH WILL
0046  7074                                                         ; WRITE AN INVALID PTE INTO
0046  7075                                                         ; GROUP 1 AND THEN DO A BUS
0046  7076                                                         ; READ FUNCTION WHICH SHOULD
0046  7077                                                         ; RESULT IN A TB MISS UTRAP.
```

```
                        0046  7078          CMPREGMSK       CSTRP,2$,,3$,,W,          ; SEE IF TRAP OCCURRED.
                        0052  7079          IFERROR         ,<<UTR>>,                 ; TB MISS UTRAP DIDN'T OCCUR.
                        0058  7080          SKIP
                        005F  7081
                        005F  7082 BEGIN_TEST_DATA
                        005F
                        005F        ;-------------------------------------------------------------------------
                        005F  7083
                        005F  7084 1$:
U 00. 7700.C800.0364.0300.0470.1801   005F  7085 ;* 00    NOP
U 01. 7700.3800.78FF.FFFF.8470.1802   006A  7089 ;* 01    LONLIT_[0E000000]                 ; TURN OFF THE CMI.
U 02. 7700.4800.5BE4.03D4.6870.1803   0075  7093 ;* 02    MEMSCAR_R[LONLIT]
U 03. 7700.3800.7F7F.FFFF.8470.1804   0080  7097 ;* 03    LONLIT_[01000000]
U 04. 7700.C800.5BE4.03D4.6070.1805   008B  7101 ;* 04    MEMSCR_R[LONLIT]
U 05. 7700.F800.7CFF.FFFF.8470.1806   0096  7105 ;* 05    LONLIT_[06000000]                 ; TURN OFF THE CACHE.
U 06. 7700.C800.5BE4.03D4.6870.1807   00A1  7109 ;* 06    MEMSCAR_R[LONLIT]
U 07. 7700.3800.7F7F.FFFF.8470.1808   00AC  7113 ;* 07    LONLIT_[01000000]
U 08. 7700.C800.5BE4.03D4.6070.1809   00B7  7117 ;* 08    MEMSCR_R[LONLIT]
U 09. 7700.F800.7FFF.FFFF.8470.180A   00C2  7121 ;* 09    LONLIT_[00000000]                 ; SET MME.
U 0A. 7700.C800.5BE4.03D4.6870.180B   00CD  7125 ;* 0A    MEMSCAR_R[LONLIT]
U 0B. 7700.B800.7F7F.FFFF.8470.180C   00D8  7129 ;* 0B    LONLIT_[01000000]
U 0C. 7700.4800.5BE4.03D4.6070.180D   00E3  7133 ;* 0C    MEMSCR_R[LONLIT]
U 0D. 7700.8800.5BE6.03D8.0470.180E   00EE  7137 ;* 0D    D_R[ZERO]                         ; zero the D register
U 0E. 7700.D800.DB13.0300.4A70.180F   00F9  7141 ;* 0E    VA_Q_D_D+ZLIT8[0]                 ; invalidate the TB at 0 &
U 0F. 7700.1800.DB12.0300.5360.1810   0104  7145 ;* 0F    VA_D_D+ZLIT8[0] INVALIDATE TB     ; 200.............
U 10. 7700.F800.7E7F.FFFF.8470.1811   010F  7149 ;* 10    LONLIT_[03000000]                 ; SET FORCE REPLACE GROUP 1
U 11. 7700.4800.5BE4.03D4.6870.1812   011A  7153 ;* 11    MEMSCAR_R[LONLIT]                 ; FORCE MISS TO GROUP 0.
U 12. 7700.3800.797F.FFFF.8470.1813   0125  7157 ;* 12    LONLIT_[0D000000]
U 13. 7700.C800.5BE4.03D4.6070.1814   0130  7161 ;* 13    MEMSCR_R[LONLIT]
U 14. 7700.7800.5555.7FFF.8470.1815   013B  7165 ;* 14    LONLIT_[55550000]                 ; MAKE VA=55550000 AND
U 15. 7700.C800.5BE4.03D4.4A70.1816   0146  7169 ;* 15    VA_R[LONLIT]                      ; PTE=00123450 (NOTE VALID BIT
U 16. 7700.B800.7FF6.E5D7.8470.1817   0151  7173 ;* 16    LONLIT_[00123450]                 ; OFF) A HIT IN GROUP 1.
U 17. 7700.C81B.5BE4.03D4.5070.1818   015C  7177 ;* 17    TB_R[LONLIT]
U 18. 7701.8800.0364.0200.0500.1819   0167  7181 ;* 18    BUS/READ,SIZE[LONG]               ; THIS READ SHOULD RESULT IN
U 19. 7300.C800.0364.0300.0470.181A   0172  7185 ;* 19    STOP.CLOCK                        ; A TB MISS UTRAP.
                                      017D  7189
                              002A    017D  7190 2$:      .WORD   ^X002A                    ; EXPECTED MICRO PC OF
                              3FFF    017F  7191 3$:      .WORD   ^X3FFF                    ; MASK FOR CMPREGMSK PSEUDO OP
                                      0181  7192                                            ; TB MISS MICRO TRAP VECTOR.
                                      0181  7193 END_TEST_DATA
                                      0181
                                      0181        ;-------------------------------------------------------------------------
                                      0181  7194
                                      0181  7195 ENDTEST
```

```
0020          .SBTTL  "TEST   02C     TB WRITE PULSE LOGIC TEST 1
0020  7197 ;*******************************************************************************
0020  7198 ;++
0020  7199 ;
0020  7200 ;       TB WRITE PULSE LOGIC TEST 1
0020  7201 ;
0020  7202 ; FUNCTIONAL DESCRIPTION:
0020  7203 ;       THIS IS A TEST OF THE TRANSLATION BUFFER WRITE LOGIC. VA=2AAA0000
0020  7204 ;       AND PTE=00555550 ARE MADE A HIT IN GROUP 1, AND VA=55550000 AND
0020  7205 ;       PTE=00333330 ARE MADE A HIT IN GROUP 0.
0020  7206 ;       THEN VA=33330000 AND PTE=00999990 ARE MADE A HIT IN THE TB WITH
0020  7207 ;       THE GROUP 0 FORCE REPLACE BIT SET. THIS SHOULD LEAVE VA=2AAA0000
0020  7208 ;       AND PTE=00555550 STILL A HIT IN GROUP 1.
0020  7209 ;
0020  7210 ;
0020  7211 ; TEST ALGORITHM:
0020  7212 ;       1.  CREATE A TEST LOOP OF 3 TO SELECT TEST ADDRESSES
0020  7213 ;       2.  LOAD TEST ADDRESS INTO RDM WD REGISTER
0020  7214 ;       3.  EXECUTE DCS PROGRAM
0020  7215 ;               a.  INVALIDATE TB
0020  7216 ;               b.  SET MISS GROUP 0 REPLACE GROUP 1
0020  7217 ;               c.  LOAD VA WITH 2AAA0000
0020  7218 ;               d.  WRITE 00555550 INTO TB GROUP 1
0020  7219 ;               e.  SET MISS GROUP 1 REPLACE GROUP 0
0020  7220 ;               f.  LOAD VA WITH 55550000
0020  7221 ;               g.  WRITE 00333330 INTO TB GROUP 0
0020  7222 ;               h.  ENABLE BOTH TB GROUPS BUT FORCE REPLACE GROUP 0
0020  7223 ;               i.  LOAD VA WITH 33330000
0020  7224 ;               j.  WRITE 00999990 INTO TB
0020  7225 ;               k.  LOAD VA WITH TEST ADDRESS FROM RDM
0020  7226 ;               l.  READ TB TO RDM WH REGISTER
0020  7227 ;       4.  COMPARE WH REGISTER TO EXPECTED PTE
0020  7228 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES.
0020  7229 ;
0020  7230 ;
0020  7231 ; TEST PATTERNS:
0020  7232 ;       NONE
0020  7233 ;
0020  7234 ;
0020  7235 ; LOGIC DESCRIPTION:
0020  7236 ;       THIS TESTS LOGIC IN THE ADK GATE ARRAY.
0020  7237 ;
0020  7238 ;
0020  7239 ; ASSUMPTIONS:
0020  7240 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
0020  7241 ;
0020  7242 ;
0020  7243 ; ERROR DESCRIPTION:
0020  7244 ;       EXPECTED PTE
0020  7245 ;       GOT PTE
0020  7246 ;
0020  7247 ;--
0020  7248 ;*******************************************************************************
0020  7249 ;///////////////////////////////////////////////////////////////////////////////
0020  7250
```

```
                     0020  7251          INITIALIZE
                     0022  7252          LOOP              I,1,3                       ;TEST LOOP OF 3
                     0029  7253          LOADREG           WDREG,1$,I,L                ;LOAD TEST ADDRESS INTO WD REG
                     0031  7254          ERRLOOP                                       ;ERROR LOOP POINT
                     0033  7255          FETCH             0                           ;EXECUTE THE UCODE
                     0038  7256          BRSTCLK           <^X1E>
                     003C  7257          CMPREGMSK         WHREG,2$,I,3$,I,L,          ;TEST FOR CORRECT PTE
                     0048  7258          IFERROR           ,<ADK>,                     ;INCORRECT PTE
                     004E  7259          ENDLOOP           I                           ;END TEST LOOP
                     0051  7260          SKIP                                          ;GO TO NEXT TEST
                     0058  7261
                     0058  7262 BEGIN_TEST_DATA
                     0058
                     0058               ;-------------------------------------------------------------------------
                     0058  7263
2AAA0000             0058  7264 1$:      .LONG    ^X2AAA0000                           ;TEST ADDRESS TABLE
33330000             005C  7265          .LONG    ^X33330000
55550000             0060  7266          .LONG    ^X55550000
                     0064  7267
00555550             0064  7268 2$:      .LONG    ^X00555550                           ;COMPARE DATA
00999990             0068  7269          .LONG    ^X00999990
00000000             006C  7270          .LONG    ^X00000000
                     0070  7271
00FFFFF8             0070  7272 3$:      .LONG    ^X00FFFFF8                           ;MASK TABLE
00FFFFF8             0074  7273          .LONG    ^X00FFFFF8
00000100             0078  7274          .LONG    ^X00000100
                     007C  7275
                     007C  7276 BEGIN_CPU_DATA
                     0002  7277
                     0002  7278 DCS_DATA
                     0001  7279
U 00. 7700.C800.0364.0300.0470.1801  0001  7280 ;% 00   NOP
U 01. 7700.8800.5BE6.03D8.0470.1802  000C  7284 ;% 01   D_R[ZERO]                       ; zero the D register
U 02. 7700.D800.DB13.0300.4A70.1803  0017  7288 ;% 02   VA_Q_D_D+ZLIT8[0]              ; invalidate the TB at 0 &
U 03. 7700.1800.DB12.0300.5360.1804  0022  7292 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB ; 200.............
U 04. 7700.F800.7E7F.FFFF.8470.1805  002D  7296 ;% 04   LONLIT_[03000000]              ; SET FORCE MISS GROUP 0 AND
U 05. 7700.4800.5BE4.03D4.6870.1806  0038  7300 ;% 05   MEMSCAR_R[LONLIT]             ; FORCE REPLACE GROUP 1.
U 06. 7700.3800.797F.FFFF.8470.1807  0043  7304 ;% 06   LONLIT_[0D000000]
U 07. 7700.4800.5BE4.03D4.6070.1808  004E  7308 ;% 07   MEMSCR_R[LONLIT]
U 08. 7700.B800.6AAA.FFFF.8470.1809  0059  7312 ;% 08   LONLIT_[2AAA0000]              ; LOAD THE VA WITH 2AAA0000.
U 09. 7700.4800.5BE4.03D4.4A70.180A  0064  7316 ;% 09   VA_R[LONLIT]
U 0A. 7700.7800.7FD5.5557.8470.180B  006F  7320 ;% 0A   LONLIT_[00555550]              ; WRITE VA=2AAA0000 AND
U 0B. 7700.C81B.5BE4.03D4.5070.180C  007A  7324 ;% 0B   TB_R[LONLIT]                  ; PTE=00555550 INTO TB GROUP 1.
U 0C. 7700.7800.7E7F.FFFF.8470.180D  0085  7328 ;% 0C   LONLIT_[03000000]              ; SET FORCE MISS GROUP 1 AND
U 0D. 7700.C800.5BE4.03D4.6870.180E  0090  7332 ;% 0D   MEMSCAR_R[LONLIT]             ; FORCE REPLACE GROUP 0.
U 0E. 7700.F800.7AFF.FFFF.8470.180F  009B  7336 ;% 0E   LONLIT_[0A000000]
U 0F. 7700.4800.5BE4.03D4.6070.1810  00A6  7340 ;% 0F   MEMSCR_R[LONLIT]
U 10. 7700.F800.5555.7FFF.8470.1811  00B1  7344 ;% 10   LONLIT_[55550000]              ; LOAD THE VA WITH 55550000.
U 11. 7700.4800.5BE4.03D4.4A70.1812  00BC  7348 ;% 11   VA_R[LONLIT]
U 12. 7700.7800.7FE6.6667.8470.1813  00C7  7352 ;% 12   LONLIT_[00333330]              ; WRITE VA=55550000 AND
U 13. 7700.C81B.5BE4.03D4.5070.1814  00D2  7356 ;% 13   TB_R[LONLIT]                  ; PTE=00333330 INTO TB GROUP 0.
U 14. 7700.7800.7E7F.FFFF.8470.1815  00DD  7360 ;% 14   LONLIT_[03000000]              ; ENABLE BOTH TB GROUPS BUT
U 15. 7700.C800.5BE4.03D4.6870.1816  00E8  7364 ;% 15   MEMSCAR_R[LONLIT]             ; FORCE REPLACE GROUP 0.
U 16. 7700.B800.7BFF.FFFF.8470.1817  00F3  7368 ;% 16   LONLIT_[08000000]
U 17. 7700.C800.5BE4.03D4.6070.1818  00FE  7372 ;% 17   MEMSCR_R[LONLIT]
```

```
U 18, 7700,7800,6666,7FFF,8470,1819  0109  7376 ;% 18   LONLIT_[33330000]                    ; LOAD THE VA WITH 33330000.
U 19, 7700,C800,5BE4,03D4,4A70,181A  0114  7380 ;% 19   VA_R[LONLIT]
U 1A, 7700,F800,7FB3,3337,8470,181B  011F  7384 ;% 1A   LONLIT_[00999990]                    ; WRITE VA=33330000 WITH
U 1B, 7700,481B,5BE4,03D4,5070,181C  012A  7388 ;% 1B   TB_R[LONLIT]                        ; PTE=00999990 INTO THE TB. THIS
U 1C, 5700,4800,0364,0300,4A70,181D  0135  7392 ;% 1C   VA_RDM                             ; LOAD THE VA WITH TEST ADDRESS.
U 1D, 6701,081F,5924,0200,05F0,181E  0140  7396 ;% 1D   RDM_TB                             ; READ OUT OF THE TB AT THIS
U 1E, 7300,C800,0364,0300,0470,181F  014B  7400 ;% 1E   NOP,STOP.CLOCK                     ; ADDRESS.
                                     0156  7404
                                     0156  7405 END_CPU_DATA
                                     007C  7406
                                     007C  7407 END_TEST_DATA
                                     007C
                                     007C       ;-------------------------------------------------------------------------
                                     007C  7408
                                     007C  7409 ENDTEST
```

```
0020        .SBTTL  "TEST   02D     TB WRITE PULSE LOGIC TEST 2
0020  7411 ;*******************************************************************************
0020  7412 ;++
0020  7413 ;
0020  7414 ;       TB WRITE PULSE LOGIC TEST 2
0020  7415 ;
0020  7416 ; FUNCTIONAL DESCRIPTION:
0020  7417 ;       THIS IS A TEST OF THE TRANSLATION BUFFER WRITE LOGIC. VA=2AAA0000
0020  7418 ;       AND PTE=00555500 ARE MADE A HIT IN GROUP 0, AND VA=55551000 AND
0020  7419 ;       PTE=00333330 ARE MADE A HIT IN GROUP 1.
0020  7420 ;       THEN VA=33330000 AND PTE=00999990 ARE MADE A HIT IN THE TB WITH
0020  7421 ;       THE GROUP 1 FORCE REPLACE BIT SET. THIS SHOULD LEAVE VA=2AAA0000
0020  7422 ;       AND PTE=00555550 STILL A HIT IN GROUP 0.
0020  7423 ;
0020  7424 ;
0020  7425 ; TEST ALGORITHM:
0020  7426 ;       1.  CREATE A TEST LOOP OF 3 TO SELECT TEST ADDRESSES
0020  7427 ;       2.  LOAD A TEST ADDRESS INTO THE RDM WD REGISTER
0020  7428 ;       3.  EXECUTE DCS PROGRAM
0020  7429 ;               a.  INVALIDATE THE TB
0020  7430 ;               b.  SET MISS GROUP 1 AND REPLACE GROUP 0
0020  7431 ;               c.  LOAD VA WITH 2AAA0000
0020  7432 ;               d.  WRITE 00555550 INTO GROUP 0
0020  7433 ;               e.  SET MISS GROUP 0 AND REPLACE GROUP 1
0020  7434 ;               f.  LOAD VA WITH 55550000
0020  7435 ;               g.  WRITE 00333330 INTO GROUP 1
0020  7436 ;               h.  ENABLE BOTH TB GROUPS BUT FORCE REPLACE GROUP 1
0020  7437 ;               i.  LOAD VA WITH 33330000
0020  7438 ;               j.  WRITE 00999990 INTO TB
0020  7439 ;               k.  LOAD VA WITH TEST ADDRESS FROM RDM WD REGISTER
0020  7440 ;               l.  READ TB TO RDM WH REGISTER
0020  7441 ;       4.  TEST WH REGISTER FOR CORRECT PTE
0020  7442 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0020  7443 ;
0020  7444 ;
0020  7445 ; TEST PATTERNS:
0020  7446 ;       NONE
0020  7447 ;
0020  7448 ;
0020  7449 ; LOGIC DESCRIPTION:
0020  7450 ;       THIS TESTS LOGIC IN THE ADK GATE ARRAY.
0020  7451 ;
0020  7452 ; ASSUMPTIONS:
0020  7453 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
0020  7454 ;
0020  7455 ;
0020  7456 ; ERROR DESCRIPTION:
0020  7457 ;       EXPECTED PTE
0020  7458 ;       GOT PTE
0020  7459 ;
0020  7460 ;--
0020  7461 ;*******************************************************************************
0020  7462 ;///////////////////////////////////////////////////////////////////////////////
0020  7463
0020  7464         INITIALIZE
```

```
                        0022  7465          LOOP            I,1,3                   ;CREATE A TEST LOOP OF 3
                        0029  7466          LOADREG         WDREG,1$,I,L            ;LOAD TEST ADDRESS INTO WD REG
                        0031  7467          ERRLOOP                                 ;ERROR LOOP POINT
                        0033  7468          FETCH           0                       ;EXECUTE THE UCODE.
                        0038  7469          BRSTCLK         <^X1E>
                        003C  7470          CMPREGMSK       WHREG,2$,I,3$,I,L,      ;TEST FOR CORRECT PTE
                        0048  7471          IFERROR         ,<<ADK>>,               ;INCORRECT PTE
                        004E  7472          ENDLOOP         I                       ;END TEST LOOP
                        0051  7473          SKIP                                    ;GO TO NEXT TEST
                        0058  7474
                        0058  7475 BEGIN_TEST_DATA
                        0058
                        0058              ;-------------------------------------------------------------------------
                        0058  7476
2AAA0000                0058  7477 1$:      .LONG   ^X2AAA0000                      ;TEST ADDRESS TABLE
33330000                005C  7478          .LONG   ^X33330000
55550000                0060  7479          .LONG   ^X55550000
                        0064  7480
00555550                0064  7481 2$:      .LONG   ^X00555550                      ;COMPARE DATA TABLE
00999990                0068  7482          .LONG   ^X00999990
00000000                006C  7483          .LONG   ^X00000000
                        0070  7484
00FFFFF8                0070  7485 3$:      .LONG   ^X00FFFFF8                      ;MASK FOR CMPREGMSK PSEUDO-OP
00FFFFF8                0074  7486          .LONG   ^X00FFFFF8
00000100                0078  7487          .LONG   ^X00000100
                        007C  7488
                        007C  7489 BEGIN_CPU_DATA
                        0002  7490
                        0002  7491 DCS_DATA
                        0001  7492
U 00, 7700,C800,0364,0300,0470,1801   0001  7493 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802   000C  7497 ;% 01   D_R[ZERO]                                ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803   0017  7501 ;% 02   VA_Q_D_D+ZLIT8[0]                       ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804   0022  7505 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB          ; 200.............
U 04, 7700,F800,7E7F,FFFF,8470,1805   002D  7509 ;% 04   LONLIT_[03000000]                       ; SET FORCE MISS GROUP 1 AND
U 05, 7700,4800,5BE4,03D4,6870,1806   0038  7513 ;% 05   MEMSCAR_R[LONLIT]                      ; FORCE REPLACE GROUP 0.
U 06, 7700,7800,7AFF,FFFF,8470,1807   0043  7517 ;% 06   LONLIT_[0A000000]
U 07, 7700,4800,5BE4,03D4,6070,1808   004E  7521 ;% 07   MEMSCR_R[LONLIT]
U 08, 7700,B800,6AAA,FFFF,8470,1809   0059  7525 ;% 08   LONLIT_[2AAA0000]                       ; LOAD THE VA WITH 2AAA0000.
U 09, 7700,4800,5BE4,03D4,4A70,180A   0064  7529 ;% 09   VA_R[LONLIT]
U 0A, 7700,7800,7FD5,5557,8470,180B   006F  7533 ;% 0A   LONLIT_[00555550]                       ; WRITE VA=2AAA0000 AND
U 0B, 7700,C81B,5BE4,03D4,5070,180C   007A  7537 ;% 0B   TB_R[LONLIT]                           ; PTE=00555550 INTO TB GROUP 0.
U 0C, 7700,7800,7E7F,FFFF,8470,180D   0085  7541 ;% 0C   LONLIT_[03000000]                       ; SET FORCE MISS GROUP 0 AND
U 0D, 7700,C800,5BE4,03D4,6870,180E   0090  7545 ;% 0D   MEMSCAR_R[LONLIT]                      ; FORCE REPLACE GROUP 1.
U 0E, 7700,B800,797F,FFFF,8470,180F   009B  7549 ;% 0E   LONLIT_[0D000000]
U 0F, 7700,4800,5BE4,03D4,6070,1810   00A6  7553 ;% 0F   MEMSCR_R[LONLIT]
U 10, 7700,F800,5555,7FFF,8470,1811   00B1  7557 ;% 10   LONLIT_[55550000]                       ; LOAD THE VA WITH 55550000.
U 11, 7700,4800,5BE4,03D4,4A70,1812   00BC  7561 ;% 11   VA_R[LONLIT]
U 12, 7700,7800,7FE6,6667,8470,1813   00C7  7565 ;% 12   LONLIT_[00333330]                       ; WRITE VA=55550000 AND
U 13, 7700,C81B,5BE4,03D4,5070,1814   00D2  7569 ;% 13   TB_R[LONLIT]                           ; PTE=00333330 INTO TB GROUP 1.
U 14, 7700,7800,7E7F,FFFF,8470,1815   00DD  7573 ;% 14   LONLIT_[03000000]                       ; ENABLE BOTH TB GROUPS BUT
U 15, 7700,C800,5BE4,03D4,6870,1816   00E8  7577 ;% 15   MEMSCAR_R[LONLIT]                      ; FORCE REPLACE GROUP 1.
U 16, 7700,F800,79FF,FFFF,8470,1817   00F3  7581 ;% 16   LONLIT_[0C000000]
U 17, 7700,C800,5BE4,03D4,6070,1818   00FE  7585 ;% 17   MEMSCR_R[LONLIT]
U 18, 7700,7800,6666,7FFF,8470,1819   0109  7589 ;% 18   LONLIT_[33330000]                       ; LOAD THE VA WITH 33330000.
```

```
U 19, 7700,C800,5BE4,03D4,4A70,181A  0114  7593 ;% 19   VA_R[LONLIT]
U 1A, 7700,F800,7FB3,3337,8470,181B  011F  7597 ;% 1A   LONLIT_[00999990]                    ; WRITE VA=33330000 WITH
U 1B, 7700,481B,5BE4,03D4,5070,181C  012A  7601 ;% 1B   TB_R[LONLIT]                        ; PTE=00999990 INTO THE TB. THIS
U 1C, 5700,4800,0364,0300,4A70,181D  0135  7605 ;% 1C   VA_RDM                             ; LOAD THE VA WITH TEST ADDRESS.
U 1D, 6701,081F,5924,0200,05F0,181E  0140  7609 ;% 1D   RDM_TB                             ; READ OUT OF THE TB AT THIS
U 1E, 7300,C800,0364,0300,0470,181F  014B  7613 ;% 1E   NOP,STOP.CLOCK                     ; ADDRESS.
                                     0156  7617
                                     0156  7618 END_CPU_DATA
                                     007C  7619
                                     007C  7620 END_TEST_DATA
                                     007C
                                     007C       ;-------------------------------------------------------------------------
                                     007C  7621
                                     007C  7622 ENDTEST
```

```
0020          .SBTTL  "TEST   02E     TB WRITE PULSE LOGIC TEST 3
0020  7624 ;*******************************************************************************
0020  7625 ;++
0020  7626 ;
0020  7627 ;       TB WRITE PULSE LOGIC TEST 3
0020  7628 ;
0020  7629 ;
0020  7630 ; FUNCTIONAL DESCRIPTION
0020  7631 ;       THIS IS A TEST OF THE TB WRITE PULSE LOGIC WHICH OVER RIDES THE
0020  7632 ;       FORCE REPLACE BIT FOR GROUP 0 WHEN A WRITE IS DONE INTO THE TB
0020  7633 ;       AT AN ADDRESS WHICH IS A HIT IN GROUP 1.
0020  7634 ;
0020  7635 ; TEST ALGORITHM
0020  7636 ;       1.  CREATE A TEST LOOP OF 2 TO SELECT TEST ADDRESSES
0020  7637 ;       2.  LOAD TEST ADDRESS INTO RDM WD REGISTER
0020  7638 ;       3.  EXECUTE DCS PROGRAM
0020  7639 ;               a.  INVALIDATE TB
0020  7640 ;               b.  SET MISS GROUP 0 AND REPLACE GROUP 1
0020  7641 ;               c.  LOAD VA WITH 2AAA0000
0020  7642 ;               d.  WRITE 00555550 INTO TB GROUP 1
0020  7643 ;               e.  SET MISS GROUP 1 AND REPLACE GROUP 0
0020  7644 ;               f.  LOAD VA WITH 55550000
0020  7645 ;               g.  WRITE 00333330 INTO TB GROUP 0
0020  7646 ;               h.  ENABLE BOTH TB GROUPS BUT FORCE REPLACE GROUP 0
0020  7647 ;               i.  LOAD VA WITH 2AAA0000
0020  7648 ;               j.  WRITE 00999990 INTO TB
0020  7649 ;               k.  LOAD VA WITH TEST ADDRESS FROM RDM
0020  7650 ;               l.  MOVE CONTENTS OF TB TO RDM WH REGISTER
0020  7651 ;       4.  TEST WH REGISTER FOR CORRECT PTE
0020  7652 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINGING TEST ADDRESSES
0020  7653 ;
0020  7654 ;
0020  7655 ; LOGIC DESCRIPTION
0020  7656 ;       THIS TESTS LOGIC IN THE ADK GATE ARRAY.
0020  7657 ;
0020  7658 ; ASSUMPTIONS
0020  7659 ;       THIS TEST ASSUMES THE DPM AND WB ARE OPERATIONAL
0020  7660 ;
0020  7661 ; ERROR DESCRIPTION
0020  7662 ;       EXPECTED PTE
0020  7663 ;       GOT PTE
0020  7664 ;
0020  7665 ;--
0020  7666 ;*******************************************************************************
0020  7667 ;///////////////////////////////////////////////////////////////////////////////
0020  7668
0020  7669          INITIALIZE
0022  7670          LOOP            I,1,2                   ;CREATE TEST LOOP OF 2
0029  7671          LOADREG         WDREG,1$,I,L            ;LOAD TEST ADDRESS INTO WD REG
0031  7672          ERRLOOP                                 ;ERROR LOOP POINT
0033  7673          FETCH           0                       ;EXECUTE THE UCODE.
0038  7674          BRSTCLK         <^X1E>
003C  7675          CMPREGMSK       WHREG,2$,I,3$,,L,       ;TEST FOR CORRECT PTE
0048  7676          IFERROR         ,<<ADK>>,               ;INCORRECT PTE
004E  7677          ENDLOOP         I                       ;END TEST LOOP
```

```
                                0051  7678          SKIP                                                ;GO TO NEXT TEST
                                0058  7679
                                0058  7680 BEGIN_TEST_DATA
                                0058
                                0058          ;-------------------------------------------------------------------------
                                0058  7681
                     2AAA0000   0058  7682 1$:      .LONG    ^X2AAA0000                                 ;TEST ADDRESS TABLE
                     55550000   005C  7683          .LONG    ^X55550000
                                0060  7684
                     00999990   0060  7685 2$:      .LONG    ^X00999990                                 ;COMPARE DATA
                     00333330   0064  7686          .LONG    ^X00333330
                                0068  7687
                     00FFFFF8   0068  7688 3$:      .LONG    ^X00FFFFF8                                 ;MASK FOR CMPREGMSK PSEUDO-OP
                                006C  7689
                                006C  7690 BEGIN_CPU_DATA
                                0002  7691
                                0002  7692 DCS_DATA
                                0001  7693
U 00. 7700,C800,0364,0300,0470,1801   0001  7694 ;* 00    NOP
U 01. 7700,8800,5BE6,03D8,0470,1802   000C  7698 ;* 01    D_R[ZERO]                                  ; zero the D register
U 02. 7700,D800,DB13,0300,4A70,1803   0017  7702 ;* 02    VA_Q_D_D+ZLIT8[0]                          ; invalidate the TB at 0 &
U 03. 7700,1800,DB12,0300,5360,1804   0022  7706 ;* 03    VA_D_D+ZLIT8[0] INVALIDATE TB              ; 200.............
U 04. 7700,F800,7E7F,FFFF,8470,1805   002D  7710 ;* 04    LONLIT_[03000000]                          ; SET FORCE MISS GROUP 0 AND
U 05. 7700,4800,5BE4,03D4,6870,1806   0038  7714 ;* 05    MEMSCAR_R[LONLIT]                          ; FORCE REPLACE GROUP 1.
U 06. 7700,3800,797F,FFFF,8470,1807   0043  7718 ;* 06    LONLIT_[0D000000]
U 07. 7700,4800,5BE4,03D4,6070,1808   004E  7722 ;* 07    MEMSCR_R[LONLIT]
U 08. 7700,B800,6AAA,FFFF,8470,1809   0059  7726 ;* 08    LONLIT_[2AAA0000]                          ; LOAD THE VA WITH 2AAA0000.
U 09. 7700,4800,5BE4,03D4,4A70,180A   0064  7730 ;* 09    VA_R[LONLIT]
U 0A. 7700,7800,7FD5,5557,8470,180B   006F  7734 ;* 0A    LONLIT_[00555550]                          ; WRITE VA=2AAA0000 AND
U 0B. 7700,C81B,5BE4,03D4,5070,180C   007A  7738 ;* 0B    TB_R[LONLIT]                               ; PTE=00555550 INTO TB GROUP 1.
U 0C. 7700,7800,7E7F,FFFF,8470,180D   0085  7742 ;* 0C    LONLIT_[03000000]                          ; SET FORCE MISS GROUP 1 AND
U 0D. 7700,C800,5BE4,03D4,6870,180E   0090  7746 ;* 0D    MEMSCAR_R[LONLIT]                          ; FORCE REPLACE GROUP 0.
U 0E. 7700,F800,7AFF,FFFF,8470,180F   009B  7750 ;* 0E    LONLIT_[0A000000]
U 0F. 7700,4800,5BE4,03D4,6070,1810   00A6  7754 ;* 0F    MEMSCR_R[LONLIT]
U 10. 7700,F800,5555,7FFF,8470,1811   00B1  7758 ;* 10    LONLIT_[55550000]                          ; LOAD THE VA WITH 55550000.
U 11. 7700,4800,5BE4,03D4,4A70,1812   00BC  7762 ;* 11    VA_R[LONLIT]
U 12. 7700,7800,7FE6,6667,8470,1813   00C7  7766 ;* 12    LONLIT_[00333330]                          ; WRITE VA=55550000 AND
U 13. 7700,C81B,5BE4,03D4,5070,1814   00D2  7770 ;* 13    TB_R[LONLIT]                               ; PTE=00333330 INTO TB GROUP 0.
U 14. 7700,7800,7E7F,FFFF,8470,1815   00DD  7774 ;* 14    LONLIT_[03000000]                          ; ENABLE BOTH TB GROUPS BUT
U 15. 7700,C800,5BE4,03D4,6870,1816   00E8  7778 ;* 15    MEMSCAR_R[LONLIT]                          ; FORCE REPLACE GROUP 0.
U 16. 7700,B800,7BFF,FFFF,8470,1817   00F3  7782 ;* 16    LONLIT_[08000000]
U 17. 7700,C800,5BE4,03D4,6070,1818   00FE  7786 ;* 17    MEMSCR_R[LONLIT]
U 18. 7700,3800,6AAA,FFFF,8470,1819   0109  7790 ;* 18    LONLIT_[2AAA0000]                          ; LOAD THE VA WITH 2AAA0000.
U 19. 7700,C800,5BE4,03D4,4A70,181A   0114  7794 ;* 19    VA_R[LONLIT]
U 1A. 7700,F800,7FB3,3337,8470,181B   011F  7798 ;* 1A    LONLIT_[00999990]                          ; WRITE VA=2AAA0000 WITH
U 1B. 7700,481B,5BE4,03D4,5070,181C   012A  7802 ;* 1B    TB_R[LONLIT]                               ; PTE=00999990 INTO THE TB.
U 1C. 5700,4800,0364,0300,4A70,181D   0135  7806 ;* 1C    VA_RDM                                     ; LOAD THE VA WITH TEST ADDRESS.
U 1D. 6701,081F,5924,0200,05F0,181E   0140  7810 ;* 1D    RDM_TB                                     ; READ OUT OF THE TB AT THIS
U 1E. 7300,C800,0364,0300,0470,181F   014B  7814 ;* 1E    NOP,STOP.CLOCK                             ; ADDRESS.
                                0156  7818
                                0156  7819 END_CPU_DATA
                                006C  7820
                                006C  7821 END_TEST_DATA
                                006C
                                006C          ;-------------------------------------------------------------------------
```

```
                006C  7822
                006C  7823 ENDTEST
```

```
0020          .SBTTL  "TEST   02F     TB WRITE PULSE LOGIC TEST 4
0020   7825 ;*****************************************************************************
0020   7826 ;++
0020   7827 ;
0020   7828 ;       TB WRITE PULSE LOGIC TEST 4
0020   7829 ;
0020   7830 ;
0020   7831 ; FUNCTIONAL DESCRIPTION
0020   7832 ;       THIS IS A TEST OF THE TB WRITE PULSE LOGIC WHICH OVER RIDES THE
0020   7833 ;       FORCE REPLACE BIT FOR GROUP 1 WHEN A WRITE IS DONE INTO THE TB
0020   7834 ;       AT AN ADDRESS WHICH IS A HIT IN GROUP 0.
0020   7835 ;
0020   7836 ; TEST ALGORITHM
0020   7837 ;       1.  CREATE A TEST LOOP OF 2 TO SELECT TEST ADDRESSES
0020   7838 ;       2.  LOAD TEST ADDRESS INTO RDM WD REGISTER
0020   7839 ;       3.  EXECUTE DCS PROGRAM
0020   7840 ;               a.  INVALIDATE TB
0020   7841 ;               b.  SET MISS GROUP 1 AND REPLACE GROUP 0
0020   7842 ;               c.  LOAD VA WITH 2AAA0000
0020   7843 ;               d.  WRITE 00555550 INTO TB GROUP 0
0020   7844 ;               e.  SET MISS GROUP 0 AND REPLACE GROUP 1
0020   7845 ;               f.  LOAD VA WITH 55550000
0020   7846 ;               g.  WRITE 00333330 INTO TB GROUP 1
0020   7847 ;               h.  ENABLE BOTH TB GROUPS BUT FORCE REPLACE GROUP 1
0020   7848 ;               i.  LOAD VA WITH 2AAA0000
0020   7849 ;               j.  WRITE 00999990 INTO TB
0020   7850 ;               k.  LOAD VA WITH TEST ADDRESS FROM RDM
0020   7851 ;               l.  READ PTE TO RDM WH REGISTER
0020   7852 ;       4.  TEST WH REGISTER FOR CORRECT PTE
0020   7853 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0020   7854 ;
0020   7855 ;
0020   7856 ; LOGIC DESCRIPTION
0020   7857 ;       THIS TESTS LOGIC IN THE ADK GATE ARRAY.
0020   7858 ;
0020   7859 ; ASSUMPTIONS
0020   7860 ;       THIS TEST ASSUMES THE DPM AND WB ARE OPERATIONAL
0020   7861 ;
0020   7862 ; ERROR DESCRIPTION
0020   7863 ;       EXPECTED PTE
0020   7864 ;       GOT PTE
0020   7865 ;
0020   7866 ;--
0020   7867 ;*****************************************************************************
0020   7868 ;/////////////////////////////////////////////////////////////////////////////
0020   7869
0020   7870          INITIALIZE
0022   7871          LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
0029   7872          LOADREG         WDREG,1$,I,L            ;LOAD TEST ADDRESS INTO WD REG
0031   7873          ERRLOOP                                 ;ERROR LOOP POINT
0033   7874          FETCH           0                       ;EXECUTE THE UCODE.
0038   7875          BRSTCLK         <^X1E>
003C   7876          CMPREGMSK       WHREG,2$,I,3$,,L,       ;CHECK THE PTE.
0048   7877          IFERROR         ,<<ADK>>,               ;INCORRECT PTE
004E   7878          ENDLOOP         I                       ;END TEST LOOP
```

```
                                0051  7879          SKIP                                    ;GO TO NEXT TEST
                                0058  7880
                                0058  7881 BEGIN_TEST_DATA
                                0058
                                0058          ;-----------------------------------------------------------------------
                                0058  7882
                    2AAA0000    0058  7883 1$:      .LONG    ^X2AAA0000                     ;TEST ADDRESS TABLE
                    55550000    005C  7884          .LONG    ^X55550000
                                0060  7885
                    00999990    0060  7886 2$:      .LONG    ^X00999990                     ;COMPARE DATA TABLE
                    00333330    0064  7887          .LONG    ^X00333330
                                0068  7888
                    00FFFFF8    0068  7889 3$:      .LONG    ^X00FFFFF8                     ;MASK FOR CMPREGMSK PSEUDO-OP
                                006C  7890
                                006C  7891 BEGIN_CPU_DATA
                                0002  7892
                                0002  7893 DCS_DATA
                                0001  7894
U 00, 7700,C800,0364,0300,0470,1801 0001  7895 ;% 00    NOP
U 01, 7700,8800,5BE6,03D8,0470,1802 000C  7899 ;% 01    D_R[ZERO]                          ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803 0017  7903 ;% 02    VA_Q_D_D+ZLIT8[0]                  ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804 0022  7907 ;% 03    VA_D_D+ZLIT8[0] INVALIDATE TB      ; 200.............
U 04, 7700,F800,7E7F,FFFF,8470,1805 002D  7911 ;% 04    LONLIT_[03000000]                  ; SET FORCE MISS GROUP 1 AND
U 05, 7700,4800,5BE4,03D4,6870,1806 0038  7915 ;% 05    MEMSCAR_R[LONLIT]                  ; FORCE REPLACE GROUP 0.
U 06, 7700,7800,7AFF,FFFF,8470,1807 0043  7919 ;% 06    LONLIT_[0A000000]
U 07, 7700,4800,5BE4,03D4,6070,1808 004E  7923 ;% 07    MEMSCR_R[LONLIT]
U 08, 7700,B800,6AAA,FFFF,8470,1809 0059  7927 ;% 08    LONLIT_[2AAA0000]                  ; LOAD THE VA WITH 2AAA0000.
U 09, 7700,4800,5BE4,03D4,4A70,180A 0064  7931 ;% 09    VA_R[LONLIT]
U 0A, 7700,7800,7FD5,5557,8470,180B 006F  7935 ;% 0A    LONLIT_[00555550]                  ; WRITE VA=2AAA0000 AND
U 0B, 7700,C81B,5BE4,03D4,5070,180C 007A  7939 ;% 0B    TB_R[LONLIT]                       ; PTE=00555550 INTO TB GROUP 0.
U 0C, 7700,7800,7E7F,FFFF,8470,180D 0085  7943 ;% 0C    LONLIT_[03000000]                  ; SET FORCE MISS GROUP 0 AND
U 0D, 7700,C800,5BE4,03D4,6870,180E 0090  7947 ;% 0D    MEMSCAR_R[LONLIT]                  ; FORCE REPLACE GROUP 1.
U 0E, 7700,B800,797F,FFFF,8470,180F 009B  7951 ;% 0E    LONLIT_[0D000000]
U 0F, 7700,4800,5BE4,03D4,6070,1810 00A6  7955 ;% 0F    MEMSCR_R[LONLIT]
U 10, 7700,F800,5555,7FFF,8470,1811 00B1  7959 ;% 10    LONLIT_[55550000]                  ; LOAD THE VA WITH 55550000.
U 11, 7700,4800,5BE4,03D4,4A70,1812 00BC  7963 ;% 11    VA_R[LONLIT]
U 12, 7700,7800,7FE6,6667,8470,1813 00C7  7967 ;% 12    LONLIT_[00333330]                  ; WRITE VA=55550000 AND
U 13, 7700,C81B,5BE4,03D4,5070,1814 00D2  7971 ;% 13    TB_R[LONLIT]                       ; PTE=00333330 INTO TB GROUP 1.
U 14, 7700,7800,7E7F,FFFF,8470,1815 00DD  7975 ;% 14    LONLIT_[03000000]                  ; ENABLE BOTH TB GROUPS BUT
U 15, 7700,C800,5BE4,03D4,6870,1816 00E8  7979 ;% 15    MEMSCAR_R[LONLIT]                  ; FORCE REPLACE GROUP 1.
U 16, 7700,F800,79FF,FFFF,8470,1817 00F3  7983 ;% 16    LONLIT_[0C000000]
U 17, 7700,C800,5BE4,03D4,6070,1818 00FE  7987 ;% 17    MEMSCR_R[LONLIT]
U 18, 7700,3800,6AAA,FFFF,8470,1819 0109  7991 ;% 18    LONLIT_[2AAA0000]                  ; LOAD THE VA WITH 2AAA0000.
U 19, 7700,C800,5BE4,03D4,4A70,181A 0114  7995 ;% 19    VA_R[LONLIT]
U 1A, 7700,F800,7FB3,3337,8470,181B 011F  7999 ;% 1A    LONLIT_[00999990]                  ; WRITE VA=2AAA0000 WITH
U 1B, 7700,481B,5BE4,03D4,5070,181C 012A  8003 ;% 1B    TB_R[LONLIT]                       ; PTE=00999990 INTO THE TB.
U 1C, 5700,4800,0364,0300,4A70,181D 0135  8007 ;% 1C    VA_RDM                             ; LOAD THE VA WITH TEST ADDRESS
U 1D, 6701,081F,5924,0200,05F0,181E 0140  8011 ;% 1D    RDM_TB                             ; READ OUT OF THE TB AT THIS
U 1E, 7300,C800,0364,0300,0470,181F 014B  8015 ;% 1E    STOP.CLOCK                         ; ADDRESS.
                                0156  8019
                                0156  8020 END_CPU_DATA
                                006C  8021
                                006C  8022 END_TEST_DATA
                                006C
                                006C          ;-----------------------------------------------------------------------
```

```
                006C  8023
                006C  8024 ENDTEST
```

```
0020          .SBTTL  "TEST   030     TB WRITE PULSE LOGIC TEST 5
0020  8026 ;*******************************************************************************
0020  8027 ;++
0020  8028 ;
0020  8029 ;       TB WRITE PULSE LOGIC TEST 5
0020  8030 ;
0020  8031 ; FUNCTIONAL DESCRIPTION
0020  8032 ;       THIS IS A TEST OF THE TRANSLATION BUFFER RANDOM REPLACE LOGIC.
0020  8033 ;       TESTED HERE IS THE RESULT OF RANDOM REPLACEMENT STARTING WITH A
0020  8034 ;       REPLACEMENT IN GROUP 0 FOLLOWED BY A REPLACEMENT IN GROUP 1.
0020  8035 ;
0020  8036 ; TEST ALGORITHM
0020  8037 ;       1.  SET UP A TEST LOOP TO SELECT ADDRESSES FOR THE TB
0020  8038 ;       2.  LOAD AN ADDRESS INTO THE RDM WD REGISTER
0020  8039 ;       3.  EXECUTE DCS PROGRAM
0020  8040 ;               a.  INVALIDATE THE TB
0020  8041 ;               b.  SET MISS GROUP 1 REPLACE GROUP 0
0020  8042 ;               c.  LOAD VA WITH 2AAA0000
0020  8043 ;               d.  WRITE TB WITH 00555550
0020  8044 ;               e.  SET MISS GROUP 0 REPLACE GROUP 1
0020  8045 ;               f.  LOAD VA WITH 55550000
0020  8046 ;               g.  WRITE TB WITH 00333330
0020  8047 ;               h.  SET RANDOM REPLACEMENT
0020  8048 ;               i.  LOAD VA WITH 33330000
0020  8049 ;               j.  WRITE TB WITH 00999990
0020  8050 ;               k.  LOAD VA WITH 0F0F0000
0020  8051 ;               l.  WRITE TB WITH 00DDDDD0
0020  8052 ;               m.  MOVE ADDRESS FROM RDM TO VA REGISTER
0020  8053 ;               n.  READ TB
0020  8054 ;       4.  TEST WH REGISTER FOR CORRECT PTE
0020  8055 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESS
0020  8056 ;
0020  8057 ;
0020  8058 ; LOGIC DESCRIPTION
0020  8059 ;       THIS TESTS LOGIC IN THE ADK GATE ARRAY.
0020  8060 ;
0020  8061 ; ASSUMPTIONS
0020  8062 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
0020  8063 ;
0020  8064 ; ERROR DESCRIPTION
0020  8065 ;       EXPECTED PTE
0020  8066 ;       GOT PTE
0020  8067 ;
0020  8068 ;--
0020  8069 ;*******************************************************************************
0020  8070 ;///////////////////////////////////////////////////////////////////////////////
0020  8071
0020  8072         INITIALIZE
0022  8073         LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
0029  8074         LOADREG         WDREG,1$,I,L            ;ADDRESS TO READ TB
0031  8075         ERRLOOP                                 ;ERROR LOOP POINT
0033  8076         FETCH           0                       ; EXECUTE THE UCODE.
0038  8077         BRSTCLK         <^X22>
003C  8078         CMPREGMSK       WHREG,2$,I,3$,,L,       ;CHECK THE PTE.
0048  8079         IFERROR         ,<<ADK>>,               ;INCORRECT PTE
```

```
                                004E  8080          ENDLOOP          I                    ;END TEST LOOP
                                0051  8081          SKIP                                  ;GO TO NEXT TEST
                                0058  8082
                                0058  8083 BEGIN_TEST_DATA
                                0058
                                0058          ;-------------------------------------------------------------------------
                                0058  8084
                    33330000    0058  8085 1$:       .LONG    ^X33330000                   ;TEST ADDRESS TABLE
                    0F0F0000    005C  8086           .LONG    ^X0F0F0000
                                0060  8087
                    00999990    0060  8088 2$:       .LONG    ^X00999990                   ;COMPARE DATA
                    00DDDDD0    0064  8089           .LONG    ^X00DDDDD0
                                0068  8090
                    00FFFFF8    0068  8091 3$:       .LONG    ^X00FFFFF8                   ;MASK FOR CMPREGMSK PSEUDO OP
                                006C  8092
                                006C  8093 BEGIN_CPU_DATA
                                0002  8094
                                0002  8095 DCS_DATA
                                0001  8096
U 00. 7700,C800,0364,0300,0470,1801 0001  8097 ;* 00    NOP
U 01. 7700,8800,5BE6,03D8,0470,1802 000C  8101 ;* 01    D_R[ZERO]                            ; zero the D register
U 02. 7700,D800,DB13,0300,4A70,1803 0017  8105 ;* 02    VA_Q_D_D+ZLIT8[0]                    ; invalidate the TB at 0 &
U 03. 7700,1800,DB12,0300,5360,1804 0022  8109 ;* 03    VA_D_D+ZLIT8[0] INVALIDATE TB        ; 200.............
U 04. 7700,F800,7E7F,FFFF,8470,1805 002D  8113 ;* 04    LONLIT_[03000000]                    ; SET FORCE MISS GROUP 1 AND
U 05. 7700,4800,5BE4,03D4,6870,1806 0038  8117 ;* 05    MEMSCAR_R[LONLIT]                    ; FORCE REPLACE GROUP 0.
U 06. 7700,7800,7AFF,FFFF,8470,1807 0043  8121 ;* 06    LONLIT_[0A000000]
U 07. 7700,4800,5BE4,03D4,6070,1808 004E  8125 ;* 07    MEMSCR_R[LONLIT]
U 08. 7700,B800,6AAA,FFFF,8470,1809 0059  8129 ;* 08    LONLIT_[2AAA0000]                    ; LOAD THE VA WITH 2AAA0000.
U 09. 7700,4800,5BE4,03D4,4A70,180A 0064  8133 ;* 09    VA_R[LONLIT]
U 0A. 7700,7800,7FD5,5557,8470,180B 006F  8137 ;* 0A    LONLIT_[00555550]                    ; WRITE VA=2AAA0000 AND
U 0B. 7700,C81B,5BE4,03D4,5070,180C 007A  8141 ;* 0B    TB_R[LONLIT]                         ; PTE=00555550 INTO TB GROUP 0.
U 0C. 7700,7800,7E7F,FFFF,8470,180D 0085  8145 ;* 0C    LONLIT_[03000000]                    ; SET FORCE MISS GROUP 0 AND
U 0D. 7700,C800,5BE4,03D4,6870,180E 0090  8149 ;* 0D    MEMSCAR_R[LONLIT]                    ; FORCE REPLACE GROUP 1.
U 0E. 7700,B800,797F,FFFF,8470,180F 009B  8153 ;* 0E    LONLIT_[0D000000]
U 0F. 7700,4800,5BE4,03D4,6070,1810 00A6  8157 ;* 0F    MEMSCR_R[LONLIT]
U 10. 7700,F800,5555,7FFF,8470,1811 00B1  8161 ;* 10    LONLIT_[55550000]                    ; LOAD THE VA WITH 55550000.
U 11. 7700,4800,5BE4,03D4,4A70,1812 00BC  8165 ;* 11    VA_R[LONLIT]
U 12. 7700,7800,7FE6,6667,8470,1813 00C7  8169 ;* 12    LONLIT_[00333330]                    ; WRITE VA=55550000 AND
U 13. 7700,C81B,5BE4,03D4,5070,1814 00D2  8173 ;* 13    TB_R[LONLIT]                         ; PTE=00333330 INTO TB GROUP 1.
U 14. 7700,7800,7E7F,FFFF,8470,1815 00DD  8177 ;* 14    LONLIT_[03000000]                    ; ENABLE BOTH TB GROUPS
U 15. 7700,C800,5BE4,03D4,6870,1816 00E8  8181 ;* 15    MEMSCAR_R[LONLIT]                    ; WITH RANDOM REPLACEMENT.
U 16. 7700,F800,7FFF,FFFF,8470,1817 00F3  8185 ;* 16    LONLIT_[00000000]
U 17. 7700,C800,5BE4,03D4,6070,1818 00FE  8189 ;* 17    MEMSCR_R[LONLIT]
U 18. 7700,F800,6666,7FFF,8470,9819 0109  8193 ;* 18    LONLIT_[33330000],CLOCK.EXT          ; LOAD THE VA WITH 33330000.
U 19. 7700,C800,5BE4,03D4,4A70,181A 0114  8197 ;* 19    VA_R[LONLIT]
U 1A. 7700,F800,7FB3,3337,8470,181B 011F  8201 ;* 1A    LONLIT_[00999990]                    ; WRITE VA=33330000 WITH
U 1B. 7700,481B,5BE4,03D4,5070,181C 012A  8205 ;* 1B    TB_R[LONLIT]                         ; PTE=00999990 INTO THE TB. THIS
                                0135  8209                                                 ; WRITE SHOULD GO INTO GROUP 0.
U 1C. 7700,7800,7878,7FFF,8470,981D 0135  8210 ;* 1C    LONLIT_[0F0F0000],CLOCK.EXT          ; WRITE VA=0F0F0000 AND
U 1D. 7700,4800,5BE4,03D4,4A70,181E 0140  8214 ;* 1D    VA_R[LONLIT]                         ; PTE=00DDDDD0 INTO THE TB.
U 1E. 7700,3800,7F91,1117,8470,181F 014B  8218 ;* 1E    LONLIT_[00DDDDD0]
U 1F. 7700,481B,5BE4,03D4,5070,1820 0156  8222 ;* 1F    TB_R[LONLIT]                         ; THIS WRITE SHOULD GO INTO
                                0161  8226                                                 ; GROUP 1
U 20. 5700,4800,0364,0300,4A70,1821 0161  8227 ;* 20    VA_RDM                               ; LOAD THE VA WITH THE TEST
U 21. 6701,081F,5924,0200,05F0,1822 016C  8231 ;* 21    RDM_TB                               ; ADDRESS AND READ THE TB AT
```

```
U 22, 7300,C800,0364,0300,0470,1823  0177  8235 ;* 22   STOP.CLOCK                                  ; THAT ADDRESS.
                                     0182  8239
                                     0182  8240 END_CPU_DATA
                                     006C  8241
                                     006C  8242 END_TEST_DATA
                                     006C
                                     006C       ;-------------------------------------------------------------------------
                                     006C  8243
                                     006C  8244 ENDTEST
```

```
                0000  8246 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN            = 00000000
$TEST_NUMBER    = 00000031
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
ARG_CNT         = 00000049 R      F9
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_HEAD      = 00000000 R      B9
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
CML             = 0000000E
COMPOSIT_LENGTH= 00000005
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              = 00000000
D1              = 00000003
D2              = 00000000
DATA_FLAG       = 00000000
DATA_LENGTH     = 00000004
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      FA
DCS_LENGTH      = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
DUM1            = FFFFFFF0
ENDTEST_LOC     = 00000071 R      F9
END_SLICE       = 00000025
END_TEST_1        00000179 R      02
END_TEST_10       00000059 R      39
END_TEST_11       000000D6 R      3F
END_TEST_12       0000007A R      45
END_TEST_13       0000007F R      4B
END_TEST_14       00000051 R      51
END_TEST_15       00000051 R      57
END_TEST_16       00000051 R      5D
END_TEST_17       00000046 R      63
END_TEST_18       0000026E R      69
END_TEST_19       000000CA R      6F
END_TEST_2        000000BE R      09
END_TEST_20       00000071 R      71
END_TEST_21       000001BA R      77
END_TEST_22       000001AF R      7D
END_TEST_23       000001BA R      83
END_TEST_24       000001AF R      89
END_TEST_25       00000182 R      8F
END_TEST_26       000000D1 R      91
END_TEST_27       00000223 R      97
END_TEST_28       000002BF R      99
END_TEST_29       0000031D R      9F
END_TEST_3        000000C6 R      0F
END_TEST_30       00000085 R      A5
END_TEST_31       000000D4 R      AB
END_TEST_32       0000006A R      B1
END_TEST_33       000000C1 R      B7
END_TEST_34       00000087 R      BD
END_TEST_35       00000087 R      C3
END_TEST_36       0000009B R      C9
END_TEST_37       0000006A R      CF
END_TEST_38       00000123 R      D5
END_TEST_39       00000123 R      D7
END_TEST_4        000000A7 R      15
END_TEST_40       0000011A R      D9
END_TEST_41       0000011A R      DB
END_TEST_42       00000198 R      DD
END_TEST_43       00000181 R      DF
END_TEST_44       0000007C R      E1
END_TEST_45       0000007C R      E7
END_TEST_46       0000006C R      ED
END_TEST_47       0000006C R      F3
END_TEST_48       0000006C R      F9
END_TEST_5        0000009E R      1B
END_TEST_6        00000090 R      21
END_TEST_7        00000062 R      27
END_TEST_8        00000058 R      2D
END_TEST_9        00000058 R      33
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          MAREG           = 00007410          OP_MASK         = 00000018
FQA             = 00000014          MASTER_HALT_EN  = 00000008          OP_NEW_TEST     = 0000001A
FRONT1          = 00007420          MAX_ARGUMENTS   = 00000010          OP_PATTERN_GEN  = 00000010
FRONT2          = 00007421          MA_KEY          = 00000038          OP_PRINT_N      = 00000036
FRONT_PNL_1     = 00007420          MC0             = 00000003          OP_PRINT_S      = 00000034
FRONT_PNL_2     = 00007421          MC1             = 00000009          OP_SAVE_INDEX   = 0000002C
FRONT_PNL_LOCK  = 00000080          MC2             = 0000000B          OP_SKIP         = 0000001C
HALT_FLAG       = 00000001          MDL             = 0000001B          OP_SUB_TEST     = 0000001E
HALT_KEY        = 00000008          MDR             = 00000018          OVERLAY_HEAD    = 00000000 R     FE
HALT_ON_MATCH   = 00000001          MDREG           = 00007414          PARITY_ERROR    = 00000020
HEAD            = FFFF7C00          MD_KEY          = 00000034          PAR_CHK_ENABL   = 00000008
HIGH            = 00000001          MD_REG_BYTE_0   = 00007414          PAR_STOP_ENABL  = 00000010
H_FROM_WBUS     = 00000010          MD_REG_BYTE_1   = 00007415          PASS_KEY        = 00000002
IB_FLAG         = 00000020          MD_REG_BYTE_2   = 00007416          PB_INIT         = 00000040
IB_KEY          = 00000012          MD_REG_BYTE_3   = 00007417          PB_KEY          = 00000036
INSTR_FLAG      = 00000004          MEC             = 0000001A          PC_KEY          = 00000030
INSTR_KEY       = 00000018          MHREG           = 00007418          PHB             = 00000007
INT             = 00000010          MH_REG_BYTE_0   = 00007418          PRK             = 0000000C
IRD             = 00000005          MH_REG_BYTE_1   = 00007419          PS_KEY          = 0000003E
I_INDEX_FLAG    = 00000000          MH_REG_BYTE_2   = 0000741A          QA_FLAG         = 00000040
I_INIT          = 00000000          MH_REG_BYTE_3   = 0000741B          QA_KEY          = 00000014
J_INDEX_FLAG    = 00000000          MIC             = 00000001          QV_FLAG         = 00000002
J_INIT          = 00000000          MICROWORD       = 0000000A          QV_KEY          = 00000028
K_INDEX_FLAG    = 00000000          MICRO_ADDR_INH  = 00000080          RDCTRL          = 0000741E
K_INIT          = 00000000          MONITOR_SIZE    = 00002C5E          RDM             = 00000004
L               = 00000004          MSQ             = 00000008          RDM_FLAG        = 00000004
LIST_END        = 0000004E R     F9 MT_KEY          = 0000002E          RDM_KEY         = 0000002A
LIST_HEAD       = 0000004A R     F9 N               = 00000001          RDM_LAST        = 00000008
LIST_LENGTH     = 00000004          NER_FLAG        = 00000004          RD_CTRL_REG     = 0000741E
LITERAL         = 00000000          NER_KEY         = 0000000C          REMOTE          = 00000001
LONG            = 00000004          NOERROR         = 00000001          RTEMP_FLAG      = 00000000
LOOP_ERROR_FLAG= 00000020           NOTEQUAL        = 00000003          RTEMP_FLAG1     = 00000000
LOOP_FLAG       = 00000002          ONEQUAL         = 00000002          RTEMP_HEAD      = 00000000 R     D1
LOOP_KEY        = 0000000A          ONERROR         = 00000000          RTEMP_LENGTH    = 00000001
LOST_FLAG       = 00000010          OP_BEGIN_LOOP   = 00000008          RT_KEY          = 0000002C
LOST_KEY        = 0000001A          OP_BURST_CLOCK  = 00000002          SAC             = 00000009
LOW             = 00000000          OP_COMPARE_REG  = 00000004          SA_CLOCK        = 00000080
M               = 0000000A          OP_COMPARE_REGM= 00000006           SA_FLAG         = 00000010
M$TEST_SIZE     = 000007A2          OP_COMPARE_VB   = 00000020          SA_KEY          = 00000010
M$TEST_SIZE_D   = 000003E2          OP_DUMPLOG      = 00000030          SA_START_STOP   = 00000080
MA0             = 00000008          OP_ENABL_STALL  = 00000038          SBE_FLAG        = 00000020
MA1             = 0000000A          OP_ENABL_TRAP   = 0000002E          SBE_KEY         = 00000042
MA2             = 0000000C          OP_END_FILE     = 00000024          SECTION_FLAG    = 00000080
MAD             = 00000013          OP_END_LOOP     = 0000000A          SF_KEY          = 00000040
MAIN32          = 0000741D          OP_END_TEST     = 0000000C          SOMM_FLAG       = 00000001
MAINT_A_TO_CMI  = 00000080          OP_ERRLOG       = 00000032          SOMM_KEY        = 00000006
MAINT_CMI_TO_MH= 00000020           OP_ERROR_LOOP   = 0000000E          SPA             = 00000002
MAINT_DCS_ENABL= 00000004           OP_FETCH        = 00000022          SRK             = 00000001
MAINT_MD_TO_CMI= 00000040           OP_IF_ERROR     = 00000012          SRM             = 00000017
MAINT_REG       = 0000741D          OP_INC_INDEX    = 0000002A          SR_KEY          = 0000003C
MAINT_STB_INH   = 00000002          OP_INIT         = 00000014          STACK_SIZE      = 00000400
MAINT_TRAP_OFF  = 00000001          OP_LOAD_DCS     = 00000000          STATUS          = 00007406
MAINT_WB_TO_WH  = 00000008          OP_LOAD_FIELD   = 00000026          STATUS_REG      = 00007406
MAINT_WD_TO_WB  = 00000010          OP_LOAD_INDEX   = 00000028          STEP_KEY        = 0000001E
MAP             = 00000012          OP_LOAD_REG     = 00000016          STOP_CLOCK      = 00000004
```

```
STROBE_CMI      = 00000040
ST_KEY          = 0000001E
TEMP            = 00000002 R     F8
TEMP1           = 0000007A
TEMP2           = 00000184
TEMP3           = 00000002
TEMP4           = 00000002 R     F8
TEST            = 00000004
TEST_FLAG       = 00000000
TEST_HEAD       = 0000001D R     F9
TEST_KEY        = 00000000
TEST_LENGTH     = 00000001
TEST_SECT_HEAD  = 00000000 R     F9
THIS_TEST       = 00000030
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 synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| X_0001_D | 00000002 ( 2.) | 01 ( 1.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0001_T | 00000180 ( 384.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0001_GDCS | 00000038 ( 56.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0001_ERTEMP | 00000011 ( 17.) | 04 ( 4.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0001_FCACHE | 00000001 ( 1.) | 05 ( 5.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0001_HFILL | 00000034 ( 52.) | 06 ( 6.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| DIRECTORY | 00000030 ( 48.) | 07 ( 7.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0002_D | 00000002 ( 2.) | 08 ( 8.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0002_T | 00000100 ( 256.) | 09 ( 9.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0002_GDCS | 00000038 ( 56.) | 0A ( 10.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0002_ERTEMP | 00000009 ( 9.) | 0B ( 11.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0002_FCACHE | 00000001 ( 1.) | 0C ( 12.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0002_HFILL | 0000003C ( 60.) | 0D ( 13.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0003_D | 00000002 ( 2.) | 0E ( 14.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0003_T | 00000100 ( 256.) | 0F ( 15.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0003_GDCS | 00000038 ( 56.) | 10 ( 16.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0003_ERTEMP | 00000009 ( 9.) | 11 ( 17.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0003_FCACHE | 00000001 ( 1.) | 12 ( 18.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0003_HFILL | 0000003C ( 60.) | 13 ( 19.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0004_D | 00000002 ( 2.) | 14 ( 20.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0004_T | 00000100 ( 256.) | 15 ( 21.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0004_GDCS | 00000064 ( 100.) | 16 ( 22.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0004_ERTEMP | 00000011 ( 17.) | 17 ( 23.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0004_FCACHE | 00000001 ( 1.) | 18 ( 24.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0004_HFILL | 00000008 ( 8.) | 19 ( 25.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0005_D | 00000002 ( 2.) | 1A ( 26.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0005_T | 00000100 ( 256.) | 1B ( 27.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0005_GDCS | 00000043 ( 67.) | 1C ( 28.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0005_ERTEMP | 00000005 ( 5.) | 1D ( 29.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0005_FCACHE | 00000001 ( 1.) | 1E ( 30.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0005_HFILL | 00000035 ( 53.) | 1F ( 31.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0006_D | 00000002 ( 2.) | 20 ( 32.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0006_T | 00000100 ( 256.) | 21 ( 33.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0006_GDCS | 0000002D ( 45.) | 22 ( 34.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0006_ERTEMP | 00000001 ( 1.) | 23 ( 35.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0006_FCACHE | 00000001 ( 1.) | 24 ( 36.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0006_HFILL | 0000004F ( 79.) | 25 ( 37.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0007_D | 00000002 ( 2.) | 26 ( 38.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0007_T | 00000080 ( 128.) | 27 ( 39.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0007_GDCS | 00000038 ( 56.) | 28 ( 40.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0007_ERTEMP | 00000005 ( 5.) | 29 ( 41.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0007_FCACHE | 00000001 ( 1.) | 2A ( 42.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0007_HFILL | 00000040 ( 64.) | 2B ( 43.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0008_D | 00000002 ( 2.) | 2C ( 44.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0008_T | 00000080 ( 128.) | 2D ( 45.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0008_GDCS | 0000002D ( 45.) | 2E ( 46.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0008_ERTEMP | 00000001 ( 1.) | 2F ( 47.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0008_FCACHE | 00000001 | ( 1.) | 30 | ( 48.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0008_HFILL | 0000004F | ( 79.) | 31 | ( 49.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0009_D | 00000002 | ( 2.) | 32 | ( 50.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0009_T | 00000080 | ( 128.) | 33 | ( 51.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0009_GDCS | 0000002D | ( 45.) | 34 | ( 52.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0009_ERTEMP | 00000001 | ( 1.) | 35 | ( 53.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0009_FCACHE | 00000001 | ( 1.) | 36 | ( 54.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0009_HFILL | 0000004F | ( 79.) | 37 | ( 55.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0010_D | 00000002 | ( 2.) | 38 | ( 56.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0010_T | 00000080 | ( 128.) | 39 | ( 57.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0010_GDCS | 0000002D | ( 45.) | 3A | ( 58.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0010_ERTEMP | 00000001 | ( 1.) | 3B | ( 59.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0010_FCACHE | 00000001 | ( 1.) | 3C | ( 60.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0010_HFILL | 0000004F | ( 79.) | 3D | ( 61.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0011_D | 00000002 | ( 2.) | 3E | ( 62.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0011_T | 00000100 | ( 256.) | 3F | ( 63.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0011_GDCS | 0000006F | ( 111.) | 40 | ( 64.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0011_ERTEMP | 00000011 | ( 17.) | 41 | ( 65.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0011_FCACHE | 00000001 | ( 1.) | 42 | ( 66.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0011_HFILL | 0000007D | ( 125.) | 43 | ( 67.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0012_D | 00000002 | ( 2.) | 44 | ( 68.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0012_T | 00000080 | ( 128.) | 45 | ( 69.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0012_GDCS | 0000002D | ( 45.) | 46 | ( 70.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0012_ERTEMP | 00000C01 | ( 1.) | 47 | ( 71.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0012_FCACHE | 00000001 | ( 1.) | 48 | ( 72.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0012_HFILL | 0000004F | ( 79.) | 49 | ( 73.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0013_D | 00000002 | ( 2.) | 4A | ( 74.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0013_T | 00000100 | ( 256.) | 4B | ( 75.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0013_GDCS | 00000043 | ( 67.) | 4C | ( 76.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0013_ERTEMP | 00000001 | ( 1.) | 4D | ( 77.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0013_FCACHE | 00000001 | ( 1.) | 4E | ( 78.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0013_HFILL | 00000039 | ( 57.) | 4F | ( 79.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0014_D | 00000002 | ( 2.) | 50 | ( 80.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0014_T | 00000080 | ( 128.) | 51 | ( 81.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0014_GDCS | 00000043 | ( 67.) | 52 | ( 82.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0014_FCACHE | 00000005 | ( 5.) | 53 | ( 83.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0014_ERTEMP | 00000001 | ( 1.) | 54 | ( 84.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0014_HFILL | 00000035 | ( 53.) | 55 | ( 85.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0015_D | 00000002 | ( 2.) | 56 | ( 86.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0015_T | 00000080 | ( 128.) | 57 | ( 87.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0015_GDCS | 00000043 | ( 67.) | 58 | ( 88.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0015_FCACHE | 00000005 | ( 5.) | 59 | ( 89.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0015_ERTEMP | 00000001 | ( 1.) | 5A | ( 90.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0015_HFILL | 00000035 | ( 53.) | 5B | ( 91.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0016_D | 00000002 | ( 2.) | 5C | ( 92.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0016_T | 00000080 | ( 128.) | 5D | ( 93.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0016_GDCS | 00000043 | ( 67.) | 5E | ( 94.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0016_FCACHE | 00000005 | ( 5.) | 5F | ( 95.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0016_ERTEMP | 00000001 | ( 1.) | 60 | ( 96.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0016_HFILL | 00000035 | ( 53.) | 61 | ( 97.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0017_D | 00000002 | ( 2.) | 62 | ( 98.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0017_T | 00000080 | ( 128.) | 63 | ( 99.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0017_GDCS | 00000038 | ( 56.) | 64 | (100.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0017_ERTEMP | 00000001 | ( 1.) | 65 | (101.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0017_FCACHE | 00000001 | ( 1.) | 66 | (102.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0017_HFILL | 00000044 | ( 68.) | 67 | (103.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0018_D | 00000002 | ( 2.) | 68 | (104.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0018_T | 00000280 | ( 640.) | 69 | (105.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0018_ERTEMP | 00000019 | ( 25.) | 6A | (106.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0018_FCACHE | 00000001 | ( 1.) | 6B | (107.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0018_GDCS | 00000001 | ( 1.) | 6C | (108.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0018_HFILL | 00000063 | ( 99.) | 6D | (109.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0019_D | 00000002 | ( 2.) | 6E | (110.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0019_T | 000000FE | ( 254.) | 6F | (111.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0020_D | 00000002 | ( 2.) | 70 | (112.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0020_T | 00000080 | ( 128.) | 71 | (113.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0020_GDCS | 00000064 | ( 100.) | 72 | (114.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0020_ERTEMP | 00000001 | ( 1.) | 73 | (115.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0020_FCACHE | 00000001 | ( 1.) | 74 | (116.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0020_HFILL | 00000018 | ( 24.) | 75 | (117.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0021_D | 00000002 | ( 2.) | 76 | (118.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0021_T | 00000200 | ( 512.) | 77 | (119.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0021_ERTEMP | 00000005 | ( 5.) | 78 | (120.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0021_FCACHE | 00000001 | ( 1.) | 79 | (121.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0021_GDCS | 00000001 | ( 1.) | 7A | (122.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0021_HFILL | 00000077 | ( 119.) | 7B | (123.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0022_D | 00000002 | ( 2.) | 7C | (124.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0022_T | 00000200 | ( 512.) | 7D | (125.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0022_ERTEMP | 00000005 | ( 5.) | 7E | (126.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0022_FCACHE | 00000001 | ( 1.) | 7F | (127.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0022_GDCS | 00000001 | ( 1.) | 80 | (128.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0022_HFILL | 00000077 | ( 119.) | 81 | (129.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0023_D | 00000002 | ( 2.) | 82 | (130.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0023_T | 00000200 | ( 512.) | 83 | (131.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0023_ERTEMP | 00000005 | ( 5.) | 84 | (132.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0023_FCACHE | 00000001 | ( 1.) | 85 | (133.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0023_GDCS | 00000001 | ( 1.) | 86 | (134.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0023_HFILL | 00000077 | ( 119.) | 87 | (135.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0024_D | 00000002 | ( 2.) | 88 | (136.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0024_T | 00000200 | ( 512.) | 89 | (137.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0024_ERTEMP | 00000005 | ( 5.) | 8A | (138.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0024_FCACHE | 00000001 | ( 1.) | 8B | (139.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0024_GDCS | 00000001 | ( 1.) | 8C | (140.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0024_HFILL | 00000077 | ( 119.) | 8D | (141.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0025_D | 00000002 | ( 2.) | 8E | (142.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0025_T | 000001FE | ( 510.) | 8F | (143.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0026_D | 00000002 | ( 2.) | 90 | (144.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0026_T | 00000100 | ( 256.) | 91 | (145.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0026_GDCS | 000000C7 | ( 199.) | 92 | (146.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0026_ERTEMP | 00000015 | ( 21.) | 93 | (147.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0026_FCACHE | 00000001 | ( 1.) | 94 | (148.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0026_HFILL | 00000021 | ( 33.) | 95 | (149.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0027_D | 00000002 | ( 2.) | 96 | (150.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0027_T | 0000027E | ( 638.) | 97 | (151.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0028_D | 00000002 | ( 2.) | 98 | (152.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0028_T | 00000300 | ( 768.) | 99 | (153.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0028_ERTEMP | 00000011 | ( 17.) | 9A | (154.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0028_FCACHE | 00000001 | ( 1.) | 9B | (155.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0028_GDCS | 00000001 | ( 1.) | 9C | (156.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0028_HFILL | 0000006B | ( 107.) | 9D | (157.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0029_D | 00000002 | ( 2.) | 9E (158.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0029_T | 00000380 | ( 896.) | 9F (159.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0029_ERTEMP | 0000000D | ( 13.) | A0 (160.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0029_FCACHE | 00000001 | ( 1.) | A1 (161.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0029_GDCS | 00000001 | ( 1.) | A2 (162.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0029_HFILL | 0000006F | ( 111.) | A3 (163.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0030_D | 00000002 | ( 2.) | A4 (164.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0030_T | 00000100 | ( 256.) | A5 (165.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0030_GDCS | 00000022 | ( 34.) | A6 (166.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0030_ERTEMP | 00000001 | ( 1.) | A7 (167.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0030_FCACHE | 00000001 | ( 1.) | A8 (168.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0030_HFILL | 0000005A | ( 90.) | A9 (169.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0031_D | 00000002 | ( 2.) | AA (170.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0031_T | 00000100 | ( 256.) | AB (171.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0031_GDCS | 0000011F | ( 287.) | AC (172.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0031_ERTEMP | 0000000D | ( 13.) | AD (173.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0031_FCACHE | 00000005 | ( 5.) | AE (174.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0031_HFILL | 0000004D | ( 77.) | AF (175.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0032_D | 00000002 | ( 2.) | B0 (176.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0032_T | 00000080 | ( 128.) | B1 (177.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0032_GDCS | 0000004E | ( 78.) | B2 (178.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0032_ERTEMP | 00000001 | ( 1.) | B3 (179.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0032_FCACHE | 00000001 | ( 1.) | B4 (180.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0032_HFILL | 0000002E | ( 46.) | B5 (181.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0033_D | 00000002 | ( 2.) | B6 (182.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0033_T | 00000100 | ( 256.) | B7 (183.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0033_GDCS | 0000004E | ( 78.) | B8 (184.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0033_FCACHE | 00000051 | ( 81.) | B9 (185.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0033_ERTEMP | 00000001 | ( 1.) | BA (186.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0033_HFILL | 0000005E | ( 94.) | BB (187.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0034_D | 00000002 | ( 2.) | BC (188.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0034_T | 00000100 | ( 256.) | BD (189.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0034_GDCS | 0000009B | ( 155.) | BE (190.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0034_ERTEMP | 00000009 | ( 9.) | BF (191.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0034_FCACHE | 00000001 | ( 1.) | C0 (192.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0034_HFILL | 00000059 | ( 89.) | C1 (193.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0035_D | 00000002 | ( 2.) | C2 (194.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0035_T | 00000100 | ( 256.) | C3 (195.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0035_GDCS | 0000004E | ( 78.) | C4 (196.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0035_ERTEMP | 00000001 | ( 1.) | C5 (197.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0035_FCACHE | 00000001 | ( 1.) | C6 (198.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0035_HFILL | 0000002E | ( 46.) | C7 (199.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0036_D | 00000002 | ( 2.) | C8 (200.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0036_T | 00000100 | ( 256.) | C9 (201.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0036_GDCS | 00000085 | ( 133.) | CA (202.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0036_ERTEMP | 00000005 | ( 5.) | CB (203.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0036_FCACHE | 00000001 | ( 1.) | CC (204.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0036_HFILL | 00000073 | ( 115.) | CD (205.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0037_D | 00000002 | ( 2.) | CE (206.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0037_T | 00000080 | ( 128.) | CF (207.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0037_GDCS | 0000004E | ( 78.) | D0 (208.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0037_ERTEMP | 00000009 | ( 9.) | D1 (209.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0037_FCACHE | 00000001 | ( 1.) | D2 (210.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0037_HFILL | 00000026 | ( 38.) | D3 (211.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0038_D | 00000002 | ( 2.) | D4 (212.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0038_T | 0000017E | ( 382.) | D5 (213.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0039_D | 00000002 | ( 2.) | D6 (214.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0039_T | 0000017E | ( 382.) | D7 (215.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0040_D | 00000002 | ( 2.) | D8 (216.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0040_T | 0000017E | ( 382.) | D9 (217.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0041_D | 00000002 | ( 2.) | DA (218.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0041_T | 0000017E | ( 382.) | DB (219.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0042_D | 00000002 | ( 2.) | DC (220.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0042_T | 000001FE | ( 510.) | DD (221.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0043_D | 00000002 | ( 2.) | DE (222.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0043_T | 000001FE | ( 510.) | DF (223.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0044_D | 00000002 | ( 2.) | E0 (224.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0044_T | 00000100 | ( 256.) | E1 (225.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0044_GDCS | 00000156 | ( 342.) | E2 (226.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0044_ERTEMP | 00000001 | ( 1.) | E3 (227.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0044_FCACHE | 00000001 | ( 1.) | E4 (228.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0044_HFILL | 00000026 | ( 38.) | E5 (229.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0045_D | 00000002 | ( 2.) | E6 (230.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0045_T | 00000100 | ( 256.) | E7 (231.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0045_GDCS | 00000156 | ( 342.) | E8 (232.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0045_ERTEMP | 00000001 | ( 1.) | E9 (233.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0045_FCACHE | 00000001 | ( 1.) | EA (234.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0045_HFILL | 00000026 | ( 38.) | EB (235.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0046_D | 00000002 | ( 2.) | EC (236.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0046_T | 00000080 | ( 128.) | ED (237.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0046_GDCS | 00000156 | ( 342.) | EE (238.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0046_ERTEMP | 00000001 | ( 1.) | EF (239.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0046_FCACHE | 00000001 | ( 1.) | F0 (240.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0046_HFILL | 00000026 | ( 38.) | F1 (241.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0047_D | 00000002 | ( 2.) | F2 (242.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0047_T | 00000080 | ( 128.) | F3 (243.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0047_GDCS | 00000156 | ( 342.) | F4 (244.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0047_ERTEMP | 00000001 | ( 1.) | F5 (245.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0047_FCACHE | 00000001 | ( 1.) | F6 (246.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0047_HFILL | 00000026 | ( 38.) | F7 (247.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0048_D | 00000002 | ( 2.) | F8 (248.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0048_T | 00000080 | ( 128.) | F9 (249.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0048_GDCS | 00000182 | ( 386.) | FA (250.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0048_ERTEMP | 00000001 | ( 1.) | FB (251.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0048_FCACHE | 00000001 | ( 1.) | FC (252.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0048_HFILL | 0000007A | ( 122.) | FD (253.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0049_D | 00000000 | ( 0.) | FE (254.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |

## Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 27 | 00:00:00.17 | 00:00:01.12 |
| Command processing | 848 | 00:00:01.03 | 00:00:01.55 |
| Pass 1 | 4957 | 00:02:21.01 | 00:02:35.55 |
| Symbol table sort | 0 | 00:00:00.63 | 00:00:00.63 |
| Pass 2 | 1381 | 00:00:42.90 | 00:00:57.70 |
| Symbol table output | 1 | 00:00:00.24 | 00:00:00.30 |

```
Psect synopsis output              3     00:00:00.96    00:00:01.14
Cross-reference output             0     00:00:00.00    00:00:00.00
Assembler run totals            7219     00:03:06.94    00:03:38.04

The working set limit was 2048 pages.
1783930 bytes (3485 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 378 non-local and 181 local symbols.
8964 source lines were read in Pass 1, producing 543 object records in Pass 2.
99 pages of virtual memory were used to define 45 macros.
```

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| DISK$UCODPACK00:[VAX750]COMETMAC.MLB;1 | 45 |
| SYS$COMMON:[SYSLIB]STARLET.MLB;2 | 0 |
| TOTALS (all libraries) | 45 |

```
2000 GETS were required to define 45 macros.

There were no errors, warnings or information messages.

MACRO UCOD$0:[VAX750.MIC]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.MIC]ECKAC1/LIS=ECKAC1/OBJ=ECKAC1
```

(1)  24  "MIC Revision History"
(1)  1  ""
(1)  3  "DIAGNOSTIC MICROCODE MACROS"
(1)  98  "  RDM Control File Macros"
(1)  109  "  Diagnostic Macros"
(1)  194  ""
(1)  195  "MODULE GATE ARRAY MAPS"
(1)  197  "  L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1)  254  "  L0002 GATE ARRAY LOCATION"
(1)  311  "  L0003 GATE ARRAY LOCATION"
(1)  369  "  L0004 GATE ARRAY LOCATION"
(1)  427  "  L0011 & L0016 GATE ARRAY LOCATION"
(1)  485  ""
(1)  490  "EQUATED SYMBOLS
(1)  493  ""
(1)  2  "TEST  031  TB WRITE PULSE LOGIC TEST 6
(1)  221  "TEST  032  TB GROUP 0 HIT COMPARATOR TEST 1
(1)  432  "TEST  033  TB GROUP 1 HIT COMPARATOR TEST 1
(1)  644  "TEST  034  TB GROUP 0 HIT COMPARATOR TEST 2
(1)  862  "TEST  035  TB GROUP 1 HIT COMPARATOR TEST 2
(1)  1080  "TEST  036  TB GROUP 0 HIT COMPARATOR TEST 3
(1)  1290  "TEST  037  TB GROUP 1 HIT COMPARATOR TEST 3
(1)  1502  "TEST  038  TB GROUP 0 HIT COMPARATOR TEST 4
(1)  1720  "TEST  039  TB GROUP 1 HIT COMPARATOR TEST 4
(1)  1940  "TEST  03A  TB GROUP 0 M BIT UTRAP TEST
(1)  2113  "TEST  03B  TB GROUP 1 M BIT UTRAP TEST
(1)  2285  "TEST  03C  TB GROUP 0 PTE MEMORY DUAL ADDRESSING TEST
(1)  2482  "TEST  03D  TB GROUP 1 PTE MEMORY DUAL ADDRESSING TEST
(1)  2679  "TEST  03E  TB GROUP 0 TAG MEMORY DUAL ADDRESSING TEST
(1)  2892  "TEST  03F  TB GROUP 1 TAG MEMORY DUAL ADDRESSING TEST
(1)  3105  "TEST  040  TB GROUP 0 PTE MEMORY MARCH TEST
(1)  3395  "TEST  041  TB GROUP 1 PTE MEMORY MARCH TEST
(1)  3685  "TEST  042  TB GROUP 0 TAG MEMORY MARCH TEST
(1)  4025  "TEST  043  TB GROUP 1 TAG MEMORY MARCH TEST
(1)  4365  "TEST  044  TB GROUP 0 VALID BIT MARCH TEST
(1)  4673  "TEST  045  TB GROUP 1 VALID BIT MARCH TEST
(1)  4983  "TEST  046  TB INVALIDATE TEST 1
(1)  5199  "TEST  047  TB INVALIDATE TEST 2
(1)  5290  "TEST  048  TB GROUP 0 TAG PARITY GENERATOR/CHECKER TEST
(1)  5534  "TEST  049  TB GROUP 1 TAG PARITY GENERATOR/CHECKER TEST
(1)  5779  "TEST  04A  TEST TB TAG PARITY ERROR MACHINE CHECK
(1)  5969  "TEST  04B  TEST TB GROUP 0 DATA PARITY GENERATOR/CHECKER
(1)  6183  "TEST  04C  TEST TB GROUP 1 DATA PARITY GENERATOR/CHECKER
(1)  6397  "TEST  04D  TEST TB DATA PARITY ERROR MACHINE CHECK
(1)  6594  "TEST  04E  TEST FOR MACHINE CHECK WITH TB MULTIPLE HIT
(1)  6762  "TEST  04F  VA_PC+WBUS+ISIZE.PC_PC+ISIZE TEST
(1)  7010  "TEST  050  TEST XB0 AND XB1 REGISTERS
(1)  7256  "TEST  051  PREFETCH INCREMENTER TEST
(1)  7761  "TEST  052  Test PCBACK While PC Increment Logic is Active
(1)  8258  "TEST  053  TEST EXECUTION BUFFER ROTATE LOGIC
(1)  8566  "TEST  054  TEST FOR XB0 MACHINE CHECK WITH TB MULTIPLE HIT
(1)  8775  "TEST  055  TEST FOR XB1 MACHINE CHECK WITH TB MULTIPLE HIT
(1)  8996  "TEST  056  TEST XB0 FOR MACHINE CHECK WITH TB TAG PARITY ERROR
(1)  9244  "TEST  057  TEST XB1 FOR MACHINE CHECK WITH TB TAG PARITY ERROR
(1)  9515  "TEST  058  TEST XB0 FOR MACHINE CHECK WITH TB DATA PARITY ERROR
(1)  9690  "TEST  059  TEST XB1 FOR MACHINE CHECK WITH TB DATA PARITY ERROR
(1)  9891  "TEST  05A  TEST FOR MICRO-TRAP ON XB0 TB MISS

ZZ-ECKAC-8.0 (1) 10284 "TEST 05C TEST XB017-FEB-1986 Fiche 2 Frame I3 Sequence 240

```
(1)    10073        "TEST   05B     TEST FOR MICRO-TRAP ON XB1 TB MISS
(1)    10284        "TEST   05C     TEST XB0 FOR MICROTRAP WITH TB ACCESS VIOLATION
(1)    10464        "TEST   05D     TEST XB1 FOR MICROTRAP WITH TB ACCESS VIOALTION
(1)    10666        "TEST   05E     TEST D CLOCK INHIBIT ON MICROTRAP
```

```
0000     1          .TITLE  ECKAC VAX-11/750 MIC MICRO DIAGNOSTICS
0000     2          .IDENT  /V08.00/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1982
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
                        0000     24          .SBTTL  "MIC Revision History"
                        0000     25 ;++
                        0000     26 ; MIC MICRO-DIAGNOSTIC PACKAGE
                        0000     27 ;
                        0000     28 ; REVISION HISTORY
                        0000     29 ;
                        0000     30 ; 00-01 Bill Landry and Dave Spain      4-FEB-1980
                        0000     31 ;       Started file.
                        0000     32 ;
                        0000     33 ; 00-02 Bill Landry and Dave Spain      5-FEB-1980
                        0000     34 ;       Fixed numerous tests to invalidate both groups of TB before using them.
                        0000     35 ;
                        0000     36 ; 00-03 Bill Landry                     6-FEB-1980
                        0000     37 ;       Fixed my typing errors in test 4C.
                        0000     38 ;
                        0000     39 ; 00-04 Bill Landry and Dave Spain      7-FEB-1980
                        0000     40 ;       Skipped subtest 7 - 10 in test 4C. Also added cache data to PC
                        0000     41 ;       increment and ADD adder tests.
                        0000     42 ;
                        0000     43 ; 00-05 Bill Landry     21-FEB-1980
                        0000     44 ;       Fixed CMPREGMSK psuedo-ops in TB tests.
                        0000     45 ;
                        0000     46 ; 00-06 Bill Landry and Dave Spain      27-FEB-1980
                        0000     47 ;       Changed a CLRTB.VA_VA macro in test 4C to CLRTB.VA_R[]. Also added
                        0000     48 ;       cache validates to test 12 to prevent random cache parity errors.
                        0000     49 ;
                        0000     50 ; 00-07 Bill Landry     3-MAR-1980
                        0000     51 ;       Added changes for revised SKIP pseudo op.
                        0000     52 ;
                        0000     53 ; 00-08 Bill Landry     13-MAR-1980
                        0000     54 ;       Fixed tests that didnot use a CMPREGMSK pseudo op for testing the
                        0000     55 ;       CSTRP register.
                        0000     56 ; 00-09 Bill Landry     25-MAR-1980
                        0000     57 ;       Added cache tests.
                        0000     58 ;
                        0000     59 ; 01-10 Dave Spain       4-APR-1980
                        0000     60 ;       Added IRD tests to official release version.
                        0000     61 ;
                        0000     62 ; 01-11 Bill Landry     4-APR-1980
                        0000     63 ;       Moved XB, PREFETCH and IRD tests to ECKAB.SRC so manufacturing can
                        0000     64 ;       run one tape for DPM.
                        0000     65 ;
                        0000     66 ; 00-12 Bill Landry     4-APR-1980
                        0000     67 ;       Let's save version 1.0 for our first SDC release
                        0000     68 ;
                        0000     69 ; 00-13 Bill Landry     8-APR-1980
                        0000     70 ;       Revised cache tests to survive parity error traps.
                        0000     71 ;
                        0000     72 ; 00-14 Bill Landry and Dave Spain      15-APR-1980
                        0000     73 ;       Reassembled using new COMETDEF.
                        0000     74 ;
                        0000     75 ; 00-15 Dave Spain      18-APR-1980
                        0000     76 ;       Changed tests 4F and 50 to call out the ACV and UTR gate arrays instead
                        0000     77 ;       of the MIC module.
                        0000     78 ;
```

```
0000     79 ; 00-16 Bill Landry     1-MAY-1980
0000     80 ;       Revised TB tests to loop on error nicely.
0000     81 ;       Deleted redundent TB tests.
0000     82 ;
0000     83 ; 00-17 Bill Landry     14-MAY-1980
0000     84 ;       Added TB parity error tests
0000     85 ;
0000     86 ; 00-18 Bill Landry     10-JUN-1980
0000     87 ;       Added RDM tests to beginning of tape.
0000     88 ;
0000     89 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     90 ;       Added diagnostic macro file to title section and submitted to
0000     91 ;       RELEASE ENGINEERING.
0000     92 ;
0000     93 ; 01-01 Dave Spain      19-JUN-1980
0000     94 ;       Fixed Prefetch Incrementer test to clear out cache data after
0000     95 ;       every prefetch so that old data from cache will not cause test
0000     96 ;       to mistake bad addresses for good ones.
0000     97 ;
0000     98 ; 01-02 Bill Landry     26-JUN-1980
0000     99 ;       Added tests that check XB trap conditions
0000    100 ;
0000    101 ; 01-03 Bill Landry     30-JUN-1980
0000    102 ;       Added tests that check XB access violations
0000    103 ;
0000    104 ; 01-04 Dave Spain      11-JUL-1980
0000    105 ;       Changed location of BEGINSA pseudo-op to fix signature analysis
0000    106 ;       problem in PC_PC+WBUS test.
0000    107 ;
0000    108 ; 01-05 Bill Landry     8-AUG-1980
0000    109 ;       Added CMI tests in honor of my vacation.
0000    110 ;
0000    111 ; 01-06 Bill Landry     25-AUG-1980
0000    112 ;       Remodeled RDM to CMI tests.
0000    113 ;
0000    114 ; 01-07 Bill Landry     23-SEP-1980
0000    115 ;       Added byte mask testing and improved testing of first memory array.
0000    116 ;
0000    117 ; 02-00 Bill Landry     29-SEP-1980
0000    118 ;       Second release to SDC
0000    119 ;
0000    120 ; 02-01 Bill Landry     5-NOV-1980
0000    121 ;       Added miscellaneous enhancements
0000    122 ;
0000    123 ; 03-00 Bill Landry     17-NOV-1980
0000    124 ;       Third release to SDC
0000    125 ;
0000    126 ; 03-01 Bill Landry     24-NOV-1980
0000    127 ;       Added tests for cache parity logic
0000    128 ;
0000    129 ; 03-02 Bill Landry     30-DEC-1980
0000    130 ;       Separated RDM and MIC tests into two files
0000    131 ;
0000    132 ; 04-00 Bill Landry     05-JAN-1981
0000    133 ;       Submitted to RELEASE ENGINEERING
```

```
0000    134 ;
0000    135 ; 04-01 Bill Landry     14-JAN-1981
0000    136 ;       Revised ECC test to fix bug caused by ECO to INT PEND logic.
0000    137 ;
0000    138 ; 04-02 Bill Landry     27-FEB-1981
0000    139 ;       Udated MDR test to include branching logic.
0000    140 ;       Fixed PC_* tests to fix bug found by field service.
0000    141 ;
0000    142 ; 05-00 Bill Landry     2-MAR-1981
0000    143 ;       Added WRITE BUS ERROR INTERRUPT test
0000    144 ;       Submitted to RELEASE ENGINEERING
0000    145 ;
0000    146 ; 05-01 Bill Landry     18-MAR-1981
0000    147 ;       Made minor changes to various tests as suggested by BT manufacturing.
0000    148 ;
0000    149 ; 05-02 Bill Landry     8-JUN-1981
0000    150 ;       Fixed bug found by Paul Magnet in TB TAG PARITY TESTS.
0000    151 ;
0000    152 ; 05-03 Bill Landry     20-AUG-1981
0000    153 ;       Fixed bugs in memory tests caused by eco to interrupt pending logic.
0000    154 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    155 ;       logic.
0000    156 ;       Replaced all references to COMET with the word CPU.
0000    157 ;       Fixed bug in CACHE TAG MEMORY DUAL ADDRESSING TEST.
0000    158 ;       Submitted to RELEASE ENGINEERING.
0000    159 ;
0000    160 ; 05-04 Bill Landry     4-NOV-1981
0000    161 ;       Moved WCS test from DPM tape to this tape
0000    162 ;       Submitted to RELEASE ENGINEERING.
0000    163 ;
0000    164 ; 05-05 Bill Landry     10-NOV-1981
0000    165 ;       Changed all references of CLRTB.VA_WB to CLRTB.VA_VA.  This fixes a
0000    166 ;       problem caused by plugging in the FPA.
0000    167 ;       Re-released to SDC. Stopped release of 5.4
0000    168 ;
0000    169 ; 05-06 Bill Landry     6-JAN-1982
0000    170 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    171 ;       logic (same as 5.3 above).
0000    172 ;
0000    173 ; 05-07 Bill Landry     23-FEB-1982
0000    174 ;       Fixed more bugs in single bit error tests found by PAUL NATUSH.
0000    175 ;       Same as revision 5.3.
0000    176 ; 05-08 Dave Shull      9-Mar-1982
0000    177 ;       Changed the the IFERROR callouts so when L0011 (MC0) is encountered
0000    178 ;       L0016 (MC1) will be included, when M8728 (MA0) is encountered M8750
0000    179 ;       (MA1) will be included, and when the MAP gate array is encountered
0000    180 ;       MAD gate array will be included.
0000    181 ; 05-09 Dave Shull      10-Mar-1982
0000    182 ;       Added a test to check MS750 controller CSR2 and determine the
0000    183 ;       controller type (ie., L0011 or L0016) and to also print a configuration
0000    184 ;       map of the array's that are installed.
0000    185 ; 06-00 Dave Shull      22-mar-1982
0000    186 ;       Changed the 4 existing memory tests to check for array 0 type then
0000    187 ;       adjust addresses accordingly and test all of the first array versus
0000    188 ;       only testing the first 256kb.
```

```
0000    189 ;       Added 4 new tests that will check to see if array 1 is present and if
0000    190 ;       it is then determine the size and test accordingly.  It looks at both
0000    191 ;       array 0 and array 1, array 0 to determine the starting address of array
0000    192 ;       1 and array 1 to determine the starting and ending address to be tested
0000    193 ; 06-01 Dave Shull      26-Mar-1982
0000    194 ;       Changed READ.SEC MICRO ORDER test so the test could ran in an infinite
0000    195 ;       loop by itself.  Added code to reload Cache everytime the test is
0000    196 ;       restarted in any of the test that are using cache data that could get
0000    197 ;       blown away.
0000    198 ; 06-02 Dave Shull      05-Apr-1982
0000    199 ;       Removed all of the BEGINSA and ENDSA pseudo's
0000    200 ;       Submitted to RELEASE ENGINEERING
0000    201 ; 07-00 Dave Shull      10-May-1982
0000    202 ;       Added new test ( name <Test PCBACK While PC Increment Logic is Active>)
0000    203 ;       to check for side-affects on PCBACK register while incrementin the PC
0000    204 ;       through the execution buffer.
0000    205 ; 07-01 Dave Shull      14-May-1982
0000    206 ;       Changed all CLRTB.VA_VA macro's to prevent translation buffer
0000    207 ;       invalidate failures when manufacturing runs the micro's with an
0000    208 ;       external clock set to fast margins.
0000    209 ; 07-02 Dave Shull      17-May-1982
0000    210 ;       Added coverage in the Read Lock Function Test for the CML gate array.
0000    211 ; 07-03 Dave Shull      14-Jun-1982
0000    212 ;       Changed CMK callouts to CML for new gate array.
0000    213 ; 08-00 Joe Zagarella   11-Feb-1986
0000    214 ;       Enhanced the following tests to be compatible with the new memory
0000    215 ;       controller(L0022) and array card(M7199):
0000    216 ;       Test write bus error interrupt
0000    217 ;       Test for controller type and array configuration
0000    218 ;       Test data integrity for address test
0000    219 ;       Main memory dual addressing test
0000    220 ;       Array 0 memory test (part1 thru part4)
0000    221 ;       Array 1 memory test (part1 thru part4)
0000    222 ;
0000    223 ;--
0000      1         .SBTTL  ""
```

```
                     3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
                     4 ;++
                     5 ; Revision History
                     6 ;
                     7 ; 28    Dave Spain      22-Jun-82
                     8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
                     9 ;       Decided not to use them.
                    10 ;
                    11 ; 27    Dave Spain      15-Jun-82
                    12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
                    13 ;
                    14 ; 26    Dave Spain      19-May-82
                    15 ;       Added M[]_FPA macro.
                    16 ;
                    17 ; 25    Rich Lewis      23-Feb-82
                    18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
                    19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
                    20 ;
                    21 ; 24    Dave Spain      02-Feb-82
                    22 ;       Added FPA_R[]+M[] macro.
                    23 ;
                    24 ; 23    Bill Landry     22-OCT-81
                    25 ;       Added PSL_RDM macro
                    26 ;
                    27 ; 22    Dave Spain       5 Oct 81
                    28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
                    29 ;
                    30 ; 21    Bill Landry     29 Sept 81
                    31 ;       Added FPA macro, FPA_M[] FPA_R[].
                    32 ;
                    33 ; 20    Dave Spain      14 Aug 81
                    34 ;       Added the TESTFPA [] macro.
                    35 ;
                    36 ; 19    Dave Spain      18 July 81
                    37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
                    38 ;
                    39 ; 18    Dave Spain      24 Sept 80
                    40 ;       Added RDM_Q macro. This macro uses the ALU.
                    41 ;
                    42 ; 17    Bill Landry     23 Sept 80
                    43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
                    44 ;
                    45 ; 16    Dave Spain      27 August 80
                    46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
                    47 ;
                    48 ; 15    Bill Landry     4 June 80
                    49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
                    50 ;       satisfy validity checks.
                    51 ;
                    52 ; 14    Dave Spain      23 May 80
                    53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
                    54 ;       Bill.
                    55 ;
                    56 ; 13    Dave Spain      14 Apr 80
                    57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

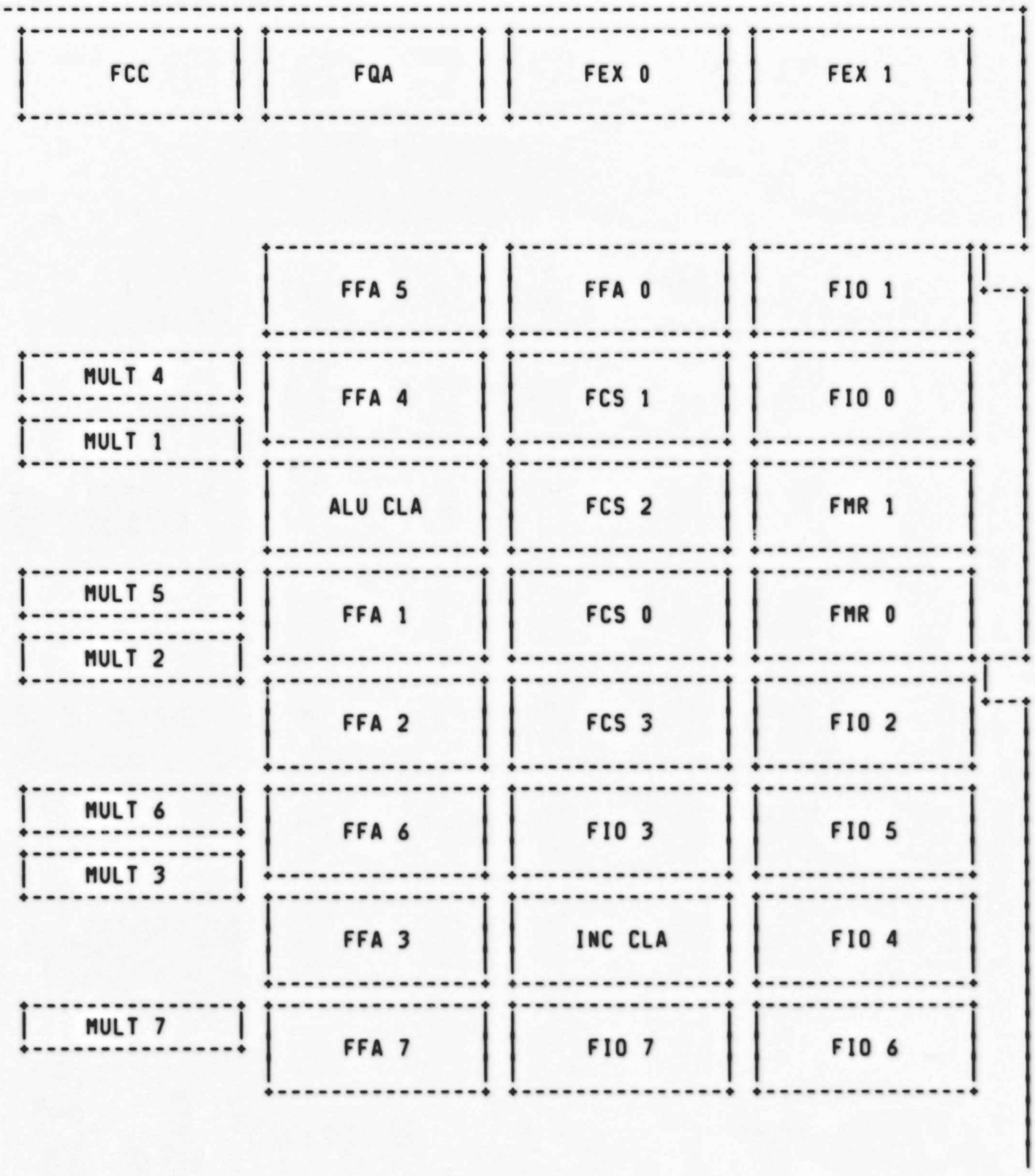
```
                     58 ;       the COMET micro-code defines to run.
                     59 ;
                     60 ; 12    Bill Landry      1 Apr 80
                     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
                     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
                     63 ;
                     64 ; 11    Dave Spain       27 Mar 80
                     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
                     66 ;
                     67 ; 10    Bill Landry      19 Mar 80
                     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
                     69 ;
                     70 ; 09    Bill Landry      19 Mar 80
                     71 ;       Deleted TB_R[].RL.P.OR.D
                     72 ;
                     73 ; 08    Bill Landry      18 Mar 80
                     74 ;       Added VA_R[].RL.P
                     75 ;
                     76 ; 07    Bill Landry      07 Mar 80
                     77 ;       Added Q_M[].RL.PTE
                     78 ;
                     79 ; 06    Dave Spain       06 Mar 80
                     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
                     81 ;
                     82 ; 05    Dave Spain       03 Mar 80
                     83 ;       Added TB_R[].RL.P macro for Bill Landry.
                     84 ;
                     85 ; 04    Dave Spain       22 Feb 80
                     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
                     87 ;
                     88 ; 03    Dave Spain       30 Jan 80
                     89 ;       Added Tom Groetzinger's macro definitions.
                     90 ;
                     91 ; 02    Dave Spain       30 Jan 80
                     92 ;       Added Tony Vezza's macro definitions.
                     93 ;
                     94 ; 01    Dave Spain       29 Jan 80
                     95 ;       Macros initiated.
                     96 ;--
                     97
                     98          .SBTTL  "       RDM Control File Macros"
                     99
                    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
                    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
                    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
                    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
                    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
                    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
                    106 ;RDM_CMI                                "RDMCF6/STROBE.CMI"
                    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
                    108
                    109          .SBTTL    "      Diagnostic Macros"
                    110
                    111 ;CLOCK.EXT                              "CLKX/XTND"
                    112 ;CLRCH.VA_RDM                           "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
                0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_ -
cont> WB"
                0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   115 ;                                        BUS/PRB.RD.PTE"
                0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   118 ;                                        BUS/PRB.RD.PTE"
                0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_ -
cont> WB"
                0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
                0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
                0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
                0000   124 ;                                        WCTRL/VA_WB"
                0000   125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
                0000   126 ;FPA_M[] FPA_R[]                       "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX -
cont> /R.S,
                0000   127 ;                                        ROT/ZERO,ALU/OR"
                0000   128 ;FPA_RDM                                "WB_RDM,FPA/FPA_DATA.WBUS"
                0000   129 ;FPA_R[]                                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_D -
cont> ATA.WBUS"
                0000   130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FP -
cont> A/FPA_DATA.WBUS"
                0000   131 ;FPA_WB                         "FPA/FPA_DATA.WBUS"
                0000   132 ;MEMSCAR_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
                0000   133 ;M[]_FPA                                "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
                0000   134 ;M[]_LONLIT                     "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
                0000   135 ;M[]_R[].OR.NIB0(M)             "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
                0000   136 ;                                        MUX/R.S,DQ1/NOP"
                0000   137 ;M[]_RDM                                "WB_RDM,SPW/MLONG,MSRC/a1"
                0000   138 ;PC_PC+RDM                      "WCTRL/PC_PC+WB,WB_RDM"
                0000   139 ;PC_PC+4 MB_XB                  "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
                0000   140 ;PC_PC+4 RDM_XB                 "RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
                0000   141 ;                                        MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
                0000   142 ;PC_RDM                         "WB_RDM,WCTRL/PC_WB"
                0000   143 ;PSL_RDM                                "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
                0000   144 ;PSL_WB                         "CCPSL/PSL_WB.CCBR_ALUS"
                0000   145 ;Q_D WX_R                       "ALPCTL/WX_R.Q_D"
                0000   146 ;Q_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
                0000   147 ;Q_M[].RL.PTE                   "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
                0000   148 ;Q_R[].AND.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
                0000   149 ;Q_R[].ASL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
                0000   150 ;Q_R[].OR.Q                     "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
                0000   151 ;Q_R[].RL.P                     "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
                0000   152 ;RDM_ALUF                       "ALPCTL/WB_ALUF,RDM_WB"
                0000   153 ;RDM_FPA                                "FPA/WBUS_FPA,RDM_WB"
                0000   154 ;RDM_LONLIT                     "WB_LONLIT,RDM_WB"
                0000   155 ;RDM_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
                0000   156 ;RDM_M[]                                "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
                0000   157 ;RDM_M[].OR.R[]                 "WB_M[a1].OR.R[a2],RDM_WB"
                0000   158 ;RDM_M[]-R[]                    "WB_M[a1]-R[a2],RDM_WB"
                0000   159 ;RDM_M[].XZ.OR.R[]              "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                0000   160 ;                               RDM_WB"
                0000   161 ;RDM_PC                         "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
                0000   162 ;RDM_PCBACK                     "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
                0000   163 ;RDM_Q                          "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
                0000   164 ;RDM_Q Q_D                      "ALPCTL/WX_Q.Q_D,RDM_WB"
```

```
0000   165 ;RDM_R[]                          "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                  "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                  "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

```
                0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
                0000   169 ;R[]_RDM                                "WB_RDM,SPW/RLONG,RSRC/a1"
                0000   170 ;R[].DT_Q                        "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
                0000   171 ;R[]_VA                          "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
                0000   172 ;R[]_WB                          "RSRC/a1,SPW/RLONG"
                0000   173 ;TB_RDM                          "WCTRL/TB_WB,WB_RDM,MSRC/VA"
                0000   174 ;TB_WB                           "WCTRL/TB_WB,MSRC/VA"
                0000   175 ;TESTFPA []                      "TESTFPA/a1"
                0000   176 ;VA_PC+LONLIT+I.PC_PC+I          "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/ -
cont> OR,
                0000   177 ;                                        ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
                0000   178 ;                                        SIZE[LONG]"
                0000   179 ;VA_Q.OR.R[]                     "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
                0000   180 ;VA_RDM                          "WB_RDM,WCTRL/VA_WB"
                0000   181 ;VA_R[].RL.P                     "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
                0000   182 ;VA_VA-ZLIT0[]                   "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
                0000   183 ;                                        MUX/M.S,ALU/A-B-CI"
                0000   184 ;VA_WB                           "WCTRL/VA_WB"
                0000   185 ;WB_FPA CCR                      "FPA/WBUS_FPA.CC"
                0000   186 ;WB_FPA TCR                      "WCTRL/FPTCR"
                0000   187 ;WB_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000   188 ;WB_LONLIT.Q_M                   "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000   189 ;WB_M[]+R[]                      "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
                0000   190 ;WB_R[].Q_D                      "RSRC/a1,ALPCTL/WX_R.Q_D"
                0000   191 ;WDR_R[]                                 "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/Z -
cont> ERO"
                0000   192 ;WRITE.LONG R[]                  "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
                0000   193 ;                                        MUX/R.S,ALU/OR,SIZE[LONG]"
                0000   194                 .SBTTL  ""
                0000   195                 .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                        .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```

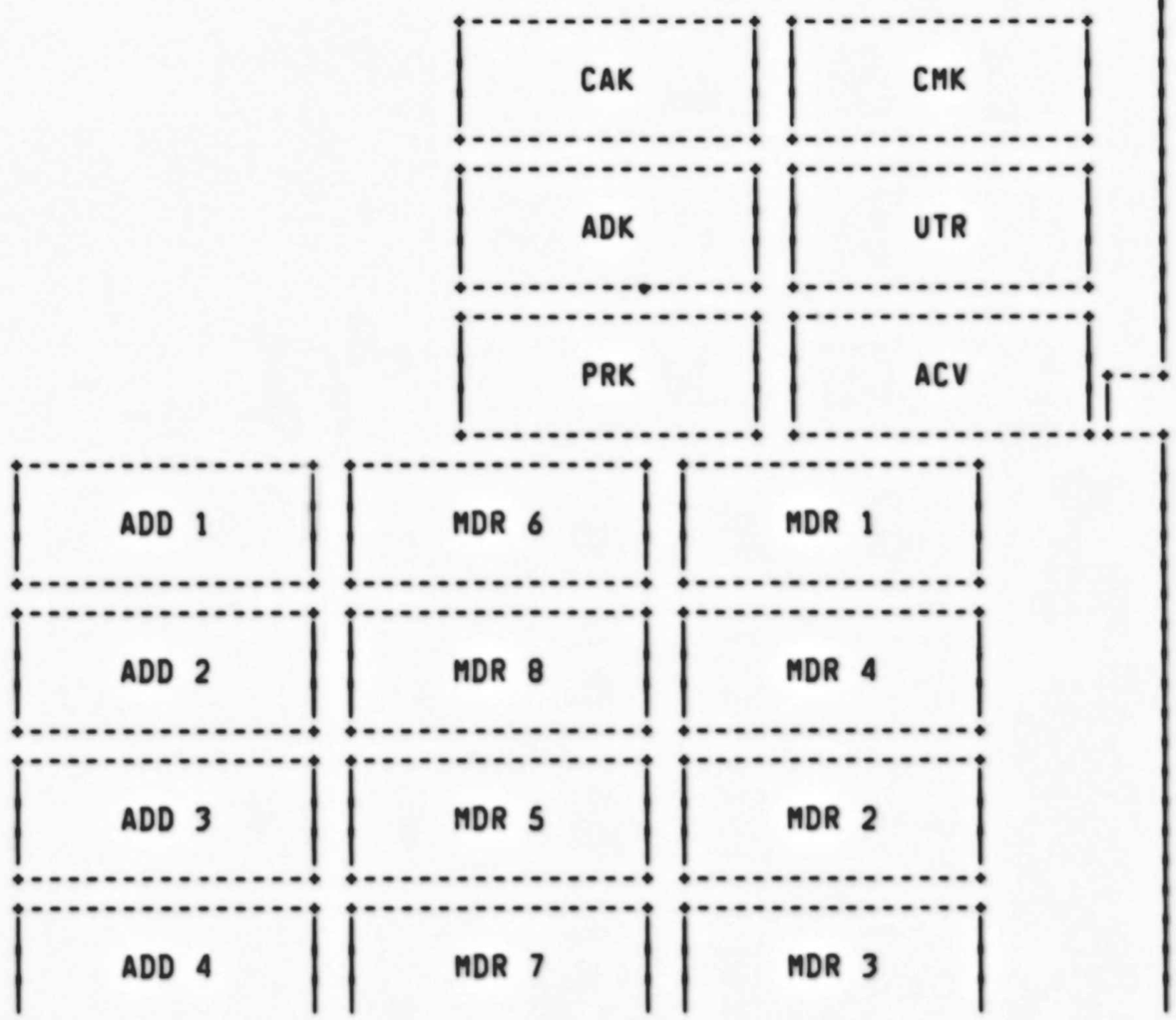


```
                0000   252 ;+-----------------------------------------------------------------------------+
```

```
0000    254          .SBTTL  "       L0002 GATE ARRAY LOCATION"
0000    255 ;+-----------------------------------------------------------------------+
0000    256 ;|                                                                       |
0000    257 ;|                                                                       |
0000    258 ;|                                                                       |
0000    259 ;|                                                                       |
0000    260 ;|                                                                       |
0000    261 ;|                                                                       |
0000    262 ;|                                                                       |
0000    263 ;|                                                                       |
0000    264 ;|                    +--------------+  +--------------+                |
0000    265 ;|                    |              |  |              |                |
0000    266 ;|                    |     PHB      |  |     MSQ      |                |
0000    267 ;|                    |              |  |              |                |
0000    268 ;|                    +--------------+  +--------------+                |
0000    269 ;|                    +--------------+  +--------------+                |
0000    270 ;|                    |              |  |              |                |
0000    271 ;|                    |     CCC      |  |     SAC      |                |
0000    272 ;|                    |              |  |              |                |
0000    273 ;|                    +--------------+  +--------------+                |
0000    274 ;|  +--------------+  +--------------+  +--------------+                |
0000    275 ;|  |              |  |              |  |              |                |
0000    276 ;|  |     SPA      |  |     SRK      |  |     IRD      |                |
0000    277 ;|  |              |  |              |  |              |                |
0000    278 ;|  +--------------+  +--------------+  +--------------+                |
0000    279 ;|  +--------------+  +--------------+  +--------------+                |
0000    280 ;|  |              |  |              |  |              |                |
0000    281 ;|  |     TOK      |  |     CLA      |  |     ALK      |                |
0000    282 ;|  |              |  |              |  |              |                |
0000    283 ;|  +--------------+  +--------------+  +--------------+                |
0000    284 ;|                    +--------------+  +--------------+                |
0000    285 ;|                    |              |  |              |                |
0000    286 ;|                    |    ALP 1     |  |    ALP 5     |                |
0000    287 ;|                    |              |  |              |                |
0000    288 ;|                    +--------------+  +--------------+                |
0000    289 ;|                    +--------------+  +--------------+                |
0000    290 ;|                    |              |  |              |                |
0000    291 ;|                    |    ALP 2     |  |    ALP 6     |                |
0000    292 ;|                    |              |  |              |                |
0000    293 ;|                    +--------------+  +--------------+                |
0000    294 ;|                    +--------------+  +--------------+                |
0000    295 ;|                    |              |  |              |                |
0000    296 ;|                    |    ALP 3     |  |    ALP 7     |                |
0000    297 ;|                    |              |  |              |                |
0000    298 ;|                    +--------------+  +--------------+                |
0000    299 ;|                    +--------------+  +--------------+                |
0000    300 ;|                    |              |  |              |                |
0000    301 ;|                    |    ALP 4     |  |    ALP 8     |                |
0000    302 ;|                    |              |  |              |                |
0000    303 ;|                    +--------------+  +--------------+                |
0000    304 ;|  +--------------+  +--------------+  +--------------+  +------------+ |
0000    305 ;|  |              |  |              |  |              |  |            | |
0000    306 ;|  |    SRM 1     |  |    SRM 2     |  |    SRM 3     |  |   SRM 4    | |
0000    307 ;|  |              |  |              |  |              |  |            | |
0000    308 ;|  +--------------+  +--------------+  +--------------+  +------------+ |
```

```
                0000   309 ;+-----------------------------------------------------------------------------+
```

```
                        311          .SBTTL  "       L0003 GATE ARRAY LOCATION"
                        312 ;+----------------------------------------------------------------------------------------------------------------------+
                        313 ;|                                                                                                                      |
                        314 ;|                                                                                                                      |
                        315 ;|                                                                                                                      |
                        316 ;|                                                                                                                      |
                        317 ;|                                                                                                                      |
                        318 ;|                                                                                                                      |
                        319 ;|                                                                                                                      |
                        320 ;|                                                                                                                      |
                        321 ;|                                                                                                                      |
                        322 ;|                                                                                                                      |
                        323 ;|                                                                                                                      |
                        324 ;|                                                                                                                      |
                        325 ;|                                                                                                                      |
                        326 ;|                                                                                                                      |
                        327 ;|                                                                                                                      |
                        328 ;|                                                                                                                      |
                        329 ;|                                                                                                                      |
                        330 ;|                                                                                                                      |
                        331 ;|                                                                                                                      |
                        332 ;|                                                                                                                      |
                        333 ;|                                                                                                                      |
                        334 ;|                                                                                                                      |
                        335 ;|                                                                                                                      |
                        336 ;|                                                                                                                      |
                        337 ;|                                                                                                                      |
                        338 ;|                                                                                                                      |
                        339 ;|                                                                                                                      |
                        340 ;|                                                                                                                      |
                        341 ;|                                                                                                                      |
                        342 ;|                                                                                                                      |
                        343 ;|                                                                                                                      |
                        344 ;|                                                                                                                      |
                        345 ;|                                                                                                                      |
                        346 ;|                                                                                                                      |
                        347 ;|                                                                                                                      |
                        348 ;|                                                                                                                      |
                        349 ;|                                                                                                                      |
                        350 ;|                                                                                                                      |
                        351 ;|                                                                                                                      |
                        352 ;|                                                                                                                      |
                        353 ;|                                                                                                                      |
                        354 ;|                                                                                                                      |
                        355 ;|                                                                                                                      |
                        356 ;|                                                                                                                      |
                        357 ;|                                                                                                                      |
                        358 ;|                                                                                                                      |
                        359 ;|                                                                                                                      |
                        360 ;|                                                                                                                      |
                        361 ;|                                                                                                                      |
                        362 ;|                                                                                                                      |
                        363 ;|                                                                                                                      |
                        364 ;|                                                                                                                      |
                        365 ;|                                                                                                                      |
```

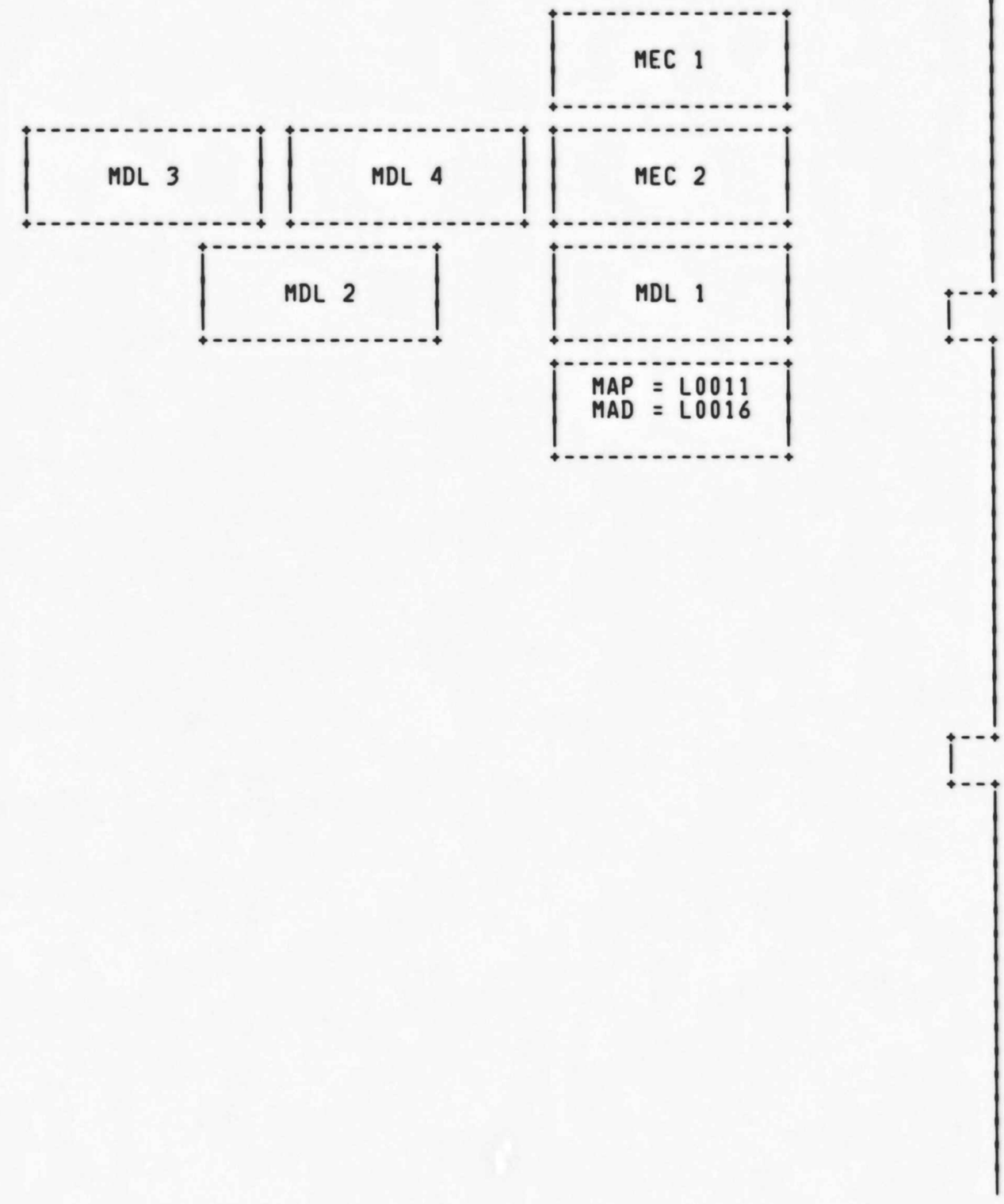


```
0000   366 ;|              +---------------+ +---------------+ +---------------+          |
0000   367 ;+-------------------------------------------------------------------------------+
```

```
                        0000    369          .SBTTL  "       L0004 GATE ARRAY LOCATION"
                        0000    370 ;+-----------------------------------------------------------------------------+
                        0000    371 ;|                                                                             |
                        0000    372 ;|                                                                             |
                        0000    373 ;|                                                                             |
                        0000    374 ;|                                                                             |
                        0000    375 ;|                                                                             |
                        0000    376 ;|                                                                             |
                        0000    377 ;|                                                                             |
                        0000    378 ;|                                                                             |
                        0000    379 ;|                                                                             |
                        0000    380 ;|                                                                             |
                        0000    381 ;|                          +---------------+                                  |
                        0000    382 ;|                          |               |                                  |
                        0000    383 ;|                          |      UCN      |                                  |
                        0000    384 ;|                          |               |                                  |
                        0000    385 ;|                          +---------------+                                  |
                        0000    386 ;|                          +---------------+                                  |
                        0000    387 ;|                          |               |                                  |
                        0000    388 ;|                          |     UDP 1     |                                  |
                        0000    389 ;|                          |               |                                  |
                        0000    390 ;|                          +---------------+                                  |
                        0000    391 ;|                          +---------------+                                  |
                        0000    392 ;|                          |               |                                  |
                        0000    393 ;|                          |     UDP 2     |                                  |
                        0000    394 ;|                          |               |                                  |
                        0000    395 ;|                          +---------------+                                  |
                        0000    396 ;|                          +---------------+                                  |
                        0000    397 ;|                          |               |                                  |
                        0000    398 ;|                          |     UDP 4     |                                  |
                        0000    399 ;|                          |               |                                  |
                        0000    400 ;|                          +---------------+                                  |
                        0000    401 ;|                          +---------------+                                  |
                        0000    402 ;|                          |               |                                  |
                        0000    403 ;|                          |     UDP 3     |                                  |
                        0000    404 ;|                          |               |                                  |
                        0000    405 ;|                          +---------------+                                  |
                        0000    406 ;|                          +---------------+                                  |
                        0000    407 ;|                          |               |                                  |
                        0000    408 ;|                          |      INT      |                                  |
                        0000    409 ;|                          |               |                                  |
                        0000    410 ;|                          +---------------+                                  |
                        0000    411 ;|                          +---------------+                                  |
                        0000    412 ;|                          |               |                                  |
                        0000    413 ;|                          |     CON 1     |                                  |
                        0000    414 ;|                          |               |                                  |
                        0000    415 ;|                          +---------------+                                  |
                        0000    416 ;|                          +---------------+                                  |
                        0000    417 ;|                          |               |                                  |
                        0000    418 ;|                          |     CON 2     |                                  |
                        0000    419 ;|                          |               |                                  |
                        0000    420 ;|                          +---------------+                                  |
                        0000    421 ;|                                                                             |
                        0000    422 ;|                                                                             |
                        0000    423 ;|                                                                             |
```

```
0000   424 ;|                                                                    |
0000   425 ;+-----------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
```



```
0000   482 ;|                                                                                |
0000   483 ;+--------------------------------------------------------------------------------+
```

```
                     0000   485          .SBTTL  ""
                     0000   486          .LIST   MC
                     0000   487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                     0000   488          .LIST   ME
                     0000   489          .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
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                              ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_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    491         .NLIST  ME
          0000    492         .LIST   MC
          0000    493         .SBTTL  ""
          0000    494         .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
0020          .SBTTL  "TEST   031     TB WRITE PULSE LOGIC TEST 6
0020       3 ;*****************************************************************************
0020       4 ;++
0020       5 ;
0020       6 ;       TB WRITE PULSE LOGIC TEST 6
0020       7 ;
0020       8 ; FUNCTIONAL DESCRIPTION
0020       9 ;       THIS IS A TEST OF THE TRANSLATION BUFFER RANDOM REPLACE LOGIC.
0020      10 ;       TESTED HERE IS THE RESULT OF RANDOM REPLACEMENT STARTING WITH A
0020      11 ;       REPLACEMENT IN GROUP 1 FOLLOWED BY A REPLACEMENT IN GROUP 0.
0020      12 ;
0020      13 ; TEST ALGORITHM
0020      14 ;       1.  CREATE A TEST LOOP OF 2 TO SELECT TEST ADDRESSES
0020      15 ;       2.  LOAD A TEST ADDRESS INTO THE RDM WD REG
0020      16 ;       3.  EXECUTE DCS PROGRAM
0020      17 ;               a.  INVALIDATE THE TB
0020      18 ;               b.  SET MISS GROUP 0 AND REPLACE GROUP 1
0020      19 ;               c.  LOAD VA WITH 2AAA0000
0020      20 ;               d.  WRITE TB GROUP 1 WITH 00555550
0020      21 ;               e.  SET MISS GROUP 1 AND REPLACE GROUP 0
0020      22 ;               f.  LOAD VA WITH 55550000
0020      23 ;               g.  WRITE TB GROUP 0 WITH 00333330
0020      24 ;               h.  ENABLE RANDOM REPLACEMENT
0020      25 ;               i.  LOAD VA WITH 33330000
0020      26 ;               j.  WRITE TB WITH 00999990
0020      27 ;               k.  LOAD VA WITH 0F0F0000
0020      28 ;               l.  WRITE TB WITH 00DDDDD0
0020      29 ;               m.  LOAD VA FROM RDM WD REGISTER
0020      30 ;               n.  READ TB TO RDM
0020      31 ;       4.  TEST RDM WH REGISTER FOR CORRECT PTE
0020      32 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESS
0020      33 ;
0020      34 ;
0020      35 ; LOGIC DESCRIPTION
0020      36 ;       THIS TESTS LOGIC IN THE ADK GATE ARRAY.
0020      37 ;
0020      38 ; ASSUMPTIONS
0020      39 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
0020      40 ;
0020      41 ; ERROR DESCRIPTION
0020      42 ;       EXPECTED PTE
0020      43 ;       GOT PTE
0020      44 ;
0020      45 ;--
0020      46 ;*****************************************************************************
0020      47 ;/////////////////////////////////////////////////////////////////////////////
0020      48
0020      49          INITIALIZE
0022      50          LOOP            I,1,2                   ;CREATE A TEST LOOP OF 2
0029      51          LOADREG         WDREG,1$,I,L            ;LOAD TEST ADDRESS
0031      52          ERRLOOP                                 ;ERROR LOOP POINT
0033      53          FETCH           0                       ;EXECUTE THE UCODE.
0038      54          BRSTCLK         <^X22>
003C      55          CMPREGMSK       WHREG,2$,I,3$,,L        ;CHECK THE PTE.
0048      56          IFERROR         ,<<ADK>>,               ;INCORRECT PTE
```

```
                                 004E    57          ENDLOOP         I                          ;END TEST LOOP
                                 0051    58          SKIP                                       ;GO TO NEXT TEST
                                 0058    59
                                 0058    60 BEGIN_TEST_DATA
                                 0058
                                 0058        ;-------------------------------------------------------------------------
                                 0058    61
                      33330000   0058    62 1$:      .LONG   ^X33330000                         ;TEST ADDRESS TABLE
                      0F0F0000   005C    63          .LONG   ^X0F0F0000
                                 0060    64
                      00999990   0060    65 2$:      .LONG   ^X00999990                         ;COMPARE DATA
                      00DDDDD0   0064    66          .LONG   ^X00DDDDD0
                                 0068    67
                      00FFFFF8   0068    68 3$:      .LONG   ^X00FFFFF8                         ;MASK FOR CMPREGMSK PSEUDO OP
                                 006C    69
                                 006C    70 BEGIN_CPU_DATA
                                 0002    71
                                 0002    72 DCS_DATA
                                 0001    73
U 00. 7700.C800.0364.0300.0470.1801   0001    74 ;* 00   NOP
U 01. 7700.8800.5BE6.03D8.0470.1802   000C    78 ;* 01   D_R[ZERO]                               ; zero the D register
U 02. 7700.D800.DB13.0300.4A70.1803   0017    82 ;* 02   VA_Q_D_D+ZLIT8[0]                      ; invalidate the TB at 0 &
U 03. 7700.1800.DB12.0300.5360.1804   0022    86 ;* 03   VA_D_D+ZLIT8[0] INVALIDATE TB         ; 200.....
U 04. 7700.F800.7E7F.FFFF.8470.1805   002D    90 ;* 04   LONLIT_[03000000]                      ; SET FORCE MISS GROUP 0 AND
U 05. 7700.4800.5BE4.03D4.6870.1806   0038    94 ;* 05   MEMSCAR_R[LONLIT]                     ; FORCE REPLACE GROUP 1.
U 06. 7700.3800.797F.FFFF.8470.1807   0043    98 ;* 06   LONLIT_[0D000000]
U 07. 7700.4800.5BE4.03D4.6070.1808   004E   102 ;* 07   MEMSCR_R[LONLIT]
U 08. 7700.B800.6AAA.FFFF.8470.1809   0059   106 ;* 08   LONLIT_[2AAA0000]                      ; LOAD THE VA WITH 2AAA0000.
U 09. 7700.4800.5BE4.03D4.4A70.180A   0064   110 ;* 09   VA_R[LONLIT]
U 0A. 7700.7800.7FD5.5557.8470.180B   006F   114 ;* 0A   LONLIT_[00555550]                      ; WRITE VA=2AAA0000 AND
U 0B. 7700.C81B.5BE4.03D4.5070.180C   007A   118 ;* 0B   TB_R[LONLIT]                          ; PTE=00555550 INTO TB GROUP 1.
U 0C. 7700.7800.7E7F.FFFF.8470.180D   0085   122 ;* 0C   LONLIT_[03000000]                      ; SET FORCE MISS GROUP 1 AND
U 0D. 7700.C800.5BE4.03D4.6870.180E   0090   126 ;* 0D   MEMSCAR_R[LONLIT]                     ; FORCE REPLACE GROUP 0.
U 0E. 7700.F800.7AFF.FFFF.8470.180F   009B   130 ;* 0E   LONLIT_[0A000000]
U 0F. 7700.4800.5BE4.03D4.6070.1810   00A6   134 ;* 0F   MEMSCR_R[LONLIT]
U 10. 7700.F800.5555.7FFF.8470.1811   00B1   138 ;* 10   LONLIT_[55550000]                      ; LOAD THE VA WITH 55550000.
U 11. 7700.4800.5BE4.03D4.4A70.1812   00BC   142 ;* 11   VA_R[LONLIT]
U 12. 7700.7800.7FE6.6667.8470.1813   00C7   146 ;* 12   LONLIT_[00333330]                      ; WRITE VA=55550000 AND
U 13. 7700.C81B.5BE4.03D4.5070.1814   00D2   150 ;* 13   TB_R[LONLIT]                          ; PTE=00333330 INTO TB GROUP 0.
U 14. 7700.7800.7E7F.FFFF.8470.1815   00DD   154 ;* 14   LONLIT_[03000000]                      ; ENABLE BOTH TB GROUPS
U 15. 7700.C800.5BE4.03D4.6870.1816   00E8   158 ;* 15   MEMSCAR_R[LONLIT]                     ; WITH RANDOM REPLACEMENT.
U 16. 7700.F800.7FFF.FFFF.8470.1817   00F3   162 ;* 16   LONLIT_[00000000]
U 17. 7700.C800.5BE4.03D4.6070.1818   00FE   166 ;* 17   MEMSCR_R[LONLIT]
U 18. 7700.F800.6666.7FFF.8470.9819   0109   170 ;* 18   LONLIT_[33330000],CLOCK.EXT           ; LOAD THE VA WITH 33330000.
U 19. 7700.C800.5BE4.03D4.4A70.181A   0114   174 ;* 19   VA_R[LONLIT]
U 1A. 7700.F800.7FB3.3337.8470.181B   011F   178 ;* 1A   LONLIT_[00999990]                      ; WRITE VA=33330000 WITH
U 1B. 7700.481B.5BE4.03D4.5070.181C   012A   182 ;* 1B   TB_R[LONLIT]                          ; PTE=00999990 INTO THE TB. THIS
                                      0135   186                                              ; WRITE SHOULD GO INTO GROUP 1.
U 1C. 7700.7800.7878.7FFF.8470.981D   0135   187 ;* 1C   LONLIT_[0F0F0000],CLOCK.EXT           ; WRITE VA=0F0F0000 AND
U 1D. 7700.4800.5BE4.03D4.4A70.181E   0140   191 ;* 1D   VA_R[LONLIT]                          ; PTE=00DDDDD0 INTO THE TB.
U 1E. 7700.3800.7F91.1117.8470.181F   014B   195 ;* 1E   LONLIT_[00DDDDD0]
U 1F. 7700.481B.5BE4.03D4.5070.1820   0156   199 ;* 1F   TB_R[LONLIT]                          ; WRITE GROUP 0
U 20. 5700.4800.0364.0300.4A70.1821   0161   203 ;* 20   VA_RDM                                ; LOAD THE VA WITH THE TEST
U 21. 6701.081F.5924.0200.05F0.1822   016C   207 ;* 21   RDM_TB                               ; ADDRESS AND READ THE TB AT
U 22. 7300.C800.0364.0300.0470.1823   0177   211 ;* 22   STOP.CLOCK                           ; THAT ADDRESS.
```

```
0182   215
0182   216 END_CPU_DATA
006C   217
006C   218 END_TEST_DATA
006C
006C       ;-----------------------------------------------------------------------------
006C   219
006C   220 ENDTEST
```

```
                        0025        .SBTTL  "TEST   032     TB GROUP 0 HIT COMPARATOR TEST 1
                        0025    222 ;*******************************************************************************
                        0025    223 ;++
                        0025    224 ;
                        0025    225 ;       TB GROUP 0 HIT COMPARATOR TEST 1
                        0025    226 ;
                        0025    227 ; FUNCTIONAL DESCRIPTION:
                        0025    228 ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 0
                        0025    229 ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
                        0025    230 ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
                        0025    231 ;
                        0025    232 ;
                        0025    233 ; TEST ALGORITHM:
                        0025    234 ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
                        0025    235 ;       2.  LOAD TEST ADDRESS INTO RDM WD REGISTER
                        0025    236 ;       3.  EXECUTE DCS PROGRAM
                        0025    237 ;               a.  INVALIDATE THE TB
                        0025    238 ;               b.  SET MISS GROUP 1 AND REPLACE GROUP 0
                        0025    239 ;               c.  LOAD TEST ADDRESS INTO VA REGISTER
                        0025    240 ;               d.  WRITE 00555550 INTO TB GROUP 0
                        0025    241 ;               e.  LOAD VA REGISTER WITH 0 (SUBTEST 2)
                        0025    242 ;               f.  READ TB TO RDM WH REGISTER
                        0025    243 ;       4.  TEST RDM WH REGISTER FOR A TB HIT IN SUBTEST 1 AND A TB
                        0025    244 ;           MISS IN SUBTEST 2
                        0025    245 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
                        0025    246 ;
                        0025    247 ;
                        0025    248 ; TEST PATTERNS:
                        0025    249 ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 1
                        0025    250 ;       THRU 0'S IN VA BITS <30:16>, THE VA TAG FIELD.
                        0025    251 ;       1.      00010000
                        0025    252 ;       2.      00020000
                        0025    253 ;       3.      00040000
                        0025    254 ;       4.      00080000
                        0025    255 ;       5.      00100000
                        0025    256 ;       6.      00200000
                        0025    257 ;       7.      00400000
                        0025    258 ;       8.      00800000
                        0025    259 ;       9.      01000000
                        0025    260 ;       10.     02000000
                        0025    261 ;       11.     04000000
                        0025    262 ;       12.     08000000
                        0025    263 ;       13.     10000000
                        0025    264 ;       14.     20000000
                        0025    265 ;       15.     40000000
                        0025    266 ;
                        0025    267 ; LOGIC DESCRIPTION:
                        0025    268 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 0.
                        0025    269 ;
                        0025    270 ;
                        0025    271 ; ASSUMPTIONS:
                        0025    272 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
                        0025    273 ;
                        0025    274 ;
                        0025    275 ; ERROR DESCRIPTION:
```

```
                     276 ;       EXPECTED PTE
                     277 ;       GOT PTE
                     278 ;       INDEX IDENTIFYING WHICH TEST PATTERN FAILED
                     279 ;
                     280 ;--
                     281 ;*****************************************************************************
                     282
                     283         SUBTEST                                         ; SUBTEST #1
                         ;/////////////////////////////////////////////////////////////////////////////
                     284 ;+
                     285 ;SUBTEST #1
                     286 ;
                     287 ;       TEST FOR TB HITS WITH ALL TEST ADDRESSES
                     288 ;-
                     289         INITIALIZE                                      ; INIT THE CPU
                     290         LOADDCS         7$,,<^X0B>,2                    ; MODIFY DCS U-CODE
                     291         LOOP            I,1,15                          ; SET UP FOR 15 ITERATIONS.
                     292         LOADREG         WDREG,1$,I,L                    ; LOAD TEST PATTERN.
                     293         ERRLOOP
                     294         FETCH           0                               ; EXECUTE THE UCODE.
                     295         BRSTCLK         <^X0C>
                     296         CMPREGMSK       WHREG,2$,,3$,,L                 ; CHECK THE PTE.
                     297         IFERROR         ,<<ADK>>,                       ; INCORRECT PTE
                     298         ENDLOOP         I                               ; END TEST LOOP
                     299
                     300         SUBTEST                                         ; SUBTEST #2
                         ;/////////////////////////////////////////////////////////////////////////////
                     301 ;+
                     302 ;SUBTEST #2
                     303 ;
                     304 ;       TEST FOR TB MISS WITH ALL TEST ADDRESSES AND ZERO
                     305 ;-
                     306         INITIALIZE                                      ; INIT THE CPU
                     307         LOADDCS         4$,,<^X0B>,4                    ; MODIFY DCS U-CODE
                     308         LOOP            I,1,15                          ; CREATE A TEST LOOP OF 15
                     309         LOADREG         WDREG,1$,I,L                    ; TEST ADDRESS TO RDM WD REG
                     310         ERRLOOP                                         ; ERROR LOOP POINT
                     311         FETCH           0                               ; START DCS PROGRAM
                     312         BRSTCLK         <^X0E>                          ; END DCS PROGRAM
                     313         CMPREGMSK       WHREG,5$,,6$,,L,                ; TEST FOR TB MISS
                     314         IFERROR         ,<<ADK>>,                       ; FAILING ARRAY
                     315         ENDLOOP         I                               ; END TEST LOOP
                     316         SKIP                                            ; GO TO NEXT TEST
                     317
                     318 BEGIN_TEST_DATA
                         ;-----------------------------------------------------------------------------
                     319
00010000             320 1$:     .LONG   ^X00010000                              ; TEST ADDRESS TABLE
00020000             321         .LONG   ^X00020000
00040000             322         .LONG   ^X00040000
00080000             323         .LONG   ^X00080000
00100000             324         .LONG   ^X00100000
```

```
                    00200000  00B6   325          .LONG   ^X00200000
                    00400000  00BA   326          .LONG   ^X00400000
                    00800000  00BE   327          .LONG   ^X00800000
                    01000000  00C2   328          .LONG   ^X01000000
                    02000000  00C6   329          .LONG   ^X02000000
                    04000000  00CA   330          .LONG   ^X04000000
                    08000000  00CE   331          .LONG   ^X08000000
                    10000000  00D2   332          .LONG   ^X10000000
                    20000000  00D6   333          .LONG   ^X20000000
                    40000000  00DA   334          .LONG   ^X40000000
                              00DE   335
                    00555550  00DE   336 2$:      .LONG   ^X00555550                      ; COMPARE DATA FOR SUBTEST #1
                    00FFFFF8  00E2   337 3$:      .LONG   ^X00FFFFF8                      ; MASK FOR CMPREGMSK PSEUDO OP
                              00E6   338
                              00E6   339 4$:
U 0B, 7700,F800,7FFF,FFFF,8470,180C   00E6   340 ;% 0B   LONLIT_[0]                      ; U-CODE FOR SUBTEST #2
U 0C, 7700,C800,5BE4,03D4,4A70,180D   00F1   344 ;% 0C   VA_R[LONLIT]
U 0D, 6701,881F,5924,0200,05F0,180E   00FC   348 ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F   0107   352 ;% 0E   NOP,STOP.CLOCK
                              0112   356
                    00000000  0112   357 5$:      .LONG   ^X00000000                      ; COMPARE DATA FOR SUBTEST #2
                    00000100  0116   358 6$:      .LONG   ^X00000100                      ; MASK FOR CMPREGMSK PSEUDO OP
                              011A   359
                              011A   360 7$:
U 0B, 6701,081F,5924,0200,05F0,180C   011A   361 ;% 0B   RDM_TB                          ; U-CODE FOR SUBTEST #1
U 0C, 7300,C800,0364,0300,0470,180D   0125   365 ;% 0C   NOP,STOP.CLOCK
                              0130   369
                              0130   370 BEGIN_CPU_DATA
                              0002   371
                              0002   372 DCS_DATA
                              0001   373
U 00, 7700,C800,0364,0300,0470,1801   0001   374 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802   000C   378 ;% 01   D_R[ZERO]                        ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803   0017   382 ;% 02   VA_Q_D_D+ZLIT8[0]               ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804   0022   386 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB  ; 200.....
U 04, 7700,F800,7E7F,FFFF,8470,1805   002D   390 ;% 04   LONLIT_[03000000]              ; SET FORCE MISS GROUP 1 AND
U 05, 7700,4800,5BE4,03D4,6870,1806   0038   394 ;% 05   MEMSCAR_R[LONLIT]             ; FORCE REPLACE GROUP 0.
U 06, 7700,7800,7AFF,FFFF,8470,1807   0043   398 ;% 06   LONLIT_[0A000000]
U 07, 7700,4800,5BE4,03D4,6070,1808   004E   402 ;% 07   MEMSCR_R[LONLIT]
U 08, 5700,4800,0364,0300,4A70,1809   0059   406 ;% 08   VA_RDM                         ; LOAD THE TEST PATTERN INTO VA.
U 09, 7700,F800,7FD5,5557,8470,180A   0064   410 ;% 09   LONLIT_[00555550]            ; MAKE THE TEST PATTERN AND
U 0A, 7700,481B,5BE4,03D4,5070,180B   006F   414 ;% 0A   TB_R[LONLIT]                  ; PTE=00555550 A HIT IN GROUP 0.
U 0B, 6701,081F,5924,0200,05F0,180C   007A   418 ;% 0B   RDM_TB                        ; READ THE TB AT THAT ADDRESS
U 0C, 7300,C800,0364,0300,0470,180D   0085   422 ;% 0C   NOP,STOP.CLOCK                ; STOP THE CPU CLOCK
                              0090   426
                              0090   427 END_CPU_DATA
                              0130   428
                              0130   429 END_TEST_DATA
                              0130
                              0130       ;-------------------------------------------------------------------------
                              0130   430
                              0130   431 ENDTEST
```

```
                        0025        .SBTTL  "TEST   033     TB GROUP 1 HIT COMPARATOR TEST 1
                        0025    433 ;*******************************************************************************
                        0025    434 ;++
                        0025    435 ;
                        0025    436 ;       TB GROUP 1 HIT COMPARATOR TEST 1
                        0025    437 ;
                        0025    438 ; FUNCTIONAL DESCRIPTION:
                        0025    439 ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 1
                        0025    440 ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
                        0025    441 ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
                        0025    442 ;
                        0025    443 ;
                        0025    444 ; TEST ALGORITHM:
                        0025    445 ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
                        0025    446 ;       2.  LOAD TEST ADDRESS INTO RDM WD REGISTER
                        0025    447 ;       3.  EXECUTE DCS PROGRAM
                        0025    448 ;               a.  INVALIDATE THE TB
                        0025    449 ;               b.  SET MISS GROUP 0 AND REPLACE GROUP 1
                        0025    450 ;               c.  LOAD TEST ADDRESS INTO VA REGISTER
                        0025    451 ;               d.  WRITE 00555550 INTO TB GROUP 1
                        0025    452 ;               e.  LOAD VA REGISTER WITH 0 (SUBTEST 2)
                        0025    453 ;               f.  READ TB TO RDM WH REGISTER
                        0025    454 ;       4.  TEST RDM WH REGISTER FOR A TB HIT IN SUBTEST 1 AND A TB
                        0025    455 ;           MISS IN SUBTEST 2
                        0025    456 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
                        0025    457 ;
                        0025    458 ;
                        0025    459 ; TEST PATTERNS:
                        0025    460 ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 1
                        0025    461 ;       THRU 0'S IN VA BITS <30:16>, THE VA TAG FIELD.
                        0025    462 ;       1.      00010000
                        0025    463 ;       2.      00020000
                        0025    464 ;       3.      00040000
                        0025    465 ;       4.      00080000
                        0025    466 ;       5.      00100000
                        0025    467 ;       6.      00200000
                        0025    468 ;       7.      00400000
                        0025    469 ;       8.      00800000
                        0025    470 ;       9.      01000000
                        0025    471 ;       10.     02000000
                        0025    472 ;       11.     04000000
                        0025    473 ;       12.     08000000
                        0025    474 ;       13.     10000000
                        0025    475 ;       14.     20000000
                        0025    476 ;       15.     40000000
                        0025    477 ;
                        0025    478 ;
                        0025    479 ; LOGIC DESCRIPTION:
                        0025    480 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 1.
                        0025    481 ;
                        0025    482 ;
                        0025    483 ; ASSUMPTIONS:
                        0025    484 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
                        0025    485 ;
                        0025    486 ;
```

```
                     0025   487 ; ERROR DESCRIPTION:
                     0025   488 ;       EXPECTED PTE
                     0025   489 ;       GOT PTE
                     0025   490 ;       INDEX IDENTIFYING WHICH TEST PATTERN FAILED
                     0025   491 ;
                     0025   492 ;--
                     0025   493 ;*******************************************************************************
                     0025   494
                     0025   495         SUBTEST                                         ; SUBTEST #1
                     0028
                     0028           ;////////////////////////////////////////////////////////////////////////////
                     0028   496 ;+
                     0028   497 ;SUBTEST #1
                     0028   498 ;
                     0028   499 ;       TEST FOR TB HITS WITH ALL TEST ADDRESSES
                     0028   500 ;-
                     0028   501         INITIALIZE                                      ; INIT THE CPU
                     002A   502         LOADDCS         7$,,<^X0B>,2                    ; MODIFY DCS U-CODE
                     0031   503         LOOP            I,1,15                          ; SET UP FOR 15 ITERATIONS.
                     0038   504         LOADREG         WDREG,1$,I,L                    ; LOAD TEST PATTERN.
                     0040   505         ERRLOOP
                     0042   506         FETCH           0                               ; EXECUTE THE UCODE.
                     0047   507         BRSTCLK         <^X0C>
                     004B   508         CMPREGMSK       WHREG,2$,,3$,,L                 ; CHECK THE PTE.
                     0057   509         IFERROR         ,<<ADK>>,                       ; INCORRET PTE
                     005D   510         ENDLOOP         I                               ; END TEST LOOP
                     0060   511
                     0060   512         SUBTEST                                         ; SUBTEST #2
                     0063
                     0063           ;////////////////////////////////////////////////////////////////////////////
                     0063   513 ;+
                     0063   514 ;SUBTEST #2
                     0063   515 ;
                     0063   516 ;       TEST FOR TB MISS WITH ALL TEST ADDRESSES AND ZERO
                     0063   517 ;-
                     0063   518         INITIALIZE                                      ; INIT THE CPU
                     0065   519         LOADDCS         4$,,<^X0B>,4                    ; MODIFY DCS U-CODE
                     006C   520         LOOP            I,1,15                          ; CREATE A TEST LOOP OF 15
                     0073   521         LOADREG         WDREG,1$,I,L                    ; TEST ADDRESS TO RDM WD REG
                     007B   522         ERRLOOP                                         ; ERROR LOOP POINT
                     007D   523         FETCH           0                               ; START DCS PROGRAM
                     0082   524         BRSTCLK         <^X0E>                          ; END DCS PROGRAM
                     0086   525         CMPREGMSK       WHREG,5$,,6$,,L,                ; TEST FOR TB MISS
                     0092   526         IFERROR         ,<<ADK>>,                       ; FAILING ARRAY
                     0098   527         ENDLOOP         I                               ; END TEST LOOP
                     009B   528         SKIP                                            ; GO TO NEXT TEST
                     00A2   529
                     00A2   530 BEGIN_TEST_DATA
                     00A2
                     00A2           ;------------------------------------------------------------------------
                     00A2   531
00010000             00A2   532 1$:      .LONG   ^X00010000                             ; TEST ADDRESS TABLE
00020000             00A6   533          .LONG   ^X00020000
00040000             00AA   534          .LONG   ^X00040000
00080000             00AE   535          .LONG   ^X00080000
```

```
                    00100000  00B2   536          .LONG   ^X00100000
                    00200000  00B6   537          .LONG   ^X00200000
                    00400000  00BA   538          .LONG   ^X00400000
                    00800000  00BE   539          .LONG   ^X00800000
                    01000000  00C2   540          .LONG   ^X01000000
                    02000000  00C6   541          .LONG   ^X02000000
                    04000000  00CA   542          .LONG   ^X04000000
                    08000000  00CE   543          .LONG   ^X08000000
                    10000000  00D2   544          .LONG   ^X10000000
                    20000000  00D6   545          .LONG   ^X20000000
                    40000000  00DA   546          .LONG   ^X40000000
                              00DE   547
                    00555550  00DE   548 2$:      .LONG   ^X00555550                      ; COMPARE DATA SUBTEST #1
                    00FFFFF8  00E2   549 3$:      .LONG   ^X00FFFFF8                      ; MASK FOR CMPREGMSK PSEUDO OP
                              00E6   550
                              00E6   551 4$:
U 0B, 7700,F800,7FFF,FFFF,8470,180C   00E6   552 ;% 0B   LONLIT_[0]                      ; U-CODE FOR SUBTEST #2
U 0C, 7700,C800,5BE4,03D4,4A70,180D   00F1   556 ;% 0C   VA_R[LONLIT]
U 0D, 6701,881F,5924,0200,05F0,180E   00FC   560 ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F   0107   564 ;% 0E   NOP,STOP.CLOCK
                              0112   568
                    00000000  0112   569 5$:      .LONG   ^X00000000                      ; COMPARE DATA SUBTEST #2
                    00000100  0116   570 6$:      .LONG   ^X00000100                      ; MASK FOR CMPREGMSK PSEUDO OP
                              011A   571
                              011A   572 7$:
U 0B, 6701,081F,5924,0200,05F0,180C   011A   573 ;% 0B   RDM_TB                          ; U-CODE FOR SUBTEST #1
U 0C, 7300,C800,0364,0300,0470,180D   0125   577 ;% 0C   NOP,STOP.CLOCK
                              0130   581
                              0130   582 BEGIN_CPU_DATA
                              0002   583
                              0002   584 DCS_DATA
                              0001   585
U 00, 7700,C800,0364,0300,0470,1801   0001   586 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802   000C   590 ;% 01   D_R[ZERO]                        ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803   0017   594 ;% 02   VA_Q_D_D+ZLIT8[0]               ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804   0022   598 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB  ; 200.....
U 04, 7700,F800,7E7F,FFFF,8470,1805   002D   602 ;% 04   LONLIT_[03000000]              ; SET FORCE MISS GROUP 0 AND
U 05, 7700,4800,5BE4,03D4,6870,1806   0038   606 ;% 05   MEMSCAR_R[LONLIT]             ; FORCE REPLACE GROUP 1.
U 06, 7700,3800,797F,FFFF,8470,1807   0043   610 ;% 06   LONLIT_[0D000000]
U 07, 7700,4800,5BE4,03D4,6070,1808   004E   614 ;% 07   MEMSCR_R[LONLIT]
U 08, 5700,4800,0364,0300,4A70,1809   0059   618 ;% 08   VA_RDM                         ; LOAD THE TEST PATTERN INTO VA.
U 09, 7700,F800,7FD5,5557,8470,180A   0064   622 ;% 09   LONLIT_[00555550]            ; MAKE THE TEST PATTERN AND
U 0A, 7700,481B,5BE4,03D4,5070,180B   006F   626 ;% 0A   TB_R[LONLIT]                  ; PTE=00555550 A HIT IN GROUP 1.
U 0B, 6701,081F,5924,0200,05F0,180C   007A   630 ;% 0B   RDM_TB                        ; ADDRESS AND READ THE TB
U 0C, 7300,C800,0364,0300,0470,180D   0085   634 ;% 0C   STOP.CLOCK                    ; STOP THE CPU CLOCK
                              0090   638
                              0090   639 END_CPU_DATA
                              0130   640
                              0130   641 END_TEST_DATA
                              0130
                              0130       ;-----------------------------------------------------------------------
                              0130   642
                              0130   643 ENDTEST
```

```
0025          .SBTTL  "TEST   034     TB GROUP 0 HIT COMPARATOR TEST 2
0025    645 ;*****************************************************************************
0025    646 ;++
0025    647 ;
0025    648 ;       TB GROUP 0 HIT COMPARATOR TEST 2
0025    649 ;
0025    650 ; FUNCTIONAL DESCRIPTION:
0025    651 ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 0
0025    652 ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
0025    653 ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
0025    654 ;
0025    655 ;
0025    656 ; TEST ALGORITHM:
0025    657 ;       SUBTEST #1:
0025    658 ;       1.  EXECUTE DCS PROGRAM
0025    659 ;               a.  INVALIDATE THE TB
0025    660 ;               b.  SET MISS GROUP 1 AND REPLACE GROUP 0
0025    661 ;               c.  LOAD VA WITH ZERO
0025    662 ;               d.  WRITE PTE OF 00999990 TO GROUP 0
0025    663 ;               e.  READ PTE BACK TO RDM WH REGISTER
0025    664 ;       2.  TEST THAT PTE IS CORRECT
0025    665 ;
0025    666 ;       SUBTEST #2
0025    667 ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
0025    668 ;       2.  LOAD A TEST ADDRESS INTO THE RDM WD REGISTER
0025    669 ;       3.  EXECUTE DCS PROGRAM
0025    670 ;               a.  INVALIDATE THE TB
0025    671 ;               b.  SET MISS GROUP 1 AND REPLACE GROUP 0
0025    672 ;               c.  LOAD VA WITH ZERO
0025    673 ;               d.  WRITE PTE OF 00999990 TO GROUP 0
0025    674 ;               e.  LOAD VA WITH TEST ADDRESS FROM RDM WD REGISTER
0025    675 ;               f.  READ TB
0025    676 ;       4.  TEST THAT THE TB READ WAS A MISS
0025    677 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST ADDRESSES
0025    678 ;
0025    679 ;
0025    680 ; TEST PATTERNS:
0025    681 ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 1
0025    682 ;       THRU 0'S IN VA BITS <30:16>, THE VA TAG FIELD.
0025    683 ;       1.      00010000
0025    684 ;       2.      00020000
0025    685 ;       3.      00040000
0025    686 ;       4.      00080000
0025    687 ;       5.      00100000
0025    688 ;       6.      00200000
0025    689 ;       7.      00400000
0025    690 ;       8.      00800000
0025    691 ;       9.      01000000
0025    692 ;       10.     02000000
0025    693 ;       11.     04000000
0025    694 ;       12.     08000000
0025    695 ;       13.     10000000
0025    696 ;       14.     20000000
0025    697 ;       15.     40000000
0025    698 ;
```

```
                     699 ;
                     700 ; LOGIC DESCRIPTION:
                     701 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 0.
                     702 ;
                     703 ;
                     704 ; ASSUMPTIONS:
                     705 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
                     706 ;
                     707 ;
                     708 ; ERROR DESCRIPTION:
                     709 ;       EXPECTED PTE
                     710 ;       GOT PTE
                     711 ;       INDEX WHICH IDENTIFIES THE FAILING TEST PATTERN
                     712 ;
                     713 ;--
                     714 ;******************************************************************************
                     715
                     716         SUBTEST                                         ; SUBTEST #1
                         ;////////////////////////////////////////////////////////////////////////////
                     717 ;+
                     718 ;SUBTEST #1
                     719 ;
                     720 ;       TEST THAT TB LOCATION ZERO IS A HIT
                     721 ;-
                     722         INITIALIZE                                      ; INIT THE CPU
                     723         LOADDCS         7$,,<^X0C>,2                    ; LOAD THE TEST UCODE.
                     724         ERRLOOP
                     725         FETCH           0                               ; EXECUTE THE UCODE.
                     726         BRSTCLK         <^X0D>
                     727         CMPREGMSK       WHREG,2$,,3$,,L,                ; CHECK THE PTE.
                     728         IFERROR         ,<<ADK>>,                       ; INCORRECT PTE
                     729
                     730         SUBTEST                                         ; SUBTEST #2
                         ;////////////////////////////////////////////////////////////////////////////
                     731 ;+
                     732 ;SUBTEST #2
                     733 ;
                     734 ;       TEST THAT THE TEST ADDRESSES ARE A MISS AT LOCATION ZERO
                     735 ;-
                     736         INITIALIZE                                      ; INIT THE CPU
                     737         LOADDCS         4$,,<^X0C>,3                    ; MODIFY DCS U-CODE
                     738         LOOP            I,1,15                          ; CREATE A TEST LOOP OF 15
                     739         LOADREG         WDREG,1$,I,L                    ; TEST ADDRESS TO RDM WD REG
                     740         ERRLOOP                                         ; ERROR LOOP POINT
                     741         FETCH           0                               ; START DCS PROGRAM
                     742         BRSTCLK         <^X0E>                          ; END DCS PROGRAM
                     743         CMPREGMSK       WHREG,5$,,6$,,L,                ; TEST FOR TB MISS
                     744         IFERROR         ,<<ADK>>,                       ; FAILING ARRAY
                     745         ENDLOOP         I                               ; END TEST LOOP
                     746         SKIP                                            ; GO TO NEXT TEST
                     747
                     748 BEGIN_TEST_DATA
```

```
                                  0090      ;-----------------------------------------------------------------------
                                  0090
                        00010000  0090      1$:     .LONG   ^X00010000                      ; TEST ADDRESS TABLE
                        00020000  0094              .LONG   ^X00020000
                        00040000  0098              .LONG   ^X00040000
                        00080000  009C              .LONG   ^X00080000
                        00100000  00A0              .LONG   ^X00100000
                        00200000  00A4              .LONG   ^X00200000
                        00400000  00A8              .LONG   ^X00400000
                        00800000  00AC              .LONG   ^X00800000
                        01000000  00B0              .LONG   ^X01000000
                        02000000  00B4              .LONG   ^X02000000
                        04000000  00B8              .LONG   ^X04000000
                        08000000  00BC              .LONG   ^X08000000
                        10000000  00C0              .LONG   ^X10000000
                        20000000  00C4              .LONG   ^X20000000
                        40000000  00C8              .LONG   ^X40000000
                                  00CC
                        00999990  00CC      2$:     .LONG   ^X00999990                      ; COMPARE DATA SUBTEST #1
                        00FFFFF8  00D0      3$:     .LONG   ^X00FFFFF8                      ; MASK FOR CMPREGMSK PSEUDO OP
                                  00D4
                                  00D4      4$:
U 0C, 5700,C800,0364,0300,4A70,180D  00D4      ;% 0C   VA_RDM                                  ; U-CODE FOR SUBTEST #2
U 0D, 6701,881F,5924,0200,05F0,180E  00DF      ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F  00EA      ;% 0E   NOP,STOP.CLOCK
                                  00F5
                        00000000  00F5      5$:     .LONG   ^X00000000                      ; COMPARE DATA SUBTEST #2
                        00000100  00F9      6$:     .LONG   ^X00000100                      ; MASK FOR CMPREGMSK PSEUDO OP
                                  00FD
                                  00FD      7$:
U 0C, 6701,881F,5924,0200,05F0,180D  00FD      ;% 0C   RDM_TB                                  ;U-CODE FOR SUBTEST #1
U 0D, 7300,C800,0364,0300,0470,180E  0108      ;% 0D   NOP,STOP.CLOCK
                                  0113
                                  0113      BEGIN_CPU_DATA
                                  0002
                                  0002      DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001      ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  000C      ;% 01   D_R[ZERO]                               ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803  0017      ;% 02   VA_Q_D_D+ZLIT8[0]                       ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804  0022      ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB           ; 200...........
U 04, 7700,F800,7E7F,FFFF,8470,1805  002D      ;% 04   LONLIT_[03000000]                       ; SET FORCE MISS GROUP 1 AND
U 05, 7700,4800,5BE4,03D4,6870,1806  0038      ;% 05   MEMSCAR_R[LONLIT]                       ; FORCE REPLACE GROUP 0.
U 06, 7700,7800,7AFF,FFFF,8470,1807  0043      ;% 06   LONLIT_[0A000000]
U 07, 7700,4800,5BE4,03D4,6070,1808  004E      ;% 07   MEMSCR_R[LONLIT]
U 08, 7700,F800,7FFF,FFFF,8470,1809  0059      ;% 08   LONLIT_[0]                              ; ZERO TO LONLIT
U 09, 7700,4800,5BE4,03D4,4A70,180A  0064      ;% 09   VA_R[LONLIT]                            ; LOAD ZERO INTO VA REGISTER
U 0A, 7700,7800,7FB3,3337,8470,180B  006F      ;% 0A   LONLIT_[00999990]                       ; PTE TO LONLIT
U 0B, 7700,C81B,5BE4,03D4,5070,180C  007A      ;% 0B   TB_R[LONLIT]                            ; WRITE TB GROUP 0
U 0C, 6701,881F,5924,0200,05F0,180D  0085      ;% 0C   RDM_TB                                  ; READ TB TO RDM WH REGISTER
U 0D, 7300,C800,0364,0300,0470,180E  0090      ;% 0D   STOP.CLOCK                              ; STOP CPU CLOCK
                                  009B
                                  009B      END_CPU_DATA
                                  0113
                                  0113      END_TEST_DATA
```

```
0113
0113        ;-----------------------------------------------------------------------
0113   860
0113   861 ENDTEST
```

```
                        0025        .SBTTL  "TEST   035     TB GROUP 1 HIT COMPARATOR TEST 2
                        0025    863 ;*******************************************************************************
                        0025    864 ;++
                        0025    865 ;
                        0025    866 ;       TB GROUP 1 HIT COMPARATOR TEST 2
                        0025    867 ;
                        0025    868 ; FUNCTIONAL DESCRIPTION:
                        0025    869 ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 1
                        0025    870 ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
                        0025    871 ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
                        0025    872 ;
                        0025    873 ;
                        0025    874 ; TEST ALGORITHM:
                        0025    875 ;       SUBTEST #1:
                        0025    876 ;       1.  EXECUTE DCS PROGRAM
                        0025    877 ;               a.  INVALIDATE THE TB
                        0025    878 ;               b.  SET MISS GROUP 0 AND REPLACE GROUP 1
                        0025    879 ;               c.  LOAD VA WITH ZERO
                        0025    880 ;               d.  WRITE PTE OF 00999990 TO GROUP 1
                        0025    881 ;               e.  READ PTE BACK TO RDM WH REGISTER
                        0025    882 ;       2.  TEST THAT PTE IS CORRECT
                        0025    883 ;
                        0025    884 ;       SUBTEST #2
                        0025    885 ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
                        0025    886 ;       2.  LOAD A TEST ADDRESS INTO THE RDM WD REGISTER
                        0025    887 ;       3.  EXECUTE DCS PROGRAM
                        0025    888 ;               a.  INVALIDATE THE TB
                        0025    889 ;               b.  SET MISS GROUP 0 AND REPLACE GROUP 1
                        0025    890 ;               c.  LOAD VA WITH ZERO
                        0025    891 ;               d.  WRITE PTE OF 00999990 TO GROUP 1
                        0025    892 ;               e.  LOAD VA WITH TEST ADDRESS FROM RDM WD REGISTER
                        0025    893 ;               f.  READ TB
                        0025    894 ;       4.  TEST THAT THE TB READ WAS A MISS
                        0025    895 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST ADDRESSES
                        0025    896 ;
                        0025    897 ;
                        0025    898 ; TEST PATTERNS:
                        0025    899 ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 1
                        0025    900 ;       THRU 0'S IN VA BITS <30:16>, THE VA TAG FIELD.
                        0025    901 ;       1.      00010000
                        0025    902 ;       2.      00020000
                        0025    903 ;       3.      00040000
                        0025    904 ;       4.      00080000
                        0025    905 ;       5.      00100000
                        0025    906 ;       6.      00200000
                        0025    907 ;       7.      00400000
                        0025    908 ;       8.      00800000
                        0025    909 ;       9.      01000000
                        0025    910 ;       10.     02000000
                        0025    911 ;       11.     04000000
                        0025    912 ;       12.     08000000
                        0025    913 ;       13.     10000000
                        0025    914 ;       14.     20000000
                        0025    915 ;       15.     40000000
                        0025    916 ;
```

```
                     0025   917 ;
                     0025   918 ; LOGIC DESCRIPTION:
                     0025   919 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 1
                     0025   920 ;
                     0025   921 ;
                     0025   922 ; ASSUMPTIONS:
                     0025   923 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
                     0025   924 ;
                     0025   925 ;
                     0025   926 ; ERROR DESCRIPTION:
                     0025   927 ;       EXPECTED PTE
                     0025   928 ;       GOT PTE
                     0025   929 ;       INDEX WHICH IDENTIFIES THE FAILING TEST PATTERN
                     0025   930 ;
                     0025   931 ;--
                     0025   932 ;*****************************************************************************
                     0025   933
                     0025   934         SUBTEST                                         ; SUBTEST #1
                     0028
                     0028           ;////////////////////////////////////////////////////////////////////////
                     0028   935 ;+
                     0028   936 ;SUBTEST #1
                     0028   937 ;
                     0028   938 ;       TEST THAT TB LOCATION ZERO IS A HIT
                     0028   939 ;-
                     0028   940         INITIALIZE                                      ; INIT THE CPU
                     002A   941         LOADDCS         7$,,<^X0C>,2                    ; LOAD THE TEST UCODE.
                     0031   942         ERRLOOP
                     0033   943         FETCH           0                               ; EXECUTE THE UCODE.
                     0038   944         BRSTCLK         <^X0D>
                     003C   945         CMPREGMSK       WHREG,2$,,3$,,L,                ; CHECK THE PTE.
                     0048   946         IFERROR         ,<<ADK>>,                       ; INCORRECT PTE
                     004E   947
                     004E   948         SUBTEST                                         ; SUBTEST #2
                     0051
                     0051           ;////////////////////////////////////////////////////////////////////////
                     0051   949 ;+
                     0051   950 ;SUBTEST #2
                     0051   951 ;
                     0051   952 ;       TEST THAT THE TEST ADDRESSES ARE A MISS AT LOCATION ZERO
                     0051   953 ;-
                     0051   954         INITIALIZE                                      ; INIT THE CPU
                     0053   955         LOADDCS         4$,,<^X0C>,3                    ; MODIFY DCS U-CODE
                     005A   956         LOOP            I,1,15                          ; CREATE A TEST LOOP OF 15
                     0061   957         LOADREG         WDREG,1$,I,L                    ; TEST ADDRESS TO RDM WD REG
                     0069   958         ERRLOOP                                         ; ERROR LOOP POINT
                     006B   959         FETCH           0                               ; START DCS PROGRAM
                     0070   960         BRSTCLK         <^X0E>                          ; END DCS PROGRAM
                     0074   961         CMPREGMSK       WHREG,5$,,6$,,L,                ; TEST FOR TB MISS
                     0080   962         IFERROR         ,<<ADK>>,                       ; FAILING ARRAY
                     0086   963         ENDLOOP         I                               ; END TEST LOOP
                     0089   964         SKIP                                            ; GO TO NEXT TEST
                     0090   965
                     0090   966 BEGIN_TEST_DATA
                     0090
```

```
                                  0090        ;-----------------------------------------------------------------------
                                  0090
                        00010000  0090        1$:     .LONG   ^X00010000                      ; TEST ADDRESS TABLE
                        00020000  0094                .LONG   ^X00020000
                        00040000  0098                .LONG   ^X00040000
                        00080000  009C                .LONG   ^X00080000
                        00100000  00A0                .LONG   ^X00100000
                        00200000  00A4                .LONG   ^X00200000
                        00400000  00A8                .LONG   ^X00400000
                        00800000  00AC                .LONG   ^X00800000
                        01000000  00B0                .LONG   ^X01000000
                        02000000  00B4                .LONG   ^X02000000
                        04000000  00B8                .LONG   ^X04000000
                        08000000  00BC                .LONG   ^X08000000
                        10000000  00C0                .LONG   ^X10000000
                        20000000  00C4                .LONG   ^X20000000
                        40000000  00C8                .LONG   ^X40000000
                                  00CC
                        00999990  00CC        2$:     .LONG   ^X00999990                      ; COMPARE DATA SUBTEST #1
                        00FFFFF8  00D0        3$:     .LONG   ^X00FFFFF8                      ; MASK FOR CMPREGMSK PSEUDO OP
                                  00D4
                                  00D4        4$:
U 0C, 5700,C800,0364,0300,4A70,180D  00D4     ;% 0C   VA_RDM                                  ; U-CODE FOR SUBTEST #2
U 0D, 6701,881F,5924,0200,05F0,180E  00DF     ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F  00EA     ;% 0E   NOP,STOP.CLOCK
                                  00F5
                        00000000  00F5        5$:     .LONG   ^X00000000                      ; COMPARE DATA SUBTEST #2
                        00000100  00F9        6$:     .LONG   ^X00000100                      ; MASK FOR CMPREGMSK PSEUDO OP
                                  00FD
                                  00FD        7$:
U 0C, 6701,881F,5924,0200,05F0,180D  00FD     ;% 0C   RDM_TB                                  ;U-CODE FOR SUBTEST #1
U 0D, 7300,C800,0364,0300,0470,180E  0108     ;% 0D   NOP,STOP.CLOCK
                                  0113
                                  0113        BEGIN_CPU_DATA
                                  0002
                                  0002        DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001     ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  000C     ;% 01   D_R[ZERO]                               ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803  0017     ;% 02   VA_Q_D_D+ZLIT8[0]                       ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804  0022     ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB           ; 200...........
U 04, 7700,F800,7E7F,FFFF,8470,1805  002D     ;% 04   LONLIT_[03000000]                       ; SET FORCE MISS GROUP 0 AND
U 05, 7700,4800,5BE4,03D4,6870,1806  0038     ;% 05   MEMSCAR_R[LONLIT]                       ; FORCE REPLACE GROUP 1
U 06, 7700,3800,797F,FFFF,8470,1807  0043     ;% 06   LONLIT_[0D000000]
U 07, 7700,4800,5BE4,03D4,6070,1808  004E     ;% 07   MEMSCR_R[LONLIT]
U 08, 7700,F800,7FFF,FFFF,8470,1809  0059     ;% 08   LONLIT_[0]                              ; ZERO TO LONLIT
U 09, 7700,4800,5BE4,03D4,4A70,180A  0064     ;% 09   VA_R[LONLIT]                            ; LOAD ZERO INTO VA REGISTER
U 0A, 7700,7800,7FB3,3337,8470,180B  006F     ;% 0A   LONLIT_[00999990]                       ; PTE TO LONLIT
U 0B, 7700,C81B,5BE4,03D4,5070,180C  007A     ;% 0B   TB_R[LONLIT]                            ; WRITE TB GROUP 1
U 0C, 6701,881F,5924,0200,05F0,180D  0085     ;% 0C   RDM_TB                                  ; READ TB TO RDM WH REGISTER
U 0D, 7300,C800,0364,0300,0470,180E  0090     ;% 0D   STOP.CLOCK                              ; STOP CPU CLOCK
                                  009B
                                  009B        END_CPU_DATA
                                  0113
                                  0113        END_TEST_DATA
```

```
                0113
                0113        ;-----------------------------------------------------------------------------
                0113  1078
                0113  1079 ENDTEST
```

```
                        .SBTTL  "TEST   036     TB GROUP 0 HIT COMPARATOR TEST 3
0025  1081 ;*******************************************************************************
0025  1082 ;++
0025  1083 ;
0025  1084 ;       TB GROUP 0 HIT COMPARATOR TEST 3
0025  1085 ;
0025  1086 ; FUNCTIONAL DESCRIPTION:
0025  1087 ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 0
0025  1088 ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
0025  1089 ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
0025  1090 ;
0025  1091 ;
0025  1092 ; TEST ALGORITHM:
0025  1093 ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
0025  1094 ;       2.  LOAD A TEST ADDRESS INTO RDM WD REGISTER
0025  1095 ;       3.  EXECUTE DCS PROGRAM
0025  1096 ;               a.  INVALIDATE TB
0025  1097 ;               b.  SET MISS GROUP 1 AND REPLACE GROUP 0
0025  1098 ;               c.  LOAD TEST ADDRESS INTO VA REGISTER FROM RDM
0025  1099 ;               d.  WRITE TB GROUP 0 WITH 00333330
0025  1100 ;               e.  LOAD VA WITH 7FFF0000 (SUBTEST 2)
0025  1101 ;               f.  READ TB TO RDM WH REGISTER
0025  1102 ;       4.  TEST WH REGISTER FOR CORRECT PTE IN SUBTEST 1.  TEST FOR
0025  1103 ;           TB MISS IN SUBTEST 2.
0025  1104 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
0025  1105 ;
0025  1106 ;
0025  1107 ; TEST PATTERNS:
0025  1108 ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 0
0025  1109 ;       THRU 1'S IN VA BITS <30:16>, THE VA TAG FIELD.
0025  1110 ;       1.      7FFE0000
0025  1111 ;       2.      7FFD0000
0025  1112 ;       3.      7FFB0000
0025  1113 ;       4.      7FF70000
0025  1114 ;       5.      7FEF0000
0025  1115 ;       6.      7FDF0000
0025  1116 ;       7.      7FBF0000
0025  1117 ;       8.      7F7F0000
0025  1118 ;       9.      7EFF0000
0025  1119 ;       10.     7DFF0000
0025  1120 ;       11.     7BFF0000
0025  1121 ;       12.     77FF0000
0025  1122 ;       13.     6FFF0000
0025  1123 ;       14.     5FFF0000
0025  1124 ;       15.     3FFF0000
0025  1125 ;
0025  1126 ;
0025  1127 ; LOGIC DESCRIPTION:
0025  1128 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 0.
0025  1129 ;
0025  1130 ;
0025  1131 ; ASSUMPTIONS:
0025  1132 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
0025  1133 ;
0025  1134 ;
```

```
                     0025  1135 ; ERROR DESCRIPTION:
                     0025  1136 ;       EXPECTED PTE
                     0025  1137 ;       GOT PTE
                     0025  1138 ;       INDEX WHICH IDENTIFIES THE FAILING TEST PATTERN
                     0025  1139 ;
                     0025  1140 ;--
                     0025  1141 ;*******************************************************************************
                     0025  1142
                     0025  1143         SUBTEST                                         ; SUBTEST #1
                     0028
                     0028       ;////////////////////////////////////////////////////////////////////////////////
                     0028  1144 ;+
                     0028  1145 ;SUBTEST #1
                     0028  1146 ;
                     0028  1147 ;       TEST FOR TB HITS WITH ALL TEST ADDRESSES
                     0028  1148 ;-
                     0028  1149         INITIALIZE                                      ; INIT THE CPU
                     002A  1150         LOADDCS         1$,,<^X0B>,2                    ; LOAD THE TEST UCODE.
                     0031  1151         LOOP            I,1,15                          ; SET UP FOR 15 ITERATIONS.
                     0038  1152         ERRLOOP
                     003A  1153         FETCH           0                               ; EXECUTE THE UCODE.
                     003F  1154         BRSTCLK         <^X0C>
                     0043  1155         CMPREGMSK       WHREG,2$,,3$,,L                 ; CHECK THE PTE.
                     004F  1156         IFERROR         ,<<ADK>>,                       ; INCORRECT PTE
                     0055  1157         ENDLOOP         I                               ; END TEST LOOP
                     0058  1158
                     0058  1159         SUBTEST                                         ; SUBTEST #2
                     005B
                     005B       ;////////////////////////////////////////////////////////////////////////////////
                     005B  1160 ;+
                     005B  1161 ;SUBTEST #2
                     005B  1162 ;
                     005B  1163 ;       TEST FOR TB MISS WITH TEST ADDRESS AGAINST 7FFF0000
                     005B  1164 ;-
                     005B  1165         INITIALIZE                                      ; INIT THE CPU
                     005D  1166         LOADDCS         4$,,<^X0B>,4                    ; MODIFY DCS U-CODE
                     0064  1167         LOOP            I,1,15                          ; CREATE TEST LOOP OF 15
                     006B  1168         ERRLOOP                                         ; ERROR LOOP POINT
                     006D  1169         FETCH           0                               ; START DCS PROGRAM
                     0072  1170         BRSTCLK         <^X0E>                          ; END DCS PROGRAM
                     0076  1171         CMPREGMSK       WHREG,5$,,6$,,L,                ; TEST FOR TB MISS
                     0082  1172         IFERROR         ,<<ADK>>,                       ; FAILING ARRAY
                     0088  1173         ENDLOOP         I                               ; END TEST LOOP
                     008B  1174         SKIP                                            ; GO TO NEXT TEST
                     0092  1175
                     0092  1176 BEGIN_TEST_DATA
                     0092
                     0092       ;-------------------------------------------------------------------------------
                     0092  1177
                     0092  1178 1$:
U 0B, 6701,081F,5924,0200,05F0,180C  0092  1179 ;% 0B   RDM_TB                          ; U-CODE FOR SUBTEST #1
U 0C, 7300,C800,0364,0300,0470,180D  009D  1183 ;% 0C   NOP,STOP.CLOCK
                     00A8  1187
                 00333330  00A8  1188 2$:    .LONG   ^X00333330                      ; COMPARE DATA SUBTEST #1
                 00FFFFF8  00AC  1189 3$:    .LONG   ^X00FFFFF8                      ; MASK FOR CMPREGMSK PSEUDO OP
```

```
                                00B0  1190
                                00B0  1191 4$:
U 0B, 7700,B800,4000,7FFF,8470,180C  00B0  1192 ;% 0B   LONLIT_[7FFF0000]                    ; U-CODE FOR SUBTEST #2
U 0C, 7700,C800,5BE4,03D4,4A70,180D  00BB  1196 ;% 0C   VA_R[LONLIT]
U 0D, 6701,881F,5924,0200,05F0,180E  00C6  1200 ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F  00D1  1204 ;% 0E   NOP,STOP.CLOCK
                                00DC  1208
                        00000000  00DC  1209 5$:     .LONG   ^X00000000                  ; COMPARE DATA SUBTEST #2
                        00000100  00E0  1210 6$:     .LONG   ^X00000100                  ; MASK FOR CMPREGMSK PSEUDO OP
                                00E4  1211
                        7FFE0000  00E4  1212 7$:     .LONG   ^X7FFE0000                  ; TEST ADDRESS TABLE
                        7FFD0000  00E8  1213         .LONG   ^X7FFD0000
                        7FFB0000  00EC  1214         .LONG   ^X7FFB0000
                        7FF70000  00F0  1215         .LONG   ^X7FF70000
                        7FEF0000  00F4  1216         .LONG   ^X7FEF0000
                        7FDF0000  00F8  1217         .LONG   ^X7FDF0000
                        7FBF0000  00FC  1218         .LONG   ^X7FBF0000
                        7F7F0000  0100  1219         .LONG   ^X7F7F0000
                        7EFF0000  0104  1220         .LONG   ^X7EFF0000
                        7DFF0000  0108  1221         .LONG   ^X7DFF0000
                        7BFF0000  010C  1222         .LONG   ^X7BFF0000
                        77FF0000  0110  1223         .LONG   ^X77FF0000
                        6FFF0000  0114  1224         .LONG   ^X6FFF0000
                        5FFF0000  0118  1225         .LONG   ^X5FFF0000
                        3FFF0000  011C  1226         .LONG   ^X3FFF0000
                                0120  1227
                                0120  1228 BEGIN_CPU_DATA
                                0002  1229
                                0002  1230 DCS_DATA
                                0001  1231
U 00, 7700,C800,0364,0300,0470,1801  0001  1232 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  000C  1236 ;% 01   D_R[ZERO]                          ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803  0017  1240 ;% 02   VA_Q_D_D+ZLIT8[0]                  ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804  0022  1244 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB     ; 200.....
U 04, 7700,F800,7E7F,FFFF,8470,1805  002D  1248 ;% 04   LONLIT_[03000000]                  ; SET FORCE MISS GROUP 1 AND
U 05, 7700,4800,5BE4,03D4,6870,1806  0038  1252 ;% 05   MEMSCAR_R[LONLIT]                 ; FORCE REPLACE GROUP 0.
U 06, 7700,7800,7AFF,FFFF,8470,1807  0043  1256 ;% 06   LONLIT_[0A000000]
U 07, 7700,4800,5BE4,03D4,6070,1808  004E  1260 ;% 07   MEMSCR_R[LONLIT]
U 08, 5700,4800,0364,0300,4A70,1809  0059  1264 ;% 08   VA_RDM                            ; LOAD THE TEST PATTERN INTO VA.
U 09, 7700,F800,7FE6,6667,8470,180A  0064  1268 ;% 09   LONLIT_[00333330]                 ; MAKE THE TEST PATTERN AND
U 0A, 7700,481B,5BE4,03D4,5070,180B  006F  1272 ;% 0A   TB_R[LONLIT]                      ; PTE=00333330 A HIT IN GROUP 0.
U 0B, 6701,081F,5924,0200,05F0,180C  007A  1276 ;% 0B   RDM_TB                           ; READ TB TO RDM
U 0C, 7300,C800,0364,0300,0470,180D  0085  1280 ;% 0C   STOP.CLOCK                       ; STOP CPU CLOCK
                                0090  1284
                                0090  1285 END_CPU_DATA
                                0120  1286
                                0120  1287 END_TEST_DATA
                                0120
                                0120       ;-------------------------------------------------------------------------
                                0120  1288
                                0120  1289 ENDTEST
```

```
0025          .SBTTL  "TEST   037     TB GROUP 1 HIT COMPARATOR TEST 3
0025  1291 ;*******************************************************************************
0025  1292 ;++
0025  1293 ;
0025  1294 ;       TB GROUP 1 HIT COMPARATOR TEST 3
0025  1295 ;
0025  1296 ; FUNCTIONAL DESCRIPTION:
0025  1297 ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 1
0025  1298 ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
0025  1299 ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
0025  1300 ;
0025  1301 ;
0025  1302 ; TEST ALGORITHM:
0025  1303 ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
0025  1304 ;       2.  LOAD A TEST ADDRESS INTO RDM WD REGISTER
0025  1305 ;       3.  EXECUTE DCS PROGRAM
0025  1306 ;               a.  INVALIDATE TB
0025  1307 ;               b.  SET MISS GROUP 0 AND REPLACE GROUP 1
0025  1308 ;               c.  LOAD TEST ADDRESS INTO VA REGISTER FROM RDM
0025  1309 ;               d.  WRITE TB GROUP 1 WITH 00333330
0025  1310 ;               e.  LOAD VA WITH 7FFF0000 (SUBTEST 2)
0025  1311 ;               f.  READ TB TO RDM WH REGISTER
0025  1312 ;       4.  TEST WH REGISTER FOR CORRECT PTE IN SUBTEST 1.  TEST FOR
0025  1313 ;           TB MISS IN SUBTEST 2.
0025  1314 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
0025  1315 ;
0025  1316 ;
0025  1317 ; TEST PATTERNS:
0025  1318 ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 0
0025  1319 ;       THRU 1'S IN VA BITS <30:16>, THE VA TAG FIELD.
0025  1320 ;       1.      7FFE0000
0025  1321 ;       2.      7FFD0000
0025  1322 ;       3.      7FFB0000
0025  1323 ;       4.      7FF70000
0025  1324 ;       5.      7FEF0000
0025  1325 ;       6.      7FDF0000
0025  1326 ;       7.      7FBF0000
0025  1327 ;       8.      7F7F0000
0025  1328 ;       9.      7EFF0000
0025  1329 ;       10.     7DFF0000
0025  1330 ;       11.     7BFF0000
0025  1331 ;       12.     77FF0000
0025  1332 ;       13.     6FFF0000
0025  1333 ;       14.     5FFF0000
0025  1334 ;       15.     3FFF0000
0025  1335 ;
0025  1336 ;
0025  1337 ; LOGIC DESCRIPTION:
0025  1338 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 1
0025  1339 ;
0025  1340 ;
0025  1341 ; ASSUMPTIONS:
0025  1342 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
0025  1343 ;
0025  1344 ;
```

```
                        0025  1345 ; ERROR DESCRIPTION:
                        0025  1346 ;       EXPECTED PTE
                        0025  1347 ;       GOT PTE
                        0025  1348 ;       INDEX WHICH IDENTIFIES THE FAILING TEST PATTERN
                        0025  1349 ;
                        0025  1350 ;--
                        0025  1351 ;*******************************************************************************
                        0025  1352
                        0025  1353         SUBTEST                                         ; SUBTEST #1
                        0028
                        0028       ;////////////////////////////////////////////////////////////////////////////
                        0028  1354 ;+
                        0028  1355 ;SUBTEST #1
                        0028  1356 ;
                        0028  1357 ;       TEST FOR TB HITS WITH ALL TEST ADDRESSES
                        0028  1358 ;-
                        0028  1359         INITIALIZE                                      ; INIT THE CPU
                        002A  1360         LOADDCS         1$,,<^X0B>,2                    ; LOAD THE TEST UCODE.
                        0031  1361         LOOP            I,1,15                          ; SET UP FOR 15 ITERATIONS.
                        0038  1362         ERRLOOP
                        003A  1363         FETCH           0                               ; EXECUTE THE UCODE.
                        003F  1364         BRSTCLK         <^X0C>
                        0043  1365         CMPREGMSK       WHREG,2$,,3$,,L                 ; CHECK THE PTE.
                        004F  1366         IFERROR         ,<<ADK>>,                       ; INCORRECT PTE
                        0055  1367         ENDLOOP         I                               ; END TEST LOOP
                        0058  1368
                        0058  1369         SUBTEST                                         ; SUBTEST #2
                        005B
                        005B       ;////////////////////////////////////////////////////////////////////////////
                        005B  1370 ;+
                        005B  1371 ;SUBTEST #2
                        005B  1372 ;
                        005B  1373 ;       TEST FOR TB MISS WITH TEST ADDRESS AGAINST 7FFF0000
                        005B  1374 ;-
                        005B  1375         INITIALIZE                                      ; INIT THE CPU
                        005D  1376         LOADDCS         4$,,<^X0B>,4                    ; MODIFY DCS U-CODE
                        0064  1377         LOOP            I,1,15                          ; CREATE TEST LOOP OF 15
                        006B  1378         ERRLOOP                                         ; ERROR LOOP POINT
                        006D  1379         FETCH           0                               ; START DCS PROGRAM
                        0072  1380         BRSTCLK         <^X0E>                          ; END DCS PROGRAM
                        0076  1381         CMPREGMSK       WHREG,5$,,6$,,L,                ; TEST FOR TB MISS
                        0082  1382         IFERROR         ,<<ADK>>,                       ; FAILING ARRAY
                        0088  1383         ENDLOOP         I                               ; END TEST LOOP
                        008B  1384         SKIP                                            ; GO TO NEXT TEST
                        0092  1385
                        0092  1386 BEGIN_TEST_DATA
                        0092
                        0092       ;-------------------------------------------------------------------------
                        0092  1387
                        0092  1388 1$:
U 0B, 6701,081F,5924,0200,05F0,180C 0092  1389 ;% 0B    RDM_TB                          ; U-CODE FOR SUBTEST #1
U 0C, 7300,C800,0364,0300,0470,180D 009D  1393 ;% 0C    NOP,STOP.CLOCK
                        00A8  1397
                  00333330  00A8  1398 2$:      .LONG    ^X00333330                     ; COMPARE DATA SUBTEST #1
                        00AC  1399
```

```
                        00FFFFF8  00AC  1400 3$:     .LONG   ^X00FFFFF8                      ; MASK FOR CMPREGMSK PSEUDO OP
                                  00B0  1401
                                  00B0  1402 4$:
U 0B, 7700,B800,4000,7FFF,8470,180C 00B0  1403 ;% 0B   LONLIT_[7FFF0000]                   ; U-CODE FOR SUBTEST #2
U 0C, 7700,C800,5BE4,03D4,4A70,180D 00BB  1407 ;% 0C   VA_R[LONLIT]
U 0D, 6701,881F,5924,0200,05F0,180E 00C6  1411 ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F 00D1  1415 ;% 0E   NOP,STOP.CLOCK
                                  00DC  1419
                        00000000  00DC  1420 5$:     .LONG   ^X00000000                      ; COMPARE DATA SUBTEST #2
                                  00E0  1421
                        00000100  00E0  1422 6$:     .LONG   ^X00000100                      ; MASK FOR CMPREGMSK PSEUDO OP
                                  00E4  1423
                        7FFE0000  00E4  1424 7$:     .LONG   ^X7FFE0000                      ; TEST ADDRESS TABLE
                        7FFD0000  00E8  1425         .LONG   ^X7FFD0000
                        7FFB0000  00EC  1426         .LONG   ^X7FFB0000
                        7FF70000  00F0  1427         .LONG   ^X7FF70000
                        7FEF0000  00F4  1428         .LONG   ^X7FEF0000
                        7FDF0000  00F8  1429         .LONG   ^X7FDF0000
                        7FBF0000  00FC  1430         .LONG   ^X7FBF0000
                        7F7F0000  0100  1431         .LONG   ^X7F7F0000
                        7EFF0000  0104  1432         .LONG   ^X7EFF0000
                        7DFF0000  0108  1433         .LONG   ^X7DFF0000
                        7BFF0000  010C  1434         .LONG   ^X7BFF0000
                        77FF0000  0110  1435         .LONG   ^X77FF0000
                        6FFF0000  0114  1436         .LONG   ^X6FFF0000
                        5FFF0000  0118  1437         .LONG   ^X5FFF0000
                        3FFF0000  011C  1438         .LONG   ^X3FFF0000
                                  0120  1439
                                  0120  1440 BEGIN_CPU_DATA
                                  0002  1441
                                  0002  1442 DCS_DATA
                                  0001  1443
U 00, 7700,C800,0364,0300,0470,1801 0001  1444 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802 000C  1448 ;% 01   D_R[ZERO]                            ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803 0017  1452 ;% 02   VA_Q_D_D+ZLIT8[0]                   ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804 0022  1456 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB       ; 200.....
U 04, 7700,F800,7E7F,FFFF,8470,1805 002D  1460 ;% 04   LONLIT_[03000000]                   ; SET FORCE MISS GROUP 0 AND
U 05, 7700,4800,5BE4,03D4,6870,1806 0038  1464 ;% 05   MEMSCAR_R[LONLIT]                   ; FORCE REPLACE GROUP 1
U 06, 7700,3800,797F,FFFF,8470,1807 0043  1468 ;% 06   LONLIT_[0D000000]
U 07, 7700,4800,5BE4,03D4,6070,1808 004E  1472 ;% 07   MEMSCR_R[LONLIT]
U 08, 5700,4800,0364,0300,4A70,1809 0059  1476 ;% 08   VA_RDM                             ; LOAD THE TEST PATTERN INTO VA.
U 09, 7700,F800,7FE6,6667,8470,180A 0064  1480 ;% 09   LONLIT_[00333330]                  ; MAKE THE TEST PATTERN AND
U 0A, 7700,481B,5BE4,03D4,5070,180B 006F  1484 ;% 0A   TB_R[LONLIT]                       ; PTE=00333330 A HIT IN GROUP 1
U 0B, 6701,081F,5924,0200,05F0,180C 007A  1488 ;% 0B   RDM_TB                            ; READ TB TO RDM
U 0C, 7300,C800,0364,0300,0470,180D 0085  1492 ;% 0C   STOP.CLOCK                        ; STOP CPU CLOCK
                                  0090  1496
                                  0090  1497 END_CPU_DATA
                                  0120  1498
                                  0120  1499 END_TEST_DATA
                                  0120
                                  0120       ;-------------------------------------------------------------------------
                                  0120  1500
                                  0120  1501 ENDTEST
```

```
0025          .SBTTL  "TEST   038     TB GROUP 0 HIT COMPARATOR TEST 4
0025  1503 ;*******************************************************************************
0025  1504 ;++
0025  1505 ;
0025  1506 ;       TB GROUP 0 HIT COMPARATOR TEST 4
0025  1507 ;
0025  1508 ; FUNCTIONAL DESCRIPTION:
0025  1509 ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 0
0025  1510 ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
0025  1511 ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
0025  1512 ;
0025  1513 ;
0025  1514 ; TEST ALGORITHM:
0025  1515 ;       SUBTEST #1
0025  1516 ;       1.  EXECUTE DCS PROGRAM
0025  1517 ;               a.  INVALIDATE TB
0025  1518 ;               b.  SET FORCE MISS GROUP 1 AND REPLACE GROUP 0
0025  1519 ;               c.  LOAD VA WITH 7FFF0000
0025  1520 ;               d.  WRITE PTE 00DDDDD0 TO GROUP 0
0025  1521 ;               e.  READ TB BACK TO RDM WH REGISTER
0025  1522 ;       2.  TEST WH REGISTER FOR CORRECT PTE
0025  1523 ;
0025  1524 ;       SUBTEST #2
0025  1525 ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
0025  1526 ;       2.  LOAD A TEST ADDRESS INTO RDM WD REGISTER
0025  1527 ;       3.  EXECUTE DCS PROGRAM
0025  1528 ;               a.  INVALIDATE TB
0025  1529 ;               b.  SET MISS GROUP 1 AND REPLACE GROUP 0
0025  1530 ;               c.  LOAD VA WITH 7FFF0000
0025  1531 ;               d.  WRITE PTE 00DDDDD0 TO GROUP 0
0025  1532 ;               e.  LOAD VA WITH TEST ADDRESS FROM RDM
0025  1533 ;               f.  READ TB
0025  1534 ;       4.  TEST WH REGISTER FOR TB MISS
0025  1535 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
0025  1536 ;
0025  1537 ;
0025  1538 ; TEST PATTERNS:
0025  1539 ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 0
0025  1540 ;       THRU 1'S IN VA BITS <30:16>, THE VA TAG FIELD.
0025  1541 ;       1.      7FFE0000
0025  1542 ;       2.      7FFD0000
0025  1543 ;       3.      7FFB0000
0025  1544 ;       4.      7FF70000
0025  1545 ;       5.      7FEF0000
0025  1546 ;       6.      7FDF0000
0025  1547 ;       7.      7FBF0000
0025  1548 ;       8.      7F7F0000
0025  1549 ;       9.      7EFF0000
0025  1550 ;       10.     7DFF0000
0025  1551 ;       11.     7BFF0000
0025  1552 ;       12.     77FF0000
0025  1553 ;       13.     6FFF0000
0025  1554 ;       14.     5FFF0000
0025  1555 ;       15.     3FFF0000
0025  1556 ;
```

```
                     1557 ;
                     1558 ; LOGIC DESCRIPTION:
                     1559 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 0.
                     1560 ;
                     1561 ;
                     1562 ; ASSUMPTIONS:
                     1563 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
                     1564 ;
                     1565 ;
                     1566 ; ERROR DESCRIPTION:
                     1567 ;       EXPECTED PTE
                     1568 ;       GOT PTE
                     1569 ;       INDEX WHICH IDENTIFIES THE FAILING TEST PATTERN (SUBTEST 2)
                     1570 ;
                     1571 ;--
                     1572 ;*****************************************************************************
                     1573
                     1574         SUBTEST                                        ; SUBTEST #1
                     0028
                     0028        ;/////////////////////////////////////////////////////////////////////////
                     1575 ;+
                     1576 ;SUBTEST #1
                     1577 ;
                     1578 ;       TEST TB LOCATION ZERO FOR A HIT WITH ADDRESS EQUAL 7FFF0000
                     1579 ;-
                     1580         INITIALIZE                                     ; INIT THE CPU
                     002A 1581    LOADDCS         1$,,<^X0C>,2                   ; LOAD THE FIRST UCODE OVERLAY
                     0031 1582    ERRLOOP
                     0033 1583    FETCH           0                              ; EXECUTE THE UCODE.
                     0038 1584    BRSTCLK         <^X0D>
                     003C 1585    CMPREGMSK       WHREG,2$,,3$,,L                ; CHECK THE PTE.
                     0048 1586    IFERROR         ,<<ADK>>,                      ; INCORRECT PTE
                     004E 1587
                     004E 1588    SUBTEST                                        ; SUBTEST #2
                     0051
                     0051        ;/////////////////////////////////////////////////////////////////////////
                     1589 ;+
                     1590 ;SUBTEST #2
                     1591 ;
                     1592 ;       TEST TB FOR MISSES WITH ALL TEST PATTERNS
                     1593 ;-
                     1594         INITIALIZE                                     ; INIT THE CPU
                     0053 1595    LOADDCS         4$,,<^X0C>,3                   ; MODIFY DCS U-CODE
                     005A 1596    LOOP            I,1,15                         ; CREATE A TEST LOOP OF 15
                     0061 1597    LOADREG         WDREG,7$,I,L                   ; LOAD TEST ADDRESS INTO WD REG
                     0069 1598    ERRLOOP                                        ; ERROR LOOP POINT
                     006B 1599    FETCH           0                              ; START DCS PROGRAM
                     0070 1600    BRSTCLK         <^X0E>                         ; END DCS PROGRAM
                     0074 1601    CMPREGMSK       WHREG,5$,,6$,,L,               ; TEST FOR TB MISS
                     0080 1602    IFERROR         ,<<ADK>>,                      ; FAILING ARRAY
                     0086 1603    ENDLOOP         I                              ; END TEST LOOP
                     0089 1604    SKIP                                           ; GO TO NEXT TEST
                     0090 1605
                     0090 1606 BEGIN_TEST_DATA
                     0090
```

```
                                0090        ;-----------------------------------------------------------------------
                                0090  1607
                                0090  1608 1$:
U 0C, 6701,881F,5924,0200,05F0,180D  0090  1609 ;% 0C   RDM_TB                                  ; U-CODE FOR SUBTEST 1
U 0D, 7300,C800,0364,0300,0470,180E  009B  1613 ;% 0D   NOP,STOP.CLOCK
                                00A6  1617
                            00DDDDD0  00A6  1618 2$:     .LONG   ^X00DDDDD0                    ; COMPARE DATA SUBTEST 1
                            00FFFFF8  00AA  1619 3$:     .LONG   ^X00FFFFF8                    ; MASK FOR CMPREGMSK PSEUDO OP
                                00AE  1620
                                00AE  1621 4$:
U 0C, 5700,C800,0364,0300,4A70,180D  00AE  1622 ;% 0C   VA_RDM                                 ; U-CODE FOR SUBTEST 2
U 0D, 6701,881F,5924,0200,05F0,180E  00B9  1626 ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F  00C4  1630 ;% 0E   NOP,STOP.CLOCK
                                00CF  1634
                            00000000  00CF  1635 5$:     .LONG   ^X00000000                    ; COMPARE DATA SUBTEST 2
                            00000100  00D3  1636 6$:     .LONG   ^X00000100                    ; MASK FOR CMPREGMSK PSEUDO OP
                                00D7  1637
                            7FFE0000  00D7  1638 7$:     .LONG   ^X7FFE0000
                            7FFD0000  00DB  1639         .LONG   ^X7FFD0000
                            7FFB0000  00DF  1640         .LONG   ^X7FFB0000
                            7FF70000  00E3  1641         .LONG   ^X7FF70000
                            7FEF0000  00E7  1642         .LONG   ^X7FEF0000
                            7FDF0000  00EB  1643         .LONG   ^X7FDF0000
                            7FBF0000  00EF  1644         .LONG   ^X7FBF0000
                            7F7F0000  00F3  1645         .LONG   ^X7F7F0000
                            7EFF0000  00F7  1646         .LONG   ^X7EFF0000
                            7DFF0000  00FB  1647         .LONG   ^X7DFF0000
                            7BFF0000  00FF  1648         .LONG   ^X7BFF0000
                            77FF0000  0103  1649         .LONG   ^X77FF0000
                            6FFF0000  0107  1650         .LONG   ^X6FFF0000
                            5FFF0000  010B  1651         .LONG   ^X5FFF0000
                            3FFF0000  010F  1652         .LONG   ^X3FFF0000
                                0113  1653
                                0113  1654 BEGIN_CPU_DATA
                                0002  1655
                                0002  1656 DCS_DATA
                                0001  1657
U 00, 7700,C800,0364,0300,0470,1801  0001  1658 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802  000C  1662 ;% 01   D_R[ZERO]                               ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803  0017  1666 ;% 02   VA_Q_D_D+ZLIT8[0]                      ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804  0022  1670 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB          ; 200.....
U 04, 7700,F800,7E7F,FFFF,8470,1805  002D  1674 ;% 04   LONLIT_[03000000]                      ; SET FORCE MISS GROUP 1 AND
U 05, 7700,4800,5BE4,03D4,6870,1806  0038  1678 ;% 05   MEMSCAR_R[LONLIT]                      ; FORCE REPLACE GROUP 0.
U 06, 7700,7800,7AFF,FFFF,8470,1807  0043  1682 ;% 06   LONLIT_[0A000000]
U 07, 7700,4800,5BE4,03D4,6070,1808  004E  1686 ;% 07   MEMSCR_R[LONLIT]
U 08, 7700,B800,4000,7FFF,8470,1809  0059  1690 ;% 08   LONLIT_[7FFF0000]
U 09, 7700,4800,5BE4,03D4,4A70,180A  0064  1694 ;% 09   VA_R[LONLIT]
U 0A, 7700,3800,7F91,1117,8470,180B  006F  1698 ;% 0A   LONLIT_[00DDDDD0]                      ; MAKE THE TEST PATTERN AND
U 0B, 7700,C81B,5BE4,03D4,5070,180C  007A  1702 ;% 0B   TB_R[LONLIT]                          ; PTE=00DDDDD0 A HIT IN GROUP 0.
U 0C, 6701,881F,5924,0200,05F0,180D  0085  1706 ;% 0C   RDM_TB                                ; ADDRESS AND READ THE TB AT
U 0D, 7300,C800,0364,0300,0470,180E  0090  1710 ;% 0D   STOP.CLOCK                            ; THAT ADDRESS.
                                009B  1714
                                009B  1715 END_CPU_DATA
                                0113  1716
                                0113  1717 END_TEST_DATA
```

```
                0113
                0113        ;-----------------------------------------------------------------------------
                0113  1718
                0113  1719 ENDTEST
```

```
                        .SBTTL  "TEST   039     TB GROUP 1 HIT COMPARATOR TEST 4
                ;*****************************************************************************
                ;++
                ;
                ;       TB GROUP 1 HIT COMPARATOR TEST 4
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;       THIS TEST INSURES THAT THE HIT AND MISS COMPARATOR FOR TB GROUP 1
                ;       IS FUNCTIONING. IT NOT ONLY CHECKS THAT HITS ARE GENERATED CORRECTLY
                ;       BUT IT ALSO CHECKS THAT MISSES OCCUR WHEN THEY SHOULD.
                ;
                ;
                ; TEST ALGORITHM:
                ;       SUBTEST #1
                ;       1.  EXECUTE DCS PROGRAM
                ;               a.  INVALIDATE TB
                ;               b.  SET FORCE MISS GROUP 0 AND REPLACE GROUP 1
                ;               c.  LOAD VA WITH 7FFF0000
                ;               d.  WRITE PTE 00DDDDD0 TO GROUP 1
                ;               e.  READ TB BACK TO RDM WH REGISTER
                ;       2.  TEST WH REGISTER FOR CORRECT PTE
                ;
                ;       SUBTEST #2
                ;       1.  CREATE A TEST LOOP OF 15 TO SELECT TEST ADDRESSES
                ;       2.  LOAD A TEST ADDRESS INTO RDM WD REGISTER
                ;       3.  EXECUTE DCS PROGRAM
                ;               a.  INVALIDATE TB
                ;               b.  SET MISS GROUP 0 AND REPLACE GROUP 1
                ;               c.  LOAD VA WITH 7FFF0000
                ;               d.  WRITE PTE 00DDDDD0 TO GROUP 1
                ;               e.  LOAD VA WITH TEST ADDRESS FROM RDM
                ;               f.  READ TB
                ;       4.  TEST WH REGISTER FOR TB MISS
                ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
                ;
                ;
                ; TEST PATTERNS:
                ;       THE TEST PATTERNS USED HERE ARE GENERATED BY FLOATING A 0
                ;       THRU 1'S IN VA BITS <30:16>, THE VA TAG FIELD.
                ;       1.      7FFE0000
                ;       2.      7FFD0000
                ;       3.      7FFB0000
                ;       4.      7FF70000
                ;       5.      7FEF0000
                ;       6.      7FDF0000
                ;       7.      7FBF0000
                ;       8.      7F7F0000
                ;       9.      7EFF0000
                ;       10.     7DFF0000
                ;       11.     7BFF0000
                ;       12.     77FF0000
                ;       13.     6FFF0000
                ;       14.     5FFF0000
                ;       15.     3FFF0000
                ;
```

```
                     0025  1775 ;
                     0025  1776 ; LOGIC DESCRIPTION:
                     0025  1777 ;       TESTED HERE ARE THE DC102 COMPARATORS FOR TB GROUP 1
                     0025  1778 ;
                     0025  1779 ;
                     0025  1780 ; ASSUMPTIONS:
                     0025  1781 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
                     0025  1782 ;
                     0025  1783 ;
                     0025  1784 ; ERROR DESCRIPTION:
                     0025  1785 ;       EXPECTED PTE
                     0025  1786 ;       GOT PTE
                     0025  1787 ;       INDEX WHICH IDENTIFIES THE FAILING TEST PATTERN (SUBTEST 2)
                     0025  1788 ;
                     0025  1789 ;--
                     0025  1790 ;*****************************************************************************
                     0025  1791
                     0025  1792         SUBTEST                                         ; SUBTEST #1
                     0028
                     0028       ;/////////////////////////////////////////////////////////////////////////////
                     0028  1793 ;+
                     0028  1794 ;SUBTEST #1
                     0028  1795 ;
                     0028  1796 ;       TEST TB LOCATION ZERO FOR A HIT WITH ADDRESS EQUAL 7FFF0000
                     0028  1797 ;-
                     0028  1798         INITIALIZE                                      ; INIT THE CPU
                     002A  1799         LOADDCS         1$,,<^X0C>,2                    ; LOAD FIRST UCODE OVERLAY.
                     0031  1800         ERRLOOP
                     0033  1801         FETCH           0                               ; EXECUTE THE UCODE.
                     0038  1802         BRSTCLK         <^X0D>
                     003C  1803         CMPREGMSK       WHREG,2$,,3$,,L                 ; CHECK THE PTE.
                     0048  1804         IFERROR         ,<<ADK>>,                       ; INCORRECT PTE
                     004E  1805
                     004E  1806         SUBTEST                                         ; SUBTEST #2
                     0051
                     0051       ;/////////////////////////////////////////////////////////////////////////////
                     0051  1807 ;+
                     0051  1808 ;SUBTEST #2
                     0051  1809 ;
                     0051  1810 ;       TEST TB FOR MISSES WITH ALL TEST PATTERNS
                     0051  1811 ;-
                     0051  1812         INITIALIZE                                      ; INIT THE CPU
                     0053  1813         LOADDCS         4$,,<^X0C>,3                    ; MODIFY DCS U-CODE
                     005A  1814         LOOP            I,1,15                          ; CREATE A TEST LOOP OF 15
                     0061  1815         LOADREG         WDREG,7$,I,L                    ; LOAD TEST ADDRESS INTO WD REG
                     0069  1816         ERRLOOP                                         ; ERROR LOOP POINT
                     006B  1817         FETCH           0                               ; START DCS PROGRAM
                     0070  1818         BRSTCLK         <^X0E>                          ; END DCS PROGRAM
                     0074  1819         CMPREGMSK       WHREG,5$,,6$,,L,                ; TEST FOR TB MISS
                     0080  1820         IFERROR         ,<<ADK>>,                       ; FAILING ARRAY
                     0086  1821         ENDLOOP         I                               ; END TEST LOOP
                     0089  1822         SKIP                                            ; GO TO NEXT TEST
                     0090  1823
                     0090  1824 BEGIN_TEST_DATA
                     0090
```

```
                                        0090        ;-----------------------------------------------------------------------
                                        0090  1825
                                        0090  1826 1$:
U 0C, 6701,881F,5924,0200,05F0,180D     0090  1827 ;% 0C   RDM_TB                                  ; U-CODE FOR SUBTEST 1
U 0D, 7300,C800,0364,0300,0470,180E     009B  1831 ;% 0D   NOP,STOP.CLOCK
                                        00A6  1835
                              00DDDDD0  00A6  1836 2$:     .LONG   ^X00DDDDD0                     ; COMPARE DATA SUBTEST 1
                                        00AA  1837
                              00FFFFF8  00AA  1838 3$:     .LONG   ^X00FFFFF8                     ; MASK FOR CMPREGMSK PSEUDO OP
                                        00AE  1839
                                        00AE  1840 4$:
U 0C, 5700,C800,0364,0300,4A70,180D     00AE  1841 ;% 0C   VA_RDM                                  ; U-CODE FOR SUBTEST 2
U 0D, 6701,881F,5924,0200,05F0,180E     00B9  1845 ;% 0D   RDM_TB
U 0E, 7300,4800,0364,0300,0470,180F     00C4  1849 ;% 0E   NOP,STOP.CLOCK
                                        00CF  1853
                              00000000  00CF  1854 5$:     .LONG   ^X00000000                     ; COMPARE DATA SUBTEST 2
                                        00D3  1855
                              00000100  00D3  1856 6$:     .LONG   ^X00000100                     ; MASK FOR CMPREGMSK PSEUDO OP
                                        00D7  1857
                              7FFE0000  00D7  1858 7$:     .LONG   ^X7FFE0000
                              7FFD0000  00DB  1859         .LONG   ^X7FFD0000
                              7FFB0000  00DF  1860         .LONG   ^X7FFB0000
                              7FF70000  00E3  1861         .LONG   ^X7FF70000
                              7FEF0000  00E7  1862         .LONG   ^X7FEF0000
                              7FDF0000  00EB  1863         .LONG   ^X7FDF0000
                              7FBF0000  00EF  1864         .LONG   ^X7FBF0000
                              7F7F0000  00F3  1865         .LONG   ^X7F7F0000
                              7EFF0000  00F7  1866         .LONG   ^X7EFF0000
                              7DFF0000  00FB  1867         .LONG   ^X7DFF0000
                              7BFF0000  00FF  1868         .LONG   ^X7BFF0000
                              77FF0000  0103  1869         .LONG   ^X77FF0000
                              6FFF0000  0107  1870         .LONG   ^X6FFF0000
                              5FFF0000  010B  1871         .LONG   ^X5FFF0000
                              3FFF0000  010F  1872         .LONG   ^X3FFF0000
                                        0113  1873
                                        0113  1874 BEGIN_CPU_DATA
                                        0002  1875
                                        0002  1876 DCS_DATA
                                        0001  1877
U 00, 7700,C800,0364,0300,0470,1801     0001  1878 ;% 00   NOP
U 01, 7700,8800,5BE6,03D8,0470,1802     000C  1882 ;% 01   D_R[ZERO]                               ; zero the D register
U 02, 7700,D800,DB13,0300,4A70,1803     0017  1886 ;% 02   VA_Q_D_D+ZLIT8[0]                       ; invalidate the TB at 0 &
U 03, 7700,1800,DB12,0300,5360,1804     0022  1890 ;% 03   VA_D_D+ZLIT8[0] INVALIDATE TB           ; 200...........
U 04, 7700,F800,7E7F,FFFF,8470,1805     002D  1894 ;% 04   LONLIT_[03000000]                       ; SET FORCE MISS GROUP 0 AND
U 05, 7700,4800,5BE4,03D4,6870,1806     0038  1898 ;% 05   MEMSCAR_R[LONLIT]                       ; FORCE REPLACE GROUP 1
U 06, 7700,3800,797F,FFFF,8470,1807     0043  1902 ;% 06   LONLIT_[0D000000]
U 07, 7700,4800,5BE4,03D4,6070,1808     004E  1906 ;% 07   MEMSCR_R[LONLIT]
U 08, 7700,B800,4000,7FFF,8470,1809     0059  1910 ;% 08   LONLIT_[7FFF0000]
U 09, 7700,4800,5BE4,03D4,4A70,180A     0064  1914 ;% 09   VA_R[LONLIT]
U 0A, 7700,3800,7F91,1117,8470,180B     006F  1918 ;% 0A   LONLIT_[00DDDDD0]                       ; MAKE THE TEST PATTERN AND
U 0B, 7700,C81B,5BE4,03D4,5070,180C     007A  1922 ;% 0B   TB_R[LONLIT]                            ; PTE=00DDDDD0 A HIT IN GROUP 1
U 0C, 6701,881F,5924,0200,05F0,180D     0085  1926 ;% 0C   RDM_TB                                  ; ADDRESS AND READ THE TB AT
U 0D, 7300,C800,0364,0300,0470,180E     0090  1930 ;% 0D   STOP.CLOCK                              ; THAT ADDRESS.
                                        009B  1934
                                        009B  1935 END_CPU_DATA
```

```
                0113  1936
                0113  1937 END_TEST_DATA
                0113
                0113       ;-----------------------------------------------------------------------
                0113  1938
                0113  1939 ENDTEST
```

```
0020          .SBTTL  "TEST   03A     TB GROUP 0 M BIT UTRAP TEST
0020  1941 ;*******************************************************************************
0020  1942 ;++
0020  1943 ;
0020  1944 ;       TB GROUP 0 M BIT UTRAP TEST
0020  1945 ;
0020  1946 ; FUNCTIONAL DESCRIPTION
0020  1947 ;
0020  1948 ;       THIS TEST CHECKS THE LOGIC WHICH GENERATES AN M BIT UTRAP
0020  1949 ;       CONDITION IN THE TRANSLATION BUFFER. A PTE IS MADE A HIT
0020  1950 ;       IN THE TRANSLATION BUFFER WITH ITS M BIT CLEAR. THEN A
0020  1951 ;       WRITE CYCLE IS PERFORMED WHICH SHOULD RESULT IN AN M BIT
0020  1952 ;       UTRAP.
0020  1953 ;
0020  1954 ; TEST ALGORITHM
0020  1955 ;       1.      DISABLE CACHE AND CMI.
0020  1956 ;       2.      INVALIDATE BOTH TB GROUPS AT INDEX LOCATION 0.
0020  1957 ;       3.      SET THE TB GROUP 1 FORCE MISS BIT AND THE TB GROUP 0
0020  1958 ;               FORCE REPLACE BIT.
0020  1959 ;       4.      LOAD THE VA WITH 55550000.
0020  1960 ;       5.      WRITE PTE=00123F50 WITH VA=55550000 INTO TB GROUP 0,
0020  1961 ;               NOTE THAT PTE=00123F50 HAS THE M BIT CLEAR.
0020  1962 ;       6.      PERFORM A WRITE CYCLE.
0020  1963 ;       7.      CHECK THAT A UTRAP OCCURRED AND THAT THE MICRO TRAP
0020  1964 ;               VECTOR IS CORRECT FOR M BIT UTRAPS.
0020  1965 ;
0020  1966 ; TEST PATTERNS
0020  1967 ;       NONE
0020  1968 ;
0020  1969 ; LOGIC DESCRIPTION
0020  1970 ;       THIS TEST CHECKS THE ABILITY OF THE UTR GATE ARRAY TO DETECT A
0020  1971 ;       WRITE OPERATION WITH THE M BIT NOT SET.
0020  1972 ;
0020  1973 ; ASSUMPTIONS
0020  1974 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
0020  1975 ;
0020  1976 ; ERROR DESCRIPTION
0020  1977 ;       EXPECTED UTRAP VECTOR
0020  1978 ;       GOT MICRO PC
0020  1979 ;
0020  1980 ;--
0020  1981 ;*******************************************************************************
0020  1982 ;///////////////////////////////////////////////////////////////////////////////
0020  1983
0020  1984          INITIALIZE
0022  1985          EXPUTRAP                                ; SET UP FOR UCODE TRAP.
0024  1986          LOADDCS         1$,,0,<^X1A>            ; LOAD TEST UCODE.
002B  1987          ERRLOOP                                 ; ERROR LOOP
002D  1988          FETCH           0                       ; EXECUTE SERIES OF MICRO
0032  1989          BRSTCLK         <^X19>                  ; INSTRUCTIONS WHICH WILL
0036  1990                                                  ; WRITE A PTE WITH M=0 INTO
0036  1991                                                  ; GROUP 0 AND THEN DO A BUS
0036  1992                                                  ; WRITE FUNCTION WHICH SHOULD
0036  1993                                                  ; RESULT IN A TB M BIT UTRAP.
0036  1994          CMPREGMSK       CSTRP,2$,,3$,,W,        ; SEE IF TRAP OCCURRED.
```

```
                        0042  1995          IFERROR         ,<<UTR>>,                       ; TB M BIT UTRAP DIDN'T OCCUR.
                        0048  1996          SKIP                                            ; GO TO NEXT TEST
                        004F  1997
                        004F  1998 BEGIN_TEST_DATA
                        004F
                        004F          ;-------------------------------------------------------------------------
                        004F  1999
                        004F  2000 1$:
U 00. 7700,C800,0364,0300,0470,1801     004F  2001 ;% 00    NOP
U 01. 7700,3800,78FF,FFFF,8470,1802     005A  2005 ;% 01    LONLIT_[0E000000]                   ; TURN OFF THE CMI.
U 02. 7700,4800,5BE4,03D4,6870,1803     0065  2009 ;% 02    MEMSCAR_R[LONLIT]
U 03. 7700,3800,7F7F,FFFF,8470,1804     0070  2013 ;% 03    LONLIT_[01000000]
U 04. 7700,C800,5BE4,03D4,6070,1805     007B  2017 ;% 04    MEMSCR_R[LONLIT]
U 05. 7700,F800,7CFF,FFFF,8470,1806     0086  2021 ;% 05    LONLIT_[06000000]                   ; TURN OFF THE CACHE.
U 06. 7700,C800,5BE4,03D4,6870,1807     0091  2025 ;% 06    MEMSCAR_R[LONLIT]
U 07. 7700,3800,7F7F,FFFF,8470,1808     009C  2029 ;% 07    LONLIT_[01000000]
U 08. 7700,C800,5BE4,03D4,6070,1809     00A7  2033 ;% 08    MEMSCR_R[LONLIT]
U 09. 7700,F800,7FFF,FFFF,8470,180A     00B2  2037 ;% 09    LONLIT_[00000000]                   ; SET MME.
U 0A. 7700,C800,5BE4,03D4,6870,180B     00BD  2041 ;% 0A    MEMSCAR_R[LONLIT]
U 0B. 7700,B800,7F7F,FFFF,8470,180C     00C8  2045 ;% 0B    LONLIT_[01000000]
U 0C. 7700,4800,5BE4,03D4,6070,180D     00D3  2049 ;% 0C    MEMSCR_R[LONLIT]
U 0D. 7700,8800,5BE6,03D8,0470,180E     00DE  2053 ;% 0D    D_R[ZERO]                           ; zero the D register
U 0E. 7700,D800,DB13,0300,4A70,180F     00E9  2057 ;% 0E    VA_Q_D_D+ZLIT8[0]                   ; invalidate the TB at 0 &
U 0F. 7700,1800,DB12,0300,5360,1810     00F4  2061 ;% 0F    VA_D_D+ZLIT8[0] INVALIDATE TB       ; 200.....
U 10. 7700,F800,7E7F,FFFF,8470,1811     00FF  2065 ;% 10    LONLIT_[03000000]                   ; SET FORCE REPLACE GROUP 0
U 11. 7700,4800,5BE4,03D4,6870,1812     010A  2069 ;% 11    MEMSCAR_R[LONLIT]                   ; FORCE MISS TO GROUP 1.
U 12. 7700,7800,7AFF,FFFF,8470,1813     0115  2073 ;% 12    LONLIT_[0A000000]
U 13. 7700,C800,5BE4,03D4,6070,1814     0120  2077 ;% 13    MEMSCR_R[LONLIT]
U 14. 7700,7800,5555,7FFF,8470,1815     012B  2081 ;% 14    LONLIT_[55550000]                   ; MAKE VA=55550000 AND
U 15. 7700,C800,5BE4,03D4,4A70,1816     0136  2085 ;% 15    VA_R[LONLIT]                        ; PTE=00123F50 (NOTE M BIT
U 16. 7700,F800,7FF6,E057,8470,1817     0141  2089 ;% 16    LONLIT_[00123F50]                   ; OFF) A HIT IN GROUP 0.
U 17. 7700,C81B,5BE4,03D4,5070,1818     014C  2093 ;% 17    TB_R[LONLIT]
U 18. 7701,4800,0364,0300,0580,1819     0157  2097 ;% 18    BUS/WRITE,VSIZE/1                   ; THIS WRITE SHOULD RESULT IN
U 19. 7300,C800,0364,0300,0470,181A     0162  2101 ;% 19    STOP.CLOCK                          ; A TB MISS UTRAP.
                                        016D  2105
                                002B    016D  2106 2$:      .WORD   ^X002B                      ; EXPECTED MICRO PC OF
                                        016F  2107
                                3FFF    016F  2108 3$:      .WORD   ^X3FFF                      ; MASK FOR CMPREGMSK PSEUDO OP
                                        0171  2109                                            ; A TB M BIT MICRO TRAP VECTOR.
                                        0171  2110 END_TEST_DATA
                                        0171
                                        0171          ;-------------------------------------------------------------------------
                                        0171  2111
                                        0171  2112 ENDTEST
```

```
                        0020        .SBTTL  "TEST   03B     TB GROUP 1 M BIT UTRAP TEST
                        0020  2114 ;*******************************************************************************
                        0020  2115 ;++
                        0020  2116 ;
                        0020  2117 ;       TB GROUP 1 M BIT UTRAP TEST
                        0020  2118 ;
                        0020  2119 ; FUNCTIONAL DESCRIPTION
                        0020  2120 ;       THIS TEST CHECKS THE LOGIC WHICH GENERATES AN M BIT UTRAP
                        0020  2121 ;       CONDITION IN THE TRANSLATION BUFFER. A PTE IS MADE A HIT
                        0020  2122 ;       IN THE TRANSLATION BUFFER WITH ITS M BIT CLEAR. THEN A
                        0020  2123 ;       WRITE CYCLE IS PERFORMED WHICH SHOULD RESULT IN AN M BIT
                        0020  2124 ;       UTRAP.
                        0020  2125 ;
                        0020  2126 ; TEST ALGORITHM
                        0020  2127 ;       1.      DISABLE CACHE AND CMI.
                        0020  2128 ;       2.      INVALIDATE BOTH TB GROUPS AT INDEX LOCATION 0.
                        0020  2129 ;       3.      SET THE TB GROUP 0 FORCE MISS BIT AND THE TB GROUP 1
                        0020  2130 ;               FORCE REPLACE BIT.
                        0020  2131 ;       4.      LOAD THE VA WITH 55550000.
                        0020  2132 ;       5.      WRITE PTE=00123F50 WITH VA=55550000 INTO TB GROUP 1,
                        0020  2133 ;               NOTE THAT PTE=00123F50 HAS THE M BIT CLEAR.
                        0020  2134 ;       6.      PERFORM A WRITE CYCLE.
                        0020  2135 ;       7.      CHECK THAT A UTRAP OCCURRED AND THAT THE MICRO TRAP
                        0020  2136 ;               VECTOR IS CORRECT FOR M BIT UTRAPS.
                        0020  2137 ;
                        0020  2138 ; TEST PATTERNS
                        0020  2139 ;       NONE
                        0020  2140 ;
                        0020  2141 ; LOGIC DESCRIPTION
                        0020  2142 ;       THIS TEST CHECKS THE ABILITY OF THE UTR GATE ARRAY TO DETECT A
                        0020  2143 ;       WRITE OPERATION WITH THE M BIT NOT SET.
                        0020  2144 ;
                        0020  2145 ; ASSUMPTIONS
                        0020  2146 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
                        0020  2147 ;
                        0020  2148 ; ERROR DESCRIPTION
                        0020  2149 ;       EXPECTED UTRAP VECTOR
                        0020  2150 ;       GOT MICRO PC
                        0020  2151 ;
                        0020  2152 ;--
                        0020  2153 ;*******************************************************************************
                        0020  2154 ;///////////////////////////////////////////////////////////////////////////////
                        0020  2155
                        0020  2156         INITIALIZE
                        0022  2157         EXPUTRAP                                ; SET UP FOR UCODE TRAP.
                        0024  2158         LOADDCS         1$,,0,<^X1A>            ; LOAD TEST UCODE.
                        002B  2159         ERRLOOP                                 ; ERROR LOOP
                        002D  2160         FETCH           0                       ; EXECUTE SERIES OF MICRO
                        0032  2161         BRSTCLK         <^X19>                  ; INSTRUCTIONS WHICH WILL
                        0036  2162                                                 ; WRITE A PTE WITH M=0 INTO
                        0036  2163                                                 ; GROUP 1 AND THEN DO A BUS
                        0036  2164                                                 ; WRITE FUNCTION WHICH SHOULD
                        0036  2165                                                 ; RESULT IN A TB M BIT UTRAP.
                        0036  2166         CMPREGMSK       CSTRP,2$,,3$,,W,        ; SEE IF TRAP OCCURRED.
                        0042  2167         IFERROR         ,<<UTR>>,               ; TB M BIT UTRAP DIDN'T OCCUR.
```

```
                                0048  2168          SKIP
                                004F  2169
                                004F  2170 BEGIN_TEST_DATA
                                004F
                                004F          ;-------------------------------------------------------------------------
                                004F  2171
                                004F  2172 1$:
U 00. 7700.C800.0364.0300.0470.1801   004F  2173 ;% 00    NOP
U 01. 7700.3800.78FF.FFFF.8470.1802   005A  2177 ;% 01    LONLIT_[0E000000]                          ; TURN OFF THE CMI.
U 02. 7700.4800.5BE4.03D4.6870.1803   0065  2181 ;% 02    MEMSCAR_R[LONLIT]
U 03. 7700.3800.7F7F.FFFF.8470.1804   0070  2185 ;% 03    LONLIT_[01000000]
U 04. 7700.C800.5BE4.03D4.6070.1805   007B  2189 ;% 04    MEMSCR_R[LONLIT]
U 05. 7700.F800.7CFF.FFFF.8470.1806   0086  2193 ;% 05    LONLIT_[06000000]                          ; TURN OFF THE CACHE.
U 06. 7700.C800.5BE4.03D4.6870.1807   0091  2197 ;% 06    MEMSCAR_R[LONLIT]
U 07. 7700.3800.7F7F.FFFF.8470.1808   009C  2201 ;% 07    LONLIT_[01000000]
U 08. 7700.C800.5BE4.03D4.6070.1809   00A7  2205 ;% 08    MEMSCR_R[LONLIT]
U 09. 7700.F800.7FFF.FFFF.8470.180A   00B2  2209 ;% 09    LONLIT_[00000000]                          ; SET MME.
U 0A. 7700.C800.5BE4.03D4.6870.180B   00BD  2213 ;% 0A    MEMSCAR_R[LONLIT]
U 0B. 7700.B800.7F7F.FFFF.8470.180C   00C8  2217 ;% 0B    LONLIT_[01000000]
U 0C. 7700.4800.5BE4.03D4.6070.180D   00D3  2221 ;% 0C    MEMSCR_R[LONLIT]
U 0D. 7700.8800.5BE6.03D8.0470.180E   00DE  2225 ;% 0D    D_R[ZERO]                                  ; zero the D register
U 0E. 7700.D800.DB13.0300.4A70.180F   00E9  2229 ;% 0E    VA_Q_D_D+ZLIT8[0]                          ; invalidate the TB at 0 &
U 0F. 7700.1800.DB12.0300.5360.1810   00F4  2233 ;% 0F    VA_D_D+ZLIT8[0] INVALIDATE TB              ; 200.....
U 10. 7700.F800.7E7F.FFFF.8470.1811   00FF  2237 ;% 10    LONLIT_[03000000]                          ; SET FORCE REPLACE GROUP 1
U 11. 7700.4800.5BE4.03D4.6870.1812   010A  2241 ;% 11    MEMSCAR_R[LONLIT]                          ; FORCE MISS TO GROUP 0.
U 12. 7700.3800.797F.FFFF.8470.1813   0115  2245 ;% 12    LONLIT_[0D000000]
U 13. 7700.C800.5BE4.03D4.6070.1814   0120  2249 ;% 13    MEMSCR_R[LONLIT]
U 14. 7700.7800.5555.7FFF.8470.1815   012B  2253 ;% 14    LONLIT_[55550000]                          ; MAKE VA=55550000 AND
U 15. 7700.C800.5BE4.03D4.4A70.1816   0136  2257 ;% 15    VA_R[LONLIT]                               ; PTE=00123F50 (NOTE M BIT
U 16. 7700.F800.7FF6.E057.8470.1817   0141  2261 ;% 16    LONLIT_[00123F50]                          ; OFF) A HIT IN GROUP 1.
U 17. 7700.C81B.5BE4.03D4.5070.1818   014C  2265 ;% 17    TB_R[LONLIT]
U 18. 7701.4800.0364.0300.0580.1819   0157  2269 ;% 18    BUS/WRITE,VSIZE/1                          ; THIS WRITE SHOULD RESULT IN
U 19. 7300.C800.0364.0300.0470.181A   0162  2273 ;% 19    STOP.CLOCK                                 ; A TB MISS UTRAP.
                                      016D  2277
                                002B  016D  2278 2$:      .WORD   ^X002B                             ; EXPECTED MICRO PC OF
                                      016F  2279
                                3FFF  016F  2280 3$:      .WORD   ^X3FFF                             ; MASK FOR CMPREGMSK PSEUDO OP
                                      0171  2281                                                   ; A TB M BIT MICRO TRAP VECTOR.
                                      0171  2282 END_TEST_DATA
                                      0171
                                      0171          ;-------------------------------------------------------------------------
                                      0171  2283
                                      0171  2284 ENDTEST
```

```
                        .SBTTL  "TEST   03C     TB GROUP 0 PTE MEMORY DUAL ADDRESSING TEST
002F   2286 ;*******************************************************************************
002F   2287 ;++
002F   2288 ;
002F   2289 ;       TB GROUP 0 PTE MEMORY DUAL ADDRESSING TEST
002F   2290 ;
002F   2291 ; FUNCTIONAL DESCRIPTION
002F   2292 ;       THIS IS A TEST OF THE ADDRESSING LINES THAT GO TO THE TB GROUP 0
002F   2293 ;       PTE MEMORY.
002F   2294 ;
002F   2295 ; TEST ALGORITHM
002F   2296 ;       1.      SET THE TB GROUP 1 FORCE MISS BIT AND SET THE TB GROUP 0
002F   2297 ;               FORCE REPLACE BIT.
002F   2298 ;       2.      LOAD THE VA WITH TEST ADDRESS 1.
002F   2299 ;       3.      WRITE PTE = 00000100 (JUST THE VALID BIT SET) WITH THE
002F   2300 ;               VA INTO TB GROUP 0.
002F   2301 ;       4.      LOAD THE VA WITH TEST ADDRESS 2.
002F   2302 ;       5.      WRITE PTE = 00FFFFF8 (ALL BITS SET) WITH THE VA INTO GROUP 0.
002F   2303 ;       6.      LOAD THE VA WITH TEST ADDRESS 1.
002F   2304 ;       7.      READ THE PTE OUT OF TB GROUP 0 AT TEST ADDRESS 1.
002F   2305 ;       8.      CHECK THIS PTE, IT SHOULD HAVE JUST THE VALID BIT SET.
002F   2306 ;       9.      REPEAT 1. THRU 8. FOR EVERY COMBINATION OF TEST ADDRESS 1 AND
002F   2307 ;               TEST ADDRESS 2 TAKEN FROM THE TABLE BELOW EXCEPT FOR WHEN
002F   2308 ;               TEST ADDRESS 1. EQUALS TEST ADDRESS 2.
002F   2309 ;
002F   2310 ; TEST PATTERNS
002F   2311 ;       PICK TEST ADDRESS 1 AND TEST ADDRESS 2 IN SUCH A WAY AS EVERY
002F   2312 ;       COMBINATION OF TWO ADDRESS TAKEN FROM THIS TABLE ARE USED:
002F   2313 ;       1.      00000000
002F   2314 ;       2.      80000000
002F   2315 ;       3.      00008000
002F   2316 ;       4.      00004000
002F   2317 ;       5.      00002000
002F   2318 ;       6.      00001000
002F   2319 ;       7.      00000800
002F   2320 ;       8.      00000400
002F   2321 ;       9.      00000200
002F   2322 ;       NOTE THAT ALL COMBINATIONS OF TWO ADDRESSES TAKEN FROM THIS
002F   2323 ;       TABLE ARE TRIED FOR TEST ADDRESS 1 AND TEST ADDRESS 2 EXCEPT
002F   2324 ;       FOR CASES WHERE TEST ADDRESS 1 EQUALS TEST ADDRESS 2.
002F   2325 ;
002F   2326 ; LOGIC DESCRIPTION
002F   2327 ;       THIS TEST INVOLVES MOSTLY DISCRETE LOGIC HANGING ON MAD BUS.
002F   2328 ;       THE ONLY GATE ARRAY THAT MIGHT AFFECT THE OUTCOME WOULD BE ADD.
002F   2329 ;
002F   2330 ; ASSUMPTIONS
002F   2331 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL.
002F   2332 ;
002F   2333 ; ERROR DESCRIPTION
002F   2334 ;       EXPECTED PTE
002F   2335 ;       GOT PTE
002F   2336 ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
002F   2337 ;               BEING USED AS TEST ADDRESS 1 WHEN THE ERROR OCCURRED.
002F   2338 ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
002F   2339 ;               BEING USED AS TEST ADDRESS 2 WHEN THE ERROR OCCURRED.
```

```
                        002F  2340 ;
                        002F  2341 ;--
                        002F  2342 ;*******************************************************************************
                        002F  2343 ;///////////////////////////////////////////////////////////////////////////////
                        002F  2344
                        002F  2345         INITIALIZE
                        0031  2346         LOADDCS          1$,,0,<^X16>                ; LOAD TEST U-CODE
                        0038  2347         LOOP             I,1,9                       ; INITIALIZE A LOOP TO PICK
                        003F  2348                                                      ; TEST ADDRESS 1.
                        003F  2349         LDFIELD          10$,I,LONG,2$,31,62,LONLIT ; MAKE TEST ADDRESS 1 PART
                        0050  2350                                                      ; OF THIS UINSTRUCTION'S LONLIT.
                        0050  2351         LOADDCS          2$,,1,1                     ; LOAD THE TEST UCODE.
                        0057  2352         LOOP             J,1,9                       ; INITIALIZE A LOOP TO PICK
                        005E  2353                                                      ; TEST ADDRESS 2.
                        005E  2354         SKIP             100$,ONEQUAL,I,J            ; IF I AND J ARE EQUAL SKIP
                        0065  2355                                                      ; THIS ITERATION.
                        0065  2356         LOADREG          WDREG,10$,J,LONG            ; GET TEST ADDRESS 2.
                        006D  2357         ERRLOOP
                        006F  2358         FETCH            0                           ; EXECUTE THE TEST UCODE.
                        0074  2359         BRSTCLK          <^X15>
                        0078  2360         CMPREGMSK        WHREG,3$,,5$,,LONG          ; CHECK THE RESULT.
                        0084  2361         IFERROR          ,<<ADD>>,                   ; DUAL ADDRESSING PROBLEM
                        008A  2362                                                      ; WITH TB GROUP 0 PTE MEMORY.
                        008A  2363 100$:   ENDLOOP          J                           ; SELECT NEXT TEST ADDRESS 2.
                        008D  2364         ENDLOOP          I                           ; SELECT NEXT TEST ADDRESS 1.
                        0090  2365         SKIP
                        0097  2366
                        0097  2367 BEGIN_TEST_DATA
                        0097
                        0097       ;-------------------------------------------------------------------------
                        0097  2368
                        0097  2369 1$:
U 00, 7700,C800,0364,0300,0470,1801     0097  2370 ;* 00    NOP
                        00A2  2374 ;
                        00A2  2375 ; Micro addresses 01-08 are used to invalidate the TB at both test address 1
                        00A2  2376 ; and test address 2.  The lonlit value that gets loaded in address 01 will
                        00A2  2377 ; be changed for each address test case.
                        00A2  2378 ;
                        00A2  2379 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     00A2  2380 ;* 01    LONLIT_[00000000]                   ; load test address 1
U 02, 7700,0800,5BE6,03D4,0470,1803     00AD  2384 ;* 02    D_R[LONLIT]                         ; store address in D
U 03, 7700,5800,DB13,0300,4A70,1804     00B8  2388 ;* 03    VA_Q_D_D+ZLIT8[0]                   ; set up VA & Q
U 04, 7700,9800,DB12,0300,5360,1805     00C3  2392 ;* 04    VA_D_D+ZLIT8[0] INVALIDATE TB       ; invalidate for test address 1
U 05, 5700,0840,0364,0300,0470,1806     00CE  2396 ;* 05    R[TEMP0]_RDM                        ; load test address 2
U 06, 7700,8800,5BE6,0300,0470,1807     00D9  2400 ;* 06    D_R[TEMP0]                          ; D <-- test address 2
U 07, 7700,1868,DB10,0300,0470,1808     00E4  2404 ;* 07    M[TEMP8]_D+ZLIT8[0]                 ; store in MTEMP8
U 08, 7700,9800,DB12,0300,5360,1809     00EF  2408 ;* 08    VA_D_D+ZLIT8[0] INVALIDATE TB       ; invalidate for test address 2
U 09, 7700,F800,7E7F,FFFF,8470,180A     00FA  2412 ;* 09    LONLIT_[03000000]                   ; SET FORCE MISSES TO TB GROUP
U 0A, 7700,C800,5BE4,03D4,6870,180B     0105  2416 ;* 0A    MEMSCAR_R[LONLIT]                   ; 1 AND REPLACEMENT OF GROUP 0.
U 0B, 7700,F800,7AFF,FFFF,8470,180C     01 0  2420 ;* 0B    LONLIT_[0A000000]
U 0C, 7700,4800,5BE4,03D4,6070,180D     01 B  2424 ;* 0C    MEMSCR_R[LONLIT]
U 0D, 7700,C800,03A4,03D8,4A70,180E     0126  2428 ;* 0D    VA_Q                                ; WRITE VA=TEST ADDRESS 1 AND
U 0E, 7700,B800,7FFF,FF7F,8470,180F     0131  2432 ;* 0E    LONLIT_[00000100]                   ; PTE=00000100 INTO GROUP 0.
U 0F, 7700,481B,5BE4,03D4,5070,1810     013C  2436 ;* 0F    TB_R[LONLIT]
U 10, 5700,4800,0364,0300,4A70,1811     0147  2440 ;* 10    VA_RDM                              ; WRITE VA=TEST ADDRESS 2 AND
```

```
U 11. 7700,B800,7F80,0003,8470,1812  0152  2444 ;% 11   LONLIT_[00FFFFF8]                    ; PTE=00FFFFF8 INTO GROUP 0.
U 12. 7700,481B,5BE4,03D4,5070,1813  015D  2448 ;% 12   TB_R[LONLIT]
U 13. 7700,4800,03A4,03D8,4A70,1814  0168  2452 ;% 13   VA_Q                               ; READ THE PTE AT VA=TEST
U 14. 6701,881F,5924,0200,05F0,1815  0173  2456 ;% 14   RDM_TB                             ; ADDRESS 1.
U 15. 7300,C800,0364,0300,0470,1816  017E  2460 ;% 15   STOP.CLOCK
                                     0189  2464
                          00000100   0189  2465 3$:     .LONG   ^X00000100
                                     018D  2466
                          00FFFFF8   018D  2467 5$:     .LONG   ^X00FFFFF8
                                     0191  2468
                          00000000   0191  2469 10$:    .LONG   ^X00000000
                          80000000   0195  2470         .LONG   ^X80000000
                          00008000   0199  2471         .LONG   ^X00008000
                          00004000   019D  2472         .LONG   ^X00004000
                          00002000   01A1  2473         .LONG   ^X00002000
                          00001000   01A5  2474         .LONG   ^X00001000
                          00000800   01A9  2475         .LONG   ^X00000800
                          00000400   01AD  2476         .LONG   ^X00000400
                          00000200   01B1  2477         .LONG   ^X00000200
                                     01B5  2478
                                     01B5  2479 END_TEST_DATA
                                     01B5
                                     01B5       ;-----------------------------------------------------------------------------
                                     01B5  2480
                                     01B5  2481 ENDTEST
```

```
                        .SBTTL  "TEST   03D     TB GROUP 1 PTE MEMORY DUAL ADDRESSING TEST
                ;*****************************************************************************
                ;++
                ;
                ;       TB GROUP 1 PTE MEMORY DUAL ADDRESSING TEST
                ;
                ; FUNCTIONAL DESCRIPTION
                ;       THIS IS A TEST OF THE ADDRESSING LINES THAT GO TO THE TB GROUP 1
                ;       PTE MEMORY.
                ;
                ; TEST ALGORITHM
                ;       1.      SET THE TB GROUP 0 FORCE MISS BIT AND SET THE TB GROUP 1
                ;               FORCE REPLACE BIT.
                ;       2.      LOAD THE VA WITH TEST ADDRESS 1.
                ;       3.      WRITE PTE = 00000100 (JUST THE VALID BIT SET) WITH THE
                ;               VA INTO TB GROUP 1.
                ;       4.      LOAD THE VA WITH TEST ADDRESS 2.
                ;       5.      WRITE PTE = 00FFFFF8 (ALL BITS SET) WITH THE VA INTO GROUP 1.
                ;       6.      LOAD THE VA WITH TEST ADDRESS 1.
                ;       7.      READ THE PTE OUT OF TB GROUP 1 AT TEST ADDRESS 1.
                ;       8.      CHECK THIS PTE, IT SHOULD HAVE JUST THE VALID BIT SET.
                ;       9.      REPEAT 1. THRU 8. FOR EVERY COMBINATION OF TEST ADDRESS 1 AND
                ;               TEST ADDRESS 2 TAKEN FROM THE TABLE BELOW EXCEPT FOR WHEN
                ;               TEST ADDRESS 1. EQUALS TEST ADDRESS 2.
                ;
                ; TEST PATTERNS
                ;       PICK TEST ADDRESS 1 AND TEST ADDRESS 2 IN SUCH A WAY AS EVERY
                ;       COMBINATION OF TWO ADDRESS TAKEN FROM THIS TABLE ARE USED:
                ;       1.      00000000
                ;       2.      80000000
                ;       3.      00008000
                ;       4.      00004000
                ;       5.      00002000
                ;       6.      00001000
                ;       7.      00000800
                ;       8.      00000400
                ;       9.      00000200
                ;       NOTE THAT ALL COMBINATIONS OF TWO ADDRESSES TAKEN FROM THIS
                ;       TABLE ARE TRIED FOR TEST ADDRESS 1 AND TEST ADDRESS 2 EXCEPT
                ;       FOR CASES WHERE TEST ADDRESS 1 EQUALS TEST ADDRESS 2.
                ;
                ; LOGIC DESCRIPTION
                ;       THIS TEST INVLOVES MOSTLY DISCRETE LOGIC HANGING ON THE MAD BUS.
                ;       THE ONLY GATE ARRAY THAT MIGHT AFFECT THE OUTCOME IS ADD.
                ;
                ; ASSUMPTIONS
                ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
                ;
                ; ERROR DESCRIPTION
                ;       EXPECTED PTE
                ;       GOT PTE
                ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
                ;               BEING USED AS TEST ADDRESS 1 WHEN THE ERROR OCCURRED.
                ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
                ;               BEING USED AS TEST ADDRESS 2 WHEN THE ERROR OCCURRED.
```

```
                002F  2537 ;
                002F  2538 ;--
                002F  2539 ;*******************************************************************************
                002F  2540 ;///////////////////////////////////////////////////////////////////////////////
                002F  2541
                002F  2542         INITIALIZE
                0031  2543         LOADDCS         1$,,0,<^X16>                    ; LOAD TEST U-CODE
                0038  2544         LOOP            I,1,9                           ; INITIALIZE A LOOP TO PICK
                003F  2545                                                         ; TEST ADDRESS 1.
                003F  2546         LDFIELD         10$,I,LONG,2$,31,62,LONLIT ; MAKE TEST ADDRESS 1 PART
                0050  2547                                                         ; OF THIS UINSTRUCTION'S LONLIT.
                0050  2548         LOADDCS         2$,,1,1                         ; LOAD THE TEST UCODE.
                0057  2549         LOOP            J,1,9                           ; INITIALIZE A LOOP TO PICK
                005E  2550                                                         ; TEST ADDRESS 2.
                005E  2551         SKIP            100$,ONEQUAL,I,J                ; IF I AND J ARE EQUAL SKIP
                0065  2552                                                         ; THIS ITERATION.
                0065  2553         LOADREG         WDREG,10$,J,LONG                        ; GET TEST ADDRESS 2.
                006D  2554         ERRLOOP
                006F  2555         FETCH           0                               ; EXECUTE THE TEST UCODE.
                0074  2556         BRSTCLK         <^X15>
                0078  2557         CMPREGMSK       WHREG,3$,,5$,,LONG              ; CHECK THE RESULT.
                0084  2558         IFERROR         ,<<ADD>>,                       ; DUAL ADDRESSING PROBLEM
                008A  2559                                                         ; WITH TB GROUP 1 PTE MEMORY.
                008A  2560 100$:   ENDLOOP         J                               ; SELECT NEXT TEST ADDRESS 2.
                008D  2561         ENDLOOP         I                               ; SELECT NEXT TEST ADDRESS 1.
                0090  2562         SKIP
                0097  2563
                0097  2564 BEGIN_TEST_DATA
                0097
                0097       ;-------------------------------------------------------------------------
                0097  2565
                0097  2566 1$:
U 00, 7700,C800,0364,0300,0470,1801   0097  2567 ;% 00   NOP
                00A2  2571 ;
                00A2  2572 ; Micro addresses 01-08 are used to invalidate the TB at both test address 1
                00A2  2573 ; and test address 2.  The lonlit value that gets loaded in address 01 will
                00A2  2574 ; be changed for each address test case.
                00A2  2575 ;
                00A2  2576 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802   00A2  2577 ;% 01   LONLIT_[00000000]                  ; load test address 1
U 02, 7700,0800,5BE6,03D4,0470,1803   00AD  2581 ;% 02   D_R[LONLIT]                       ; store address in D
U 03, 7700,5800,DB13,0300,4A70,1804   00B8  2585 ;% 03   VA_Q_D_D+ZLIT8[0]                ; set up VA & Q
U 04, 7700,9800,DB12,0300,5360,1805   00C3  2589 ;% 04   VA_D_D+ZLIT8[0] INVALIDATE TB    ; invalidate for test address 1
U 05, 5700,0840,0364,0300,0470,1806   00CE  2593 ;% 05   R[TEMP0]_RDM                     ; load test address 2
U 06, 7700,8800,5BE6,0300,0470,1807   00D9  2597 ;% 06   D_R[TEMP0]                      ; D <-- test address 2
U 07, 7700,1868,DB10,0300,0470,1808   00E4  2601 ;% 07   M[TEMP8]_D+ZLIT8[0]             ; store in MTEMP8
U 08, 7700,9800,DB12,0300,5360,1809   00EF  2605 ;% 08   VA_D_D+ZLIT8[0] INVALIDATE TB    ; invalidate for test address 2
U 09, 7700,F800,7E7F,FFFF,8470,180A   00FA  2609 ;% 09   LONLIT_[03000000]               ; SET FORCE MISSES TO TB GROUP
U 0A, 7700,C800,5BE4,03D4,6870,180B   0105  2613 ;% 0A   MEMSCAR_R[LONLIT]               ; 0 AND REPLACEMENT OF GROUP 1.
U 0B, 7700,B800,797F,FFFF,8470,180C   0110  2617 ;% 0B   LONLIT_[0D000000]
U 0C, 7700,4800,5BE4,03D4,6070,180D   011B  2621 ;% 0C   MEMSCR_R[LONLIT]
U 0D, 7700,C800,03A4,03D8,4A70,180E   0126  2625 ;% 0D   VA_Q                           ; WRITE VA=TEST ADDRESS 1 AND
U 0E, 7700,B800,7FFF,FF7F,8470,180F   0131  2629 ;% 0E   LONLIT_[00000100]              ; PTE=00000100 INTO GROUP 1.
U 0F, 7700,481B,5BE4,03D4,5070,1810   013C  2633 ;% 0F   TB_R[LONLIT]
U 10, 5700,4800,0364,0300,4A70,1811   0147  2637 ;% 10   VA_RDM                         ; WRITE VA=TEST ADDRESS 2 AND
```

```
U 11. 7700.B800.7F80.0003.8470.1812  0152  2641 ;% 11   LONLIT_[00FFFFF8]                    ; PTE=00FFFFF8 INTO GROUP 1.
U 12. 7700.481B.5BE4.03D4.5070.1813  015D  2645 ;% 12   TB_R[LONLIT]
U 13. 7700.4800.03A4.03D8.4A70.1814  0168  2649 ;% 13   VA_Q                               ; READ THE PTE AT VA=TEST
U 14. 6701.881F.5924.0200.05F0.1815  0173  2653 ;% 14   RDM_TB                             ; ADDRESS 1.
U 15. 7300.C800.0364.0300.0470.1816  017E  2657 ;% 15   STOP.CLOCK
                                     0189  2661
                          00000100   0189  2662 3$:     .LONG   ^X00000100
                                     018D  2663
                          00FFFFF8   018D  2664 5$:     .LONG   ^X00FFFFF8
                                     0191  2665
                          00000000   0191  2666 10$:    .LONG   ^X00000000
                          80000000   0195  2667         .LONG   ^X80000000
                          00008000   0199  2668         .LONG   ^X00008000
                          00004000   019D  2669         .LONG   ^X00004000
                          00002000   01A1  2670         .LONG   ^X00002000
                          00001000   01A5  2671         .LONG   ^X00001000
                          00000800   01A9  2672         .LONG   ^X00000800
                          00000400   01AD  2673         .LONG   ^X00000400
                          00000200   01B1  2674         .LONG   ^X00000200
                                     01B5  2675
                                     01B5  2676 END_TEST_DATA
                                     01B5
                                     01B5       ;-----------------------------------------------------------------------------
                                     01B5  2677
                                     01B5  2678 ENDTEST
```

```
                        002F        .SBTTL  "TEST   03E     TB GROUP 0 TAG MEMORY DUAL ADDRESSING TEST
                        002F  2680 ;*******************************************************************************
                        002F  2681 ;++
                        002F  2682 ;
                        002F  2683 ;       TB GROUP 0 TAG MEMORY DUAL ADDRESSING TEST
                        002F  2684 ;
                        002F  2685 ; FUNCTIONAL DESCRIPTION
                        002F  2686 ;       THIS IS A TEST OF THE ADDRESSING LINES THAT GO TO THE TB GROUP 0
                        002F  2687 ;       TAG MEMORY.
                        002F  2688 ;
                        002F  2689 ; TEST ALGORITHM
                        002F  2690 ;       1.      SET THE TB GROUP 1 FORCE MISS BIT AND SET THE TB GROUP 0
                        002F  2691 ;               FORCE REPLACE BIT.
                        002F  2692 ;       2.      LOAD THE VA WITH TEST ADDRESS 1.
                        002F  2693 ;       3.      WRITE PTE = 00000100 (JUST THE VALID BIT SET) WITH THE
                        002F  2694 ;               VA INTO TB GROUP 0.
                        002F  2695 ;       4.      LOAD THE VA WITH TEST ADDRESS 2.
                        002F  2696 ;       5.      WRITE PTE = 00000100 (JUST THE VALID BIT SET) WITH THE
                        002F  2697 ;               VA INTO TB GROUP 0.
                        002F  2698 ;       6.      LOAD THE VA WITH TEST ADDRESS 1.
                        002F  2699 ;       7.      READ THE PTE OUT OF TB GROUP 0 AT TEST ADDRESS 1.
                        002F  2700 ;       8.      CHECK THIS PTE, IT SHOULD HAVE JUST THE VALID BIT SET.
                        002F  2701 ;               IT SHOULD BE A HIT IN GROUP 0.
                        002F  2702 ;       9.      REPEAT 1. THRU 8. FOR EVERY COMBINATION OF TEST ADDRESS
                        002F  2703 ;               1 AND TEST ADDRESS 2 TAKEN FROM THE TABLES BELOW.
                        002F  2704 ;
                        002F  2705 ; TEST PATTERNS
                        002F  2706 ;       PICK TEST ADDRESS 1 FROM THIS TABLE:
                        002F  2707 ;       1.      00000000
                        002F  2708 ;       2.      80000000
                        002F  2709 ;       3.      00008000
                        002F  2710 ;       4.      00004000
                        002F  2711 ;       5.      00002000
                        002F  2712 ;       6.      00001000
                        002F  2713 ;       7.      00000800
                        002F  2714 ;       8.      00000400
                        002F  2715 ;       9.      00000200
                        002F  2716 ;       PICK TEST ADDRESS 2 FROM THIS TABLE:
                        002F  2717 ;       1.      7FFF0000
                        002F  2718 ;       2.      FFFF0000
                        002F  2719 ;       3.      7FFF8000
                        002F  2720 ;       4.      7FFF4000
                        002F  2721 ;       5.      7FFF2000
                        002F  2722 ;       6.      7FFF1000
                        002F  2723 ;       7.      7FFF0800
                        002F  2724 ;       8.      7FFF0400
                        002F  2725 ;       9.      7FFF0200
                        002F  2726 ;
                        002F  2727 ; LOGIC DESCRIPTION
                        002F  2728 ;       THIS TEST INVLOVES MOSTLY DISCRETE LOGIC HANGING ON THE MAD BUS.
                        002F  2729 ;       THE ONLY GATE ARRAY THAT MIGHT AFFECT YOU IS ADD.
                        002F  2730 ;
                        002F  2731 ; ASSUMPTIONS
                        002F  2732 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
                        002F  2733 ;
```

```
                     002F  2734 ; ERROR DESCRIPTION
                     002F  2735 ;       EXPECTED PTE
                     002F  2736 ;       GOT PTE
                     002F  2737 ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
                     002F  2738 ;             BEING USED AS TEST ADDRESS 1 WHEN THE ERROR OCCURRED.
                     002F  2739 ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
                     002F  2740 ;             BEING USED AS TEST ADDRESS 2 WHEN THE ERROR OCCURRED.
                     002F  2741 ;--
                     002F  2742 ;*******************************************************************************
                     002F  2743 ;///////////////////////////////////////////////////////////////////////////////
                     002F  2744
                     002F  2745         INITIALIZE
                     0031  2746         LOADDCS         1$,,0,<^X16>                ; LOAD THE TEST UCODE.
                     0038  2747         LOOP            I,1,9                       ; INITIALIZE A LOOP TO PICK
                     003F  2748                                                     ; TEST ADDRESS 1.
                     003F  2749         LDFIELD         10$,I,LONG,2$,31,62,LONLIT ; MAKE TEST ADDRESS 1 PART
                     0050  2750                                                     ; OF THIS UINSTRUCTION'S LONLIT.
                     0050  2751         LOADDCS         2$,,1,1                     ; LOAD THE TEST UCODE.
                     0057  2752         LOOP            J,1,9                       ; INITIALIZE A LOOP TO PICK
                     005E  2753                                                     ; TEST ADDRESS 2.
                     005E  2754         SKIP            100$,ONEQUAL,I,J            ; IF I AND J ARE EQUAL SKIP
                     0065  2755                                                     ; THIS ITERATION.
                     0065  2756         LOADREG         WDREG,20$,J,LONG            ; GET TEST ADDRESS 2.
                     006D  2757         ERRLOOP
                     006F  2758         FETCH           0                           ; EXECUTE THE TEST UCODE.
                     0074  2759         BRSTCLK         <^X15>
                     0078  2760         CMPREGMSK       WHREG,3$,,5$,,LONG          ; CHECK THE RESULT.
                     0084  2761         IFERROR         ,<<ADD>>,                   ; DUAL ADDRESSING PROBLEM
                     008A  2762                                                     ; WITH TB GROUP 0 TAG MEMORY.
                     008A  2763 100$:   ENDLOOP         J                           ; SELECT NEXT TEST ADDRESS 2.
                     008D  2764         ENDLOOP         I                           ; SELECT NEXT TEST ADDRESS 1.
                     0090  2765         SKIP
                     0097  2766
                     0097  2767 BEGIN_TEST_DATA
                     0097
                     0097          ;-------------------------------------------------------------------------
                     0097  2768
                     0097  2769 1$:
U 00, 7700,C800,0364,0300,0470,1801  0097  2770 ;% 00    NOP
                     00A2  2774 ;
                     00A2  2775 ; Micro addresses 01-08 are used to invalidate the TB at both test address 1
                     00A2  2776 ; and test address 2.  The lonlit value that gets loaded in address 01 will
                     00A2  2777 ; be changed for each address test case.
                     00A2  2778 ;
                     00A2  2779 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802  00A2  2780 ;% 01    LONLIT_[00000000]                  ; load test address 1
U 02, 7700,0800,5BE6,03D4,0470,1803  00AD  2784 ;% 02    D_R[LONLIT]                        ; store address in D
U 03, 7700,5800,DB13,0300,4A70,1804  00B8  2788 ;% 03    VA_Q_D_D+ZLIT8[0]                  ; set up VA & Q
U 04, 7700,9800,DB12,0300,5360,1805  00C3  2792 ;% 04    VA_D_D+ZLIT8[0] INVALIDATE TB      ; invalidate for test address 1
U 05, 5700,0840,0364,0300,0470,1806  00CE  2796 ;% 05    R[TEMP0]_RDM                       ; load test address 2
U 06, 7700,8800,5BE6,0300,0470,1807  00D9  2800 ;% 06    D_R[TEMP0]                         ; D <-- test address 2
U 07, 7700,1868,DB10,0300,0470,1808  00E4  2804 ;% 07    M[TEMP8]_D+ZLIT8[0]                ; store in MTEMP8
U 08, 7700,9800,DB12,0300,5360,1809  00EF  2808 ;% 08    VA_D_D+ZLIT8[0] INVALIDATE TB      ; invalidate for test address 2
U 09, 7700,F800,7E7F,FFFF,8470,180A  00FA  2812 ;% 09    LONLIT_[03000000]                  ; SET FORCE MISSES TO TB GROUP
U 0A, 7700,C800,5BE4,03D4,6870,180B  0105  2816 ;% 0A    MEMSCAR_R[LONLIT]                  ; 1 AND REPLACEMENT OF GROUP 0.
```

```
U 0B. 7700.F800.7AFF.FFFF.8470.180C  0110  2820 ;% 0B   LONLIT_[0A000000]
U 0C. 7700.4800.5BE4.03D4.6070.180D  011B  2824 ;% 0C   MEMSCR_R[LONLIT]
U 0D. 7700.C800.03A4.03D8.4A70.180E  0126  2828 ;% 0D   VA_Q                    ; WRITE VA=TEST ADDRESS 1 AND
U 0E. 7700.B800.7FFF.FF7F.8470.180F  0131  2832 ;% 0E   LONLIT_[00000100]      ; PTE=00000100 INTO GROUP 0.
U 0F. 7700.481B.5BE4.03D4.5070.1810  013C  2836 ;% 0F   TB_R[LONLIT]
U 10. 5700.4800.0364.0300.4A70.1811  0147  2840 ;% 10   VA_RDM                  ; WRITE VA=TEST ADDRESS 2 AND
U 11. 7700.B800.7FFF.FF7F.8470.1812  0152  2844 ;% 11   LONLIT_[00000100]      ; PTE=00000100 INTO GROUP 0.
U 12. 7700.481B.5BE4.03D4.5070.1813  015D  2848 ;% 12   TB_R[LONLIT]
U 13. 7700.4800.03A4.03D8.4A70.1814  0168  2852 ;% 13   VA_Q                    ; READ THE PTE AT VA=TEST
U 14. 6701.881F.5924.0200.05F0.1815  0173  2856 ;% 14   RDM_TB                  ; ADDRESS 1.
U 15. 7300.C800.0364.0300.0470.1816  017E  2860 ;% 15   STOP.CLOCK
                                     0189  2864
                         00000100    0189  2865 3$:     .LONG   ^X00000100
                                     018D  2866
                         00FFFFF8    018D  2867 5$:     .LONG   ^X00FFFFF8
                                     0191  2868
                         00000000    0191  2869 10$:    .LONG   ^X00000000
                         80000000    0195  2870         .LONG   ^X80000000
                         00008000    0199  2871         .LONG   ^X00008000
                         00004000    019D  2872         .LONG   ^X00004000
                         00002000    01A1  2873         .LONG   ^X00002000
                         00001000    01A5  2874         .LONG   ^X00001000
                         00000800    01A9  2875         .LONG   ^X00000800
                         00000400    01AD  2876         .LONG   ^X00000400
                         00000200    01B1  2877         .LONG   ^X00000200
                                     01B5  2878
                         7FFF0000    01B5  2879 20$:    .LONG   ^X7FFF0000
                         FFFF0000    01B9  2880         .LONG   ^XFFFF0000
                         7FFF8000    01BD  2881         .LONG   ^X7FFF8000
                         7FFF4000    01C1  2882         .LONG   ^X7FFF4000
                         7FFF2000    01C5  2883         .LONG   ^X7FFF2000
                         7FFF1000    01C9  2884         .LONG   ^X7FFF1000
                         7FFF0800    01CD  2885         .LONG   ^X7FFF0800
                         7FFF0400    01D1  2886         .LONG   ^X7FFF0400
                         7FFF0200    01D5  2887         .LONG   ^X7FFF0200
                                     01D9  2888
                                     01D9  2889 END_TEST_DATA
                                     01D9
                                     01D9       ;-----------------------------------------------------------------------------
                                     01D9  2890
                                     01D9  2891 ENDTEST
```

```
                        002F        .SBTTL  "TEST   03F     TB GROUP 1 TAG MEMORY DUAL ADDRESSING TEST
                        002F   2893 ;*******************************************************************************
                        002F   2894 ;++
                        002F   2895 ;
                        002F   2896 ;       TB GROUP 1 TAG MEMORY DUAL ADDRESSING TEST
                        002F   2897 ;
                        002F   2898 ; FUNCTIONAL DESCRIPTION
                        002F   2899 ;       THIS IS A TEST OF THE ADDRESSING LINES THAT GO TO THE TB GROUP 1
                        002F   2900 ;       TAG MEMORY.
                        002F   2901 ;
                        002F   2902 ; TEST ALGORITHM
                        002F   2903 ;       1.      SET THE TB GROUP 0 FORCE MISS BIT AND SET THE TB GROUP 1
                        002F   2904 ;               FORCE REPLACE BIT.
                        002F   2905 ;       2.      LOAD THE VA WITH TEST ADDRESS 1.
                        002F   2906 ;       3.      WRITE PTE = 00000100 (JUST THE VALID BIT SET) WITH THE
                        002F   2907 ;               VA INTO TB GROUP 1.
                        002F   2908 ;       4.      LOAD THE VA WITH TEST ADDRESS 2.
                        002F   2909 ;       5.      WRITE PTE = 00000100 (JUST THE VALID BIT SET) WITH THE
                        002F   2910 ;               VA INTO TB GROUP 1.
                        002F   2911 ;       6.      LOAD THE VA WITH TEST ADDRESS 1.
                        002F   2912 ;       7.      READ THE PTE OUT OF TB GROUP 1 AT TEST ADDRESS 1.
                        002F   2913 ;       8.      CHECK THIS PTE, IT SHOULD HAVE JUST THE VALID BIT SET.
                        002F   2914 ;               IT SHOULD BE A HIT IN GROUP 1.
                        002F   2915 ;       9.      REPEAT 1. THRU 8. FOR EVERY COMBINATION OF TEST ADDRESS
                        002F   2916 ;               1 AND TEST ADDRESS 2 TAKEN FROM THE TABLES BELOW.
                        002F   2917 ;
                        002F   2918 ; TEST PATTERNS
                        002F   2919 ;       PICK TEST ADDRESS 1 FROM THIS TABLE:
                        002F   2920 ;       1.      00000000
                        002F   2921 ;       2.      80000000
                        002F   2922 ;       3.      00008000
                        002F   2923 ;       4.      00004000
                        002F   2924 ;       5.      00002000
                        002F   2925 ;       6.      00001000
                        002F   2926 ;       7.      00000800
                        002F   2927 ;       8.      00000400
                        002F   2928 ;       9.      00000200
                        002F   2929 ;       PICK TEST ADDRESS 2 FROM THIS TABLE:
                        002F   2930 ;       1.      7FFF0000
                        002F   2931 ;       2.      FFFF0000
                        002F   2932 ;       3.      7FFF8000
                        002F   2933 ;       4.      7FFF4000
                        002F   2934 ;       5.      7FFF2000
                        002F   2935 ;       6.      7FFF1000
                        002F   2936 ;       7.      7FFF0800
                        002F   2937 ;       8.      7FFF0400
                        002F   2938 ;       9.      7FFF0200
                        002F   2939 ;
                        002F   2940 ; LOGIC DESCRIPTION
                        002F   2941 ;       THIS TEST INVOLVES MOSTLY DISCRETE LOGIC HANGING ON THE MAD BUS.
                        002F   2942 ;       THE ONLY GATE ARRAY THAT MIGHT AFFECT YOU IS ADD.
                        002F   2943 ;
                        002F   2944 ; ASSUMPTIONS
                        002F   2945 ;       THIS TEST ASSUMES THE WBUS AND DPM ARE OPERATIONAL
                        002F   2946 ;
```

```
                        002F  2947 ; ERROR DESCRIPTION
                        002F  2948 ;       EXPECTED PTE
                        002F  2949 ;       GOT PTE
                        002F  2950 ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
                        002F  2951 ;             BEING USED AS TEST ADDRESS 1 WHEN THE ERROR OCCURRED.
                        002F  2952 ;       INDEX INDICATING WHICH ADDRESS FROM THE ABOVE TABLE WAS
                        002F  2953 ;             BEING USED AS TEST ADDRESS 2 WHEN THE ERROR OCCURRED.
                        002F  2954 ;--
                        002F  2955 ;*******************************************************************************
                        002F  2956 ;///////////////////////////////////////////////////////////////////////////////
                        002F  2957
                        002F  2958         INITIALIZE
                        0031  2959         LOADDCS         1$,,0,<^X16>                ; LOAD THE TEST UCODE.
                        0038  2960         LOOP            I,1,9                       ; INITIALIZE A LOOP TO PICK
                        003F  2961                                                     ; TEST ADDRESS 1.
                        003F  2962         LDFIELD         10$,I,LONG,2$,31,62,LONLIT ; MAKE TEST ADDRESS 1 PART
                        0050  2963                                                     ; OF THIS UINSTRUCTION'S LONLIT.
                        0050  2964         LOADDCS         2$,,1,1                     ; LOAD THE TEST UCODE.
                        0057  2965         LOOP            J,1,9                       ; INITIALIZE A LOOP TO PICK
                        005E  2966                                                     ; TEST ADDRESS 2.
                        005E  2967         SKIP            100$,ONEQUAL,I,J            ; IF I AND J ARE EQUAL SKIP
                        0065  2968                                                     ; THIS ITERATION.
                        0065  2969         LOADREG         WDREG,20$,J,LONG            ; GET TEST ADDRESS 2.
                        006D  2970         ERRLOOP
                        006F  2971         FETCH           0                           ; EXECUTE THE TEST UCODE.
                        0074  2972         BRSTCLK         <^X15>
                        0078  2973         CMPREGMSK       WHREG,3$,,5$,,LONG          ; CHECK THE RESULT.
                        0084  2974         IFERROR         ,<<ADD>>,                   ; DUAL ADDRESSING PROBLEM
                        008A  2975                                                     ; WITH TB GROUP 1 TAG MEMORY.
                        008A  2976 100$:   ENDLOOP         J                           ; SELECT NEXT TEST ADDRESS 2.
                        008D  2977         ENDLOOP         I                           ; SELECT NEXT TEST ADDRESS 1.
                        0090  2978         SKIP
                        0097  2979
                        0097  2980 BEGIN_TEST_DATA
                        0097
                        0097         ;-------------------------------------------------------------------------
                        0097  2981
                        0097  2982 1$:
U 00, 7700,C800,0364,0300,0470,1801     0097  2983 ;% 00    NOP
                        00A2  2987 ;
                        00A2  2988 ; Micro addresses 01-08 are used to invalidate the TB at both test address 1
                        00A2  2989 ; and test address 2.  The lonlit value that gets loaded in address 01 will
                        00A2  2990 ; be changed for each address test case.
                        00A2  2991 ;
                        00A2  2992 2$:
U 01, 7700,7800,7FFF,FFFF,8470,1802     00A2  2993 ;% 01    LONLIT_[00000000]                   ; load test address 1
U 02, 7700,0800,5BE6,03D4,0470,1803     00AD  2997 ;% 02    D_R[LONLIT]                         ; store address in D
U 03, 7700,5800,DB13,0300,4A70,1804     00B8  3001 ;% 03    VA_Q_D_D+ZLIT8[0]                   ; set up VA & Q
U 04, 7700,9800,DB12,0300,5360,1805     00C3  3005 ;% 04    VA_D_D+ZLIT8[0] INVALIDATE TB       ; invalidate for test address 1
U 05, 5700,0840,0364,0300,0470,1806     00CE  3009 ;% 05    R[TEMP0]_RDM                        ; load test address 2
U 06, 7700,8800,5BE6,0300,0470,1807     00D9  3013 ;% 06    D_R[TEMP0]                          ; D <-- test address 2
U 07, 7700,1868,DB10,0300,0470,1808     00E4  3017 ;% 07    M[TEMP8]_D+ZLIT8[0]                 ; store in MTEMP8
U 08, 7700,9800,DB12,0300,5360,1809     00EF  3021 ;% 08    VA_D_D+ZLIT8[0] INVALIDATE TB       ; invalidate for test address 2
U 09, 7700,F800,7E7F,FFFF,8470,180A     00FA  3025 ;% 09    LONLIT_[03000000]                   ; SET FORCE MISSES TO TB GROUP
U 0A, 7700,C800,5BE4,03D4,6870,180B     0105  3029 ;% 0A    MEMSCAR_R[LONLIT]                   ; 0 AND REPLACEMENT OF GROUP 1.
```

```
U 0B. 7700.F800.7AFF.FFFF.8470.180C  0110  3033 ;* 0B   LONLIT_[0A000000]
U 0C. 7700.4800.5BE4.03D4.6070.180D  011B  3037 ;* 0C   MEMSCR_R[LONLIT]
U 0D. 7700.C800.03A4.03D8.4A70.180E  0126  3041 ;* 0D   VA_Q                    ; WRITE VA=TEST ADDRESS 1 AND
U 0E. 7700.B800.7FFF.FF7F.8470.180F  0131  3045 ;* 0E   LONLIT_[00000100]      ; PTE=00000100 INTO GROUP 1.
U 0F. 7700.481B.5BE4.03D4.5070.1810  013C  3049 ;* 0F   TB_R[LONLIT]
U 10. 5700.4800.0364.0300.4A70.1811  0147  3053 ;* 10   VA_RDM                  ; WRITE VA=TEST ADDRESS 2 AND
U 11. 7700.B800.7FFF.FF7F.8470.1812  0152  3057 ;* 11   LONLIT_[00000100]      ; PTE=00000100 INTO GROUP 1.
U 12. 7700.481B.5BE4.03D4.5070.1813  015D  3061 ;* 12   TB_R[LONLIT]
U 13. 7700.4800.03A4.03D8.4A70.1814  0168  3065 ;* 13   VA_Q                    ; READ THE PTE AT VA=TEST
U 14. 6701.881F.5924.0200.05F0.1815  0173  3069 ;* 14   RDM_TB                  ; ADDRESS 1.
U 15. 7300.C800.0364.0300.0470.1816  017E  3073 ;* 15   STOP.CLOCK
                                     0189  3077
                          00000100   0189  3078 3$:     .LONG   ^X00000100
                                     018D  3079
                          00FFFFF8   018D  3080 5$:     .LONG   ^X00FFFFF8
                                     0191  3081
                          00000000   0191  3082 10$:    .LONG   ^X00000000
                          80000000   0195  3083         .LONG   ^X80000000
                          00008000   0199  3084         .LONG   ^X00008000
                          00004000   019D  3085         .LONG   ^X00004000
                          00002000   01A1  3086         .LONG   ^X00002000
                          00001000   01A5  3087         .LONG   ^X00001000
                          00000800   01A9  3088         .LONG   ^X00000800
                          00000400   01AD  3089         .LONG   ^X00000400
                          00000200   01B1  3090         .LONG   ^X00000200
                                     01B5  3091
                          7FFF0000   01B5  3092 20$:    .LONG   ^X7FFF0000
                          FFFF0000   01B9  3093         .LONG   ^XFFFF0000
                          7FFF8000   01BD  3094         .LONG   ^X7FFF8000
                          7FFF4000   01C1  3095         .LONG   ^X7FFF4000
                          7FFF2000   01C5  3096         .LONG   ^X7FFF2000
                          7FFF1000   01C9  3097         .LONG   ^X7FFF1000
                          7FFF0800   01CD  3098         .LONG   ^X7FFF0800
                          7FFF0400   01D1  3099         .LONG   ^X7FFF0400
                          7FFF0200   01D5  3100         .LONG   ^X7FFF0200
                                     01D9  3101
                                     01D9  3102 END_TEST_DATA
                                     01D9
                                     01D9       ;-----------------------------------------------------------------------------
                                     01D9  3103
                                     01D9  3104 ENDTEST
```

```
                        0025        .SBTTL  "TEST   040     TB GROUP 0 PTE MEMORY MARCH TEST
                        0025  3106 ;*******************************************************************************
                        0025  3107 ;++
                        0025  3108 ;
                        0025  3109 ;       TB GROUP 0 PTE MEMORY MARCH TEST
                        0025  3110 ;
                        0025  3111 ; FUNCTIONAL DESCRIPTION
                        0025  3112 ;       THIS IS A DYNAMIC TEST OF TB GROUP 0'S DATA STORE. THIS
                        0025  3113 ;       WILL TEST ADDRESSING AND DATA INTEGRITY INSIDE THE TB
                        0025  3114 ;       DATA STORE RAM CHIPS.
                        0025  3115 ;
                        0025  3116 ; TEST ALGORITHM
                        0025  3117 ;       1.  CREATE A LOOP OF 256 TO VALIDATE ALL GROUP 0 LOCATIONS
                        0025  3118 ;       2.  PUT INDEX VALUE I INTO RDM WD REGISTER
                        0025  3119 ;       3.  EXECUTE DCS PROGRAM
                        0025  3120 ;               a.  SET MISS GROUP 1 AND REPLACE GROUP 0
                        0025  3121 ;               b.  MOVE INDEX VALUE TO TEMP1
                        0025  3122 ;               c.  ROTATE INDEX VALUE LEFT BY 9 TO FORM TAG FIELD
                        0025  3123 ;               d.  MASK OUT OTHER BITS AND STORE TAG IN TEMP 0
                        0025  3124 ;               e.  ROTATE INDEX VALUE LEFT BY 18 TO FORM BIT 31 OF TAG
                        0025  3125 ;               f.  MASK OUT OTHER BITS AND STORE IN Q REGISTER
                        0025  3126 ;               g.  "OR" TEMP0 AND Q TO FORM FINAL TAG ADDRESS
                        0025  3127 ;               h.  MOVE TAG ADDRESS TO VA REGISTER AND TEMP0
                        0025  3128 ;               i.  WRITE TB WITH 00000100
                        0025  3129 ;       4.  GO TO STEP 2 UNTIL TB IS COMPLETELY WRITTEN
                        0025  3130 ;       5.  LOAD U-CODE TO READ AND WRITE TB
                        0025  3131 ;       6.  CREATE A LOOP OF 9 TO SELECT TEST PATTERNS (I)
                        0025  3132 ;       7.  CREATE A SECOND LOOP OF 256 TO ADDRESS TB WITH (J)
                        0025  3133 ;       8.  MOVE INDEX (J) TO RDM WD REGISTER
                        0025  3134 ;       9.  EXECUTE DCS PROGRAM
                        0025  3135 ;               a.  FORM TAG FIELD AS IN LAST SET OF U-CODE
                        0025  3136 ;               b.  READ TB AT THAT LOCATION AND MOVE TO RDM WH REGISTER
                        0025  3137 ;      10.  TEST WH REGISTER FOR CORRECT DATA
                        0025  3138 ;      11.  IF NO ERROR LOAD RDM WD REGISTER WITH NEXT TEST PATTERN
                        0025  3139 ;      12.  EXECUTE DCS PROGRAM
                        0025  3140 ;               a.  WRITE TEST PATTERN TO TB
                        0025  3141 ;      13.  GO TO STEP 7 UNTIL ALL TB LOCATIONS ARE READ AND WRITTEN
                        0025  3142 ;      14.  GO TO STEP 6 UNTIL ALL TEST PATTERNS ARE USED
                        0025  3143 ;      15.  REPEAT STEPS 6 THROUGH 14 WITH LOOP J DECENDING
                        0025  3144 ;
                        0025  3145 ; TEST PATTERNS
                        0025  3146 ;       NOTE THAT THESE DATA PATTERNS ARE USED AS TB PTE'S WHICH ARE
                        0025  3147 ;       IN THE INTERNAL FORMAT FOR CPU PTE'S. NOTE ALSO THAT ALL
                        0025  3148 ;       OF THEM HAVE THE VALID BIT SET.
                        0025  3149 ;       0.      00000100
                        C025  3150 ;       1.      00FFFFF8
                        0025  3151 ;       2.      00111188
                        0025  3152 ;       3.      00EEEF70
                        0025  3153 ;       4.      00333398
                        0025  3154 ;       5.      00CCCD60
                        0025  3155 ;       6.      007777B8
                        0025  3156 ;       7.      00888940
                        0025  3157 ;       8.      00FFFFF8
                        0025  3158 ;       9.      00000100
                        0025  3159 ;
```

```
                     ; LOGIC DESCRIPTION
                     ;       TESTED HERE ARE ALL THE RAM CHIPS IN TB GROUP 0'S DATA
                     ;       STORE MEMORY, INCLUDING THE CHIP WHICH STORES THE PARITY
                     ;       BITS. NOTE THAT IF THIS TEST FAILS IT IS ONE OF THESE RAM
                     ;       CHIPS WHICH IS MOST LIKELY AT FAULT.
                     ;
                     ; ASSUMPTIONS
                     ;       ALL LOGIC USED TO PERFORM THIS TEST WHICH IS EXTERNAL TO
                     ;       TB GROUP 0'S DATA STORE RAM CHIPS IS ASSUMED TO HAVE BEEN
                     ;       PREVIOUSLY TESTED. THIS IMPLIES THAT IF A FAILURE OCCURS WHILE THIS
                     ;       TEST IS RUNNING IT IS ONE OF THESE RAM CHIPS WHICH IS AT FAULT.
                     ;
                     ; ERROR DESCRIPTION
                     ;       EXPECTED PTE (IN CPU'S INTERNAL FORMAT)
                     ;       GOT PTE
                     ;       INDEX I (POINTS AT TEST PATTERN BEING USED)
                     ;       INDEX J (USED TO FORM TB ADDRESS)
                     ;       RTEMP0 (CONTAINS ADDRESS BEING READ AT TIME OF FAILURE)
                     ;--
                     ;*****************************************************************************
                     ;/////////////////////////////////////////////////////////////////////////////

                             INITIALIZE
0027 3183            LOADDCS         1$,,0,<^X10>            ; LOAD UCODE TO WRITE
002E 3184                                                    ; PTE=00000100 INTO ALL
002E 3185                                                    ; GROUP 0 LOCATIONS.
002E 3186            LOOP            I,1,256                 ; THIS LOOP WRITES VA=00000000
0035 3187            SAVEINDEX       I,2$                    ; PTE=00000100 INTO EVERY
003A 3188            LOADREG         WDREG,2$,,L             ; TB GROUP 0 MEMORY LOCATION.
0042 3189            FETCH           0                       ; START DCS PROGRAM
0047 3190            BRSTCLK         <^X0F>                  ; END DCS PROGRAM
004B 3191            ENDLOOP         I                       ; CONTINUE UNTIL ALL 256
004E 3192                                                    ; LOCATIONS HAVE BEEN WRITTEN.
004E 3193            LOADDCS         3$,,0,<^X10>            ; LOAD U-CODE TO READ AND WRITE
0055 3194                                                    ; TB
0055 3195            LOOP            I,1,<^X9>               ; INITIALIZE A LOOP THRU
005C 3196                                                    ; THE TEST PATTERNS FOR PTE'S.
005C 3197            LOOP            J,1,256                 ; INITIALIZE A LOOP THRU EVERY
0063 3198                                                    ; TB INDEX LOCATION IN THE
0063 3199                                                    ; FORWARD DIRECTION.
0063 3200            SAVEINDEX       J,2$                    ; LOAD THE TB INDEX VALUE.
0068 3201            LOADREG         WDREG,2$,,L
0070 3202            ERRLOOP                                 ; ERROR LOOP POINT
0072 3203            FETCH           0                       ; EXECUTE THE TEST UCODE
0077 3204            BRSTCLK         <^X0C>
007B 3205            CMPREGMSK       WHREG,4$,I,6$,,L        ; SEE IF THE PTE READ IS CORRECT.
0087 3206            IFERROR         1,,<<MIC>>              ; THE PTE RETURNED WAS INCORRECT.
008D 3207            LOADREG         WDREG,5$,I,L            ; NEW WRITE TEST PATTERN
0095 3208            FETCH           <^X0D>                  ; START DCS PROGRAM
009A 3209            BRSTCLK         <^X0F>                  ; END DCS PROGRAM
009E 3210            ENDLOOP         J                       ; LOOP THRU ALL INDEX LOCATIONS.
00A1 3211            ENDLOOP         I                       ; LOOP THRU ALL PTE TEST PATTERNS.
00A4 3212            LOOP            I,1,<^X9>               ; INITIALIZE A LOOP THRU
00AB 3213                                                    ; THE TEST PATTERNS FOR PTE'S.
00AB 3214            LOOP            J,256,1                 ; INITIALIZE A LOOP THRU EVERY
```

```
                     00B2  3215                                                ; TB INDEX LOCATION IN THE
                     00B2  3216                                                ; BACKWARD DIRECTION.
                     00B2  3217          SAVEINDEX       J,2$                  ; LOAD THE TB INDEX VALUE.
                     00B7  3218          LOADREG         WDREG,2$,,L
                     00BF  3219          ERRLOOP                               ; ERROR LOOP POINT
                     00C1  3220          FETCH           0                     ; EXECUTE THE TEST UCODE.
                     00C6  3221          BRSTCLK         <^X0C>
                     00CA  3222          CMPREGMSK       WHREG,4$,I,6$,,L      ; SEE IF THE PTE READ IS CORRECT.
                     00D6  3223          IFERROR         1,,<<MIC>>            ; THE PTE RETURNED WAS INCORRECT.
                     00DC  3224          LOADREG         WDREG,5$,I,L          ; NEW WRITE TEST PATTERN
                     00E4  3225          FETCH           <^X0D>                ; START DCS PROGRAM
                     00E9  3226          BRSTCLK         <^X0F>                ; END DCS PROGRAM
                     00ED  3227          ENDLOOP         J                     ; LOOP THRU ALL INDEX LOCATIONS.
                     00F0  3228          ENDLOOP         I                     ; LOOP THRU ALL PTE TEST PATTERNS.
                     00F3  3229          SKIP                                  ; GO TO NEXT TEST
                     00FA  3230
                     00FA  3231 BEGIN_TEST_DATA
                     00FA
                     00FA        ;-----------------------------------------------------------------------
                     00FA  3232
                     00FA  3233 1$:
U 00, 7700,C800,0364,0300,0470,1801  00FA  3234 ;* 00    NOP
U 01, 7700,8800,5BE4,0308,6870,1802  0105  3238 ;* 01    MEMSCAR_R[TEMP2]                  ; SET MISS GROUP 1 AND
U 02, 7700,C800,5BE4,030C,6070,1803  0110  3242 ;* 02    MEMSCR_R[TEMP3]                   ; REPLACE GROUP 0
U 03, 5700,C840,0364,0304,0470,1804  011B  3246 ;* 03    R[TEMP1]_RDM                      ; LOOP COUNT TO TEMP1
U 04, 7700,4801,4771,0300,0470,1805  0126  3250 ;* 04    Q_M[TEMP1].RL.PTE                 ; ROTATE LOOP COUNT LEFT 9
U 05, 7700,B800,7FFF,80FF,8470,1806  0131  3254 ;* 05    LONLIT_[0000FE00]                 ; MASK ROTATED LOOP COUNT
U 06, 7700,C800,03A1,03D4,0470,1807  013C  3258 ;* 06    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 07, 7700,C860,03A4,03D8,0470,1808  0147  3262 ;* 07    M[TEMP0]_Q                        ; MOVE Q TO TEMP0
U 08, 7700,9800,EF64,030C,0470,1809  0152  3266 ;* 08    PL_[00000018]                     ; SET PLATCH TO 18
U 09, 7700,4800,8B71,0304,0470,180A  015D  3270 ;* 09    Q_R[TEMP1].RL.P                   ; ROTATE LOOP COUNT LEFT 18
U 0A, 7700,3800,3FFF,FFFF,8470,180B  0168  3274 ;* 0A    LONLIT_[80000000]                 ; MASK ROTATED LOOP COUNT
U 0B, 7700,4800,03A1,03D4,0470,180C  0173  3278 ;* 0B    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 0C, 7700,8840,03A4,0300,4A70,180D  017E  3282 ;* 0C    VA_Q.OR.R[TEMP0],SPW/RLONG        ; CREATE TB TEST ADDRESS
U 0D, 7700,3800,7FFF,FF7F,8470,180E  0189  3286 ;* 0D    LONLIT_[00000100]                 ; WRITE DATA
U 0E, 7700,C81B,5BE4,03D4,5070,180F  0194  3290 ;* 0E    TB_R[LONLIT]                      ; WRITE TB ACCORDING TO VA
U 0F, 7300,C800,0364,0300,0470,1810  019F  3294 ;* 0F    NOP,STOP.CLOCK                    ; STOP CPU CLOCK
                     01AA  3298
                00000000  01AA  3299 2$: .LONG       ^X00000000                    ; TEMPORARY STORAGE
                     01AE  3300
                     01AE  3301 3$:
U 00, 7700,C800,0364,0300,0470,1801  01AE  3302 ;* 00    NOP
U 01, 5700,C840,0364,0304,0470,1802  01B9  3306 ;* 01    R[TEMP1]_RDM                      ; MOVE LOOP COUNT TO TEMP1
U 02, 7700,4801,4771,0300,0470,1803  01C4  3310 ;* 02    Q_M[TEMP1].RL.PTE                 ; ROTATE LOOP COUNT LEFT 9
U 03, 7700,3800,7FFF,80FF,8470,1804  01CF  3314 ;* 03    LONLIT_[0000FE00]                 ; MASK ROTATED LOOP COUNT
U 04, 7700,4800,03A1,03D4,0470,1805  01DA  3318 ;* 04    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 05, 7700,4860,03A4,03D8,0470,1806  01E5  3322 ;* 05    M[TEMP0]_Q                        ; MOVE Q TO TEMP0
U 06, 7700,1800,EF64,030C,0470,1807  01F0  3326 ;* 06    PL_[00000018]                     ; PLATCH GETS 18
U 07, 7700,C800,8B71,0304,0470,1808  01FB  3330 ;* 07    Q_R[TEMP1].RL.P                   ; ROTATE LOOP COUNT LEFT 18
U 08, 7700,B800,3FFF,FFFF,8470,1809  0206  3334 ;* 08    LONLIT_[80000000]                 ; MASK ROTATED LOOP COUNT
U 09, 7700,4800,03A1,03D4,0470,180A  0211  3338 ;* 09    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 0A, 7700,8840,03A4,0300,4A70,180B  021C  3342 ;* 0A    VA_Q.OR.R[TEMP0],SPW/RLONG        ; LOAD TEST ADDRESS INTO VA
U 0B, 6701,081F,5924,0200,05F0,180C  0227  3346 ;* 0B    RDM_TB                            ; READ TB TO RDM WH REGISTER
U 0C, 7300,C800,0364,0300,0470,180D  0232  3350 ;* 0C    NOP,STOP.CLOCK                    ; STOP CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E  023D  3354 ;* 0D    NOP
```

```
U 0E, 5700,C81B,0364,0300,5070,180F  0248  3358 ;% 0E   TB_RDM                          ; WRITE NEXT TEST PATTERN
U 0F, 7300,C800,0364,0300,0470,1810  0253  3362 ;% 0F   NOP,STOP.CLOCK                  ; STOP CPU CLOCK
                                     025E  3366
                          00000100   025E  3367 4$:     .LONG   ^X00000100             ; TEST PATTERN TABLE
                                     0262  3368
                          00FFFFF8   0262  3369 5$:     .LONG   ^X00FFFFF8
                          00111188   0266  3370         .LONG   ^X00111188
                          00EEEF70   026A  3371         .LONG   ^X00EEEF70
                          00333398   026E  3372         .LONG   ^X00333398
                          00CCCD60   0272  3373         .LONG   ^X00CCCD60
                          007777B8   0276  3374         .LONG   ^X007777B8
                          00888940   027A  3375         .LONG   ^X00888940
                          00FFFFF8   027E  3376         .LONG   ^X00FFFFF8
                          00000100   0282  3377         .LONG   ^X00000100
                                     0286  3378
                          00FFFFF8   0286  3379 6$:     .LONG   ^X00FFFFF8             ; MASK FOR CMPREGMSK PSEUDO OP
                                     028A  3380
                                     028A  3381 BEGIN_CPU_DATA
                                     0002  3382
                                     0002  3383 RTEMP_DATA
                                     0001  3384
                          00000000   0001  3385         .LONG   ^X00000000             ; TEMPORARY STORAGE/RTEMP0
                          00000000   0005  3386         .LONG   ^X00000000             ; TEMPORARY STORAGE/RTEMP1
                          03000000   0009  3387         .LONG   ^X03000000             ; TB GROUP DISABLE REGISTER ADD
                          0A000000   000D  3388         .LONG   ^X0A000000             ; DISABLE GROUP 1
                                     0011  3389
                                     0011  3390 END_CPU_DATA
                                     028A  3391
                                     028A  3392 END_TEST_DATA
                                     028A
                                     028A       ;-----------------------------------------------------------------------
                                     028A  3393
                                     028A  3394 ENDTEST
```

```
                        0025        .SBTTL  "TEST   041     TB GROUP 1 PTE MEMORY MARCH TEST
                        0025  3396 ;*******************************************************************************
                        0025  3397 ;++
                        0025  3398 ;
                        0025  3399 ;       TB GROUP 1 PTE MEMORY MARCH TEST
                        0025  3400 ;
                        0025  3401 ; FUNCTIONAL DESCRIPTION
                        0025  3402 ;       THIS IS A DYNAMIC TEST OF TB GROUP 1'S DATA STORE. THIS
                        0025  3403 ;       WILL TEST ADDRESSING AND DATA INTEGRITY INSIDE THE TB
                        0025  3404 ;       DATA STORE RAM CHIPS.
                        0025  3405 ;
                        0025  3406 ; TEST ALGORITHM
                        0025  3407 ;       1.  CREATE A LOOP OF 256 TO VALIDATE ALL GROUP 1 LOCATIONS
                        0025  3408 ;       2.  PUT INDEX VALUE I INTO RDM WD REGISTER
                        0025  3409 ;       3.  EXECUTE DCS PROGRAM
                        0025  3410 ;               a.  SET MISS GROUP 0 AND REPLACE GROUP 1
                        0025  3411 ;               b.  MOVE INDEX VALUE TO TEMP1
                        0025  3412 ;               c.  ROTATE INDEX VALUE LEFT BY 9 TO FORM TAG FIELD
                        0025  3413 ;               d.  MASK OUT OTHER BITS AND STORE TAG IN TEMP 0
                        0025  3414 ;               e.  ROTATE INDEX VALUE LEFT BY 18 TO FORM BIT 31 OF TAG
                        0025  3415 ;               f.  MASK OUT OTHER BITS AND STORE IN Q REGISTER
                        0025  3416 ;               g.  "OR" TEMP0 AND Q TO FORM FINAL TAG ADDRESS
                        0025  3417 ;               h.  MOVE TAG ADDRESS TO VA REGISTER AND TEMP0
                        0025  3418 ;               i.  WRITE TB WITH 00000100
                        0025  3419 ;       4.  GO TO STEP 2 UNTIL TB IS COMPLETELY WRITTEN
                        0025  3420 ;       5.  LOAD U-CODE TO READ AND WRITE TB
                        0025  3421 ;       6.  CREATE A LOOP OF 9 TO SELECT TEST PATTERNS (I)
                        0025  3422 ;       7.  CREATE A SECOND LOOP OF 256 TO ADDRESS TB WITH (J)
                        0025  3423 ;       8.  MOVE INDEX (J) TO RDM WD REGISTER
                        0025  3424 ;       9.  EXECUTE DCS PROGRAM
                        0025  3425 ;               a.  FORM TAG FIELD AS IN LAST SET OF U-CODE
                        0025  3426 ;               b.  READ TB AT THAT LOCATION AND MOVE TO RDM WH REGISTER
                        0025  3427 ;      10.  TEST WH REGISTER FOR CORRECT DATA
                        0025  3428 ;      11.  IF NO ERROR LOAD RDM WD REGISTER WITH NEXT TEST PATTERN
                        0025  3429 ;      12.  EXECUTE DCS PROGRAM
                        0025  3430 ;               a.  WRITE TEST PATTERN TO TB
                        0025  3431 ;      13.  GO TO STEP 7 UNTIL ALL TB LOCATIONS ARE READ AND WRITTEN
                        0025  3432 ;      14.  GO TO STEP 6 UNTIL ALL TEST PATTERNS ARE USED
                        0025  3433 ;      15.  REPEAT STEPS 6 THROUGH 14 WITH LOOP J DECENDING
                        0025  3434 ;
                        0025  3435 ; TEST PATTERNS
                        0025  3436 ;       NOTE THAT THESE DATA PATTERNS ARE USED AS TB PTE'S WHICH ARE
                        0025  3437 ;       IN THE INTERNAL FORMAT FOR CPU PTE'S. NOTE ALSO THAT ALL
                        0025  3438 ;       OF THEM HAVE THE VALID BIT SET.
                        0025  3439 ;       0.      00000100
                        0025  3440 ;       1.      00FFFFF8
                        0025  3441 ;       2.      00111188
                        0025  3442 ;       3.      00EEEF70
                        0025  3443 ;       4.      00333398
                        0025  3444 ;       5.      00CCCD60
                        0025  3445 ;       6.      007777B8
                        0025  3446 ;       7.      00888940
                        0025  3447 ;       8.      00FFFFF8
                        0025  3448 ;       9.      00000100
                        0025  3449 ;
```

```
                     ; LOGIC DESCRIPTION
                     ;       TESTED HERE ARE ALL THE RAM CHIPS IN TB GROUP 1'S DATA
                     ;       STORE MEMORY, INCLUDING THE CHIP WHICH STORES THE PARITY
                     ;       BITS. NOTE THAT IF THIS TEST FAILS IT IS ONE OF THESE RAM
                     ;       CHIPS WHICH IS MOST LIKELY AT FAULT.
                     ;
                     ; ASSUMPTIONS
                     ;       ALL LOGIC USED TO PERFORM THIS TEST WHICH IS EXTERNAL TO
                     ;       TB GROUP 1'S DATA STORE RAM CHIPS IS ASSUMED TO HAVE BEEN
                     ;       PREVIOUSLY TESTED. THIS IMPLIES THAT IF A FAILURE OCCURS WHILE THIS
                     ;       TEST IS RUNNING IT IS ONE OF THESE RAM CHIPS WHICH IS AT FAULT.
                     ;
                     ; ERROR DESCRIPTION
                     ;       EXPECTED PTE (IN CPU'S INTERNAL FORMAT)
                     ;       GOT PTE
                     ;       INDEX I (POINTS AT TEST PATTERN BEING USED)
                     ;       INDEX J (USED TO FORM TB ADDRESS)
                     ;       RTEMP0 (CONTAINS ADDRESS BEING READ AT TIME OF FAILURE)
                     ;--
                     ;*****************************************************************************
                     ;/////////////////////////////////////////////////////////////////////////////

                             INITIALIZE
                             LOADDCS         1$,,0,<^X10>            ; LOAD UCODE TO WRITE
                                                                     ; PTE=00000100 INTO ALL
                                                                     ; GROUP 1 LOCATIONS.
                     LOOP    LOOP            I,1,256                 ; THIS LOOP WRITES VA=00000000
                     SAVEINDEX       SAVEINDEX       I,2$            ; PTE=00000100 INTO EVERY
                     LOADREG         LOADREG         WDREG,2$,,L     ; TB GROUP 1 MEMORY LOCATION.
                             FETCH           0                       ; START DCS PROGRAM
                             BRSTCLK         <^X0F>                  ; END DCS PROGRAM
                             ENDLOOP         I                       ; CONTINUE UNTIL ALL 256
                                                                     ; LOCATIONS HAVE BEEN WRITTEN.
                             LOADDCS         3$,,0,<^X10>            ; LOAD U-CODE TO READ AND WRITE
                                                                     ; TB
                             LOOP            I,1,<^X9>               ; INITIALIZE A LOOP THRU
                                                                     ; THE TEST PATTERNS FOR PTE'S.
                             LOOP            J,1,256                 ; INITIALIZE A LOOP THRU EVERY
                                                                     ; TB INDEX LOCATION IN THE
                                                                     ; FORWARD DIRECTION.
                             SAVEINDEX       J,2$                    ; LOAD THE TB INDEX VALUE.
                             LOADREG         WDREG,2$,,L
                             ERRLOOP                                 ; ERROR LOOP POINT
                             FETCH           0                       ; EXECUTE THE TEST UCODE
                             BRSTCLK         <^X0C>
                             CMPREGMSK       WHREG,4$,I,6$,,L        ; SEE IF THE PTE READ IS CORRECT.
                             IFERROR         1,,<<MIC>>              ; THE PTE RETURNED WAS INCORRECT.
                             LOADREG         WDREG,5$,I,L            ; NEW WRITE TEST PATTERN
                             FETCH           <^X0D>                  ; START DCS PROGRAM
                             BRSTCLK         <^X0F>                  ; END DCS PROGRAM
                             ENDLOOP         J                       ; LOOP THRU ALL INDEX LOCATIONS.
                             ENDLOOP         I                       ; LOOP THRU ALL PTE TEST PATTERNS.
                             LOOP            I,1,<^X9>               ; INITIALIZE A LOOP THRU
                                                                     ; THE TEST PATTERNS FOR PTE'S.
                             LOOP            J,256,1                 ; INITIALIZE A LOOP THRU EVERY
```

```
                        00B2  3505                                                    ; TB INDEX LOCATION IN THE
                        00B2  3506                                                    ; BACKWARD DIRECTION.
                        00B2  3507          SAVEINDEX       J,2$                      ; LOAD THE TB INDEX VALUE.
                        00B7  3508          LOADREG         WDREG,2$,,L
                        00BF  3509          ERRLOOP                                   ; ERROR LOOP POINT
                        00C1  3510          FETCH           0                         ; EXECUTE THE TEST UCODE.
                        00C6  3511          BRSTCLK         <^X0C>
                        00CA  3512          CMPREGMSK       WHREG,4$,I,6$,,L          ; SEE IF THE PTE READ IS CORRECT.
                        00D6  3513          IFERROR         1,,<<MIC>>                ; THE PTE RETURNED WAS INCORRECT.
                        00DC  3514          LOADREG         WDREG,5$,I,L              ; NEW WRITE TEST PATTERN
                        00E4  3515          FETCH           <^X0D>                    ; START DCS PROGRAM
                        00E9  3516          BRSTCLK         <^X0F>                    ; END DCS PROGRAM
                        00ED  3517          ENDLOOP         J                         ; LOOP THRU ALL INDEX LOCATIONS.
                        00F0  3518          ENDLOOP         I                         ; LOOP THRU ALL PTE TEST PATTERNS.
                        00F3  3519          SKIP                                      ; GO TO NEXT TEST
                        00FA  3520
                        00FA  3521 BEGIN_TEST_DATA
                        00FA
                        00FA        ;-------------------------------------------------------------------------
                        00FA  3522
                        00FA  3523 1$:
U 00, 7700,C800,0364,0300,0470,1801     00FA  3524 ;% 00    NOP
U 01, 7700,8800,5BE4,0308,6870,1802     0105  3528 ;% 01    MEMSCAR_R[TEMP2]                  ; SET MISS GROUP 0 AND
U 02, 7700,C800,5BE4,030C,6070,1803     0110  3532 ;% 02    MEMSCR_R[TEMP3]                   ; REPLACE GROUP 1
U 03, 5700,C840,0364,0304,0470,1804     011B  3536 ;% 03    R[TEMP1]_RDM                      ; LOOP COUNT TO TEMP1
U 04, 7700,4801,4771,0300,0470,1805     0126  3540 ;% 04    Q_M[TEMP1].RL.PTE                 ; ROTATE LOOP COUNT LEFT 9
U 05, 7700,B800,7FFF,80FF,8470,1806     0131  3544 ;% 05    LONLIT_[0000FE00]                 ; MASK ROTATED LOOP COUNT
U 06, 7700,C800,03A1,03D4,0470,1807     013C  3548 ;% 06    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 07, 7700,C860,03A4,03D8,0470,1808     0147  3552 ;% 07    M[TEMP0]_Q                        ; MOVE Q TO TEMP0
U 08, 7700,9800,EF64,030C,0470,1809     0152  3556 ;% 08    PL_[00000018]                     ; SET PLATCH TO 18
U 09, 7700,4800,8B71,0304,0470,180A     015D  3560 ;% 09    Q_R[TEMP1].RL.P                   ; ROTATE LOOP COUNT LEFT 18
U 0A, 7700,3800,3FFF,FFFF,8470,180B     0168  3564 ;% 0A    LONLIT_[80000000]                 ; MASK ROTATED LOOP COUNT
U 0B, 7700,4800,03A1,03D4,0470,180C     0173  3568 ;% 0B    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 0C, 7700,8840,03A4,0300,4A70,180D     017E  3572 ;% 0C    VA_Q.OR.R[TEMP0],SPW/RLONG        ; CREATE TB TEST ADDRESS
U 0D, 7700,3800,7FFF,FF7F,8470,180E     0189  3576 ;% 0D    LONLIT_[00000100]                 ; WRITE DATA
U 0E, 7700,C81B,5BE4,03D4,5070,180F     0194  3580 ;% 0E    TB_R[LONLIT]                      ; WRITE TB ACCORDING TO VA
U 0F, 7300,C800,0364,0300,0470,1810     019F  3584 ;% 0F    NOP,STOP.CLOCK                    ; STOP CPU CLOCK
                                        01AA  3588
                        00000000        01AA  3589 2$: .LONG        ^X00000000            ; TEMPORARY STORAGE
                                        01AE  3590
                                        01AE  3591 3$:
U 00, 7700,C800,0364,0300,0470,1801     01AE  3592 ;% 00    NOP
U 01, 5700,C840,0364,0304,0470,1802     01B9  3596 ;% 01    R[TEMP1]_RDM                      ; MOVE LOOP COUNT TO TEMP1
U 02, 7700,4801,4771,0300,0470,1803     01C4  3600 ;% 02    Q_M[TEMP1].RL.PTE                 ; ROTATE LOOP COUNT LEFT 9
U 03, 7700,3800,7FFF,80FF,8470,1804     01CF  3604 ;% 03    LONLIT_[0000FE00]                 ; MASK ROTATED LOOP COUNT
U 04, 7700,4800,03A1,03D4,0470,1805     01DA  3608 ;% 04    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 05, 7700,4860,03A4,03D8,0470,1806     01E5  3612 ;% 05    M[TEMP0]_Q                        ; MOVE Q TO TEMP0
U 06, 7700,1800,EF64,030C,0470,1807     01F0  3616 ;% 06    PL_[00000018]                     ; PLATCH GETS 18
U 07, 7700,C800,8B71,0304,0470,1808     01FB  3620 ;% 07    Q_R[TEMP1].RL.P                   ; ROTATE LOOP COUNT LEFT 18
U 08, 7700,B800,3FFF,FFFF,8470,1809     0206  3624 ;% 08    LONLIT_[80000000]                 ; MASK ROTATED LOOP COUNT
U 09, 7700,4800,03A1,03D4,0470,180A     0211  3628 ;% 09    Q_R[LONLIT].AND.Q                 ; AND STORE IN Q REGISTER
U 0A, 7700,8840,03A4,0300,4A70,180B     021C  3632 ;% 0A    VA_Q.OR.R[TEMP0],SPW/RLONG        ; LOAD TEST ADDRESS INTO VA
U 0B, 6701,081F,5924,0200,05F0,180C     0227  3636 ;% 0B    RDM_TB                            ; READ TB TO RDM WH REGISTER
U 0C, 7300,C800,0364,0300,0470,180D     0232  3640 ;% 0C    NOP,STOP.CLOCK                    ; STOP CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E     023D  3644 ;% 0D    NOP
```

```
U 0E, 5700,C81B,0364,0300,5070,180F  0248  3648 ;* 0E   TB_RDM                          ; WRITE NEXT TEST PATTERN
U 0F, 7300,C800,0364,0300,0470,1810  0253  3652 ;* 0F   NOP,STOP.CLOCK                  ; STOP CPU CLOCK
                                     025E  3656
                          00000100   025E  3657 4$:     .LONG   ^X00000100            ; TEST PATTERN TABLE
                                     0262  3658
                          00FFFFF8   0262  3659 5$:     .LONG   ^X00FFFFF8
                          00111188   0266  3660         .LONG   ^X00111188
                          00EEEF70   026A  3661         .LONG   ^X00EEEF70
                          00333398   026E  3662         .LONG   ^X00333398
                          00CCCD60   0272  3663         .LONG   ^X00CCCD60
                          007777B8   0276  3664         .LONG   ^X007777B8
                          00888940   027A  3665         .LONG   ^X00888940
                          00FFFFF8   027E  3666         .LONG   ^X00FFFFF8
                          00000100   0282  3667         .LONG   ^X00000100
                                     0286  3668
                          00FFFFF8   0286  3669 6$:     .LONG   ^X00FFFFF8            ; MASK FOR CMPREGMSK PSEUDO OP
                                     028A  3670
                                     028A  3671 BEGIN_CPU_DATA
                                     0002  3672
                                     0002  3673 RTEMP_DATA
                                     0001  3674
                          00000000   0001  3675         .LONG   ^X00000000            ; TEMPORARY STORAGE/RTEMP0
                          00000000   0005  3676         .LONG   ^X00000000            ; TEMPORARY STORAGE/RTEMP1
                          03000000   0009  3677         .LONG   ^X03000000            ; TB GROUP DISABLE REGISTER ADD
                          0D000000   000D  3678         .LONG   ^X0D000000            ; DISABLE GROUP 0
                                     0011  3679
                                     0011  3680 END_CPU_DATA
                                     028A  3681
                                     028A  3682 END_TEST_DATA
                                     028A
                                     028A       ;-----------------------------------------------------------------------------
                                     028A  3683
                                     028A  3684 ENDTEST
```

```
                        0025        .SBTTL  "TEST   042     TB GROUP 0 TAG MEMORY MARCH TEST
                        0025  3686 ;*******************************************************************************
                        0025  3687 ;++
                        0025  3688 ;
                        0025  3689 ;       TB GROUP 0 TAG MEMORY MARCH TEST
                        0025  3690 ;
                        0025  3691 ; FUNCTIONAL DESCRIPTION
                        0025  3692 ;       THIS IS A DYNAMIC TEST OF TB GROUP 0'S TAG STORE. THIS
                        0025  3693 ;       WILL TEST ADDRESSING AND DATA INTEGRITY INSIDE THE TB
                        0025  3694 ;       TAG STORE RAM CHIPS. ALL BITS IN THIS MEMORY ARE TESTED EXCEPT
                        0025  3695 ;       THE TB GROUP 0 VALID BIT.
                        0025  3696 ;
                        0025  3697 ; TEST ALGORITHM
                        0025  3698 ;       1.  CREATE A LOOP OF 256 TO VALIDATE ALL TB LOCATIONS
                        0025  3699 ;       2.  LOAD LOOP COUNT INTO RDM WD REGISTER
                        0025  3700 ;       3.  EXECUTE DCS PROGRAM
                        0025  3701 ;               a.  SET MISS GROUP 1 AND REPLACE GROUP 0
                        0025  3702 ;               b.  MOVE LOOP COUNT TO RTEMP1
                        0025  3703 ;               c.  SHIFT LOOP COUNT LEFT 9
                        0025  3704 ;               d.  MASK ROTATED LOOP COUNT AND SAVE IN RTEMP0
                        0025  3705 ;               e.  SHIFT LOOP COUNT LEFT 18
                        0025  3706 ;               f.  MASK ROTATED LOOP COUNT AND SAVE IN THE Q REGISTER
                        0025  3707 ;               g.  FORM TB ADDRESS AND MOVE TO VA REGISTER
                        0025  3708 ;               h.  SHIFT LOOP COUNT LEFT 9 AND SET VALID BIT
                        0025  3709 ;               i.  WRITE PTE ACCORDING TO VA
                        0025  3710 ;       4.  GO TO STEP 2 FOR REMAINING TB LOCATIONS
                        0025  3711 ;       5.  LOAD NEW DCS U-CODE
                        0025  3712 ;       6.  CREATE A LOOP OF 9 TO SELECT TEST PATTERNS (I)
                        0025  3713 ;       7.  MOVE TEST TAG TO DCS U-WORD
                        0025  3714 ;       8.  CREATE A LOOP TO READ THEN REWRITE ALL TB LOCATIONS (J)
                        0025  3715 ;       9.  LOAD LOOP COUNT (J) INTO RDM WD REGISTER
                        0025  3716 ;      10.  EXECUTE DCS PROGRAM
                        0025  3717 ;               a.  MOVE LOOP COUNT TO RTEMP5
                        0025  3718 ;               b.  SHIFT LOOP COUNT LEFT 9
                        0025  3719 ;               c.  MASK ROTATED LOOP COUNT AND STORE IN RTEMP6
                        0025  3720 ;               d.  SHIFT LOOP COUNT LEFT 18
                        0025  3721 ;               e.  MASK ROTATED LOOP COUNT AND STORE IN THE Q REGISTER
                        0025  3722 ;               f.  FORM TB INDEX AND STORE IN THE Q REGISTER
                        0025  3723 ;               g.  ADD TAG FIELD TO INDEX AND STORE IN VA AND RTEMP0
                        0025  3724 ;               h.  READ TB ACCORDING TO VA AND SAVE IN RTEMP1
                        0025  3725 ;      11.  TEST RDM WH REGISTER FOR A VALID PTE
                        0025  3726 ;      12.  IF NO ERROR LOAD RDM WD REGISTER WITH NEW TAG TEST PATTERN
                        0025  3727 ;      13.  EXECUTE DCS PROGRAM
                        0025  3728 ;               a.  MOVE NEW TAG FIELD TO RTEMP5
                        0025  3729 ;               b.  FORM NEW TB ADDRESS AND MOVE TO VA REGISTER
                        0025  3730 ;               c.  WRITE TB ACCORDING TO VA
                        0025  3731 ;      14.  GO TO STEP 9 FOR REMAINING TB LOCATIONS
                        0025  3732 ;      15.  GO TO STEP 7 FOR REMAINING TEST PATTERNS
                        0025  3733 ;      16.  REPEAT STEPS 6 THROUGH 15 WITH LOOP J DECREMENTING
                        0025  3734 ;
                        0025  3735 ; TEST PATTERNS
                        0025  3736 ;       EACH OF THE FOLLOWING DATA PATTERNS IS USED IN FOUR PASSES
                        0025  3737 ;       OF THIS TEST. FIRST THE PATTERN IS WRITTEN INTO THE TB TAG MEMORY WHILE
                        0025  3738 ;       THE TB INDEX IS BEING INCREMENTED. THEN THE PATTERN IS READ
                        0025  3739 ;       OUT OF THE TB TAG MEMORY WHILE THE TB INDEX IS BEING INCREMENTED. THEN
```

```
                     ;       THE PATTERN IN WRITTEN INTO THE TB TAG MEMORY WHILE THE TB INDEX IS
                     ;       BEING DECREMENTED. FINALLY THE PATTERN IS READ OUT OF THE
                     ;       TB TAG MEMORY WHILE THE TB INDEX IS BEING DECREMENTED.
                     ;       NOTE THAT THESE DATA PATTERNS ARE USED AS TB TAG'S.
                     ;       0.      00000000
                     ;       1.      7FFF0000
                     ;       2.      11110000
                     ;       3.      6EEE0000
                     ;       4.      33330000
                     ;       5.      4CCC0000
                     ;       6.      77770000
                     ;       7.      08880000
                     ;       8.      7FFF0000
                     ;       9.      00000000
                     ;
                     ; LOGIC DESCRIPTION
                     ;       TESTED HERE ARE ALL THE RAM CHIPS IN TB GROUP 0'S TAG
                     ;       STORE MEMORY EXCEPT THE VALID BIT.
                     ;       NOTE THAT IF THIS TEST FAILS IT IS ONE OF THESE RAM
                     ;       CHIPS WHICH IS MOST LIKELY AT FAULT.
                     ;
                     ; ASSUMPTIONS
                     ;       ALL LOGIC USED TO PERFORM THIS TEST WHICH IS EXTERNAL TO
                     ;       TB GROUP 0'S TAG STORE RAM CHIPS IS ASSUMED TO HAVE BEEN
                     ;       PREVIOUSLY TESTED. THIS IMPLIES THAT IF A FAILURE OCCURS WHILE THIS
                     ;       TEST IS RUNNING IT IS ONE OF THESE RAM CHIPS WHICH IS AT FAULT.
                     ;
                     ; ERROR DESCRIPTION
                     ;       VALID BIT SET IN PTE
                     ;       RECEIVED VALID BIT
                     ;       INDEX I
                     ;       INDEX J
                     ;       R0 = VA BEING READ
                     ;       R1 = PTE READ
                     ;               R1 SHOULD CONTAIN THE ADDRESS BEING READ IN BITS <16:09>.
                     ;               IF IT CONTAINS SOMEOTHER ADDRESS, THEN THAT LOCATION HAS
                     ;               OVERWRITTEN IT.  POSSIBLE INDICATION OF CROSSED ADDRESS LINES.
                     ;
                     ;--
                     ;*****************************************************************************
                     ;/////////////////////////////////////////////////////////////////////////////
                     
                             INITIALIZE
                             LOADDCS         1$,,0,<^X11>            ; LOAD UCODE TO WRITE
                                                                     ; PTE=00000100 INTO ALL
                                                                     ; GROUP 0 LOCATIONS.
                             LOOP            I,1,256                 ; THIS LOOP WRITES
                             SAVEINDEX       I,2$                    ; PTE=00000100 INTO EVERY
                             LOADREG         WDREG,2$,,L             ; TB GROUP 0 MEMORY LOCATION.
                             FETCH           0                       ; START DCS PROGRAM
                             BRSTCLK         <^X10>                  ; END DCS PROGRAM
                             ENDLOOP         I                       ; CONTINUE UNTIL ALL 256
                                                                     ; LOCATIONS HAVE BEEN WRITTEN.
                             LOADDCS         3$,,0,<^X14>            ; LOAD U-CODE TO READ AND THEN
                                                                     ; REWRITE TB
```

```
                0055  3795          LOOP            I,1,9                           ; INITIALIZE A LOOP THRU
                005C  3796                                                        ; THE TEST PATTERNS FOR TAG'S.
                005C  3797          LDFIELD         5$,I,L,4$,31,62,LON,            ; PUT THE CURRENT TAG
                006D  3798                                                        ; INTO THE UWORD LONLIT FIELD.
                006D  3799          LOADDCS         4$,,<^X0B>,1                    ; LOAD THE TEST UCODE.
                0074  3800          LOOP            J,1,256                         ; INITIALIZE A LOOP THRU EVERY
                007B  3801                                                        ; TB INDEX LOCATION IN THE
                007B  3802                                                        ; FORWARD DIRECTION.
                007B  3803          SAVEINDEX       J,2$                            ; LOAD THE TB INDEX VALUE.
                0080  3804          LOADREG         WDREG,2$,,L
                0088  3805          ERRLOOP                                       ; ERROR LOOP POINT
                008A  3806          FETCH           0                             ; EXECUTE THE TEST UCODE.
                008F  3807          BRSTCLK         <^X0E>
                0093  3808          CMPREGMSK       WHREG,7$,,7$,,L,              ; SEE IF THE PTE READ IS VALID
                009F  3809          IFERROR         2,,<<MIC>>                    ; THE PTE RETURNED WAS INCORRECT.
                00A5  3810          LOADREG         WDREG,6$,I,L                  ; NEW TAG TEST PATTERN
                00AD  3811          FETCH           <^X0F>                        ; START DCS PROGRAM
                00B2  3812          BRSTCLK         <^X13>                        ; END DCS PROGRAM
                00B6  3813          ENDLOOP         J                             ; LOOP THRU ALL INDEX LOCATIONS.
                00B9  3814          ENDLOOP         I                             ; LOOP THRU ALL TAG TEST PATTERNS.
                00BC  3815          LOOP            I,1,9                         ; INITIALIZE A LOOP THRU
                00C3  3816                                                        ; THE TEST PATTERNS FOR TAG'S.
                00C3  3817          LDFIELD         5$,I,L,4$,31,62,LON,          ; PUT THE CURRENT TAG
                00D4  3818                                                        ; INTO THE UWORD LONLIT FIELD.
                00D4  3819          LOADDCS         4$,,<^X0B>,1                  ; LOAD THE TEST UCODE.
                00DB  3820          LOOP            J,256,1                       ; INITIALIZE A LOOP THRU EVERY
                00E2  3821                                                        ; TB INDEX LOCATION IN THE
                00E2  3822                                                        ; BACKWARD DIRECTION.
                00E2  3823          SAVEINDEX       J,2$                          ; LOAD THE TB INDEX VALUE.
                00E7  3824          LOADREG         WDREG,2$,,L
                00EF  3825          ERRLOOP                                       ; ERROR LOOP POINT
                00F1  3826          FETCH           0                             ; EXECUTE THE TEST UCODE.
                00F6  3827          BRSTCLK         <^X0E>
                00FA  3828          CMPREGMSK       WHREG,7$,,7$,,L,              ; SEE IF THE PTE READ IS VALID
                0106  3829          IFERROR         2,,<<MIC>>                    ; THE PTE RETURNED WAS INCORRECT.
                010C  3830          LOADREG         WDREG,6$,I,L                  ; NEW TAG TEST PATTERN
                0114  3831          FETCH           <^X0F>                        ; START DCS PROGRAM
                0119  3832          BRSTCLK         <^X13>                        ; END DCS PROGRAM
                011D  3833          ENDLOOP         J                             ; LOOP THRU ALL INDEX LOCATIONS.
                0120  3834          ENDLOOP         I                             ; LOOP THRU ALL TAG TEST PATTERNS.
                0123  3835          SKIP                                          ; GO TO NEXT TEST
                012A  3836
                012A  3837 BEGIN_TEST_DATA
                012A
                012A        ;-------------------------------------------------------------------------
                012A  3838
                012A  3839 1$:
U 00, 7700,C800,0364,0300,0470,1801   012A  3840 ;% 00    NOP                           ; U-CODE TO VALIDATE TB
U 01, 7700,C800,5BE4,030C,6870,1802   0135  3844 ;% 01    MEMSCAR_R[TEMP3]              ; SET MISS GROUP 1 AND
U 02, 7700,8800,5BE4,0310,6070,1803   0140  3848 ;% 02    MEMSCR_R[TEMP4]               ; REPLACE GROUP 0
U 03, 5700,C840,0364,0304,0470,1804   014B  3852 ;% 03    R[TEMP1]_RDM                  ; MOVE LOOP COUNT TO RTEMP1
U 04, 7700,4801,4771,0300,0470,1805   0156  3856 ;% 04    Q_M[TEMP1].RL.PTE             ; SHIFT LOOP COUNT LEFT 9
U 05, 7700,B800,7FFF,80FF,8470,1806   0161  3860 ;% 05    LONLIT_[0000FE00]             ; MASK ROTATED LOOP COUNT
U 06, 7700,C800,03A1,03D4,0470,1807   016C  3864 ;% 06    Q_R[LONLIT].AND.Q
U 07, 7700,C860,03A4,03D8,0470,1808   0177  3868 ;% 07    M[TEMP0]_Q                    ; SAVE IN RTEMP0
```

```
U 08, 7700,9800,EF64,030C,0470,1809  0182  3872 ;% 08   PL_[00000018]                    ; PLATCH GETS 18
U 09, 7700,4800,8B71,0304,0470,180A  018D  3876 ;% 09   Q_R[TEMP1].RL.P                 ; SHIFT LOOP COUNT LEFT 18
U 0A, 7700,3800,3FFF,FFFF,8470,180B  0198  3880 ;% 0A   LONLIT_[80000000]              ; MASK ROTATED LOOP COUNT
U 0B, 7700,4800,03A1,03D4,0470,180C  01A3  3884 ;% 0B   Q_R[LONLIT].AND.Q
U 0C, 7700,C800,03A4,0300,4A70,180D  01AE  3888 ;% 0C   VA_Q.OR.R[TEMP0]               ; MOVE TB ADDRESS TO VA
U 0D, 7700,C801,4771,0300,0470,180E  01B9  3892 ;% 0D   Q_M[TEMP1].RL.PTE             ; SHIFT LOOP COUNT LEFT 9
U 0E, 7700,4800,03A5,0308,0470,180F  01C4  3896 ;% 0E   Q_R[TEMP2].OR.Q              ; SET VALID BIT IN PTE
U 0F, 7700,481B,03A4,03D8,5070,1810  01CF  3900 ;% 0F   TB_WB,WB_Q,MSRC/VA           ; WRITE TB
U 10, 7300,4800,0364,0300,0470,1811  01DA  3904 ;% 10   NOP,STOP.CLOCK               ; STOP CPU CLOCK
                                     01E5  3908
                          00000000   01E5  3909 2$:     .LONG   ^X00000000          ; TEMPORARY STORAGE
                                     01E9  3910
                                     01E9  3911 3$:
U 00, 7700,C800,0364,0300,0470,1801  01E9  3912 ;% 00   NOP                          ; U-CODE TO READ THEN WRITE TB
U 01, 5700,8840,0364,0314,0470,1802  01F4  3916 ;% 01   R[TEMP5]_RDM                 ; MOVE LOOP COUNT TO RTEMP5
U 02, 7700,0805,4771,0300,0470,1803  01FF  3920 ;% 02   Q_M[TEMP5].RL.PTE           ; SHIFT LOOP COUNT LEFT 9
U 03, 7700,3800,7FFF,80FF,8470,1804  020A  3924 ;% 03   LONLIT_[0000FE00]           ; MASK ROTATED LOOP COUNT
U 04, 7700,4800,03A1,03D4,0470,1805  0215  3928 ;% 04   Q_R[LONLIT].AND.Q
U 05, 7700,4866,03A4,03D8,0470,1806  0220  3932 ;% 05   M[TEMP6]_Q                   ; SAVE IN RTEMP6
U 06, 7700,1800,EF64,030C,0470,1807  022B  3936 ;% 06   PL_[00000018]              ; PLATCH GETS 18
U 07, 7700,8800,8B71,0314,0470,1808  0236  3940 ;% 07   Q_R[TEMP5].RL.P            ; ROTATE LOOP COUNT LEFT 18
U 08, 7700,B800,3FFF,FFFF,8470,1809  0241  3944 ;% 08   LONLIT_[80000000]          ; MASK ROTATED LOOP COUNT
U 09, 7700,4800,03A1,03D4,0470,180A  024C  3948 ;% 09   Q_R[LONLIT].AND.Q
U 0A, 7700,8800,03A5,0318,0470,180B  0257  3952 ;% 0A   Q_R[TEMP6].OR.Q             ; CREATE TB INDEX
                                     0262  3956 4$:
U 0B, 7700,F800,7FFF,FFFF,8470,180C  0262  3957 ;% 0B   LONLIT_[0]                  ; TAG FIELD FOR TB
U 0C, 7700,8840,03A4,03D4,4A70,180D  026D  3961 ;% 0C   VA_Q.OR.R[LONLIT],SPW/RLONG  ; VA AND TEMP0 GET TB ADDRESS
U 0D, 6701,C85F,5924,0304,05F0,180E  0278  3965 ;% 0D   RDM_WB,R[TEMP1]_TB          ; READ TB TO RDM AND TEMP1
U 0E, 7300,4800,0364,0300,0470,180F  0283  3969 ;% 0E   NOP,STOP.CLOCK              ; STOP CPU CLOCK
U 0F, 7700,C800,0364,0300,0470,1810  028E  3973 ;% 0F   NOP
U 10, 5700,0840,0364,0314,0470,1811  0299  3977 ;% 10   R[TEMP5]_RDM                 ; NEW TAG FIELD FOR WRITE
U 11, 7700,4800,03A4,0314,4A70,1812  02A4  3981 ;% 11   VA_Q.OR.R[TEMP5]            ; VA GETS NEW TB ADDRESS
U 12, 7700,081B,5BE4,0304,5070,1813  02AF  3985 ;% 12   TB_R[TEMP1]                 ; WRITE TB
U 13, 7300,4800,0364,0300,0470,1814  02BA  3989 ;% 13   NOP,STOP.CLOCK              ; STOP CPU CLOCK
                                     02C5  3993
                          00000000   02C5  3994 5$:     .LONG   ^X00000000          ; TEST PATTERN TABLE
                                     02C9  3995
                          7FFF0000   02C9  3996 6$:     .LONG   ^X7FFF0000
                          11110000   02CD  3997         .LONG   ^X11110000
                          6EEE0000   02D1  3998         .LONG   ^X6EEE0000
                          33330000   02D5  3999         .LONG   ^X33330000
                          4CCC0000   02D9  4000         .LONG   ^X4CCC0000
                          77770000   02DD  4001         .LONG   ^X77770000
                          08880000   02E1  4002         .LONG   ^X08880000
                          7FFF0000   02E5  4003         .LONG   ^X7FFF0000
                          00000000   02E9  4004         .LONG   ^X00000000
                                     02ED  4005
                          00000100   02ED  4006 7$:     .LONG   ^X00000100          ; COMPARE DATA
                                     02F1  4007
                                     02F1  4008 BEGIN_CPU_DATA
                                     0002  4009
                                     0002  4010 RTEMP_DATA
                                     0001  4011
                          00000000   0001  4012 .LONG   ^X00000000          ;ADDRESS READ/RTEMP 0
                          00000000   0005  4013 .LONG   ^X00000000          ;DATA READ/RTEMP 1
```

```
00000100  0009  4014 .LONG   ^X00000100                    ;VALID BIT/RTEMP 2
03000000  000D  4015 .LONG   ^X03000000                    ;TB DISABLE REG ADD/RTEMP 3
0A000000  0011  4016 .LONG   ^X0A000000                    ;DISABLE GROUP 1/RTEMP 4
00000000  0015  4017 .LONG   ^X00000000                    ;TEMPORARY STORAGE/RTEMP 5
00000000  0019  4018 .LONG   ^X00000000                    ;TEMPORARY STORAGE/RTEMP 6
          001D  4019
          001D  4020 END_CPU_DATA
          02F1  4021
          02F1  4022 END_TEST_DATA
          02F1
          02F1       ;-------------------------------------------------------------------------
          02F1  4023
          02F1  4024 ENDTEST
```

```
0025          .SBTTL  "TEST   043     TB GROUP 1 TAG MEMORY MARCH TEST
0025  4026 ;*******************************************************************************
0025  4027 ;++
0025  4028 ;
0025  4029 ;       TB GROUP 1 TAG MEMORY MARCH TEST
0025  4030 ;
0025  4031 ; FUNCTIONAL DESCRIPTION
0025  4032 ;       THIS IS A DYNAMIC TEST OF TB GROUP 1'S TAG STORE. THIS
0025  4033 ;       WILL TEST ADDRESSING AND DATA INTEGRITY INSIDE THE TB
0025  4034 ;       TAG STORE RAM CHIPS. ALL BITS IN THIS MEMORY ARE TESTED EXCEPT
0025  4035 ;       THE TB GROUP 1 VALID BIT.
0025  4036 ;
0025  4037 ; TEST ALGORITHM
0025  4038 ;       1.  CREATE A LOOP OF 256 TO VALIDATE ALL TB LOCATIONS
0025  4039 ;       2.  LOAD LOOP COUNT INTO RDM WD REGISTER
0025  4040 ;       3.  EXECUTE DCS PROGRAM
0025  4041 ;               a.  SET MISS GROUP 0 AND REPLACE GROUP 1
0025  4042 ;               b.  MOVE LOOP COUNT TO RTEMP1
0025  4043 ;               c.  SHIFT LOOP COUNT LEFT 9
0025  4044 ;               d.  MASK ROTATED LOOP COUNT AND SAVE IN RTEMP0
0025  4045 ;               e.  SHIFT LOOP COUNT LEFT 18
0025  4046 ;               f.  MASK ROTATED LOOP COUNT AND SAVE IN THE Q REGISTER
0025  4047 ;               g.  FORM TB ADDRESS AND MOVE TO VA REGISTER
0025  4048 ;               h.  SHIFT LOOP COUNT LEFT 9 AND SET VALID BIT
0025  4049 ;               i.  WRITE PTE ACCORDING TO VA
0025  4050 ;       4.  GO TO STEP 2 FOR REMAINING TB LOCATIONS
0025  4051 ;       5.  LOAD NEW DCS U-CODE
0025  4052 ;       6.  CREATE A LOOP OF 9 TO SELECT TEST PATTERNS (I)
0025  4053 ;       7.  MOVE TEST TAG TO DCS U-WORD
0025  4054 ;       8.  CREATE A LOOP TO READ THEN REWRITE ALL TB LOCATIONS (J)
0025  4055 ;       9.  LOAD LOOP COUNT (J) INTO RDM WD REGISTER
0025  4056 ;      10.  EXECUTE DCS PROGRAM
0025  4057 ;               a.  MOVE LOOP COUNT TO RTEMP5
0025  4058 ;               b.  SHIFT LOOP COUNT LEFT 9
0025  4059 ;               c.  MASK ROTATED LOOP COUNT AND STORE IN RTEMP6
0025  4060 ;               d.  SHIFT LOOP COUNT LEFT 18
0025  4061 ;               e.  MASK ROTATED LOOP COUNT AND STORE IN THE Q REGISTER
0025  4062 ;               f.  FORM TB INDEX AND STORE IN THE Q REGISTER
0025  4063 ;               g.  ADD TAG FIELD TO INDEX AND STORE IN VA AND RTEMP0
0025  4064 ;               h.  READ TB ACCORDING TO VA AND SAVE IN RTEMP1
0025  4065 ;      11.  TEST RDM WH REGISTER FOR A VALID PTE
0025  4066 ;      12.  IF NO ERROR LOAD RDM WD REGISTER WITH NEW TAG TEST PATTERN
0025  4067 ;      13.  EXECUTE DCS PROGRAM
0025  4068 ;               a.  MOVE NEW TAG FIELD TO RTEMP5
0025  4069 ;               b.  FORM NEW TB ADDRESS AND MOVE TO VA REGISTER
0025  4070 ;               c.  WRITE TB ACCORDING TO VA
0025  4071 ;      14.  GO TO STEP 9 FOR REMAINING TB LOCATIONS
0025  4072 ;      15.  GO TO STEP 7 FOR REMAINING TEST PATTERNS
0025  4073 ;      16.  REPEAT STEPS 6 THROUGH 15 WITH LOOP J DECREMENTING
0025  4074 ;
0025  4075 ; TEST PATTERNS
0025  4076 ;       EACH OF THE FOLLOWING DATA PATTERNS IS USED IN FOUR PASSES
0025  4077 ;       OF THIS TEST. FIRST THE PATTERN IS WRITTEN INTO THE TB TAG MEMORY WHILE
0025  4078 ;       THE TB INDEX IS BEING INCREMENTED. THEN THE PATTERN IS READ
0025  4079 ;       OUT OF THE TB TAG MEMORY WHILE THE TB INDEX IS BEING INCREMENTED. THEN
```

```
0025  4080 ;       THE PATTERN IN WRITTEN INTO THE TB TAG MEMORY WHILE THE TB INDEX IS
0025  4081 ;       BEING DECREMENTED. FINALLY THE PATTERN IS READ OUT OF THE
0025  4082 ;       TB TAG MEMORY WHILE THE TB INDEX IS BEING DECREMENTED.
0025  4083 ;       NOTE THAT THESE DATA PATTERNS ARE USED AS TB TAG'S.
0025  4084 ;       0.      00000000
0025  4085 ;       1.      7FFF0000
0025  4086 ;       2.      11110000
0025  4087 ;       3.      6EEE0000
0025  4088 ;       4.      33330000
0025  4089 ;       5.      4CCC0000
0025  4090 ;       6.      77770000
0025  4091 ;       7.      08880000
0025  4092 ;       8.      7FFF0000
0025  4093 ;       9.      00000000
0025  4094 ;
0025  4095 ; LOGIC DESCRIPTION
0025  4096 ;       TESTED HERE ARE ALL THE RAM CHIPS IN TB GROUP 1'S TAG
0025  4097 ;       STORE MEMORY EXCEPT THE VALID BIT.
0025  4098 ;       NOTE THAT IF THIS TEST FAILS IT IS ONE OF THESE RAM
0025  4099 ;       CHIPS WHICH IS MOST LIKELY AT FAULT.
0025  4100 ;
0025  4101 ; ASSUMPTIONS
0025  4102 ;       ALL LOGIC USED TO PERFORM THIS TEST WHICH IS EXTERNAL TO
0025  4103 ;       TB GROUP 1'S TAG STORE RAM CHIPS IS ASSUMED TO HAVE BEEN
0025  4104 ;       PREVIOUSLY TESTED. THIS IMPLIES THAT IF A FAILURE OCCURS WHILE THIS
0025  4105 ;       TEST IS RUNNING IT IS ONE OF THESE RAM CHIPS WHICH IS AT FAULT.
0025  4106 ;
0025  4107 ; ERROR DESCRIPTION
0025  4108 ;       VALID BIT SET IN PTE
0025  4109 ;       RECEIVED VALID BIT
0025  4110 ;       INDEX I
0025  4111 ;       INDEX J
0025  4112 ;       R0 = VA BEING READ
0025  4113 ;       R1 = PTE READ
0025  4114 ;               R1 SHOULD CONTAIN THE ADDRESS BEING READ IN BITS <16:09>.
0025  4115 ;               IF IT CONTAINS SOMEOTHER ADDRESS, THEN THAT LOCATION HAS
0025  4116 ;               OVERWRITTEN IT.  POSSIBLE INDICATION OF CROSSED ADDRESS LINES.
0025  4117 ;
0025  4118 ;--
0025  4119 ;*****************************************************************************
0025  4120 ;/////////////////////////////////////////////////////////////////////////////
0025  4121
0025  4122         INITIALIZE
0027  4123         LOADDCS         1$,,0,<^X11>            ; LOAD UCODE TO WRITE
002E  4124                                                 ; PTE=00000100 INTO ALL
002E  4125                                                 ; GROUP 1 LOCATIONS.
002E  4126         LOOP            I,1,256                 ; THIS LOOP WRITES
0035  4127         SAVEINDEX       I,2$                    ; PTE=00000100 INTO EVERY
003A  4128         LOADREG         WDREG,2$,,L             ; TB GROUP 1 MEMORY LOCATION.
0042  4129         FETCH           0                       ; START DCS PROGRAM
0047  4130         BRSTCLK         <^X10>                  ; END DCS PROGRAM
004B  4131         ENDLOOP         I                       ; CONTINUE UNTIL ALL 256
004E  4132                                                 ; LOCATIONS HAVE BEEN WRITTEN.
004E  4133         LOADDCS         3$,,0,<^X14>            ; LOAD U-CODE TO READ AND THEN
0055  4134                                                 ; REWRITE TB
```

```
                0055  4135          LOOP            I,1,9                   ; INITIALIZE A LOOP THRU
                005C  4136                                                  ; THE TEST PATTERNS FOR TAG'S.
                005C  4137          LDFIELD         5$,I,L,4$,31,62,LON,    ; PUT THE CURRENT TAG
                006D  4138                                                  ; INTO THE UWORD LONLIT FIELD.
                006D  4139          LOADDCS         4$,,<^X0B>,1            ; LOAD THE TEST UCODE.
                0074  4140          LOOP            J,1,256                 ; INITIALIZE A LOOP THRU EVERY
                007B  4141                                                  ; TB INDEX LOCATION IN THE
                007B  4142                                                  ; FORWARD DIRECTION.
                007B  4143          SAVEINDEX       J,2$                    ; LOAD THE TB INDEX VALUE.
                0080  4144          LOADREG         WDREG,2$,,L
                0088  4145          ERRLOOP                                 ; ERROR LOOP POINT
                008A  4146          FETCH           0                       ; EXECUTE THE TEST UCODE.
                008F  4147          BRSTCLK         <^X0E>
                0093  4148          CMPREGMSK       WHREG,7$,,7$,,L,        ; SEE IF THE PTE READ IS VALID
                009F  4149          IFERROR         2,,<<MIC>>              ; THE PTE RETURNED WAS INCORRECT.
                00A5  4150          LOADREG         WDREG,6$,I,L            ; NEW TAG TEST PATTERN
                00AD  4151          FETCH           <^X0F>                  ; START DCS PROGRAM
                00B2  4152          BRSTCLK         <^X13>                  ; END DCS PROGRAM
                00B6  4153          ENDLOOP         J                       ; LOOP THRU ALL INDEX LOCATIONS.
                00B9  4154          ENDLOOP         I                       ; LOOP THRU ALL TAG TEST PATTERNS.
                00BC  4155          LOOP            I,1,9                   ; INITIALIZE A LOOP THRU
                00C3  4156                                                  ; THE TEST PATTERNS FOR TAG'S.
                00C3  4157          LDFIELD         5$,I,L,4$,31,62,LON,    ; PUT THE CURRENT TAG
                00D4  4158                                                  ; INTO THE UWORD LONLIT FIELD.
                00D4  4159          LOADDCS         4$,,<^X0B>,1            ; LOAD THE TEST UCODE.
                00DB  4160          LOOP            J,256,1                 ; INITIALIZE A LOOP THRU EVERY
                00E2  4161                                                  ; TB INDEX LOCATION IN THE
                00E2  4162                                                  ; BACKWARD DIRECTION.
                00E2  4163          SAVEINDEX       J,2$                    ; LOAD THE TB INDEX VALUE.
                00E7  4164          LOADREG         WDREG,2$,,L
                00EF  4165          ERRLOOP                                 ; ERROR LOOP POINT
                00F1  4166          FETCH           0                       ; EXECUTE THE TEST UCODE.
                00F6  4167          BRSTCLK         <^X0E>
                00FA  4168          CMPREGMSK       WHREG,7$,,7$,,L,        ; SEE IF THE PTE READ IS VALID
                0106  4169          IFERROR         2,,<<MIC>>              ; THE PTE RETURNED WAS INCORRECT.
                010C  4170          LOADREG         WDREG,6$,I,L            ; NEW TAG TEST PATTERN
                0114  4171          FETCH           <^X0F>                  ; START DCS PROGRAM
                0119  4172          BRSTCLK         <^X13>                  ; END DCS PROGRAM
                011D  4173          ENDLOOP         J                       ; LOOP THRU ALL INDEX LOCATIONS.
                0120  4174          ENDLOOP         I                       ; LOOP THRU ALL TAG TEST PATTERNS.
                0123  4175          SKIP                                    ; GO TO NEXT TEST
                012A  4176
                012A  4177 BEGIN_TEST_DATA
                012A
                012A        ;-------------------------------------------------------------------------
                012A  4178
                012A  4179 1$:
U 00, 7700,C800,0364,0300,0470,1801   012A  4180 ;% 00    NOP                     ; U-CODE TO VALIDATE TB
U 01, 7700,C800,5BE4,030C,6870,1802   0135  4184 ;% 01    MEMSCAR_R[TEMP3]        ; SET MISS GROUP 0 AND
U 02, 7700,8800,5BE4,0310,6070,1803   0140  4188 ;% 02    MEMSCR_R[TEMP4]         ; REPLACE GROUP 1
U 03, 5700,C840,0364,0304,0470,1804   014B  4192 ;% 03    R[TEMP1]_RDM            ; MOVE LOOP COUNT TO RTEMP1
U 04, 7700,4801,4771,0300,0470,1805   0156  4196 ;% 04    Q_M[TEMP1].RL.PTE       ; SHIFT LOOP COUNT LEFT 9
U 05, 7700,B800,7FFF,80FF,8470,1806   0161  4200 ;% 05    LONLIT_[0000FE00]       ; MASK ROTATED LOOP COUNT
U 06, 7700,C800,03A1,03D4,0470,1807   016C  4204 ;% 06    Q_R[LONLIT].AND.Q
U 07, 7700,C860,03A4,03D8,0470,1808   0177  4208 ;% 07    M[TEMP0]_Q              ; SAVE IN RTEMP0
```

```
U 08. 7700,9800,EF64,030C,0470,1809  0182  4212 ;% 08   PL_[00000018]                    ; PLATCH GETS 18
U 09. 7700,4800,8B71,0304,0470,180A  018D  4216 ;% 09   Q_R[TEMP1].RL.P                 ; SHIFT LOOP COUNT LEFT 18
U 0A. 7700,3800,3FFF,FFFF,8470,180B  0198  4220 ;% 0A   LONLIT_[80000000]              ; MASK ROTATED LOOP COUNT
U 0B. 7700,4800,03A1,03D4,0470,180C  01A3  4224 ;% 0B   Q_R[LONLIT].AND.Q
U 0C. 7700,C800,03A4,0300,4A70,180D  01AE  4228 ;% 0C   VA_Q.OR.R[TEMP0]                ; MOVE TB ADDRESS TO VA
U 0D. 7700,C801,4771,0300,0470,180E  01B9  4232 ;% 0D   Q_M[TEMP1].RL.PTE             ; SHIFT LOOP COUNT LEFT 9
U 0E. 7700,4800,03A5,0308,0470,180F  01C4  4236 ;% 0E   Q_R[TEMP2].OR.Q               ; SET VALID BIT IN PTE
U 0F. 7700,481B,03A4,03D8,5070,1810  01CF  4240 ;% 0F   TB_WB,WB_Q,MSRC/VA            ; WRITE TB
U 10. 7300,4800,0364,0300,0470,1811  01DA  4244 ;% 10   NOP,STOP.CLOCK                ; STOP CPU CLOCK
                                     01E5  4248
                         00000000    01E5  4249 2$:     .LONG   ^X00000000           ; TEMPORARY STORAGE
                                     01E9  4250
                                     01E9  4251 3$:
U 00. 7700,C800,0364,0300,0470,1801  01E9  4252 ;% 00   NOP                           ; U-CODE TO READ THEN WRITE TB
U 01. 5700,8840,0364,0314,0470,1802  01F4  4256 ;% 01   R[TEMP5]_RDM                  ; MOVE LOOP COUNT TO RTEMP5
U 02. 7700,0805,4771,0300,0470,1803  01FF  4260 ;% 02   Q_M[TEMP5].RL.PTE            ; SHIFT LOOP COUNT LEFT 9
U 03. 7700,3800,7FFF,80FF,8470,1804  020A  4264 ;% 03   LONLIT_[0000FE00]           ; MASK ROTATED LOOP COUNT
U 04. 7700,4800,03A1,03D4,0470,1805  0215  4268 ;% 04   Q_R[LONLIT].AND.Q
U 05. 7700,4866,03A4,03D8,0470,1806  0220  4272 ;% 05   M[TEMP6]_Q                   ; SAVE IN RTEMP6
U 06. 7700,1800,EF64,030C,0470,1807  022B  4276 ;% 06   PL_[00000018]                ; PLATCH GETS 18
U 07. 7700,8800,8B71,0314,0470,1808  0236  4280 ;% 07   Q_R[TEMP5].RL.P             ; ROTATE LOOP COUNT LEFT 18
U 08. 7700,B800,3FFF,FFFF,8470,1809  0241  4284 ;% 08   LONLIT_[80000000]           ; MASK ROTATED LOOP COUNT
U 09. 7700,4800,03A1,03D4,0470,180A  024C  4288 ;% 09   Q_R[LONLIT].AND.Q
U 0A. 7700,8800,03A5,0318,0470,180B  0257  4292 ;% 0A   Q_R[TEMP6].OR.Q             ; CREATE TB INDEX
                                     0262  4296 4$:
U 0B. 7700,F800,7FFF,FFFF,8470,180C  0262  4297 ;% 0B   LONLIT_[0]                   ; TAG FIELD FOR TB
U 0C. 7700,8840,03A4,03D4,4A70,180D  026D  4301 ;% 0C   VA_Q.OR.R[LONLIT],SPW/RLONG  ; VA AND TEMP0 GET TB ADDRESS
U 0D. 6701,C85F,5924,0304,05F0,180E  0278  4305 ;% 0D   RDM_WB,R[TEMP1]_TB          ; READ TB TO RDM AND TEMP1
U 0E. 7300,4800,0364,0300,0470,180F  0283  4309 ;% 0E   NOP,STOP.CLOCK              ; STOP CPU CLOCK
U 0F. 7700,C800,0364,0300,0470,1810  028E  4313 ;% 0F   NOP
U 10. 5700,0840,0364,0314,0470,1811  0299  4317 ;% 10   R[TEMP5]_RDM                 ; NEW TAG FIELD FOR WRITE
U 11. 7700,4800,03A4,0314,4A70,1812  02A4  4321 ;% 11   VA_Q.OR.R[TEMP5]            ; VA GETS NEW TB ADDRESS
U 12. 7700,081B,5BE4,0304,5070,1813  02AF  4325 ;% 12   TB_R[TEMP1]                 ; WRITE TB
U 13. 7300,4800,0364,0300,0470,1814  02BA  4329 ;% 13   NOP,STOP.CLOCK              ; STOP CPU CLOCK
                                     02C5  4333
                         00000000    02C5  4334 5$:     .LONG   ^X00000000           ; TEST PATTERN TABLE
                                     02C9  4335
                         7FFF0000    02C9  4336 6$:     .LONG   ^X7FFF0000
                         11110000    02CD  4337         .LONG   ^X11110000
                         6EEE0000    02D1  4338         .LONG   ^X6EEE0000
                         33330000    02D5  4339         .LONG   ^X33330000
                         4CCC0000    02D9  4340         .LONG   ^X4CCC0000
                         77770000    02DD  4341         .LONG   ^X77770000
                         08880000    02E1  4342         .LONG   ^X08880000
                         7FFF0000    02E5  4343         .LONG   ^X7FFF0000
                         00000000    02E9  4344         .LONG   ^X00000000
                                     02ED  4345
                         00000100    02ED  4346 7$:     .LONG   ^X00000100           ; COMPARE DATA
                                     02F1  4347
                                     02F1  4348 BEGIN_CPU_DATA
                                     0002  4349
                                     0002  4350 RTEMP_DATA
                                     0001  4351
                         00000000    0001  4352 .LONG   ^X00000000          ;ADDRESS READ/RTEMP 0
                         00000000    0005  4353 .LONG   ^X00000000          ;DATA READ/RTEMP 1
```

```
00000100  0009  4354 .LONG   ^X00000100                    ;VALID BIT/RTEMP 2
03000000  000D  4355 .LONG   ^X03000000                    ;TB DISABLE REG ADD/RTEMP 3
0D000000  0011  4356 .LONG   ^X0D000000                    ;DISABLE GROUP 0/RTEMP 4
00000000  0015  4357 .LONG   ^X00000000                    ;TEMPORARY STORAGE/RTEMP 5
00000000  0019  4358 .LONG   ^X00000000                    ;TEMPORARY STORAGE/RTEMP 6
          001D  4359
          001D  4360 END_CPU_DATA
          02F1  4361
          02F1  4362 END_TEST_DATA
          02F1
          02F1       ;-------------------------------------------------------------------------
          02F1  4363
          02F1  4364 ENDTEST
```

```
                     0024        .SBTTL  "TEST   044     TB GROUP 0 VALID BIT MARCH TEST
                     0024  4366 ;*****************************************************************************
                     0024  4367 ;++
                     0024  4368 ;
                     0024  4369 ;       TB GROUP 0 VALID BIT MARCH TEST
                     0024  4370 ;
                     0024  4371 ; FUNCTIONAL DESCRIPTION
                     0024  4372 ;       THIS IS A DYNAMIC TEST OF TB GROUP 0'S VALID BIT. THIS
                     0024  4373 ;       WILL TEST ADDRESSING AND DATA INTEGRITY INSIDE THE TB
                     0024  4374 ;       VALID BIT RAM CHIP.
                     0024  4375 ;
                     0024  4376 ; TEST ALGORITHM
                     0024  4377 ;       1.  CREATE A LOOP OF 256 TO VALIDATE ALL TB LOCATIONS
                     0024  4378 ;       2.  PUT LOOP COUNT IN RDM WD REGISTER
                     0024  4379 ;       3.  EXECUTE DCS PROGRAM
                     0024  4380 ;               a.  SET MISS GROUP 1 AND REPLACE GROUP 0
                     0024  4381 ;               b.  MOVE LOOP COUNT TO RTEMP1
                     0024  4382 ;               c.  SHIFT LOOP COUNT LEFT 9
                     0024  4383 ;               d.  MASK ROTATED LOOP COUNT AND STORE IN RTEMP0
                     0024  4384 ;               e.  SHIFT LOOP COUNT LEFT 18
                     0024  4385 ;               f.  MASK ROTATED LOOP COUNT
                     0024  4386 ;               g.  FORM TB ADDRESS AND MOVE TO VA REGISTER
                     0024  4387 ;               h.  SHIFT LOOP COUNT LEFT 9
                     0024  4388 ;               i.  SET VALID BIT
                     0024  4389 ;               j.  WRITE TB ACCORDING TO VA
                     0024  4390 ;       4.  GO TO STEP 2 FOR REMAINING TB LOCATIONS
                     0024  4391 ;       5.  LOAD NEW DCS U-CODE
                     0024  4392 ;       6.  CREATE A LOOP TO SELECT TEST PATTERNS (I)
                     0024  4393 ;       7.  MODIFY DCS U-CODE ACCORDING TO LOOP (I)
                     0024  4394 ;       8.  CREATE A LOOP TO SELECT ALL TB LOCATIONS (J)
                     0024  4395 ;       9.  LOAD LOOP COUNT INTO RDM WD REGISTER
                     0024  4396 ;      10.  EXECUTE DCS PROGRAM
                     0024  4397 ;               a.  LOAD LOOP COUNT INTO RTEMP6
                     0024  4398 ;               b.  SHIFT LOOP COUNT LEFT 9
                     0024  4399 ;               c.  MASK ROTATED LOOP COUNT AND SAVE IN RTEMP0
                     0024  4400 ;               d.  SHIFT LOOP COUNT LEFT 18
                     0024  4401 ;               e.  MASK ROTATED LOOP COUNT
                     0024  4402 ;               f.  FORM TB ADDRESS AND MOVE TO VA AND RTEMP0
                     0024  4403 ;               g.  READ TB TO RDM WH REGISTER AND RTEMP1
                     0024  4404 ;      11.  TEST RDM WH REGISTER FOR CORRECT VALID BIT
                     0024  4405 ;      12.  IF NO ERROR EXECUTE DCS PROGRAM
                     0024  4406 ;               a.  MOVE MASK TO Q REGISTER
                     0024  4407 ;               b.  COMPLEMENT VALID BIT
                     0024  4408 ;               c.  REWRITE TB
                     0024  4409 ;      13.  GO TO STEP 9 FOR REMAINING TB LOCATIONS
                     0024  4410 ;      14.  GO TO STEP 7 FOR REMAINING TEST PATTERNS
                     0024  4411 ;      15.  REPEAT STEPS 6 TO 14 WITH LOOP J DECREMENTING
                     0024  4412 ;
                     0024  4413 ; TEST PATTERNS
                     0024  4414 ;       LOOP I = 1      VALID BIT SET
                     0024  4415 ;       LOOP I = 2      VALID BIT NOT SET
                     0024  4416 ;
                     0024  4417 ; LOGIC DESCRIPTION
                     0024  4418 ;       THIS TEST SHOULD ONLY BE TESTING THE VALID BIT RAM CHIP.
                     0024  4419 ;
```

```
                     4420 ; ASSUMPTIONS
                     4421 ;       ALL LOGIC EXTERNAL TO THE TB GROUP 0 VALID BIT RAM
                     4422 ;       WHICH IS USED IN THIS TEST IS ASSUMED TO BE OPERATIONAL.
                     4423 ;       THIS IMPLIES THAT IF A FAILURE OCCURS IN THIS TEST
                     4424 ;       IT IS THIS RAM WHICH IS A FAULT.
                     4425 ;
                     4426 ; ERROR DESCRIPTION
                     4427 ;       EXPECTED VALID BIT
                     4428 ;       RECEIVED VALID BIT
                     4429 ;       LOOP COUNT I
                     4430 ;       LOOP COUNT J
                     4431 ;       R0 = TB ADDRESS BEING READ
                     4432 ;       R1 = DATA IN THAT ADDRESS
                     4433 ;               NOTE: R1 BITS <16:09> SHOULD CONTAIN THE TB ADDRESS READ.
                     4434 ;               IF THEY ARE DIFFERENT THEN SOMEBODY OVERWROTE THAT LOCATION.
                     4435 ;
                     4436 ;--
                     4437 ;*******************************************************************************
                     4438 ;///////////////////////////////////////////////////////////////////////////////
                     4439
                     4440         INITIALIZE
0026                 4441         LOADDCS         1$,,0,<^X11>            ; LOAD UCODE TO WRITE
002D                 4442                                                 ; PTE=00000100 INTO ALL
002D                 4443                                                 ; GROUP 0 LOCATIONS.
002D                 4444         LOOP            I,1,256                 ; THIS LOOP WRITES VA=00000000
0034                 4445         SAVEINDEX       I,2$                    ; PTE=00000100 INTO EVERY
0039                 4446         LOADREG         WDREG,2$,,L             ; TB GROUP 0 MEMORY LOCATION.
0041                 4447         FETCH           0
0046                 4448         BRSTCLK         <^X10>
004A                 4449         ENDLOOP         I                       ; CONTINUE UNTIL ALL 256
004D                 4450                                                 ; LOCATIONS HAVE BEEN WRITTEN.
004D                 4451         LOADDCS         3$,,0,<^X12>            ; LOAD NEW U-CODE
0054                 4452         LOOP            I,1,2                   ; INITIALIZE A LOOP THRU
005B                 4453                                                 ; THE TEST PATTERNS FOR PTE'S.
005B                 4454         LOADDCS         4$,I,<^X0E>,2           ; LOAD THE TEST UCODE.
0062                 4455         LOOP            J,1,256                 ; INITIALIZE A LOOP THRU EVERY
0069                 4456                                                 ; TB INDEX LOCATION IN THE
0069                 4457                                                 ; FORWARD DIRECTION.
0069                 4458         SAVEINDEX       J,2$                    ; LOAD THE TB INDEX VALUE.
006E                 4459         LOADREG         WDREG,2$,,L
0076                 4460         ERRLOOP                                 ; ERROR LOOP POINT
0078                 4461         FETCH           0                       ; EXECUTE THE TEST UCODE.
007D                 4462         BRSTCLK         <^X0C>
0081                 4463         CMPREGMSK       WHREG,5$,I,5$,,L        ; SEE IF THE PTE READ IS CORRECT.
008D                 4464         IFERROR         2,,<<MIC>>              ; THE PTE RETURNED WAS INCORRECT.
0093                 4465         FETCH           <^X0D>                  ; START DCS PROGRAM
0098                 4466         BRSTCLK         <^X11>                  ; END DCS PROGRAM
009C                 4467         ENDLOOP         J                       ; LOOP THRU ALL INDEX LOCATIONS.
009F                 4468         ENDLOOP         I                       ; LOOP THRU ALL PTE TEST PATTERNS.
00A2                 4469         LOOP            I,1,2                   ; INITIALIZE A LOOP THRU
00A9                 4470                                                 ; THE TEST PATTERNS FOR PTE'S.
00A9                 4471         LOADDCS         4$,I,<^X0E>,2           ; LOAD THE TEST UCODE.
00B0                 4472         LOOP            J,256,1                 ; INITIALIZE A LOOP THRU EVERY
00B7                 4473                                                 ; TB INDEX LOCATION IN THE
00B7                 4474                                                 ; BACKWARD DIRECTION.
```

```
                        00B7  4475          SAVEINDEX       J,2$                            ; LOAD THE TB INDEX VALUE.
                        00BC  4476          LOADREG         WDREG,2$,,L
                        00C4  4477          ERRLOOP                                         ; ERROR LOOP POINT
                        00C6  4478          FETCH           0                               ; EXECUTE THE TEST UCODE.
                        00CB  4479          BRSTCLK         <^X0C>
                        00CF  4480          CMPREGMSK       WHREG,5$,I,5$,,L                ; SEE IF THE PTE READ IS CORRECT.
                        00DB  4481          IFERROR         2,,<<MIC>>                      ; THE PTE RETURNED WAS INCORRECT.
                        00E1  4482          FETCH           <^X0D>                          ; START DCS PROGRAM
                        00E6  4483          BRSTCLK         <^X11>                          ; END DCS PROGRAM
                        00EA  4484          ENDLOOP         J                               ; LOOP THRU ALL INDEX LOCATIONS.
                        00ED  4485          ENDLOOP         I                               ; LOOP THRU ALL PTE TEST PATTERNS.
                        00F0  4486          SKIP                                            ; GO TO NEXT TEST
                        00F7  4487
                        00F7  4488 BEGIN_TEST_DATA
                        00F7
                        00F7          ;-------------------------------------------------------------------------
                        00F7  4489
                        00F7  4490 1$:
U 00. 7700,C800,0364,0300,0470,1801     00F7  4491 ;% 00    NOP
U 01. 7700,8800,5BE4,0308,6870,1802     0102  4495 ;% 01    MEMSCAR_R[TEMP2]                ; TB DISABLE REGISTER ADDRESS
U 02. 7700,C800,5BE4,030C,6070,1803     010D  4499 ;% 02    MEMSCR_R[TEMP3]                 ; MISS GROUP 1 REPLACE GROUP 0
U 03. 5700,C840,0364,0304,0470,1804     0118  4503 ;% 03    R[TEMP1]_RDM                    ; LOOP COUNT TO RTEMP1
U 04. 7700,4801,4771,0300,0470,1805     0123  4507 ;% 04    Q_M[TEMP1].RL.PTE               ; SHIFT LOOP COUNT LEFT 9
U 05. 7700,B800,7FFF,80FF,8470,1806     012E  4511 ;% 05    LONLIT_[0000FE00]               ; MASK ROTATED LOOP COUNT
U 06. 7700,C800,03A1,03D4,0470,1807     0139  4515 ;% 06    Q_R[LONLIT].AND.Q               ; AND STORE
U 07. 7700,C860,03A4,03D8,0470,1808     0144  4519 ;% 07    M[TEMP0]_Q                      ; IN RTEMP0
U 08. 7700,9800,EF64,030C,0470,1809     014F  4523 ;% 08    PL_[00000018]                   ; PLATCH GETS 18
U 09. 7700,4800,8B71,0304,0470,180A     015A  4527 ;% 09    Q_R[TEMP1].RL.P                 ; SHIFT LOOP COUNT LEFT 18
U 0A. 7700,3800,3FFF,FFFF,8470,180B     0165  4531 ;% 0A    LONLIT_[80000000]               ; MASK ROTATED LOOP COUNT
U 0B. 7700,4800,03A1,03D4,0470,180C     0170  4535 ;% 0B    Q_R[LONLIT].AND.Q
U 0C. 7700,C800,03A4,0300,4A70,180D     017B  4539 ;% 0C    VA_Q.OR.R[TEMP0]                ; FORM TB ADDRESS FOR VA REG
U 0D. 7700,C801,4771,0300,0470,180E     0186  4543 ;% 0D    Q_M[TEMP1].RL.PTE               ; SHIFT LOOP COUNT LEFT 9
U 0E. 7700,4800,03A5,0310,0470,180F     0191  4547 ;% 0E    Q_R[TEMP4].OR.Q                 ; SET VALID BIT
U 0F. 7700,481B,03A4,03D8,5070,1810     019C  4551 ;% 0F    TB_WB,WB_Q,MSRC/VA              ; WRITE TB
U 10. 7300,4800,0364,0300,0470,1811     01A7  4555 ;% 10    NOP,STOP.CLOCK                  ; STOP CPU CLOCK
                                        01B2  4559
                        00000000        01B2  4560 2$:      .LONG   ^X00000000              ; TEMPORARY STORAGE
                                        01B6  4561
                                        01B6  4562 3$:
U 00. 7700,C800,0364,0300,0470,1801     01B6  4563 ;% 00    NOP                             ; U-CODE TO READ AND REWRITE TB
U 01. 5700,8840,0364,0318,0470,1802     01C1  4567 ;% 01    R[TEMP6]_RDM                    ; LOOP COUNT TO RTEMP6
U 02. 7700,0806,4771,0300,0470,1803     01CC  4571 ;% 02    Q_M[TEMP6].RL.PTE               ; ROTATE LOOP COUNT LEFT 9
U 03. 7700,3800,7FFF,80FF,8470,1804     01D7  4575 ;% 03    LONLIT_[0000FE00]               ; MASK ROTATED LOOP COUNT
U 04. 7700,4800,03A1,03D4,0470,1805     01E2  4579 ;% 04    Q_R[LONLIT].AND.Q               ; AND SAVE
U 05. 7700,4860,03A4,03D8,0470,1806     01ED  4583 ;% 05    M[TEMP0]_Q                      ; IN RTEMP0
U 06. 7700,1800,EF64,030C,0470,1807     01F8  4587 ;% 06    PL_[00000018]                   ; PLATCH GETS 18
U 07. 7700,8800,8B71,0318,0470,1808     0203  4591 ;% 07    Q_R[TEMP6].RL.P                 ; SHIFT LOOP COUNT LEFT 18
U 08. 7700,B800,3FFF,FFFF,8470,1809     020E  4595 ;% 08    LONLIT_[80000000]               ; MASK ROTATED LOOP COUNT
U 09. 7700,4800,03A1,03D4,0470,180A     0219  4599 ;% 09    Q_R[LONLIT].AND.Q
U 0A. 7700,8840,03A4,0300,4A70,180B     0224  4603 ;% 0A    VA_Q.OR.R[TEMP0],SPW/RLONG      ; TB ADDRESS TO VA AND RTEMP0
U 0B. 6701,485F,5924,0304,05F0,180C     022F  4607 ;% 0B    RDM_WB,R[TEMP1]_TB              ; RDM WH REG GETS TB DATA
U 0C. 7300,C800,0364,0300,0470,180D     023A  4611 ;% 0C    NOP,STOP.CLOCK                  ; STOP CPU CLOCK
U 0D. 7700,C800,0364,0300,0470,180E     0245  4615 ;% 0D    NOP
U 0E. 7700,0800,5BE5,0314,0470,180F     0250  4619 ;% 0E    Q_R[TEMP5]                      ; MASK FOR WRITE TEST PATTERN
U 0F. 7700,8800,03A1,0304,0470,1810     025B  4623 ;% 0F    Q_R[TEMP1].AND.Q                ; SET UP TEST PATTERN
```

```
U 10, 7700,C81B,03A4,03D8,5070,1811  0266  4627 ;% 10   TB_WB,WB_Q,MSRC/VA                    ; WRITE NEW TEST PATTERN
U 11, 7300,4800,0364,0300,0470,1812  0271  4631 ;% 11   NOP,STOP.CLOCK                        ; STOP CPU CLOCK
                                     027C  4635 4$:
U 0E, 7700,0800,5BE5,0314,0470,180F  027C  4636 ;% 0E   Q_R[TEMP5]                          ; U-CODE LOOP I=1
U 0F, 7700,8800,03A1,0304,0470,1810  0287  4640 ;% 0F   Q_R[TEMP1].AND.Q
U 0E, 7700,4800,5BE5,0310,0470,180F  0292  4644 ;% 0E   Q_R[TEMP4]                          ; U-CODE LOOP I=2
U 0F, 7700,C800,03A5,0304,0470,1810  029D  4648 ;% 0F   Q_R[TEMP1].OR.Q
                                     02A8  4652
                         00000100    02A8  4653 5$:     .LONG   ^X00000100                ; COMPARE DATA
                         00000000    02AC  4654         .LONG   ^X00000000
                                     02B0  4655
                                     02B0  4656 BEGIN_CPU_DATA
                                     0002  4657
                                     0002  4658 RTEMP_DATA
                                     0001  4659
                         00000000    0001  4660         .LONG   ^X00000000                ; RTEMP0
                         00000000    0005  4661         .LONG   ^X00000000                ; RTEMP1
                         03000000    0009  4662         .LONG   ^X03000000                ; TB DISABLE REG ADDRESS
                         0A000000    000D  4663         .LONG   ^X0A000000                ; DISABLE GROUP 1
                         00000100    0011  4664         .LONG   ^X00000100                ; SET VALID BIT
                         00FFFE00    0015  4665         .LONG   ^X00FFFE00                ; CLEAR VALID BIT
                         00000000    0019  4666         .LONG   ^X00000000                ; RTEMP6
                                     001D  4667
                                     001D  4668 END_CPU_DATA
                                     02B0  4669
                                     02B0  4670 END_TEST_DATA
                                     02B0
                                     02B0       ;-----------------------------------------------------------------------------
                                     02B0  4671
                                     02B0  4672 ENDTEST
```

```
0024        .SBTTL  "TEST   045     TB GROUP 1 VALID BIT MARCH TEST
0024  4674 ;*******************************************************************************
0024  4675 ;++
0024  4676 ;
0024  4677 ;       TB GROUP 1 VALID BIT MARCH TEST
0024  4678 ;
0024  4679 ; FUNCTIONAL DESCRIPTION
0024  4680 ;       THIS IS A DYNAMIC TEST OF TB GROUP 1'S VALID BIT. THIS
0024  4681 ;       WILL TEST ADDRESSING AND DATA INTEGRITY INSIDE THE TB
0024  4682 ;       VALID BIT RAM CHIP.
0024  4683 ;
0024  4684 ; TEST ALGORITHM
0024  4685 ;       1.  CREATE A LOOP OF 256 TO VALIDATE ALL TB LOCATIONS
0024  4686 ;       2.  PUT LOOP COUNT IN RDM WD REGISTER
0024  4687 ;       3.  EXECUTE DCS PROGRAM
0024  4688 ;               a.  SET MISS GROUP 0 AND REPLACE GROUP 1
0024  4689 ;               b.  MOVE LOOP COUNT TO RTEMP1
0024  4690 ;               c.  SHIFT LOOP COUNT LEFT 9
0024  4691 ;               d.  MASK ROTATED LOOP COUNT AND STORE IN RTEMP0
0024  4692 ;               e.  SHIFT LOOP COUNT LEFT 18
0024  4693 ;               f.  MASK ROTATED LOOP COUNT
0024  4694 ;               g.  FORM TB ADDRESS AND MOVE TO VA REGISTER
0024  4695 ;               h.  SHIFT LOOP COUNT LEFT 9
0024  4696 ;               i.  SET VALID BIT
0024  4697 ;               j.  WRITE TB ACCORDING TO VA
0024  4698 ;       4.  GO TO STEP 2 FOR REMAINING TB LOCATIONS
0024  4699 ;       5.  LOAD NEW DCS U-CODE
0024  4700 ;       6.  CREATE A LOOP TO SELECT TEST PATTERNS (I)
0024  4701 ;       7.  MODIFY DCS U-CODE ACCORDING TO LOOP (I)
0024  4702 ;       8.  CREATE A LOOP TO SELECT ALL TB LOCATIONS (J)
0024  4703 ;       9.  LOAD LOOP COUNT INTO RDM WD REGISTER
0024  4704 ;      10.  EXECUTE DCS PROGRAM
0024  4705 ;               a.  LOAD LOOP COUNT INTO RTEMP6
0024  4706 ;               b.  SHIFT LOOP COUNT LEFT 9
0024  4707 ;               c.  MASK ROTATED LOOP COUNT AND SAVE IN RTEMP0
0024  4708 ;               d.  SHIFT LOOP COUNT LEFT 18
0024  4709 ;               e.  MASK ROTATED LOOP COUNT
0024  4710 ;               f.  FORM TB ADDRESS AND MOVE TO VA AND RTEMP0
0024  4711 ;               g.  READ TB TO RDM WH REGISTER AND RTEMP1
0024  4712 ;      11.  TEST RDM WH REGISTER FOR CORRECT VALID BIT
0024  4713 ;      12.  IF NO ERROR EXECUTE DCS PROGRAM
0024  4714 ;               a.  MOVE MASK TO Q REGISTER
0024  4715 ;               b.  COMPLEMENT VALID BIT
0024  4716 ;               c.  REWRITE TB
0024  4717 ;      13.  GO TO STEP 9 FOR REMAINING TB LOCATIONS
0024  4718 ;      14.  GO TO STEP 7 FOR REMAINING TEST PATTERNS
0024  4719 ;      15.  REPEAT STEPS 6 TO 14 WITH LOOP J DECREMENTING
0024  4720 ;
0024  4721 ; TEST PATTERNS
0024  4722 ;       LOOP I = 1      VALID BIT SET
0024  4723 ;       LOOP I = 2      VALID BIT NOT SET
0024  4724 ;
0024  4725 ; LOGIC DESCRIPTION
0024  4726 ;       THIS TEST SHOULD ONLY BE TESTING THE VALID BIT RAM CHIP.
0024  4727 ;
```

```
                     4728 ; ASSUMPTIONS
                     4729 ;       ALL LOGIC EXTERNAL TO THE TB GROUP 1 VALID BIT RAM
                     4730 ;       WHICH IS USED IN THIS TEST IS ASSUMED TO BE OPERATIONAL.
                     4731 ;       THIS IMPLIES THAT IF A FAILURE OCCURS IN THIS TEST
                     4732 ;       IT IS THIS RAM WHICH IS A FAULT.
                     4733 ;
                     4734 ; ERROR DESCRIPTION
                     4735 ;       EXPECTED VALID BIT
                     4736 ;       RECEIVED VALID BIT
                     4737 ;       LOOP COUNT I
                     4738 ;       LOOP COUNT J
                     4739 ;       R0 = TB ADDRESS BEING READ
                     4740 ;       R1 = DATA IN THAT ADDRESS
                     4741 ;               NOTE: R1 BITS <16:09> SHOULD CONTAIN THE TB ADDRESS READ.
                     4742 ;               IF THEY ARE DIFFERENT THEN SOMEBODY OVERWROTE THAT LOCATION.
                     4743 ;
                     4744 ;--
                     4745 ;*****************************************************************************
                     4746 ;/////////////////////////////////////////////////////////////////////////////
0024                 4747
0024                 4748         INITIALIZE
0026                 4749         LOADDCS         1$,,0,<^X11>            ; LOAD UCODE TO WRITE
002D                 4750                                                 ; PTE=00000100 INTO ALL
002D                 4751                                                 ; GROUP 1 LOCATIONS.
002D                 4752         LOOP            I,1,256                 ; THIS LOOP WRITES VA=00000000
0034                 4753         SAVEINDEX       I,2$                    ; PTE=00000100 INTO EVERY
0039                 4754         LOADREG         WDREG,2$,,L             ; TB GROUP 1 MEMORY LOCATION.
0041                 4755         FETCH           0
0046                 4756         BRSTCLK         <^X10>
004A                 4757         ENDLOOP         I                       ; CONTINUE UNTIL ALL 256
004D                 4758                                                 ; LOCATIONS HAVE BEEN WRITTEN.
004D                 4759         LOADDCS         3$,,0,<^X12>            ; LOAD NEW U-CODE
0054                 4760         LOOP            I,1,2                   ; INITIALIZE A LOOP THRU
005B                 4761                                                 ; THE TEST PATTERNS FOR PTE'S.
005B                 4762         LOADDCS         4$,I,<^X0E>,2           ; LOAD THE TEST UCODE.
0062                 4763         LOOP            J,1,256                 ; INITIALIZE A LOOP THRU EVERY
0069                 4764                                                 ; TB INDEX LOCATION IN THE
0069                 4765                                                 ; FORWARD DIRECTION.
0069                 4766         SAVEINDEX       J,2$                    ; LOAD THE TB INDEX VALUE.
006E                 4767         LOADREG         WDREG,2$,,L
0076                 4768         ERRLOOP                                 ; ERROR LOOP POINT
0078                 4769         FETCH           0                       ; EXECUTE THE TEST UCODE.
007D                 4770         BRSTCLK         <^X0C>
0081                 4771         CMPREGMSK       WHREG,5$,I,5$,,L        ; SEE IF THE PTE READ IS CORRECT.
008D                 4772         IFERROR         2,,<<MIC>>              ; THE PTE RETURNED WAS INCORRECT.
0093                 4773         FETCH           <^X0D>                  ; START DCS PROGRAM
0098                 4774         BRSTCLK         <^X11>                  ; END DCS PROGRAM
009C                 4775         ENDLOOP         J                       ; LOOP THRU ALL INDEX LOCATIONS.
009F                 4776         ENDLOOP         I                       ; LOOP THRU ALL PTE TEST PATTERNS.
00A2                 4777         LOOP            I,1,2                   ; INITIALIZE A LOOP THRU
00A9                 4778                                                 ; THE TEST PATTERNS FOR PTE'S.
00A9                 4779         LOADDCS         4$,I,<^X0E>,2           ; LOAD THE TEST UCODE.
00B0                 4780         LOOP            J,256,1                 ; INITIALIZE A LOOP THRU EVERY
00B7                 4781                                                 ; TB INDEX LOCATION IN THE
00B7                 4782                                                 ; BACKWARD DIRECTION.
```

```
                        00B7  4783          SAVEINDEX       J,2$                    ; LOAD THE TB INDEX VALUE.
                        00BC  4784          LOADREG         WDREG,2$,,L
                        00C4  4785          ERRLOOP                                 ; ERROR LOOP POINT
                        00C6  4786          FETCH           0                       ; EXECUTE THE TEST UCODE.
                        00CB  4787          BRSTCLK         <^X0C>
                        00CF  4788          CMPREGMSK       WHREG,5$,I,5$,,L        ; SEE IF THE PTE READ IS CORRECT.
                        00DB  4789          IFERROR         2,,<<MIC>>              ; THE PTE RETURNED WAS INCORRECT.
                        00E1  4790          FETCH           <^X0D>                  ; START DCS PROGRAM
                        00E6  4791          BRSTCLK         <^X11>                  ; END DCS PROGRAM
                        00EA  4792          ENDLOOP         J                       ; LOOP THRU ALL INDEX LOCATIONS.
                        00ED  4793          ENDLOOP         I                       ; LOOP THRU ALL PTE TEST PATTERNS.
                        00F0  4794          SKIP                                    ; GO TO NEXT TEST
                        00F7  4795
                        00F7  4796 BEGIN_TEST_DATA
                        00F7
                        00F7                ;-------------------------------------------------------------------------
                        00F7  4797
                        00F7  4798 1$:
U 00, 7700,C800,0364,0300,0470,1801     00F7  4799 ;% 00    NOP
U 01, 7700,8800,5BE4,0308,6870,1802     0102  4803 ;% 01    MEMSCAR_R[TEMP2]                ; TB DISABLE REGISTER ADDRESS
U 02, 7700,C800,5BE4,030C,6070,1803     010D  4807 ;% 02    MEMSCR_R[TEMP3]                 ; MISS GROUP 0 REPLACE GROUP 1
U 03, 5700,C840,0364,0304,0470,1804     0118  4811 ;% 03    R[TEMP1]_RDM                    ; LOOP COUNT TO RTEMP1
U 04, 7700,4801,4771,0300,0470,1805     0123  4815 ;% 04    Q_M[TEMP1].RL.PTE               ; SHIFT LOOP COUNT LEFT 9
U 05, 7700,B800,7FFF,80FF,8470,1806     012E  4819 ;% 05    LONLIT_[0000FE00]               ; MASK ROTATED LOOP COUNT
U 06, 7700,C800,03A1,03D4,0470,1807     0139  4823 ;% 06    Q_R[LONLIT].AND.Q               ; AND STORE
U 07, 7700,C860,03A4,03D8,0470,1808     0144  4827 ;% 07    M[TEMP0]_Q                      ; IN RTEMP0
U 08, 7700,9800,EF64,030C,0470,1809     014F  4831 ;% 08    PL_[00000018]                   ; PLATCH GETS 18
U 09, 7700,4800,8B71,0304,0470,180A     015A  4835 ;% 09    Q_R[TEMP1].RL.P                 ; SHIFT LOOP COUNT LEFT 18
U 0A, 7700,3800,3FFF,FFFF,8470,180B     0165  4839 ;% 0A    LONLIT_[80000000]               ; MASK ROTATED LOOP COUNT
U 0B, 7700,4800,03A1,03D4,0470,180C     0170  4843 ;% 0B    Q_R[LONLIT].AND.Q
U 0C, 7700,C800,03A4,0300,4A70,180D     017B  4847 ;% 0C    VA_Q.OR.R[TEMP0]                ; FORM TB ADDRESS FOR VA REG
U 0D, 7700,C801,4771,0300,0470,180E     0186  4851 ;% 0D    Q_M[TEMP1].RL.PTE               ; SHIFT LOOP COUNT LEFT 9
U 0E, 7700,4800,03A5,0310,0470,180F     0191  4855 ;% 0E    Q_R[TEMP4].OR.Q                 ; SET VALID BIT
U 0F, 7700,481B,03A4,03D8,5070,1810     019C  4859 ;% 0F    TB_WB,WB_Q,MSRC/VA              ; WRITE TB
U 10, 7300,4800,0364,0300,0470,1811     01A7  4863 ;% 10    NOP,STOP.CLOCK                  ; STOP CPU CLOCK
                                        01B2  4867
                            00000000    01B2  4868 2$:      .LONG   ^X00000000              ; TEMPORARY STORAGE
                                        01B6  4869
                                        01B6  4870 3$:
U 00, 7700,C800,0364,0300,0470,1801     01B6  4871 ;% 00    NOP                             ; U-CODE TO READ AND REWRITE TB
U 01, 5700,8840,0364,0318,0470,1802     01C1  4875 ;% 01    R[TEMP6]_RDM                    ; LOOP COUNT TO RTEMP6
U 02, 7700,0806,4771,0300,0470,1803     01CC  4879 ;% 02    Q_M[TEMP6].RL.PTE               ; ROTATE LOOP COUNT LEFT 9
U 03, 7700,3800,7FFF,80FF,8470,1804     01D7  4883 ;% 03    LONLIT_[0000FE00]               ; MASK ROTATED LOOP COUNT
U 04, 7700,4800,03A1,03D4,0470,1805     01E2  4887 ;% 04    Q_R[LONLIT].AND.Q               ; AND SAVE
U 05, 7700,4860,03A4,03D8,0470,1806     01ED  4891 ;% 05    M[TEMP0]_Q                      ; IN RTEMP0
U 06, 7700,1800,EF64,030C,0470,1807     01F8  4895 ;% 06    PL_[00000018]                   ; PLATCH GETS 18
U 07, 7700,8800,8B71,0318,0470,1808     0203  4899 ;% 07    Q_R[TEMP6].RL.P                 ; SHIFT LOOP COUNT LEFT 18
U 08, 7700,B800,3FFF,FFFF,8470,1809     020E  4903 ;% 08    LONLIT_[80000000]               ; MASK ROTATED LOOP COUNT
U 09, 7700,4800,03A1,03D4,0470,180A     0219  4907 ;% 09    Q_R[LONLIT].AND.Q
U 0A, 7700,8840,03A4,0300,4A70,180B     0224  4911 ;% 0A    VA_Q.OR.R[TEMP0],SPW/RLONG      ; TB ADDRESS TO VA AND RTEMP0
U 0B, 6701,485F,5924,0304,05F0,180C     022F  4915 ;% 0B    RDM_WB,R[TEMP1]_TB              ; RDM WH REG GETS TB DATA
U 0C, 7300,C800,0364,0300,0470,180D     023A  4919 ;% 0C    NOP,STOP.CLOCK                  ; STOP CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E     0245  4923 ;% 0D    NOP
U 0E, 7700,0800,5BE5,0314,0470,180F     0250  4927 ;% 0E    Q_R[TEMP5]                      ; MASK FOR WRITE TEST PATTERN
U 0F, 7700,8800,03A1,0304,0470,1810     025B  4931 ;% 0F    Q_R[TEMP1].AND.Q                ; SET UP TEST PATTERN
```

```
U 10, 7700,C81B,03A4,03D8,5070,1811  0266  4935 ;% 10   TB_WB,WB_Q,MSRC/VA                    ; WRITE NEW TEST PATTERN
U 11, 7300,4800,0364,0300,0470,1812  0271  4939 ;% 11   NOP,STOP.CLOCK                        ; STOP CPU CLOCK
                                     027C  4943
                                     027C  4944 4$:
U 0E, 7700,0800,5BE5,0314,0470,180F  027C  4945 ;% 0E   Q_R[TEMP5]                          ; U-CODE LOOP I=1
U 0F, 7700,8800,03A1,0304,0470,1810  0287  4949 ;% 0F   Q_R[TEMP1].AND.Q
U 0E, 7700,4800,5BE5,0310,0470,180F  0292  4953 ;% 0E   Q_R[TEMP4]                          ; U-CODE LOOP I=2
U 0F, 7700,C800,03A5,0304,0470,1810  029D  4957 ;% 0F   Q_R[TEMP1].OR.Q
                                     02A8  4961
                          00000100   02A8  4962 5$:     .LONG   ^X00000100                 ; COMPARE DATA
                          00000000   02AC  4963         .LONG   ^X00000000
                                     02B0  4964
                                     02B0  4965
                                     02B0  4966 BEGIN_CPU_DATA
                                     0002  4967
                                     0002  4968 RTEMP_DATA
                                     0001  4969
                          00000000   0001  4970         .LONG   ^X00000000                 ; RTEMP0
                          00000000   0005  4971         .LONG   ^X00000000                 ; RTEMP1
                          03000000   0009  4972         .LONG   ^X03000000                 ; TB DISABLE REG ADDRESS
                          0D000000   000D  4973         .LONG   ^X0D000000                 ; DISABLE GROUP 0
                          00000100   0011  4974         .LONG   ^X00000100                 ; SET VALID BIT
                          00FFFE00   0015  4975         .LONG   ^X00FFFE00                 ; CLEAR VALID BIT
                          00000000   0019  4976         .LONG   ^X00000000                 ; RTEMP6
                                     001D  4977
                                     001D  4978 END_CPU_DATA
                                     02B0  4979
                                     02B0  4980 END_TEST_DATA
                                     02B0
                                     02B0       ;-----------------------------------------------------------------------------
                                     02B0  4981
                                     02B0  4982 ENDTEST
```

```
0019          .SBTTL  "TEST   046     TB INVALIDATE TEST 1
0019  4984 ;*******************************************************************************
0019  4985 ;++
0019  4986 ;
0019  4987 ; FUNCTIONAL DESCRIPTION
0019  4988 ;
0019  4989 ;       This test is used to explicitly test the signal FORCE MA 09 H which is
0019  4990 ;       logically ORed in during TB invalidate to insure that two TB locations
0019  4991 ;       are invalidated with every WCTRL/CLRTB.VA_WB micro-order.
0019  4992 ;
0019  4993 ; TEST ALGORITHM
0019  4994 ;
0019  4995 ;         1. Initialize the CPU.
0019  4996 ;         2. Disable the CMI and enable Memory Management.
0019  4997 ;         3. Loop for I = 1,2.
0019  4998 ;         4. Loop for J = 1,2.
0019  4999 ;         5. Load the RDM D register with a VA address from 1$.
0019  5000 ;         6. Execute the following micro-instructions:
0019  5001 ;                 00 NOP                      ; WAIT FOR A STABLE CLOCK
0019  5002 ;                 01 LONLIT_[03000000]        ; ADDRESS OF TB MEMSCR TO LONLIT
0019  5003 ;                 02 MEMSCAR_R[LONLIT]        ; PLACE ADDRESS INTO MEMSCAR
0019  5004 ;                 03 LONLIT_[0A000000]        ; TB GROUP 1 OFF  TB GROUP 0 ON
0019  5005 ;                 04 MEMSCR_R[LONLIT]         ;                 "          "
0019  5006 ;                 05 LONLIT_[00000148]        ; ROTATED PTE FOR TB LOADING
0019  5007 ;                 06 VA_R[TEMP1]              ; VA ADDRESS OF 200 HEX
0019  5008 ;                 07 TB_R[LONLIT]             ; LOAD PTE INTO TB AT 200 HEX
0019  5009 ;                 08 VA_R[TEMP0]              ; VA ADDRESS OF 0 HEX
0019  5010 ;                 09 TB_R[LONLIT]             ; LOAD PTE INTO TB AT 0 HEX
0019  5011 ;                 0A CLRTB VA_VA              ; CLEAR TB AT 0 AND 200 HEX
0019  5012 ;                 0B VA_RDM                   ; VA ADDRESS FROM RDM
0019  5013 ;                 0C RDM_M[TB]                ; READ PTE FROM TB TO RDM
0019  5014 ;                 0D NOP,STOP.CLOCK           ; STOP
0019  5015 ;         7. Compare the PTE valid bit from the CPU with zero.
0019  5016 ;         8. If the two are not equal signal an error.
0019  5017 ;         9. Repeat loop J.
0019  5018 ;        10. Disable TB group 0, enable TB group 1 by changing DCS location 3.
0019  5019 ;        11. Repeat loop I.
0019  5020 ;        12. End of test.
0019  5021 ;
0019  5022 ; LOGIC DESCRIPTION
0019  5023 ;
0019  5024 ;       TBS
0019  5025 ;
0019  5026 ; ASSUMPTIONS
0019  5027 ;
0019  5028 ;       This test assumes that the DPM and the translation buffer have been
0019  5029 ;       successfully tested prior to running this test.
0019  5030 ;
0019  5031 ; ERROR DESCRIPTION
0019  5032 ;
0019  5033 ;       On error this test will call out the ADD and ACV gate arrays on the
0019  5034 ;       MIC module. The following information will also be typed out:
0019  5035 ;
0019  5036 ;       Expected Data
0019  5037 ;       Received Data
```

```
                                0019  5038 ;       Loop I
                                0019  5039 ;       Loop J
                                0019  5040 ;
                                0019  5041 ;--
                                0019  5042 ;*******************************************************************************
                                0019  5043 ;///////////////////////////////////////////////////////////////////////////////
                                0019  5044
                                0019  5045         INITIALIZE                                  ; INITIALIZE THE CPU
                                001B  5046         LOADDCS         4$,,0,10                    ; MICRO-CODE FOR CMI/MME
                                0022  5047         FETCH           0                           ; CMI = OFF AND
                                0027  5048         BRSTCLK         9                           ; MEMORY MANAGEMENT = ON
                                002B  5049         LOADDCS         5$,,0,<^X10>                ; LOAD TEST U-CODE INTO DCS
                                0032  5050         LOOP            I,1,2                       ; I=1 TB GR 0, I=2 TB GR 1
                                0039  5051         LOOP            J,1,2                       ; J=1 VA=0, J=2 VA=200
                                0040  5052         LOADREG         WDREG,1$,J,LONG             ; VA ADDRESS FOR TB READ
                                0048  5053         ERRLOOP                                     ; ERROR LOOP HERE
                                004A  5054         FETCH           0                           ; LOAD PTE'S AT 0 AND 200 HEX
                                004F  5055         BRSTCLK         <^X0F>                      ; AND THEN INVALIDATE BOTH
                                0053  5056         CMPREGMSK       WHREG,2$,,3$,,LONG          ; INVALIDATED PTE READ BACK?
                                005F  5057         IFERROR         ,<<ADD><ACV>>,              ; IF NOT SIGNAL AN ERROR
                                0066  5058         ENDLOOP         J                           ; OTHERWISE CONTINUE
                                0069  5059         LOADDCS         6$,,3,1                     ; TB GR 1 ON, TB GR 0 OFF
                                0070  5060         ENDLOOP         I                           ; DO LOOP J AGAIN
                                0073  5061         SKIP                                        ; END OF TEST
                                007A  5062
                                007A  5063 BEGIN_TEST_DATA
                                007A
                                007A       ;-----------------------------------------------------------------------
                                007A  5064
                        00000000  007A  5065 1$:     .LONG   ^X 0                ; VA ADDRESS 0 HEX
                        00000200  007E  5066         .LONG   ^X 200              ; VA ADDRESS 200 HEX
                                0082  5067
                        00000000  0082  5068 2$:     .LONG   ^X 0                ; INVALID PTE
                                0086  5069
                        00000100  0086  5070 3$:     .LONG   ^X 100              ; MASK FOR REGISTER COMPARISON
                                008A  5071
                                008A  5072 4$:
U 00, 7700,C800,0364,0300,0470,1801  008A  5073 ;% 00   NOP
U 01, 7700,3800,78FF,FFFF,8470,1802  0095  5077 ;% 01   LONLIT_[0E000000]           ; DISABLE CMI
U 02, 7700,4800,5BE4,03D4,6870,1803  00A0  5081 ;% 02   MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804  00AB  5085 ;% 03   LONLIT_[01000000]
U 04, 7700,C800,5BE4,03D4,6070,1805  00B6  5089 ;% 04   MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806  00C1  5093 ;% 05   LONLIT_[0]                  ; ENABLE MEMORY MANAGEMENT
U 06, 7700,C800,5BE4,03D4,6870,1807  00CC  5097 ;% 06   MEMSCAR_R[LONLIT]
U 07, 7700,3800,7F7F,FFFF,8470,1808  00D7  5101 ;% 07   LONLIT_[01000000]
U 08, 7700,C800,5BE4,03D4,6070,1809  00E2  5105 ;% 08   MEMSCR_R[LONLIT]
U 09, 7300,4800,0364,0300,0470,180A  00ED  5109 ;% 09   STOP.CLOCK
                                00F8  5113
                                00F8  5114 5$:
U 00, 7700,C800,0364,0300,0470,1801  00F8  5115 ;% 00   NOP                         ; WAIT FOR A STABLE CLOCK
U 01, 7700,7800,7E7F,FFFF,8470,1802  0103  5119 ;% 01   LONLIT_[03000000]           ; ADDRESS OF TB MEMSCR TO LONLIT
U 02, 7700,4800,5BE4,03D4,6870,1803  010E  5123 ;% 02   MEMSCAR_R[LONLIT]           ; PLACE ADDRESS INTO MEMSCAR
U 03, 7700,7800,7AFF,FFFF,8470,1804  0119  5127 ;% 03   LONLIT_[0A000000]           ; TB GROUP 1 OFF  TB GROUP 0 ON
U 04, 7700,C800,5BE4,03D4,6070,1805  0124  5131 ;% 04   MEMSCR_R[LONLIT]            ;        "                "
U 05, 7700,B800,7FFF,FF5B,8470,1806  012F  5135 ;% 05   LONLIT_[00000148]           ; ROTATED PTE FOR TB LOADING
```

```
U 06. 7700.8800.5BE4.0304.4A70.1807  013A  5139 ;% 06   VA_R[TEMP1]                ; VA ADDRESS OF 200 HEX
U 07. 7700.481B.5BE4.03D4.5070.1808  0145  5143 ;% 07   TB_R[LONLIT]               ; LOAD PTE INTO TB AT 200 HEX
U 08. 7700.4800.5BE4.0300.4A70.1809  0150  5147 ;% 08   VA_R[TEMP0]               ; VA ADDRESS OF 0 HEX
U 09. 7700.C81B.5BE4.03D4.5070.180A  015B  5151 ;% 09   TB_R[LONLIT]               ; LOAD PTE INTO TB AT 0 HEX
U 0A. 7700.8800.5BE6.03D8.0470.180B  0166  5155 ;% 0A   D_R[ZERO]                  ; zero the D register
U 0B. 7700.D800.DB13.0300.4A70.180C  0171  5159 ;% 0B   VA_Q_D_D+ZLIT8[0]         ; set va to 0
U 0C. 7700.1800.DB12.0300.5360.180D  017C  5163 ;% 0C   VA_D_D+ZLIT8[0] INVALIDATE TB
                                     0187  5167                                  ; invalidate the TB at locations 0 & 200
U 0D. 5700.C800.0364.0300.4A70.180E  0187  5168 ;% 0D   VA_RDM                    ; VA ADDRESS FROM RDM
U 0E. 6700.481F.5924.0300.0470.180F  0192  5172 ;% 0E   RDM_M[TB]                 ; READ PTE FROM TB TO RDM
U 0F. 7300.C800.0364.0300.0470.1810  019D  5176 ;% 0F   NOP,STOP.CLOCK            ; STOP
                                     01A8  5180
                                     01A8  5181 6$:
U 03. 7700.3800.797F.FFFF.8470.1804  01A8  5182 ;% 03   LONLIT_[0D000000]         ; TB GROUP 1 ON   TB GROUP 0 OFF
                                     01B3  5186
                                     01B3  5187 BEGIN_CPU_DATA
                                     0002  5188
                                     0002  5189 RTEMP_DATA
                                     0001  5190
                          00000000   0001  5191          .LONG   ^X 0             ; R[TEMP0]
                          00000200   0005  5192          .LONG   ^X 200           ; R[TEMP1]
                                     0009  5193
                                     0009  5194 END_CPU_DATA
                                     01B3  5195
                                     01B3  5196 END_TEST_DATA
                                     01B3
                                     01B3       ;-----------------------------------------------------------------------
                                     01B3  5197
                                     01B3  5198 ENDTEST
```

```
                        .SBTTL  "TEST   047     TB INVALIDATE TEST 2
                5200 ;*******************************************************************************
                5201 ;++
                5202 ;
                5203 ;       TB INVALIDATE TEST 2
                5204 ;
                5205 ; FUNCTIONAL DESCRIPTION
                5206 ;       THIS IS A PARITIAL TEST OF THE TB INVALIDATE FUNCTION, WCTRL/CLRTB.VA.
                5207 ;       CHECKED HERE IS THAT THE VA GETS LOADED OF THE WB.
                5208 ;
                5209 ; TEST ALGORITHM
                5210 ;       1.      LOAD THE VA WITH 00000000.
                5211 ;       2.      EXECUTE THE FUNCTION WCTRL/CLRTB.VA WITH THE TEST PATTERN
                5212 ;                       ON THE WBUS.
                5213 ;       3.      READ THE VA BACK INTO THE RDM.
                5214 ;       4.      REPEAT 1. THRU 3. ONCE FOR EACH TEST DATA PATTERN LISTED BELOW.
                5215 ;
                5216 ; TEST PATTERNS
                5217 ;       THIS TEST TRIES TO LOAD THE FOLLOWING DATA PATTERNS INTO THE VA:
                5218 ;       1.      AAAAAAAA
                5219 ;       2.      55555555
                5220 ;       3.      33333333
                5221 ;       4.      0F0F0F0F
                5222 ;       5.      00FF00FF
                5223 ;       6.      0000FFFF
                5224 ;
                5225 ; LOGIC DESCRIPTION
                5226 ;       TESTED HERE IS LOGIC WHICH CONTROLS THE LOADING OF THE VA DURING
                5227 ;       A TB INVALIDATE.
                5228 ;
                5229 ; ASSUMPTIONS
                5230 ;
                5231 ; ERROR DESCRIPTION
                5232 ;       EXPECTED VA
                5233 ;       GOT VA
                5234 ;       INDEX IDENTIFING WHICH OF THE ABOVE TEST DATA PATTERNS WAS BEING
                5235 ;               USED WHEN THE ERROR OCCURRED.
                5236 ;
                5237 ;--
                5238 ;*******************************************************************************
                5239 ;///////////////////////////////////////////////////////////////////////////////
                5240
                5241            INITIALIZE
001B            5242            LOADDCS         1$,,0,<^X06>            ; LOAD THE MICRO CODE.
0022            5243            LOOP            I,1,6                   ; LOOP 6 TIMES.
0029            5244            SA01PAT         I,4$,,                  ; GET TEST DATA PATTERN AND LOAD
0031            5245            LOADREG         WDREG,4$,,LONG          ; IT INTO THE RDM D REGISTER.
0039            5246            ERRLOOP
003B            5247            FETCH           0                       ; EXECUTE THE MICRO INSTRUCTIONS.
0040            5248            BRSTCLK         <^X05>
0044            5249            CMPREG          WHREG,4$,,LONG          ; CHECK THE RESULT.
004D            5250            IFERROR         ,,<MIC>                 ; THE VA WAS INCORRECT AFTER
0053            5251                                                    ; BEING LOADED BY THE
0053            5252                                                    ; WCTRL/CLRTB.VA FUNCTION.
0053            5253            ENDLOOP         I
```

```
                                0056  5254          SKIP
                                005D  5255
                                005D  5256 BEGIN_TEST_DATA
                                005D
                                005D          ;-------------------------------------------------------------------------
                                005D  5257
                                005D  5258 1$:
U 00. 7700,C800,0364,0300,0470,1801  005D  5259 ;* 00    NOP
U 01. 7700,7800,7FFF,FFFF,8470,1802  0068  5263 ;* 01    LONLIT_[00000000]                    ; INITIALIZE THE VA TO 0.
U 02. 7700,4800,5BE4,03D4,4A70,1803  0073  5267 ;* 02    VA_R[LONLIT]
U 03. 5700,C800,0364,0300,5270,1804  007E  5271 ;* 03    WB_RDM,WCTRL/CLRTB.VA_WB             ; THIS SHOULD LOAD THE TEST
                                0089  5275                                               ; PATTERN INTO THE VA.
U 04. 6700,081B,0024,03D8,0470,1805  0089  5276 ;* 04    RDM_VA                               ; READ BACK THE VA.
U 05. 7300,4800,0364,0300,0470,1806  0094  5280 ;* 05    STOP.CLOCK
                                009F  5284
                         00000000  009F  5285 4$:      .LONG   ^X00000000
                                00A3  5286
                                00A3  5287 END_TEST_DATA
                                00A3
                                00A3          ;-------------------------------------------------------------------------
                                00A3  5288
                                00A3  5289 ENDTEST
```

```
                        .SBTTL  "TEST   048     TB GROUP 0 TAG PARITY GENERATOR/CHECKER TEST
0031   5291 ;*****************************************************************************
0031   5292 ;++
0031   5293 ;
0031   5294 ; FUNCTIONAL DESCRIPTION:
0031   5295 ;
0031   5296 ;       THIS TEST VERIFIES THAT THE TB GROUP 0 TAG PARITY GENERATOR/CHECKER
0031   5297 ;       CAN CREATE AS WELL AS DETECT BAD PARITY.  IN ADDITION THE "TB PARITY
0031   5298 ;       ERROR REGISTER" AND THE "MACHINE CHECK ERROR SUMMARY REGISTER" IN
0031   5299 ;       THE "UTR" GATE ARRAY ARE TESTED.
0031   5300 ;
0031   5301 ;
0031   5302 ; TEST ALGORITHM:
0031   5303 ;
0031   5304 ;       1.  CREATE A TEST LOOP TO SELECT TEST PATTERNS
0031   5305 ;       2.  LOAD LONLIT WITH DATA FOR VALID BIT
0031   5306 ;       3.  LOAD RDM WD REGISTER WITH TAG TEST PATTERN
0031   5307 ;       4.  EXECUTE DCS PROGRAM
0031   5308 ;               a.  SET MEMORY MANAGEMENT ON
0031   5309 ;               b.  DISABLE TB GROUP 1
0031   5310 ;               c.  LOAD VA REGISTER WITH TAG TEST PATTERN
0031   5311 ;               d.  LOAD LONGLIT REGISTER WITH SELECTED VALID BIT
0031   5312 ;               e.  WRITE TB GROUP 0 WHILE FORCING A PARITY ERROR
0031   5313 ;               f.  PERFORM A READ.  THIS SHOULD CAUSE A MACHINE CHECK
0031   5314 ;       5.  EXECUTE DCS PROGRAM
0031   5315 ;               a.  READ "TB GROUP PARITY ERROR REGISTER" TO RDM WH REG
0031   5316 ;       6.  TEST WH REGISTER FOR A PARITY ERROR IN GROUP 0 TAG FIELD
0031   5317 ;       7.  IF NO ERROR EXECUTE DCS PROGRAM
0031   5318 ;               a.  READ "MACHINE CHECK ERROR SUMMARY REG" TO RDM WH REG
0031   5319 ;               b.  WRITE ZEROS TO THE "MACHINE CHECK ERROR SUMMARY REG"
0031   5320 ;       8.  TEST WH REGISTER FOR A TB ERROR
0031   5321 ;       9.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0031   5322 ;
0031   5323 ;
0031   5324 ; TEST PATTERNS:
0031   5325 ;
0031   5326 ;       ITERATION       TAG ADDRESS     VALID BIT
0031   5327 ;           1            2AAA0000           1
0031   5328 ;           2            55550000           0
0031   5329 ;           3            33330000           0
0031   5330 ;           4            0F0F0000           0
0031   5331 ;           5            00FF0000           0
0031   5332 ;
0031   5333 ;
0031   5334 ; LOGIC DESCRIPTION:
0031   5335 ;
0031   5336 ;       THIS TEST ATTEMPTS TO TEST THE ABILITY OF THE TB GROUP 0 PARITY
0031   5337 ;       GENERATOR/CHECKER TO CREATE AND DETECT PARITY ERRORS.  THE "MISC/
0031   5338 ;       FORCE.TB" MICRO ORDER IS USED TO FORCE PARITY ERRORS IN THE TAG FIELD.
0031   5339 ;       THE ONLY WAY TO BE SURE THE PROPER PARITY ERROR OCCURED IS TO CHECK THE
0031   5340 ;       "TB GROUP PARITY ERROR REGISTER" AND THE "MACHINE CHECK ERROR SUMMARY
0031   5341 ;       REGISTER" IN THE "UTR" GATE ARRAY.  IF THE TEST FAILS YOU SHOULD FIRST
0031   5342 ;       REPLACE THE "UTR" GATE ARRAY.  IF THAT DOESN'T HELP THEN YOU'LL HAVE TO
0031   5343 ;       SCOPE THE PARITY GENERATOR/CHECKER LOGIC TO FIND THE BAD CHIP.
0031   5344 ;
```

```
                     0031  5345 ;
                     0031  5346 ; ASSUMPTIONS:
                     0031  5347 ;
                     0031  5348 ;       THIS TEST ASSUMES THE DATA INTEGRITY TESTS FOR THE TB HAVE BEEN RUN
                     0031  5349 ;       SUCCESSFULLY.
                     0031  5350 ;
                     0031  5351 ;
                     0031  5352 ; ERROR DESCRIPTION:
                     0031  5353 ;
                     0031  5354 ;       NOTE: THERE ARE TWO "IFERROR" STATEMENTS IN THIS TEST; SO CHECK THE
                     0031  5355 ;       FAILING PC TO SEE WHICH ERROR MESSAGE APPLIES.
                     0031  5356 ;
                     0031  5357 ;       FIRST IFERROR:
                     0031  5358 ;               EXPECTED CONTENTS OF "TB GROUP PARITY ERROR REGISTER"
                     0031  5359 ;               RECEIVED CONTENTS OF "TB GROUP PARITY ERROR REGISTER"
                     0031  5360 ;
                     0031  5361 ;       SECOND IFERROR:
                     0031  5362 ;               EXPECTED CONTENTS OF "MACHINE CHECK ERROR SUMMARY REGISTER"
                     0031  5363 ;               RECEIVED CONTENTS OF "MACHINE CHECK ERROR SUMMARY REGISTER"
                     0031  5364 ;
                     0031  5365 ;--
                     0031  5366 ;*******************************************************************************
                     0031  5367 ;///////////////////////////////////////////////////////////////////////////////
                     0031  5368
                     0031  5369         INITIALIZE                                      ;INIT THE CPU
                     0033  5370         EXPUTRAP                                        ;EXPECT MACHINE CHECK UTRAP
                     0035  5371         LOOP            I,1,5                           ;TEST LOOP POINT
                     003C  5372         LDFIELD         4$,I,L,5$,31,62,LON,            ;MODIFY LONLIT FIELD AND
                     004D  5373         LOADDCS         5$,,09,1                        ;LOAD INTO DCS ADDRESS 07
                     0054  5374         LOADREG         WDREG,1$,I,L                    ;LOAD TAG PATTERN INTO WD REG
                     005C  5375         ERRLOOP                                         ;ERROR LOOP POINT
                     005E  5376         FETCH           0                               ;START DCS PROGRAM
                     0063  5377         BRSTCLK         <^X0C>,INH                      ;END DCS PROGRAM
                     0067  5378         FETCH           <^X0D>                          ;START DCS PROGRAM
                     006C  5379         BRSTCLK         <^X10>,INH                      ;END DCS PROGRAM
                     0070  5380         CMPREGMSK       WHREG,2$,,3$,,L,                ;TEST TB PARITY ERROR
                     007C  5381         IFERROR         ,<<UTR>>,                       ;NO PARITY ERROR
                     0082  5382         FETCH           <^X11>                          ;START DCS PROGRAM
                     0087  5383         BRSTCLK         <^X18>,INH                      ;END DCS PROGRAM
                     008B  5384         CMPREGMSK       WHREG,2$,,6$,,L,                ;TEST FOR MACHINE CHECK
                     0097  5385         IFERROR         ,<<UTR>>,                       ;NO MACHINE CHECK
                     009D  5386         ENDLOOP         I                               ;END TEST LOOP
                     00A0  5387         SKIP                                            ;GO TO NEXT TEST
                     00A7  5388
                     00A7  5389 BEGIN_TEST_DATA
                     00A7
                     00A7               ;-------------------------------------------------------------------------
                     00A7  5390
2AAA0000             00A7  5391 1$:     .LONG   ^X2AAA0000                              ;TAG TEST PATTERN TABLE
55550000             00AB  5392         .LONG   ^X55550000
33330000             00AF  5393         .LONG   ^X33330000
0F0F0000             00B3  5394         .LONG   ^X0F0F0000
00FF0000             00B7  5395         .LONG   ^X00FF0000
                     00BB  5396
04000000             00BB  5397 2$:     .LONG   ^X04000000                              ;COMPARE DATA
```

```
                          0C000000  00BF  5398 3$:     .LONG   ^X0C000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                                    00C3  5399
                          00000100  00C3  5400 4$:     .LONG   ^X00000100                      ;VALID BIT TABLE
                          00000000  00C7  5401         .LONG   ^X00000000
                          00000000  00CB  5402         .LONG   ^X00000000
                          00000000  00CF  5403         .LONG   ^X00000000
                          00000000  00D3  5404         .LONG   ^X00000000
                                    00D7  5405
                                    00D7  5406 5$:
U 09, 7700,F800,7FFF,FFFF,8470,180A 00D7  5407 ;% 09   LONLIT_[0]                             ;MODIFY DCS ADDRESS 07
                                    00E2  5411
                          0F000000  00E2  5412 6$:     .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                                    00E6  5413
                                    00E6  5414 BEGIN_CPU_DATA
                                    0002  5415
                                    0002  5416 DCS_DATA
                                    0001  5417
U 00, 7700,C800,0364,0300,0470,1801 0001  5418 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  5422 ;% 01   MEMSCAR_R[TEMP0]                        ;SET MME ON
U 02, 7700,C800,5BE4,0314,6070,1803 0017  5426 ;% 02   MEMSCR_R[TEMP5]
U 03, 7700,C800,5BE4,030C,6870,1804 0022  5430 ;% 03   MEMSCAR_R[TEMP3]                        ;DISABLE TB GROUP 1
U 04, 7700,8800,5BE4,0310,6070,1805 002D  5434 ;% 04   MEMSCR_R[TEMP4]
U 05, 5700,0840,0364,0318,0470,1806 0038  5438 ;% 05   R[TEMP6]_RDM                           ; get tag test pattern
U 06, 7700,8800,5BE6,0318,0470,1807 0043  5442 ;% 06   D_R[TEMP6]                             ; D <-- tag test pattern
U 07, 7700,1868,DB10,0300,0470,1808 004E  5446 ;% 07   M[TEMP8]_D+ZLIT8[0]                    ; store in mtemp8
U 08, 7700,9800,DB12,0300,5360,1809 0059  5450 ;% 08   VA_D_D+ZLIT8[0] INVALIDATE TB         ; invalidate TB
U 09, 7700,F800,7FFF,FFFF,8470,180A 0064  5454 ;% 09   LONLIT_[0]                            ;VALID BIT TO LONGLIT REGISTER
U 0A, 7700,CF1B,5BE4,03D4,5070,180B 006F  5458 ;% 0A   TB_R[LONLIT],MISC/FORCE.TB            ;WRITE TB WITH PARITY ERROR
U 0B, 7700,C800,0364,0300,0510,180C 007A  5462 ;% 0B   READ.LONG                             ;PERFORM READ
U 0C, 7300,C800,0364,0300,0470,180D 0085  5466 ;% 0C   NOP,STOP.CLOCK                        ;STOP THE CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E 0090  5470 ;% 0D   NOP
U 0E, 7700,0800,5BE4,0308,6870,180F 009B  5474 ;% 0E   MEMSCAR_R[TEMP2]                       ;READ "TB GROUP PARITY ERROR
U 0F, 6700,C800,0364,0300,6470,1810 00A6  5478 ;% 0F   RDM_WB,WCTRL/MEMSCR                    ;REGISTER" TO RDM WH REGISTER
U 10, 7300,4800,0364,0300,0470,1811 00B1  5482 ;% 10   NOP,STOP.CLOCK                        ;STOP THE CPU CLOCK
U 11, 7700,4800,0364,0300,0470,1812 00BC  5486 ;% 11   NOP
U 12, 7700,8800,5BE4,0304,6870,1813 00C7  5490 ;% 12   MEMSCAR_R[TEMP1]                       ;READ "MACHINE CHECK ERROR
U 13, 6700,4800,0364,0300,6470,1814 00D2  5494 ;% 13   RDM_WB,WCTRL/MEMSCR                    ;SUMMARY REG" TO RDM WH REG
U 14, 7700,4800,5BE4,03D8,6070,1815 00DD  5498 ;% 14   MEMSCR_R[ZERO]                        ;CLEAR MACHINE CHECK REG
U 15, 7700,8800,5BE6,0318,0470,1816 00E8  5502 ;% 15   D_R[TEMP6]                             ; D <-- get addr. of bad parity
U 16, 7700,9868,DB10,0300,0470,1817 00F3  5506 ;% 16   M[TEMP8]_D+ZLIT8[0]                    ; mtemp8 <-- D
U 17, 7700,9800,DB12,0300,5360,1818 00FE  5510 ;% 17   VA_D_D+ZLIT8[0] INVALIDATE TB         ; clear out parity error
U 18, 7300,C800,0364,0300,0470,1819 0109  5514 ;% 18   NOP,STOP.CLOCK                        ;STOP THE CPU CLOCK
                                    0114  5518
                                    0114  5519 RTEMP_DATA
                                    0001  5520
                          00000000  0001  5521         .LONG   ^X00000000                      ;PHY/VIR REGISTER ADDRESS
                          08000000  0005  5522         .LONG   ^X08000000                      ;MACHINE CHECK REG ADDRESS
                          0D000000  0009  5523         .LONG   ^X0D000000                      ;TB PARITY ERROR REG ADDRESS
                          03000000  000D  5524         .LONG   ^X03000000                      ;TB GROUP DISABLE REG ADD
                          0A000000  0011  5525         .LONG   ^X0A000000                      ;DISABLE GROUP 1
                          01000000  0015  5526         .LONG   ^X01000000                      ;SET MME ON
                          00000000  0019  5527         .LONG   ^X00000000                      ;storage location for va
                                    001D  5528
                                    001D  5529 END_CPU_DATA
                                    00E6  5530
```

```
00E6  5531 END_TEST_DATA
00E6
00E6       ;-----------------------------------------------------------------------
00E6  5532
00E6  5533 ENDTEST
```

```
                        .SBTTL  "TEST   049     TB GROUP 1 TAG PARITY GENERATOR/CHECKER TEST
0031  5535 ;*******************************************************************************
0031  5536 ;++
0031  5537 ;
0031  5538 ; FUNCTIONAL DESCRIPTION:
0031  5539 ;
0031  5540 ;       THIS TEST VERIFIES THAT THE TB GROUP 1 TAG PARITY GENERATOR/CHECKER
0031  5541 ;       CAN CREATE AS WELL AS DETECT BAD PARITY.  IN ADDITION THE "TB PARITY
0031  5542 ;       ERROR REGISTER" AND THE "MACHINE CHECK ERROR SUMMARY REGISTER" IN
0031  5543 ;       THE "UTR" GATE ARRAY ARE TESTED.
0031  5544 ;
0031  5545 ;
0031  5546 ; TEST ALGORITHM:
0031  5547 ;
0031  5548 ;       1.  CREATE A TEST LOOP TO SELECT TEST PATTERNS
0031  5549 ;       2.  LOAD LONLIT WITH DATA FOR VALID BIT
0031  5550 ;       3.  LOAD RDM WD REGISTER WITH TAG TEST PATTERN
0031  5551 ;       4.  EXECUTE DCS PROGRAM
0031  5552 ;               a.  SET MEMORY MANAGEMENT ON
0031  5553 ;               b.  DISABLE TB GROUP 0
0031  5554 ;               c.  LOAD VA REGISTER WITH TAG TEST PATTERN
0031  5555 ;               d.  LOAD LONGLIT REGISTER WITH SELECTED VALID BIT
0031  5556 ;               e.  WRITE TB GROUP 1 WHILE FORCING A PARITY ERROR
0031  5557 ;               f.  PERFORM A READ.  THIS SHOULD CAUSE A MACHINE CHECK
0031  5558 ;       5.  EXECUTE DCS PROGRAM
0031  5559 ;               a.  READ "TB GROUP PARITY ERROR REGISTER" TO RDM WH REG
0031  5560 ;       6.  TEST WH REGISTER FOR A PARITY ERROR IN GROUP 1 TAG FIELD
0031  5561 ;       7.  IF NO ERROR EXECUTE DCS PROGRAM
0031  5562 ;               a.  READ "MACHINE CHECK ERROR SUMMARY REG" TO RDM WH REG
0031  5563 ;               b.  WRITE ZEROS TO THE "MACHINE CHECK ERROR SUMMARY REG"
0031  5564 ;       8.  TEST WH REGISTER FOR A TB ERROR
0031  5565 ;       9.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0031  5566 ;
0031  5567 ;
0031  5568 ; TEST PATTERNS:
0031  5569 ;
0031  5570 ;       ITERATION       TAG ADDRESS     VALID BIT
0031  5571 ;           1            2AAA0000           1
0031  5572 ;           2            55550000           0
0031  5573 ;           3            33330000           0
0031  5574 ;           4            0F0F0000           0
0031  5575 ;           5            00FF0000           0
0031  5576 ;
0031  5577 ;
0031  5578 ; LOGIC DESCRIPTION:
0031  5579 ;
0031  5580 ;       THIS TEST ATTEMPTS TO TEST THE ABILITY OF THE TB GROUP 1 PARITY
0031  5581 ;       GENERATOR/CHECKER TO CREATE AND DETECT PARITY ERRORS.  THE "MISC/
0031  5582 ;       FORCE.TB" MICRO ORDER IS USED TO FORCE PARITY ERRORS IN THE TAG FIELD.
0031  5583 ;       THE ONLY WAY TO BE SURE THE PROPER PARITY ERROR OCCURED IS TO CHECK THE
0031  5584 ;       "TB GROUP PARITY ERROR REGISTER" AND THE "MACHINE CHECK ERROR SUMMARY
0031  5585 ;       REGISTER" IN THE "UTR" GATE ARRAY.  IF THE TEST FAILS YOU SHOULD FIRST
0031  5586 ;       REPLACE THE "UTR" GATE ARRAY.  IF THAT DOESN'T HELP THEN YOU'LL HAVE TO
0031  5587 ;       SCOPE THE PARITY GENERATOR/CHECKER LOGIC TO FIND THE BAD CHIP.
0031  5588 ;
```

```
                     0031  5589 ;
                     0031  5590 ; ASSUMPTIONS:
                     0031  5591 ;
                     0031  5592 ;       THIS TEST ASSUMES THE DATA INTEGRITY TESTS FOR THE TB HAVE BEEN RUN
                     0031  5593 ;       SUCCESSFULLY.
                     0031  5594 ;
                     0031  5595 ;
                     0031  5596 ; ERROR DESCRIPTION:
                     0031  5597 ;
                     0031  5598 ;       NOTE: THERE ARE TWO "IFERROR" STATEMENTS IN THIS TEST; SO CHECK THE
                     0031  5599 ;       FAILING PC TO SEE WHICH ERROR MESSAGE APPLIES.
                     0031  5600 ;
                     0031  5601 ;       FIRST IFERROR:
                     0031  5602 ;               EXPECTED CONTENTS OF "TB GROUP PARITY ERROR REGISTER"
                     0031  5603 ;               RECEIVED CONTENTS OF "TB GROUP PARITY ERROR REGISTER"
                     0031  5604 ;
                     0031  5605 ;       SECOND IFERROR:
                     0031  5606 ;               EXPECTED CONTENTS OF "MACHINE CHECK ERROR SUMMARY REGISTER"
                     0031  5607 ;               RECEIVED CONTENTS OF "MACHINE CHECK ERROR SUMMARY REGISTER"
                     0031  5608 ;
                     0031  5609 ;--
                     0031  5610 ;*******************************************************************************
                     0031  5611 ;///////////////////////////////////////////////////////////////////////////////
                     0031  5612
                     0031  5613         INITIALIZE                                      ;INIT THE CPU
                     0033  5614         EXPUTRAP                                        ;EXPECT MACHINE CHECK UTRAP
                     0035  5615         LOOP            I,1,5                           ;TEST LOOP POINT
                     003C  5616         LDFIELD         4$,I,L,5$,31,62,LON,            ;MODIFY LONLIT FIELD AND
                     004D  5617         LOADDCS         5$,,09,1                        ;LOAD INTO DCS ADDRESS 07
                     0054  5618         LOADREG         WDREG,1$,I,L                    ;LOAD TAG PATTERN INTO WD REG
                     005C  5619         ERRLOOP                                         ;ERROR LOOP POINT
                     005E  5620         FETCH           0                               ;START DCS PROGRAM
                     0063  5621         BRSTCLK         <^X0C>,INH                      ;END DCS PROGRAM
                     0067  5622         FETCH           <^X0D>                          ;START DCS PROGRAM
                     006C  5623         BRSTCLK         <^X10>,INH                      ;END DCS PROGRAM
                     0070  5624         CMPREGMSK       WHREG,2$,,3$,,L,                ;TEST TB PARITY ERROR
                     007C  5625         IFERROR         ,<<UTR>>,                       ;NO PARITY ERROR
                     0082  5626         FETCH           <^X11>                          ;START DCS PROGRAM
                     0087  5627         BRSTCLK         <^X18>,INH                      ;END DCS PROGRAM
                     008B  5628         CMPREGMSK       WHREG,7$,,6$,,L,                ;TEST FOR MACHINE CHECK
                     0097  5629         IFERROR         ,<<UTR>>,                       ;NO MACHINE CHECK
                     009D  5630         ENDLOOP         I                               ;END TEST LOOP
                     00A0  5631         SKIP                                            ;GO TO NEXT TEST
                     00A7  5632
                     00A7  5633 BEGIN_TEST_DATA
                     00A7
                     00A7               ;-----------------------------------------------------------------------
                     00A7  5634
2AAA0000             00A7  5635 1$:      .LONG   ^X2AAA0000                              ;TAG TEST PATTERN TABLE
55550000             00AB  5636          .LONG   ^X55550000
33330000             00AF  5637          .LONG   ^X33330000
0F0F0000             00B3  5638          .LONG   ^X0F0F0000
00FF0000             00B7  5639          .LONG   ^X00FF0000
                     00BB  5640
08000000             00BB  5641 2$:      .LONG   ^X08000000                              ;COMPARE DATA
```

```
                    0C000000  00BF  5642 3$:      .LONG   ^X0C000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                              00C3  5643
                    00000100  00C3  5644 4$:      .LONG   ^X00000100                      ;VALID BIT TABLE
                    00000000  00C7  5645          .LONG   ^X00000000
                    00000000  00CB  5646          .LONG   ^X00000000
                    00000000  00CF  5647          .LONG   ^X00000000
                    00000000  00D3  5648          .LONG   ^X00000000
                              00D7  5649
                              00D7  5650 5$:
U 09, 7700,F800,7FFF,FFFF,8470,180A   00D7  5651 ;% 09   LONLIT_[0]                       ;MODIFY DCS ADDRESS 07
                              00E2  5655
                    0F000000  00E2  5656 6$:      .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                    04000000  00E6  5657 7$:      .LONG   ^X04000000                      ;COMPARE DATA
                              00EA  5658
                              00EA  5659 BEGIN_CPU_DATA
                              0002  5660
                              0002  5661 DCS_DATA
                              0001  5662
U 00, 7700,C800,0364,0300,0470,1801   0001  5663 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C  5667 ;% 01   MEMSCAR_R[TEMP0]                 ;SET MME ON
U 02, 7700,C800,5BE4,0314,6070,1803   0017  5671 ;% 02   MEMSCR_R[TEMP5]
U 03, 7700,C800,5BE4,030C,6870,1804   0022  5675 ;% 03   MEMSCAR_R[TEMP3]                ;DISABLE TB GROUP 0
U 04, 7700,8800,5BE4,0310,6070,1805   002D  5679 ;% 04   MEMSCR_R[TEMP4]
U 05, 5700,0840,0364,0318,0470,1806   0038  5683 ;% 05   R[TEMP6]_RDM                    ; get tag test pattern
U 06, 7700,8800,5BE6,0318,0470,1807   0043  5687 ;% 06   D_R[TEMP6]                      ; D <-- tag test pattern
U 07, 7700,1868,DB10,0300,0470,1808   004E  5691 ;% 07   M[TEMP8]_D+ZLIT8[0]            ; store in mtemp8
U 08, 7700,9800,DB12,0300,5360,1809   0059  5695 ;% 08   VA_D_D+ZLIT8[0] INVALIDATE TB  ; invalidate TB
U 09, 7700,F800,7FFF,FFFF,8470,180A   0064  5699 ;% 09   LONLIT_[0]                      ;VALID BIT TO LONGLIT REGISTER
U 0A, 7700,CF1B,5BE4,03D4,5070,180B   006F  5703 ;% 0A   TB_R[LONLIT],MISC/FORCE.TB      ;WRITE TB WITH PARITY ERROR
U 0B, 7700,C800,0364,0300,0510,180C   007A  5707 ;% 0B   READ.LONG                       ;PERFORM READ
U 0C, 7300,C800,0364,0300,0470,180D   0085  5711 ;% 0C   NOP,STOP.CLOCK                 ;STOP THE CPU CLOCK
U 0D, 7700,C800,0364,0300,0470,180E   0090  5715 ;% 0D   NOP
U 0E, 7700,0800,5BE4,0308,6870,180F   009B  5719 ;% 0E   MEMSCAR_R[TEMP2]                ;READ "TB GROUP PARITY ERROR
U 0F, 6700,C800,0364,0300,6470,1810   00A6  5723 ;% 0F   RDM_WB,WCTRL/MEMSCR             ;REGISTER" TO RDM WH REGISTER
U 10, 7300,4800,0364,0300,0470,1811   00B1  5727 ;% 10   NOP,STOP.CLOCK
U 11, 7700,4800,0364,0300,0470,1812   00BC  5731 ;% 11   NOP
U 12, 7700,8800,5BE4,0304,6870,1813   00C7  5735 ;% 12   MEMSCAR_R[TEMP1]                ;READ "MACHINE CHECK ERROR
U 13, 6700,4800,0364,0300,6470,1814   00D2  5739 ;% 13   RDM_WB,WCTRL/MEMSCR             ;SUMMARY REG" TO RDM WH REG
U 14, 7700,4800,5BE4,03D8,6070,1815   00DD  5743 ;% 14   MEMSCR_R[ZERO]                  ;CLEAR MACHINE CHECK REG
U 15, 7700,8800,5BE6,0318,0470,1816   00E8  5747 ;% 15   D_R[TEMP6]                      ; D <-- get addr. of bad parity
U 16, 7700,9868,DB10,0300,0470,1817   00F3  5751 ;% 16   M[TEMP8]_D+ZLIT8[0]            ; mtemp8 <-- D
U 17, 7700,9800,DB12,0300,5360,1818   00FE  5755 ;% 17   VA_D_D+ZLIT8[0] INVALIDATE TB  ; clear out parity error
U 18, 7300,C800,0364,0300,0470,1819   0109  5759 ;% 18   NOP,STOP.CLOCK                 ;STOP THE CPU CLOCK
                              0114  5763
                              0114  5764 RTEMP_DATA
                              0001  5765
                    00000000  0001  5766          .LONG   ^X00000000                      ;PHY/VIR REGISTER ADDRESS
                    08000000  0005  5767          .LONG   ^X08000000                      ;MACHINE CHECK REG ADDRESS
                    0D000000  0009  5768          .LONG   ^X0D000000                      ;TB PARITY ERROR REG ADDRESS
                    03000000  000D  5769          .LONG   ^X03000000                      ;TB GROUP DISABLE REG ADD
                    0D000000  0011  5770          .LONG   ^X0D000000                      ;DISABLE GROUP 0
                    01000000  0015  5771          .LONG   ^X01000000                      ;SET MME ON
                    00000000  0019  5772          .LONG   ^X00000000                      ;storage location for va
                              001D  5773
                              001D  5774 END_CPU_DATA
```

```
                00EA  5775
                00EA  5776 END_TEST_DATA
                00EA
                00EA        ;-----------------------------------------------------------------------------
                00EA  5777
                00EA  5778 ENDTEST
```

```
                        .SBTTL  "TEST   04A     TEST TB TAG PARITY ERROR MACHINE CHECK
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST VERIFIES THAT A TB TAG PARITY ERROR PRODUCES A MACHINE CHECK
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  MODIFY DCS U-CODE TO SELECT THE PROPER TB GROUP
                ;       2.  EXECUTE DCS PROGRAM
                ;               a.  SET MEMORY MANAGEMENT ON
                ;               b.  DISABLE 1 TB GROUP
                ;               c.  LOAD AN ADDRESS OF ZERO INTO THE VA REGISTER
                ;               d.  CLEAR THIS LOCATION IN THE TB
                ;               e.  WRITE THE TB WITH ZERO WHILE FORCING A PARITY ERROR
                ;               f.  PERFORM A READ OPERATION WHICH SHOULD CAUSE A MACHINE CHECK
                ;       3.  TEST THE CS ADDRESS FOR 28 (MACHINE CHECK)
                ;       4.  IF NO ERROR EXECUTE MORE DCS CODE
                ;               a.  CLEAR THE TB PARITY ERROR
                ;               b.  CLEAR THE "MACHINE CHECK ERROR SUMMARY REGISTER"
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       VA = 00000000
                ;       TB DATA = 00000000
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       THIS TEST INSURES THAT CPU PERFORMS A MACHINE CHECK MICRO TRAP
                ;       WHEN IT ENCOUNTERS A PARITY ERROR IN THE TB.  THE ONLY GATE ARRAY
                ;       THAT SHOULD AFFECT THIS TEST IS "UTR".
                ;
                ;
                ; ASSUMPTIONS:
                ;
                ;       THIS TEST ASSUMES THE TB DATA INTEGRITY TESTS HAVE RUN SUCCESSFULLY
                ;
                ; ERROR DESCRIPTION:
                ;
                ;       EXPECTED CONTENTS OF CSTRP REGISTER
                ;       RECEIVED CONTENTS OF CSTRP REGISTER
                ;
                ;--
                ;*****************************************************************************

                        SUBTEST                                         ;SUBTEST #1

                ;/////////////////////////////////////////////////////////////////////////////
                ;*
```

```
                     002E  5832 ;SUBTEST #1
                     002E  5833 ;
                     002E  5834 ;       GROUP 0 TAG PARITY ERROR
                     002E  5835 ;-
                     002E  5836         INITIALIZE                              ;INIT THE CPU
                     0030  5837         LOADDCS         3$,,4,1                 ;MODIFY U-CODE FOR THIS TEST
                     0037  5838         EXPUTRAP                                ;EXPECT A MACHINE CHECK
                     0039  5839         ERRLOOP                                 ;ERROR LOOP POINT
                     003B  5840         FETCH           0                       ;START DCS PROGRAM
                     0040  5841         BRSTCLK         <^X0A>                  ;END DCS PROGRAM
                     0044  5842         CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST FOR MACHINE CHECK
                     0050  5843         IFERROR         ,<<UTR>>,               ;DID NOT MACHINE CHECK
                     0056  5844         FETCH           <^X0B>                  ;START DCS PROGRAM
                     005B  5845         BRSTCLK         <^X10>                  ;END DCS PROGRAM
                     005F  5846
                     005F  5847         SUBTEST                                 ;SUBTEST #2
                     0062
                     0062       ;////////////////////////////////////////////////////////////////////////////
                     0062  5848 ;+
                     0062  5849 ;SUBTEST #2
                     0062  5850 ;
                     0062  5851 ;       GROUP 1 TAG PARITY ERROR
                     0062  5852 ;-
                     0062  5853         INITIALIZE                              ;INIT THE CPU
                     0064  5854         LOADDCS         4$,,4,1                 ;MODIFY U-CODE FOR THIS TEST
                     006B  5855         EXPUTRAP                                ;EXPECT MACHINE CHECK
                     006D  5856         ERRLOOP                                 ;ERROR LOOP POINT
                     006F  5857         FETCH           0                       ;START DCS PROGRAM
                     0074  5858         BRSTCLK         <^X0A>                  ;END DCS PROGRAM
                     0078  5859         CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST FOR MACHINE CHECK
                     0084  5860         IFERROR         ,<<UTR>>,               ;NO MACHINE CHECK
                     008A  5861         FETCH           <^X0B>                  ;START DCS PROGRAM
                     008F  5862         BRSTCLK         <^X10>                  ;END DCS PROGRAM
                     0093  5863         SKIP                                    ;GO TO NEXT TEST
                     009A  5864
                     009A  5865 BEGIN_TEST_DATA
                     009A
                     009A       ;-------------------------------------------------------------------------
                     009A  5866
                0028 009A  5867 1$:     .WORD   ^X0028                          ;ADDRESS OF MACHINE CHECK
                3FFF 009C  5868 2$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                     009E  5869
                     009E  5870 3$:
U 04, 7700,C800,5BE4,030C,6070,1805  009E  5871 ;% 04   MEMSCR_R[TEMP3]          ;U-CODE SUBTEST #1
                     00A9  5875
                     00A9  5876 4$:
U 04, 7700,8800,5BE4,0310,6070,1805  00A9  5877 ;% 04   MEMSCR_R[TEMP4]          ;U-CODE SUBTEST #2
                     00B4  5881
                     00B4  5882 BEGIN_CPU_DATA
                     0002  5883
                     0002  5884 DCS_DATA
                     0001  5885
U 00, 7700,C800,0364,0300,0470,1801  0001  5886 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  5890 ;% 01   MEMSCAR_R[TEMP0]          ;SET MME ON
U 02, 7700,8800,5BE4,0304,6070,1803  0017  5894 ;% 02   MEMSCR_R[TEMP1]
```

```
U 03. 7700.8800.5BE4.0308.6870.1804  0022  5898 ;% 03   MEMSCAR_R[TEMP2]                          ;DISABLE 1 TB GROUP
U 04. 7700.C800.5BE4.030C.6070.1805  002D  5902 ;% 04   MEMSCR_R[TEMP3]
U 05. 7700.0800.5BE6.03D8.0470.1806  0038  5906 ;% 05   D_R[ZERO]                                ; D <-- 00000000
U 06. 7700.5800.DB13.0300.4A70.1807  0043  5910 ;% 06   VA_Q_D_D+ZLIT8[0]                        ; VA <-- address <00000000>
U 07. 7700.1800.DB12.0300.5360.1808  004E  5914 ;% 07   VA_D_D+ZLIT8[0] INVALIDATE TB          ; flush the TB at VA
U 08. 7700.4F1B.5BE4.03D8.5070.1809  0059  5918 ;% 08   TB_R[ZERO],MISC/FORCE.TB               ;WRITE A PARITY ERROR
U 09. 7700.C800.0364.0300.0510.180A  0064  5922 ;% 09   READ.LONG                              ;CAUSE A MACHINE CHECK
U 0A. 7300.C800.0364.0300.0470.180B  006F  5926 ;% 0A   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
U 0B. 7700.4800.0364.0300.0470.180C  007A  5930 ;% 0B   NOP
U 0C. 7700.5800.DB13.0300.4A70.180D  0085  5934 ;% 0C   VA_Q_D_D+ZLIT8[0]                        ; set the va to zero
U 0D. 7700.1800.DB12.0300.5360.180E  0090  5938 ;% 0D   VA_D_D+ZLIT8[0] INVALIDATE TB          ; clear out parity error
U 0E. 7700.4800.5BE4.0314.6870.180F  009B  5942 ;% 0E   MEMSCAR_R[TEMP5]                       ;CLEAR THE MACHINE CHECK
U 0F. 7700.4800.5BE4.03D8.6070.1810  00A6  5946 ;% 0F   MEMSCR_R[ZERO]
U 10. 7300.4800.0364.0300.0470.1811  00B1  5950 ;% 10   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                     00BC  5954
                                     00BC  5955 RTEMP_DATA
                                     0001  5956
                           00000000  0001  5957          .LONG   ^X00000000                  ;PHY/VIR REG ADDRESS
                           01000000  0005  5958          .LONG   ^X01000000                  ;SET MME ON
                           03000000  0009  5959          .LONG   ^X03000000                  ;TB GROUP DISABLE REG ADD
                           0A000000  000D  5960          .LONG   ^X0A000000                  ;DISABLE GROUP 1
                           0D000000  0011  5961          .LONG   ^X0D000000                  ;DISABLE GROUP 0
                           08000000  0015  5962          .LONG   ^X08000000                  ;MACHINE CHECK REGISTER ADD
                                     0019  5963
                                     0019  5964 END_CPU_DATA
                                     00B4  5965
                                     00B4  5966 END_TEST_DATA
                                     00B4
                                     00B4          ;-----------------------------------------------------------------------------
                                     00B4  5967
                                     00B4  5968 ENDTEST
```

```
                        .SBTTL  "TEST   04B     TEST TB GROUP 0 DATA PARITY GENERATOR/CHECKER
                5970 ;*******************************************************************************
                5971 ;++
                5972 ;
                5973 ; FUNCTIONAL DESCRIPTION:
                5974 ;
                5975 ;       THIS TEST VERIFIES THE ABILITY OF THE TB GROUP 0 DATA PARITY LOGIC
                5976 ;       TO GENERATE AS WELL AS DETECT PARITY ERRORS.
                5977 ;
                5978 ; TEST ALGORITHM:
                5979 ;
                5980 ;       1.  CREATE A TEST LOOP OF 6 TO SELECT TEST PTE'S
                5981 ;       2.  PUT A TEST PATTERN INTO THE RDM WD REGISTER
                5982 ;       3.  EXECUTE DCS PROGRAM
                5983 ;               a.  SET MEMORY MANAGEMENT ON
                5984 ;               b.  DISABLE GROUP 1
                5985 ;               c.  LOAD AN ADDRESS OF ZERO INTO THE VA REGISTER
                5986 ;               d.  CLEAR THE TB
                5987 ;               e.  WRITE THE TEST PATTERN WHILE FORCING A PARITY ERROR
                5988 ;               f.  PERFORM A READ OPERATION
                5989 ;       4.  EXECUTE MORE DCS U-CODE
                5990 ;               a.  READ THE "TB GROUP PARITY ERROR REGISTER" TO THE WH REG
                5991 ;       5.  TEST THE WH REGISTER FOR GROUP 0 PARITY ERROR
                5992 ;       6.  IF NO ERROR EXECUTE MORE DCS U-CODE
                5993 ;               a.  READ THE "MACHINE CHECK ERROR SUMMARY REG" TO THE WH REG
                5994 ;               b.  CLEAR THE TB PARITY ERROR
                5995 ;               c.  CLEAR THE "MACHINE CHECK ERROR SUMMARY REG"
                5996 ;       7.  TEST THE WH REGISTER FOR A TB ERROR
                5997 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
                5998 ;
                5999 ;
                6000 ; TEST PATTERNS:
                6001 ;
                6002 ;       ITERATION       TEST PATTERN
                6003 ;           1             00AAABA8
                6004 ;           2             00555550
                6005 ;           3             00333330
                6006 ;           4             000F0F08
                6007 ;           5             0000FF00
                6008 ;           6             00000100
                6009 ;
                6010 ;
                6011 ; LOGIC DESCRIPTION:
                6012 ;
                6013 ;       THIS TEST FORCES PARITY ERRORS USING THE "MISC/FORCE.TB" MICRO ORDER.
                6014 ;       THE "TB GROUP PARITY ERROR REGISTER" AND THE "MACHINE CHECK ERROR
                6015 ;       SUMMARY REGISTER" ARE TESTED TO SEE IF THE FAILURE IS DETECTED
                6016 ;       PROPERLY.  IF REPLACING THE "UTR" GATE ARRAY DOES NOT HELP THEN YOU'LL
                6017 ;       HAVE TO SCOPE THE TB DATA PARITY GENERATOR/CHECKER TO ISOLATE THE FAULT.
                6018 ;
                6019 ;
                6020 ; ASSUMPTIONS:
                6021 ;
                6022 ;       THIS TEST ASSUMES THE TB DATA INTEGRITY TESTS HAVE RUN SUCCESSFULLY
                6023 ;
```

```
                     0032  6024 ;
                     0032  6025 ; ERROR DESCRIPTION:
                     0032  6026 ;
                     0032  6027 ;       NOTE: THERE ARE TWO IFERROR STATEMENTS IN THIS TEST.  SO CHECK THE PC
                     0032  6028 ;       TO DETERMINE WHICH ONE HAS FAILED.
                     0032  6029 ;
                     0032  6030 ;       FIRST IFERROR:
                     0032  6031 ;       EXPECTED "TB GROUP PARITY ERROR REGISTER" CONTENTS
                     0032  6032 ;       RECEIVED "TB GROUP PARITY ERROR REGISTER" CONTENTS
                     0032  6033 ;       LOOP COUNT I
                     0032  6034 ;
                     0032  6035 ;       SECOND IFERROR:
                     0032  6036 ;       EXPECTED "MACHINE CHECK ERROR SUMMARY REGISTER" CONTENTS
                     0032  6037 ;       RECEIVED "MACHINE CHECK ERROR SUMMARY REGISTER" CONTENTS
                     0032  6038 ;       LOOP COUNT I
                     0032  6039 ;
                     0032  6040 ;--
                     0032  6041 ;*******************************************************************************
                     0032  6042 ;///////////////////////////////////////////////////////////////////////////////
                     0032  6043
                     0032  6044         INITIALIZE                                  ;INIT THE CPU
                     0034  6045         EXPUTRAP                                    ;EXPECT A MACHINE CHECK
                     0036  6046         LOOP            I,1,6                       ;TEST LOOP OF 6
                     003D  6047         LOADREG         WDREG,1$,I,L                ;LOAD TEST PATTERN INTO WD REG
                     0045  6048         ERRLOOP                                     ;ERROR LOOP POINT
                     0047  6049         FETCH           0                           ;START DCS PROGRAM
                     004C  6050         BRSTCLK         <^X0A>,INH                  ;END DCS PROGRAM
                     0050  6051         FETCH           <^X0B>                      ;START DCS PROGRAM
                     0055  6052         BRSTCLK         <^X0E>,INH                  ;END DCS PROGRAM
                     0059  6053         CMPREGMSK       WHREG,2$,,3$,,L,            ;TEST TB ERROR REGISTER
                     0065  6054         IFERROR         ,<<UTR>>,                   ;NO PARITY ERROR
                     006B  6055         FETCH           <^X0F>                      ;START DCS PROGRAM
                     0070  6056         BRSTCLK         <^X15>                      ;END DCS PROGRAM
                     0074  6057         CMPREGMSK       WHREG,4$,,5$,,L,            ;TEST FOR MACHINE CHECK
                     0080  6058         IFERROR         ,<<UTR>>,                   ;NO MACHINE CHECK
                     0086  6059         ENDLOOP         I                           ;END TEST LOOP
                     0089  6060         SKIP                                        ;GO TO NEXT TEST
                     0090  6061
                     0090  6062 BEGIN_TEST_DATA
                     0090
                     0090         ;-------------------------------------------------------------------------
                     0090  6063
00AAABA8             0090  6064 1$:      .LONG    ^X00AAABA8                        ;TEST PATTERN TABLE
00555550             0094  6065          .LONG    ^X00555550
00333330             0098  6066          .LONG    ^X00333330
000F0F08             009C  6067          .LONG    ^X000F0F08
0000FF00             00A0  6068          .LONG    ^X0000FF00
00000100             00A4  6069          .LONG    ^X00000100
                     00A8  6070
01000000             00A8  6071 2$:      .LONG    ^X01000000                        ;TB ERROR REG CONTENTS
03000000             00AC  6072 3$:      .LONG    ^X03000000                        ;MASK FOR CMPREGMSK PSEUDO OP
04000000             00B0  6073 4$:      .LONG    ^X04000000                        ;MACHINE CHECK REG CONTENTS
0F000000             00B4  6074 5$:      .LONG    ^X0F000000                        ;MASK FOR CMPREGMSK PSEUDO OP
                     00B8  6075
                     00B8  6076 BEGIN_CPU_DATA
```

```
                                0002  6077
                                0002  6078 DCS_DATA
                                0001  6079
U 00. 7700,C800,0364,0300,0470,1801   0001  6080 ;% 00    NOP
U 01. 7700,C800,5BE4,0300,6870,1802   000C  6084 ;% 01    MEMSCAR_R[TEMP0]                    ;SET MME ON
U 02. 7700,8800,5BE4,0304,6070,1803   0017  6088 ;% 02    MEMSCR_R[TEMP1]
U 03. 7700,8800,5BE4,0308,6870,1804   0022  6092 ;% 03    MEMSCAR_R[TEMP2]                    ;DISABLE GROUP 1
U 04. 7700,C800,5BE4,030C,6070,1805   002D  6096 ;% 04    MEMSCR_R[TEMP3]
U 05. 7700,0800,5BE6,03D8,0470,1806   0038  6100 ;% 05    D_R[ZERO]                           ; D <-- 00000000
U 06. 7700,5800,DB13,0300,4A70,1807   0043  6104 ;% 06    VA_Q_D_D+ZLIT8[0]                   ; VA <-- address <00000000>
U 07. 7700,1800,DB12,0300,5360,1808   004E  6108 ;% 07    VA_D_D+ZLIT8[0] INVALIDATE TB       ; Flush the TB at VA
U 08. 5700,4F1B,0364,0300,5070,1809   0059  6112 ;% 08    TB_RDM,MISC/FORCE.TB                ;WRITE TEST PATTERN WITH ERROR
U 09. 7700,C800,0364,0300,0510,180A   0064  6116 ;% 09    READ.LONG                           ;CAUSE MACHINE CHECK
U 0A. 7300,C800,0364,0300,0470,180B   006F  6120 ;% 0A    NOP,STOP.CLOCK                      ;STOP CPU CLOCK
U 0B. 7700,4800,0364,0300,0470,180C   007A  6124 ;% 0B    NOP
U 0C. 7700,8800,5BE4,0310,6870,180D   0085  6128 ;% 0C    MEMSCAR_R[TEMP4]                    ;READ TB GROUP PARITY ERROR REG
U 0D. 6700,C800,0364,0300,6470,180E   0090  6132 ;% 0D    RDM_WB,WCTRL/MEMSCR
U 0E. 7300,4800,0364,0300,0470,180F   009B  6136 ;% 0E    NOP,STOP.CLOCK                      ;STOP CPU CLOCK
U 0F. 7700,C800,0364,0300,0470,1810   00A6  6140 ;% 0F    NOP
U 10. 7700,4800,5BE4,0314,6870,1811   00B1  6144 ;% 10    MEMSCAR_R[TEMP5]                    ;READ MACHINE CHECK SUMMARY REG
U 11. 6700,4800,0364,0300,6470,1812   00BC  6148 ;% 11    RDM_WB,WCTRL/MEMSCR
U 12. 7700,5800,DB13,0300,4A70,1813   00C7  6152 ;% 12    VA_Q_D_D+ZLIT8[0]                   ; VA <-- address (00000000)
U 13. 7700,9800,DB12,0300,5360,1814   00D2  6156 ;% 13    VA_D_D+ZLIT8[0] INVALIDATE TB       ; Clear the parity error at VA
U 14. 7700,4800,5BE4,03D8,6070,1815   00DD  6160 ;% 14    MEMSCR_R[ZERO]                      ;CLEAR THE MACHINE CHECK
U 15. 7300,C800,0364,0300,0470,1816   00E8  6164 ;% 15    NOP,STOP.CLOCK                      ;STOP CPU CLOCK
                                      00F3  6168
                                      00F3  6169 RTEMP_DATA
                                      0001  6170
                        00000000      0001  6171          .LONG   ^X00000000                ;PHY/VIR REGISTER ADDRESS
                        01000000      0005  6172          .LONG   ^X01000000                ;SET MME ON
                        03000000      0009  6173          .LONG   ^X03000000                ;TB GROUP DISABLE REG ADDRESS
                        0A000000      000D  6174          .LONG   ^X0A000000                ;DISABLE GROUP 1
                        0D000000      0011  6175          .LONG   ^X0D000000                ;TB GROUP PARITY ERROR REG ADD
                        08000000      0015  6176          .LONG   ^X08000000                ;MACHINE CHECK ERROR REG ADD
                                      0019  6177
                                      0019  6178 END_CPU_DATA
                                      00B8  6179
                                      00B8  6180 END_TEST_DATA
                                      00B8
                                      00B8        ;-----------------------------------------------------------------------------
                                      00B8  6181
                                      00B8  6182 ENDTEST
```

```
0032          .SBTTL  "TEST   04C     TEST TB GROUP 1 DATA PARITY GENERATOR/CHECKER
0032  6184 ;*******************************************************************************
0032  6185 ;++
0032  6186 ;
0032  6187 ; FUNCTIONAL DESCRIPTION:
0032  6188 ;
0032  6189 ;       THIS TEST VERIFIES THE ABILITY OF THE TB GROUP 1 DATA PARITY LOGIC
0032  6190 ;       TO GENERATE AS WELL AS DETECT PARITY ERRORS.
0032  6191 ;
0032  6192 ; TEST ALGORITHM:
0032  6193 ;
0032  6194 ;       1.  CREATE A TEST LOOP OF 6 TO SELECT TEST PTE'S
0032  6195 ;       2.  PUT A TEST PATTERN INTO THE RDM WD REGISTER
0032  6196 ;       3.  EXECUTE DCS PROGRAM
0032  6197 ;               a.  SET MEMORY MANAGEMENT ON
0032  6198 ;               b.  DISABLE GROUP 0
0032  6199 ;               c.  LOAD AN ADDRESS OF ZERO INTO THE VA REGISTER
0032  6200 ;               d.  CLEAR THE TB
0032  6201 ;               e.  WRITE THE TEST PATTERN WHILE FORCING A PARITY ERROR
0032  6202 ;               f.  PERFORM A READ OPERATION
0032  6203 ;       4.  EXECUTE MORE DCS U-CODE
0032  6204 ;               a.  READ THE "TB GROUP PARITY ERROR REGISTER" TO THE WH REG
0032  6205 ;       5.  TEST THE WH REGISTER FOR GROUP 1 PARITY ERROR
0032  6206 ;       6.  IF NO ERROR EXECUTE MORE DCS U-CODE
0032  6207 ;               a.  READ THE "MACHINE CHECK ERROR SUMMARY REG" TO THE WH REG
0032  6208 ;               b.  CLEAR THE TB PARITY ERROR
0032  6209 ;               c.  CLEAR THE "MACHINE CHECK ERROR SUMMARY REG"
0032  6210 ;       7.  TEST THE WH REGISTER FOR A TB ERROR
0032  6211 ;       8.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
0032  6212 ;
0032  6213 ;
0032  6214 ; TEST PATTERNS:
0032  6215 ;
0032  6216 ;       ITERATION       TEST PATTERN
0032  6217 ;           1             00AAABA8
0032  6218 ;           2             00555550
0032  6219 ;           3             00333330
0032  6220 ;           4             000F0F08
0032  6221 ;           5             0000FF00
0032  6222 ;           6             00000100
0032  6223 ;
0032  6224 ;
0032  6225 ; LOGIC DESCRIPTION:
0032  6226 ;
0032  6227 ;       THIS TEST FORCES PARITY ERRORS USING THE "MISC/FORCE.TB" MICRO ORDER.
0032  6228 ;       THE "TB GROUP PARITY ERROR REGISTER" AND THE "MACHINE CHECK ERROR
0032  6229 ;       SUMMARY REGISTER" ARE TESTED TO SEE IF THE FAILURE IS DETECTED
0032  6230 ;       PROPERLY.  IF REPLACING THE "UTR" GATE ARRAY DOES NOT HELP THEN YOU'LL
0032  6231 ;       HAVE TO SCOPE THE TB DATA PARITY GENERATOR/CHECKER TO ISOLATE THE FAULT.
0032  6232 ;
0032  6233 ;
0032  6234 ; ASSUMPTIONS:
0032  6235 ;
0032  6236 ;       THIS TEST ASSUMES THE TB DATA INTEGRITY TESTS HAVE RUN SUCCESSFULLY
0032  6237 ;
```

```
                     0032  6238 ;
                     0032  6239 ; ERROR DESCRIPTION:
                     0032  6240 ;
                     0032  6241 ;       NOTE: THERE ARE TWO IFERROR STATEMENTS IN THIS TEST.  SO CHECK THE PC
                     0032  6242 ;       TO DETERMINE WHICH ONE HAS FAILED.
                     0032  6243 ;
                     0032  6244 ;       FIRST IFERROR:
                     0032  6245 ;       EXPECTED "TB GROUP PARITY ERROR REGISTER" CONTENTS
                     0032  6246 ;       RECEIVED "TB GROUP PARITY ERROR REGISTER" CONTENTS
                     0032  6247 ;       LOOP COUNT I
                     0032  6248 ;
                     0032  6249 ;       SECOND IFERROR:
                     0032  6250 ;       EXPECTED "MACHINE CHECK ERROR SUMMARY REGISTER" CONTENTS
                     0032  6251 ;       RECEIVED "MACHINE CHECK ERROR SUMMARY REGISTER" CONTENTS
                     0032  6252 ;       LOOP COUNT I
                     0032  6253 ;
                     0032  6254 ;--
                     0032  6255 ;*******************************************************************************
                     0032  6256 ;///////////////////////////////////////////////////////////////////////////////
                     0032  6257
                     0032  6258         INITIALIZE                                  ;INIT THE CPU
                     0034  6259         EXPUTRAP                                    ;EXPECT A MACHINE CHECK
                     0036  6260         LOOP            I,1,6                       ;TEST LOOP OF 6
                     003D  6261         LOADREG         WDREG,1$,I,L                ;LOAD TEST PATTERN INTO WD REG
                     0045  6262         ERRLOOP                                     ;ERROR LOOP POINT
                     0047  6263         FETCH           0                           ;START DCS PROGRAM
                     004C  6264         BRSTCLK         <^X0A>,INH                  ;END DCS PROGRAM
                     0050  6265         FETCH           <^X0B>                      ;START DCS PROGRAM
                     0055  6266         BRSTCLK         <^X0E>,INH                  ;END DCS PROGRAM
                     0059  6267         CMPREGMSK       WHREG,2$,,3$,,L,            ;TEST TB ERROR REGISTER
                     0065  6268         IFERROR         ,<<UTR>>,                   ;NO PARITY ERROR
                     006B  6269         FETCH           <^X0F>                      ;START DCS PROGRAM
                     0070  6270         BRSTCLK         <^X15>                      ;END DCS PROGRAM
                     0074  6271         CMPREGMSK       WHREG,4$,,5$,,L,            ;TEST FOR MACHINE CHECK
                     0080  6272         IFERROR         ,<<UTR>>,                   ;NO MACHINE CHECK
                     0086  6273         ENDLOOP         I                           ;END TEST LOOP
                     0089  6274         SKIP                                        ;GO TO NEXT TEST
                     0090  6275
                     0090  6276 BEGIN_TEST_DATA
                     0090
                     0090       ;-------------------------------------------------------------------------
                     0090  6277
00AAABA8             0090  6278 1$:      .LONG    ^X00AAABA8                         ;TEST PATTERN TABLE
00555550             0094  6279          .LONG    ^X00555550
00333330             0098  6280          .LONG    ^X00333330
000F0F08             009C  6281          .LONG    ^X000F0F08
0000FF00             00A0  6282          .LONG    ^X0000FF00
00000100             00A4  6283          .LONG    ^X00000100
                     00A8  6284
02000000             00A8  6285 2$:      .LONG    ^X02000000                         ;TB ERROR REG CONTENTS
03000000             00AC  6286 3$:      .LONG    ^X03000000                         ;MASK FOR CMPREGMSK PSEUDO OP
04000000             00B0  6287 4$:      .LONG    ^X04000000                         ;MACHINE CHECK REG CONTENTS
0F000000             00B4  6288 5$:      .LONG    ^X0F000000                         ;MASK FOR CMPREGMSK PSEUDO OP
                     00B8  6289
                     00B8  6290 BEGIN_CPU_DATA
```

```
                                0002  6291
                                0002  6292 DCS_DATA
                                0001  6293
U 00. 7700.C800.0364.0300.0470.1801   0001  6294 ;% 00   NOP
U 01. 7700.C800.5BE4.0300.6870.1802   000C  6298 ;% 01   MEMSCAR_R[TEMP0]                    ;SET MME ON
U 02. 7700.8800.5BE4.0304.6070.1803   0017  6302 ;% 02   MEMSCR_R[TEMP1]
U 03. 7700.8800.5BE4.0308.6870.1804   0022  6306 ;% 03   MEMSCAR_R[TEMP2]                   ;DISABLE GROUP 0
U 04. 7700.C800.5BE4.030C.6070.1805   002D  6310 ;% 04   MEMSCR_R[TEMP3]
U 05. 7700.0800.5BE6.03D8.0470.1806   0038  6314 ;% 05   D_R[ZERO]                         ; D <-- 00000000
U 06. 7700.5800.DB13.0300.4A70.1807   0043  6318 ;% 06   VA_Q_D_D+ZLIT8[0]                  ; VA <-- address <00000000>
U 07. 7700.1800.DB12.0300.5360.1808   004E  6322 ;% 07   VA_D_D+ZLIT8[0] INVALIDATE TB     ; Flush the TB at VA
U 08. 5700.4F1B.0364.0300.5070.1809   0059  6326 ;% 08   TB_RDM,MISC/FORCE.TB              ;WRITE TEST PATTERN WITH ERROR
U 09. 7700.C800.0364.0300.0510.180A   0064  6330 ;% 09   READ.LONG                         ;CAUSE MACHINE CHECK
U 0A. 7300.C800.0364.0300.0470.180B   006F  6334 ;% 0A   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
U 0B. 7700.4800.0364.0300.0470.180C   007A  6338 ;% 0B   NOP
U 0C. 7700.8800.5BE4.0310.6870.180D   0085  6342 ;% 0C   MEMSCAR_R[TEMP4]                   ;READ TB GROUP PARITY ERROR REG
U 0D. 6700.C800.0364.0300.6470.180E   0090  6346 ;% 0D   RDM_WB,WCTRL/MEMSCR
U 0E. 7300.4800.0364.0300.0470.180F   009B  6350 ;% 0E   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
U 0F. 7700.C800.0364.0300.0470.1810   00A6  6354 ;% 0F   NOP
U 10. 7700.4800.5BE4.0314.6870.1811   00B1  6358 ;% 10   MEMSCAR_R[TEMP5]                   ;READ MACHINE CHECK SUMMARY REG
U 11. 6700.4800.0364.0300.6470.1812   00BC  6362 ;% 11   RDM_WB,WCTRL/MEMSCR
U 12. 7700.5800.DB13.0300.4A70.1813   00C7  6366 ;% 12   VA_Q_D_D+ZLIT8[0]                  ; VA <-- address (00000000)
U 13. 7700.9800.DB12.0300.5360.1814   00D2  6370 ;% 13   VA_D_D+ZLIT8[0] INVALIDATE TB     ; Clear the parity error at VA
U 14. 7700.4800.5BE4.03D8.6070.1815   00DD  6374 ;% 14   MEMSCR_R[ZERO]                     ;CLEAR THE MACHINE CHECK
U 15. 7300.C800.0364.0300.0470.1816   00E8  6378 ;% 15   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
                                00F3  6382
                                00F3  6383 RTEMP_DATA
                                0001  6384
                      00000000  0001  6385         .LONG   ^X00000000                    ;PHY/VIR REGISTER ADDRESS
                      01000000  0005  6386         .LONG   ^X01000000                    ;SET MME ON
                      03000000  0009  6387         .LONG   ^X03000000                    ;TB GROUP DISABLE REG ADDRESS
                      0D000000  000D  6388         .LONG   ^X0D000000                    ;DISABLE GROUP 0
                      0D000000  0011  6389         .LONG   ^X0D000000                    ;TB GROUP PARITY ERROR REG ADD
                      08000000  0015  6390         .LONG   ^X08000000                    ;MACHINE CHECK ERROR REG ADD
                                0019  6391
                                0019  6392 END_CPU_DATA
                                00B8  6393
                                00B8  6394 END_TEST_DATA
                                00B8
                                00B8        ;-----------------------------------------------------------------------------
                                00B8  6395
                                00B8  6396 ENDTEST
```

```
002C          .SBTTL  "TEST   04D     TEST TB DATA PARITY ERROR MACHINE CHECK
002C  6398 ;*******************************************************************************
002C  6399 ;++
002C  6400 ;
002C  6401 ; FUNCTIONAL DESCRIPTION:
002C  6402 ;
002C  6403 ;       THIS TEST WILL VERIFY THAT CPU EXECUTES A MACHINE CHECK WHEN A TB
002C  6404 ;       DATA PARITY ERROR IS ENCOUNTERED.
002C  6405 ;
002C  6406 ;
002C  6407 ; TEST ALGORITHM:
002C  6408 ;
002C  6409 ;       1.  MODIFY DCS U-CODE FOR SUBTEST
002C  6410 ;       2.  EXECUTE DCS PROGRAM
002C  6411 ;               a.  SET MEMORY MANAGEMENT ON
002C  6412 ;               b.  DISABLE ONE TB GROUP
002C  6413 ;               c.  LOAD VA REGISTER WITH ZERO
002C  6414 ;               d.  CLEAR THE TB
002C  6415 ;               e.  SET VALID BIT IN TB LOCATION ZERO
002C  6416 ;               f.  PERFORM A READ WHILE FORCING A TB DATA PARITY ERROR
002C  6417 ;       3.  TEST CS ADDRESS FOR A MACHINE CHECK
002C  6418 ;       4.  IF NO ERROR EXECUTE DCS PROGRAM
002C  6419 ;               a.  CLEAR THE TB
002C  6420 ;               b.  CLEAR THE MACHINE CHECK ERROR SUMMARY REGISTER
002C  6421 ;
002C  6422 ;
002C  6423 ; TEST PATTERNS:
002C  6424 ;
002C  6425 ;       VA = 00000000
002C  6426 ;       TB DATA = 00000100
002C  6427 ;
002C  6428 ;
002C  6429 ; LOGIC DESCRIPTION:
002C  6430 ;
002C  6431 ;       THIS TEST PERFORMS A READ OPERATION WHILE FORCING A TB DATA PARITY
002C  6432 ;       ERROR.  THIS SHOULD RESULT IN THE "UTR" GATE ARRAY FORCING A MACHINE
002C  6433 ;       CHECK MICROTRAP.
002C  6434 ;
002C  6435 ;
002C  6436 ; ASSUMPTIONS:
002C  6437 ;
002C  6438 ;       THIS TEST ASSUMES THE TB DATA INTEGRITY TESTS HAVE RUN SUCCESSFULLY
002C  6439 ;
002C  6440 ;
002C  6441 ; ERROR DESCRIPTION:
002C  6442 ;
002C  6443 ;       EXPECTED CSTRP REGISTER CONTENTS
002C  6444 ;       RECEIVED CSTRP REGISTER CONTENTS
002C  6445 ;
002C  6446 ;--
002C  6447 ;*******************************************************************************
002C  6448 ;///////////////////////////////////////////////////////////////////////////////
002C  6449 ;+
002C  6450 ;SUBTEST #1
002C  6451 ;
```

```
                     002C  6452 ;       GROUP 0 DATA PARITY ERROR
                     002C  6453 ;-
                     002C  6454         SUBTEST                                         ;SUBTEST #1
                     002F
                     002F       ;////////////////////////////////////////////////////////////////////////
                     002F  6455         INITIALIZE                                      ;INIT THE CPU
                     0031  6456         LOADDCS         3$,,4,1                         ;MODIFY U-CODE FOR THIS TEST
                     0038  6457         EXPUTRAP                                        ;EXPECT A MACHINE CHECK
                     003A  6458         ERRLOOP                                         ;ERROR LOOP POINT
                     003C  6459         FETCH           0                               ;START DCS PROGRAM
                     0041  6460         BRSTCLK         <^X0A>                          ;END DCS PROGRAM
                     0045  6461         CMPREGMSK       CSTRP,1$,,2$,,W,                ;TEST FOR MACHINE CHECK
                     0051  6462         IFERROR         ,<<UTR>>,                       ;DID NOT MACHINE CHECK
                     0057  6463         FETCH           <^X0B>                          ;START DCS PROGRAM
                     005C  6464         BRSTCLK         <^X10>                          ;END DCS PROGRAM
                     0060  6465
                     0060  6466
                     0060  6467 ;////////////////////////////////////////////////////////////////////////
                     0060  6468 ;+
                     0060  6469 ;SUBTEST #2
                     0060  6470 ;
                     0060  6471 ;       GROUP 1 DATA PARITY ERROR
                     0060  6472 ;-
                     0060  6473         SUBTEST                                         ;SUBTEST #2
                     0063
                     0063       ;////////////////////////////////////////////////////////////////////////
                     0063  6474         INITIALIZE                                      ;INIT THE CPU
                     0065  6475         LOADDCS         4$,,4,1                         ;MODIFY U-CODE FOR THIS TEST
                     006C  6476         EXPUTRAP                                        ;EXPECT MACHINE CHECK
                     006E  6477         ERRLOOP                                         ;ERROR LOOP POINT
                     0070  6478         FETCH           0                               ;START DCS PROGRAM
                     0075  6479         BRSTCLK         <^X0A>                          ;END DCS PROGRAM
                     0079  6480         CMPREGMSK       CSTRP,1$,,2$,,W,                ;TEST FOR MACHINE CHECK
                     0085  6481         IFERROR         ,<<UTR>>,                       ;DID NOT MACHINE CHECK
                     008B  6482         FETCH           <^X0B>                          ;START DCS PROGRAM
                     0090  6483         BRSTCLK         <^X10>                          ;END DCS PROGRAM
                     0094  6484         SKIP                                            ;GO TO NEXT TEST
                     009B  6485
                     009B  6486
                     009B  6487
                     009B  6488 BEGIN_TEST_DATA
                     009B
                     009B       ;-------------------------------------------------------------------------
                     009B  6489
                0028 009B  6490 1$:     .WORD   ^X0028                                  ;ADDRESS OF MACHINE CHECK
                3FFF 009D  6491 2$:     .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO OP
                     009F  6492 3$:
U 04, 7700,4800,5BE4,030C,6870,1805  009F  6493 ;% 04   MEMSCAR_R[TEMP3]                 ;U-CODE SUBTEST #1
                     00AA  6497 4$:
U 04, 7700,0800,5BE4,0310,6870,1805  00AA  6498 ;% 04   MEMSCAR_R[TEMP4]                 ;U-CODE SUBTEST #2
                     00B5  6502
                     00B5  6503
                     00B5  6504
                     00B5  6505 BEGIN_CPU_DATA
                     0002  6506
```

```
                                0002  6507 DCS_DATA
                                0001  6508
U 00. 7700.C800.0364.0300.0470.1801  0001  6509 ;* 00   NOP
U 01. 7700.C800.5BE4.0300.6870.1802  000C  6513 ;* 01   MEMSCAR_R[TEMP0]                    ;SET MME ON
U 02. 7700.8800.5BE4.0304.6070.1803  0017  6517 ;* 02   MEMSCR_R[TEMP1]
U 03. 7700.8800.5BE4.0308.6870.1804  0022  6521 ;* 03   MEMSCAR_R[TEMP2]                   ;DISABLE 1 TB GROUP
U 04. 7700.C800.5BE4.030C.6070.1805  002D  6525 ;* 04   MEMSCR_R[TEMP3]
U 05. 7700.0800.5BE6.03D8.0470.1806  0038  6529 ;* 05   D_R[ZERO]                          ; D <-- 00000000
U 06. 7700.5800.DB13.0300.4A70.1807  0043  6533 ;* 06   VA_Q_D_D+ZLIT8[0]                  ; VA <-- address <00000000>
U 07. 7700.1800.DB12.0300.5360.1808  004E  6537 ;* 07   VA_D_D+ZLIT8[0] INVALIDATE TB     ; Flush the TB at VA
U 08. 7700.C81B.5BE4.0318.5070.1809  0059  6541 ;* 08   TB_R[TEMP6]                        ;VALIDATE LOCATION ZERO
U 09. 7700.4F00.0364.0300.0510.180A  0064  6545 ;* 09   READ.LONG,MISC/FORCE.TB           ;READ AND FORCE PARITY ERROR
U 0A. 7300.C800.0364.0300.0470.180B  006F  6549 ;* 0A   NOP,STOP.CLOCK                    ;STOP THE CPU CLOCK
U 0B. 7700.4800.0364.0300.0470.180C  007A  6553 ;* 0B   NOP
U 0C. 7700.5800.DB13.0300.4A70.180D  0085  6557 ;* 0C   VA_Q_D_D+ZLIT8[0]                  ; VA <-- address (00000000)
U 0D. 7700.1800.DB12.0300.5360.180E  0090  6561 ;* 0D   VA_D_D+ZLIT8[0] INVALIDATE TB     ; Clear the parity error at VA
U 0E. 7700.4800.5BE4.0314.6870.180F  009B  6565 ;* 0E   MEMSCAR_R[TEMP5]                   ;CLEAR THE MACHINE CHECK
U 0F. 7700.4800.5BE4.03D8.6070.1810  00A6  6569 ;* 0F   MEMSCR_R[ZERO]
U 10. 7300.4800.0364.0300.0470.1811  00B1  6573 ;* 10   NOP,STOP.CLOCK                    ;STOP THE CPU CLOCK
                                00BC  6577
                                00BC  6578 RTEMP_DATA
                                0001  6579
                    00000000    0001  6580          .LONG   ^X00000000                  ;PHY/VIR REGISTER ADDRESS
                    01000000    0005  6581          .LONG   ^X01000000                  ;SET MME ON
                    03000000    0009  6582          .LONG   ^X03000000                  ;TB GROUP DISABLE REG ADDRESS
                    0A000000    000D  6583          .LONG   ^X0A000000                  ;DISABLE GROUP 1
                    0D000000    0011  6584          .LONG   ^X0D000000                  ;DISABLE GROUP 0
                    08000000    0015  6585          .LONG   ^X08000000                  ;MACHINE CHECK REGISTER ADDRESS
                    00000100    0019  6586          .LONG   ^X00000100                  ;VALID PTE
                                001D  6587
                                001D  6588 END_CPU_DATA
                                00B5  6589
                                00B5  6590
                                00B5  6591 END_TEST_DATA
                                00B5
                                00B5       ;-----------------------------------------------------------------------------
                                00B5  6592
                                00B5  6593 ENDTEST
```

```
                        .SBTTL  "TEST   04E     TEST FOR MACHINE CHECK WITH TB MULTIPLE HIT
0030   6595 ;*******************************************************************************
0030   6596 ;++
0030   6597 ;
0030   6598 ; FUNCTIONAL DESCRIPTION:
0030   6599 ;
0030   6600 ;       THIS TEST INSURES THAT CPU CAN DETECT A TB MULTIPLE HIT AND PRODUCE
0030   6601 ;       A MACHINE CHECK MICROTRAP AS A RESULT.
0030   6602 ;
0030   6603 ;
0030   6604 ; TEST ALGORITHM:
0030   6605 ;
0030   6606 ;       1.  EXECUTE DCS PROGRAM
0030   6607 ;               a.  SET MEMORY MANAGEMENT ON
0030   6608 ;               b.  LOAD ADDRESS INTO VA REGISTER (00000100)
0030   6609 ;               c.  PERFORM A CLEAR TB MICRO ORDER. SINCE BIT 8 WILL BE SET ON
0030   6610 ;                   THE WBUS BOTH TB GROUPS WILL END-UP BEING VALIDATED.
0030   6611 ;               d.  PERFORM A READ WHICH WILL CAUSE A MACHINE CHECK
0030   6612 ;       2.  TEST THE MICRO VECTOR LINES AT THE VBUS
0030   6613 ;       3.  IF NO ERROR TEST THE CS ADDRESS FOR 28 (MACHINE CHECK)
0030   6614 ;       4.  EXECUTE MORE DCS U-CODE
0030   6615 ;               a.  READ THE MACHINE CHECK ERROR SUMMARY REGISTER TO THE RDM
0030   6616 ;               b.  CLEAR THE MULTIPLE HIT
0030   6617 ;               c.  CLEAR THE MACHINE CHECK ERROR SUMMARY REGISTER
0030   6618 ;       5.  TEST THE WH REGISTER FOR TB ERROR
0030   6619 ;       6.  IF NO ERROR GO TO NEXT TEST
0030   6620 ;
0030   6621 ;
0030   6622 ; LOGIC DESCRIPTION:
0030   6623 ;
0030   6624 ;       THIS TEST MOSTLY INVOLVES LOGIC IN THE "UTR" GATE ARRAY.  THE MACHINE
0030   6625 ;       CHECK LOGIC IN UTR SHOULD DETECT A MULTIPLE TB HIT AND ASSERT A MACHINE
0030   6626 ;       CHECK MICROTRAP.  THE ONLY TRICKY PART OF THIS TEST IS WHERE BIT 8 (THE
0030   6627 ;       VALID BIT) IS SET ON THE WBUS DURING A CLEAR TB MICRO ORDER.  THIS
0030   6628 ;       RESULTS IN VALIDATING RATHER THAN CLEARING BOTH TB GROUPS.
0030   6629 ;
0030   6630 ;
0030   6631 ; ASSUMPTIONS:
0030   6632 ;
0030   6633 ;       THIS TEST ASSUMES THE TB DATA INTEGRITY TESTS HAVE RUN SUCCESSFULLY
0030   6634 ;
0030   6635 ;
0030   6636 ; ERROR DESCRIPTION:
0030   6637 ;
0030   6638 ;       NOTE: THERE ARE THREE IFERROR STATEMENTS IN THIS TEST.  SO YOU WILL
0030   6639 ;       HAVE TO CHECK THE FAILING PC TO DETERMINE WHICH ERROR PRINTOUT IS
0030   6640 ;       VALID.
0030   6641 ;
0030   6642 ;       FIRST IFERROR:
0030   6643 ;       EXPECTED VBUS DATA (MICRO VECTOR LINES)
0030   6644 ;       RECEIVED VBUS DATA
0030   6645 ;
0030   6646 ;       SECOND IFERROR:
0030   6647 ;       EXPECTED CS ADDRESS
0030   6648 ;       RECEIVED CS ADDRESS
```

```
                        0030  6649 ;
                        0030  6650 ;       THIRD IFERROR:
                        0030  6651 ;       EXPECTED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                        0030  6652 ;       RECEIVED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                        0030  6653 ;
                        0030  6654 ;--
                        0030  6655 ;*******************************************************************************
                        0030  6656
                        0030  6657
                        0030  6658
                        0030  6659          INITIALIZE                              ;INIT THE CPU
                        0032  6660          EXPUTRAP                                ;EXPECT A MACHINE CHECK
                        0034  6661          ERRLOOP                                 ;ERROR LOOP POINT
                        0036  6662          FETCH           0                       ;START DCS PROGRAM
                        003B  6663          BRSTCLK         6                       ;END DCS PROGRAM
                        003F  6664          CMPVBUS         5$                      ;TEST MICRO VECTOR LINES
                        0044  6665          IFERROR         ,<<UTR>>,               ;WRONG VECTOR
                        004A  6666          CMPREGMSK       CSTRP,3$,,4$,,W,        ;TEST THE CS ADDRESS
                        0056  6667          IFERROR         ,,<<DPM>>               ;NOT A MACHINE CHECK
                        005C  6668          FETCH           7                       ;START DCS PROGRAM
                        0061  6669          BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                        0065  6670          CMPREGMSK       WHREG,1$,,2$,,L,        ;TEST FOR TB ERROR
                        0071  6671          IFERROR         ,<<UTR>>,               ;INCORRECT ERROR
                        0077  6672          SKIP                                    ;GO TO NEXT TEST
                        007E  6673
                        007E  6674
                        007E  6675
                        007E  6676
                        007E  6677 BEGIN_TEST_DATA
                        007E
                        007E       ;-------------------------------------------------------------------------
                        007E  6678
                04000000  007E  6679 1$:     .LONG   ^X04000000                      ;MACHINE CHECK REG DATA
                0F000000  0082  6680 2$:     .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                    0028  0086  6681 3$:     .WORD   ^X0028                          ;MACHINE CHECK U-TRAP
                    3FFF  0088  6682 4$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                      04  008A  6683 5$:     .BYTE   ^X4                             ;VBUS MICRO VECTOR LINES
                          008B  6684          VBUSDATA        <^X08>,1
                          008C  6685          VBUSDATA        <^X09>,0
                          008D  6686          VBUSDATA        <^X0A>,0
                          008E  6687          VBUSDATA        <^X0B>,0
                          008F  6688
                          008F  6689
                          008F  6690
                          008F  6691 BEGIN_CPU_DATA
                          0002  6692
                          0002  6693 DCS_DATA
                          0001  6694
U 00, 7700,C800,0364,0300,0470,1801   0001  6695 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C  6699 ;% 01   MEMSCAR_R[TEMP0]                ;SET MME ON
U 02, 7700,8800,5BE4,0304,6070,1803   0017  6703 ;% 02   MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,4A70,1804   0022  6707 ;% 03   VA_R[TEMP2]                    ;LOAD VA REGISTER
U 04, 7700,0800,5BE4,0308,5360,1805   002D  6711 ;% 04   CLRTB.VA_R[TEMP2]             ;VALIDATE BOTH TB GROUPS
U 05, 7600,C800,0364,0300,0510,1806   0038  6715 ;% 05   READ.LONG,STROBE.V.BUS        ;READ AND LOAD VBUS
U 06, 7300,C800,0364,0300,0470,1807   0043  6719 ;% 06   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
```

```
U 07, 7700,C800,0364,0300,0470,1808  004E  6723 ;% 07   NOP
U 08, 7700,4800,5BE4,030C,6870,1809  0059  6727 ;% 08   MEMSCAR_R[TEMP3]                    ;READ MACHINE CHECK REGISTER
U 09, 6700,4800,0364,0300,6470,180A  0064  6731 ;% 09   RDM_WB,WCTRL/MEMSCR
U 0A, 7700,C800,5BE4,03D8,5360,180B  006F  6735 ;% 0A   CLRTB.VA_WB,WB_R[ZERO]              ;FLUSH TB
U 0B, 7700,C800,5BE4,03D8,6070,180C  007A  6739 ;% 0B   MEMSCR_R[ZERO]                     ;CLEAR MACHINE CHECK REGISTER
U 0C, 7300,C800,0364,0300,0470,180D  0085  6743 ;% 0C   NOP,STOP.CLOCK                    ;STOP THE CPU CLOCK
                                     0090  6747
                                     0090  6748 RTEMP_DATA
                                     0001  6749
                           00000000  0001  6750         .LONG   ^X00000000              ;PHY/VIR REGISTER ADDRESS
                           01000000  0005  6751         .LONG   ^X01000000              ;SET MME ON
                           00000100  0009  6752         .LONG   ^X00000100              ;VA ADDRESS
                           08000000  000D  6753         .LONG   ^X08000000              ;MACHINE CHECK REGISTER ADDRESS
                                     0011  6754
                                     0011  6755 END_CPU_DATA
                                     008F  6756
                                     008F  6757
                                     008F  6758
                                     008F  6759 END_TEST_DATA
                                     008F
                                     008F       ;-----------------------------------------------------------------------
                                     008F  6760
                                     008F  6761 ENDTEST
```

```
0026        .SBTTL  "TEST   04F     VA_PC+WBUS+ISIZE.PC_PC+ISIZE TEST
0026  6763 ;*******************************************************************************
0026  6764 ;++
0026  6765 ;
0026  6766 ; FUNCTIONAL DESCRIPTION
0026  6767 ;
0026  6768 ;       This test examines the data integrity of the VA_PC+WBUS+ISIZE and
0026  6769 ;       PC_PC+ISIZE micro-instruction. This test has been divided into two
0026  6770 ;       subtests.
0026  6771 ;
0026  6772 ;       The first subtest loads six longword patterns into the PC and two
0026  6773 ;       longword patterns onto the WBUS. After the micro-instruction has exe-
0026  6774 ;       cuted, the VA is examined to make sure that it contains the sum of
0026  6775 ;       the PC, WBUS and ISIZE.
0026  6776 ;
0026  6777 ;       The second subtest uses the same longword patterns for the PC and the
0026  6778 ;       WBUS. But after the test micro-instruction has executed, the PC is
0026  6779 ;       examined to see if it incremented by the current value of ISIZE.
0026  6780 ;
0026  6781 ; TEST ALGORITHM
0026  6782 ;
0026  6783 ;       1. Initialize the CPU.
0026  6784 ;       2. Load DCS with the micro-instructions for the test.
0026  6785 ;       3. Load the index K with a value of 1.
0026  6786 ;       4. Prepare a loop of three iterations on I.
0026  6787 ;       5. Load the micro instruction field for DTYPE = 1,2,4 depending on I.
0026  6788 ;       6. Reload the modified micro instruction into DCS.
0026  6789 ;       7. Prepare a loop of two iterations on J.
0026  6790 ;       8. Load the RDM D register with the PC data pattern indexed by K.
0026  6791 ;       9. Execute the micro instruction:
0026  6792 ;               1. PC_RDM
0026  6793 ;      10. Load the RDM D register with the WBUS data pattern indexed by K.
0026  6794 ;      11. Execute the micro instructions:
0026  6795 ;               2. VA_PC+WB+I.PC_PC+I,WB_RDM
0026  6796 ;               3. RDM_VA
0026  6797 ;      12. Compare the RDM H register with the correct data indexed by K.
0026  6798 ;      13. Signal an error if the two are not equal.
0026  6799 ;      14. Otherwise execute the micro instrucion:
0026  6800 ;               4. RDM_PC
0026  6801 ;      15. Compare the RDM H register with the correct data indexed by K.
0026  6802 ;      16. Signal an error if the two are not equal.
0026  6803 ;      17. Increment K.
0026  6804 ;      18. Looped two times on J?  If not go to STEP 8.
0026  6805 ;      19. Looped three times on I?  If not go to STEP 3.
0026  6806 ;      20. Test completed.
0026  6807 ;
0026  6808 ; TEST DATA
0026  6809 ;
0026  6810 ;       The following data patterns are required for this test:
0026  6811 ;
0026  6812 ;       Input data
0026  6813 ;
0026  6814 ;       I  J  K   ISIZE              PC                      WBUS
0026  6815 ;
0026  6816 ;       1  1  1     1          5 5 5 5 5 5 5 4         5 5 5 5 5 5 5 5
```

```
0026 ;       1  2  2     1          A A A A A A A 9        A A A A A A A B
0026 ;       2  1  3     2          5 5 5 5 5 5 5 3        5 5 5 5 5 5 5 5
0026 ;       2  2  4     2          A A A A A A A 8        A A A A A A A B
0026 ;       3  1  5     4          5 5 5 5 5 5 5 1        5 5 5 5 5 5 5 5
0026 ;       3  2  6     4          A A A A A A A 6        A A A A A A A B
0026 ;
0026 ;       Result data
0026 ;
0026 ;       I  J  K   ISIZE              PC                      VA
0026 ;
0026 ;       1  1  1     1          5 5 5 5 5 5 5 5        A A A A A A A A
0026 ;       1  2  2     1          A A A A A A A A        5 5 5 5 5 5 5 5
0026 ;       2  1  3     2          5 5 5 5 5 5 5 5        A A A A A A A A
0026 ;       2  2  4     2          A A A A A A A A        5 5 5 5 5 5 5 5
0026 ;       3  1  5     4          5 5 5 5 5 5 5 5        A A A A A A A A
0026 ;       3  2  6     4          A A A A A A A A        5 5 5 5 5 5 5 5
0026 ;
0026 ; LOGIC DESCRIPTION
0026 ;
0026 ;       Alternating bit patterns provide a test for stuck at one or zero
0026 ;       conditions in the address path.
0026 ;
0026 ; ASSUMPTIONS
0026 ;
0026 ;       This test assumes that the DPM, MBUS, WBUS, PC, VA and the adder
0026 ;       internal to the ADD gate arrays have been successfully tested.
0026 ;
0026 ; ERROR DESCRIPTION
0026 ;
0026 ;       Below is a table of possible error symptoms and their most likely
0026 ;       causes:
0026 ;
0026 ;               Symptom                      Probable Cause
0026 ;       +-----------------------+-----------------------------------+
0026 ;       | Bad carries, and/or   | A defective adder in the ADD slice for|
0026 ;       | stuck bits within a   | that byte. More than one defective ADD|
0026 ;       | particular byte.      | slice will cause groups of bad bytes. |
0026 ;       +-----------------------+-----------------------------------+
0026 ;       | Bad sums across byte  | A defective 74LS182 chip on the MIC   |
0026 ;       | boundaries due to an  | module.                               |
0026 ;       | incorrect carry.      |                                       |
0026 ;       +-----------------------+-----------------------------------+
0026 ;       | Byte other that the   | The CHIP ID signal to that ADD gate   |
0026 ;       | least significant byte| array has been accidentally shorted to|
0026 ;       | of the PC is being    | ground.                               |
0026 ;       | incremented by ISIZE. |                                       |
0026 ;       +-----------------------+-----------------------------------+
0026 ;       | VA contains random    | Defects in the ASRC decode logic. Bad |
0026 ;       | data or improper      | ASRC selections can cause whatever is |
0026 ;       | PC increments.        | currently on the WBUS to be added to  |
0026 ;       |                       | the PC, or an improper increment.     |
0026 ;       +-----------------------+-----------------------------------+
0026 ;       | Sum of PC is the      | A defective ADK gate array can cause a|
0026 ;       | increment of VA,      | register other that the PC to be      |
0026 ;       | VA SAVE or PC BACKUP. | incremented.                          |
```

```
0026  ;       +-----------------------+-----------------------------------+
0026  ;       | One particular bad sum| A defective PRK gate array will cause |
0026  ;       | coming from the adder.| the wrong register to be read from the|
0026  ;       |                       | ADD gate array.                      |
0026  ;       +-----------------------+-----------------------------------+
0026  ;
0026  ;       On error this test will call out the ADD, ADK and PRK gate arrays on
0026  ;       the MIC module. The following information will also by typed out:
0026  ;
0026  ;       Expected Data:
0026  ;       Received Data:
0026  ;       Index I:
0026  ;       Index J:
0026  ;       Index K:
0026  ;
0026  ;--
0026  ;*******************************************************************************
0026
0026          SUBTEST         1
0029
0029      ;////////////////////////////////////////////////////////////////////////////
0029  ;+
0029  ; Subtest 1
0029  ;-
0029          INITIALIZE                                  ; INITIALIZE THE CPU
002B          LOADDCS         6$,,0,8                     ; U-CODE FOR PC/VA/LONLIT LOAD
0032          LDINDEX         K,1                         ; INITIALIZE INDEX FOR TABLE 2
0039          LOOP            I,1,3                       ; INDEX FOR TABLE 1
0040          LDFIELD         1$,I,BYTE,8$,40,41          ; DTYPE FIELD LOAD
0051          LOADDCS         8$,,5,1                     ; SET DTYPE FIELD IN DCS
0058          LOOP            J,1,2                       ; INDEX FOR TABLES 3/4/5
005F          LDFIELD         3$,J,LONG,7$,31,62,LONLIT ; LONLIT FIELD LOAD
0070          LOADDCS         7$,,4,1                     ; SET LONLIT FIELD IN DCS
0077          LOADREG         WDREG,2$,K,LONG             ; LOAD TABLE 2 DATA INTO WDREG
007F          ERRLOOP                                     ; ERROR LOOP HERE
0081          FETCH           0                           ; EXECUTE VA AND PC LOAD
0086          BRSTCLK         7                           ;        "        "
008A          CMPREG          WHREG,4$,J,LONG             ; DOES WHREG = CORRECT DATA?
0093          IFERROR         ,<<ADD><ADK><PRK>>          ; IF NOT SIGNAL ERROR
009B          INCINDEX        K                           ; OTHERWISE INCREMENT INDEX K
009E          ENDLOOP         J                           ; CONTINUE LOOPING TILL J = 2
00A1          ENDLOOP         I                           ; CONTINUE LOOPING TILL I = 3
00A4
00A4          SUBTEST         2
00A7
00A7      ;////////////////////////////////////////////////////////////////////////////
00A7  ;+
00A7  ; Subtest 2
00A7  ;-
00A7          LOADDCS         9$,,6,1
00AE          LDINDEX         K,1
00B5          LOOP            I,1,3
00BC          LDFIELD         1$,I,BYTE,8$,40,41
00CD          LOADDCS         8$,,5,1
00D4          LOOP            J,1,2
```

```
                        00DB  6923          LDFIELD         3$,J,LONG,7$,31,62,LONLIT
                        00EC  6924          LOADDCS         7$,,4,1
                        00F3  6925          LOADREG         WDREG,2$,K,LONG
                        00FB  6926          ERRLOOP
                        00FD  6927          FETCH           0
                        0102  6928          BRSTCLK         7
                        0106  6929          CMPREG          WHREG,5$,J,LONG
                        010F  6930          IFERROR         ,<<ADD><ADK><PRK>>,
                        0117  6931          INCINDEX        K
                        011A  6932          ENDLOOP         J
                        011D  6933          ENDLOOP         I
                        0120  6934          SKIP
                        0127  6935
                        0127  6936 BEGIN_TEST_DATA
                        0127
                        0127          ;-----------------------------------------------------------------------
                        0127  6937
                     00 0127  6938 1$:      .BYTE   0               ; ISIZE values
                     01 0128  6939          .BYTE   1
                     02 0129  6940          .BYTE   2
                        012A  6941
                55555554 012A  6942 2$:      .LONG   ^X 55555554     ; PC inputs for ISIZE = byte
                AAAAAAA9 012E  6943          .LONG   ^X AAAAAAA9
                55555553 0132  6944          .LONG   ^X 55555553     ; ISIZE = word
                AAAAAAA8 0136  6945          .LONG   ^X AAAAAAA8
                55555551 013A  6946          .LONG   ^X 55555551     ; ISIZE = long
                AAAAAAA6 013E  6947          .LONG   ^X AAAAAAA6
                         0142  6948
                55555555 0142  6949 3$:      .LONG   ^X 55555555     ; WBUS inputs
                AAAAAAAB 0146  6950          .LONG   ^X AAAAAAAB
                         014A  6951
                AAAAAAAA 014A  6952 4$:      .LONG   ^X AAAAAAAA     ; VA results
                55555555 014E  6953          .LONG   ^X 55555555
                         0152  6954
                55555555 0152  6955 5$:      .LONG   ^X 55555555     ; PC results
                AAAAAAAA 0156  6956          .LONG   ^X AAAAAAAA
                         015A  6957
                         015A  6958 6$:
U 00, 7700,C800,0364,0300,0470,1801  015A  6959 ;% 00    NOP
U 01, 7700,7800,7FFF,FFFF,8470,1802  0165  6963 ;% 01    LONLIT_[0]
U 02, 7700,4800,5BE4,03D4,4A70,1803  0170  6967 ;% 02    VA_R[LONLIT]
U 03, 5700,4800,0364,0300,4870,1804  017B  6971 ;% 03    PC_RDM
                         0186  6975 7$:
U 04, 7700,F800,7FFF,FFFF,8470,1805  0186  6976 ;% 04    LONLIT_[0]
                         0191  6980 8$:
U 05, 7701,8817,5BE4,02D6,4070,1806  0191  6981 ;% 05    VA_PC+LONLIT+I.PC_PC+I
U 06, 6700,881B,0024,03D8,0470,1807  019C  6985 ;% 06    RDM_VA
U 07, 7300,C800,0364,0300,0470,1808  01A7  6989 ;% 07    STOP.CLOCK
                         01B2  6993 9$:
U 06, 6700,C81A,0024,03D8,0470,1807  01B2  6994 ;% 06    RDM_PC
                         01BD  6998
                         01BD  6999 BEGIN_CPU_DATA
                         0002  7000
                         0002  7001 CACHE_DATA
                         0001  7002
```

```
00000000  0001  7003          .LONG   ^X 00000000
          0005  7004
          0005  7005 END_CPU_DATA
          01BD  7006
          01BD  7007 END_TEST_DATA
          01BD
          01BD       ;-----------------------------------------------------------------------------
          01BD  7008
          01BD  7009 ENDTEST
```

```
                        .SBTTL  "TEST   050     TEST XB0 AND XB1 REGISTERS
001F   7011 ;*******************************************************************************
001F   7012 ;++
001F   7013 ;
001F   7014 ; FUNCTIONAL DESCRIPTION:
001F   7015 ;
001F   7016 ;       THIS TEST CONSISTS OF 2 SUBTESTS TO CHECK THE DATA INTEGRITY OF THE
001F   7017 ;       EXECUTION BUFFERS.  6 TEST PATTERNS ARE WRITTEN TO AND READ BACK
001F   7018 ;       FROM EACH REGISTER.
001F   7019 ;
001F   7020 ;
001F   7021 ; TEST ALGORITHM:
001F   7022 ;
001F   7023 ;       1.  CREATE A TEST LOOP
001F   7024 ;       2.  LOAD DCS WITH MICRO WORDS FOR THIS TEST
001F   7025 ;       3.  EXECUTE DCS PROGRAM
001F   7026 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
001F   7027 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
001F   7028 ;                2.  SET MEMORY MANAGEMENT OFF
001F   7029 ;                3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
001F   7030 ;                4.  SET CMI OFF
001F   7031 ;                5.  LOAD VIRTUAL ADDRESS REGISTER WITH 0'S
001F   7032 ;                6.  PUT TEST PATTERN INTO LONGLIT REGISTER
001F   7033 ;                7.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 0
001F   7034 ;                8.  INCREMENT VITUAL ADDRESS REGISTER BY 4
001F   7035 ;                9.  PUT TEST PATTERN INTO LONGLIT REGISTER
001F   7036 ;               10.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 4
001F   7037 ;               11.  LOAD PROGRAM COUNTER WITH 0'S
001F   7038 ;               12.  READ DESIRED TEST PATTERN FROM XB'S TO RDM
001F   7039 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
001F   7040 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ITERATIONS
001F   7041 ;
001F   7042 ;
001F   7043 ; TEST PATTERNS:
001F   7044 ;
001F   7045 ;       ITERATION           XB0           XB1
001F   7046 ;
001F   7047 ;           1               AAAAAAAA      0000FFFF
001F   7048 ;           2               55555555      00FF00FF
001F   7049 ;           3               33333333      0F0F0F0F
001F   7050 ;           4               0F0F0F0F      33333333
001F   7051 ;           5               00FF00FF      55555555
001F   7052 ;           6               0000FFFF      AAAAAAAA
001F   7053 ;
001F   7054 ;
001F   7055 ; LOGIC DESCRIPTION:
001F   7056 ;
001F   7057 ;       BY DIRECTLY LOADING THE PC THIS TEST INVOKES THE PREFETCH MECHANISM
001F   7058 ;       TO LOAD THE XB'S.  IN THE EVENT OF ERRORS THE "MDR" GATE ARRAY IS A
001F   7059 ;       GOOD FIRST CHOICE TO SWAP.  IT CONTAINS MOST OF THE DATA PATH
001F   7060 ;       INVOLVED IN THIS TEST.  SECOND CHOICE SHOULD BE "ADD" WHICH IS
001F   7061 ;       RESPONSIBLE FOR THE ADDRESSING.  IF PROBLEMS STILL PERSIST, THEN YOU
001F   7062 ;       MIGHT TRY SWAPPING "PRK" WHICH CONTROLS THE PREFETCH LOGIC.
001F   7063 ;       TO HELP ISOLATE PROBLEMS THE FIRST SUBTEST EXAMINES THE MSRC XB H
001F   7064 ;       SIGNAL ON THE VBUS FOR YOU.
```

```
                        ;       IF XB1 CONSISTANTLY CONTAINS XB0 DATA YOU MIGHT CHECK THE LATCH MA
                        ;       SIGNAL COMING FROM THE PRK GATE ARRAY.  THIS SIGNAL CAN MESS-UP THE
                        ;       PREFETCH LOGIC WHICH IS INVOKED WHEN THE PC IS LOADED.
                        ;
                        ; ASSUMPTIONS:
                        ;
                        ;       THIS TEST ASSUMES THE VA, PC, CACHE, WBUS, AND MBUS ARE OPERATIONAL.
                        ;
                        ; ERROR DESCRIPTION:
                        ;
                        ;       SUBTEST #1 FIRST IFERROR:
                        ;               EXPECTED VBUS DATA (MSRC XB L)
                        ;               RECEIVED VBUS DATA
                        ;               LOOP COUNT I
                        ;                       (BITS <7:0> = BIT POSITION)
                        ;                       (BITS <8> = BIT STATE)
                        ;
                        ;       SUBTEST #1 SECOND IFERROR AND ONLY IFERROR IN ALL OTHERS:
                        ;               EXPECTED XB DATA
                        ;               RECEIVED XB DATA
                        ;               LOOP COUNT I
                        ;
                        ;--
                        ;*****************************************************************************
                        
                                SUBTEST                                         ;SUBTEST #1
                        
                        ;/////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       TEST XB0
                        ;-
                                INITIALIZE                                      ;INIT CPU
                                LOADDCS         5$,,15,1                        ;MODIFY DCS ADD OF
                                LOOP            I,1,6                           ;CREATE A TEST LOOP
                                LDFIELD         1$,I,L,3$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                                LOADDCS         3$,,06,1                        ;TEST PATTERN TO DCS
                                LDFIELD         2$,I,L,3$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                                LOADDCS         3$,,10,1                        ;TEST PATTERN TO DCS
                                ERRLOOP                                         ;ERROR LOOP
                                FETCH           0                               ;START DCS PROGRAM
                                BRSTCLK         <^X0F>                          ;END DCS PROGRAM
                                CMPVBUS         6$                              ;TEST MSRC XB SIGNAL
                                IFERROR                                         ;SIGNAL NOT PRESENT
                                CMPREG          WHREG,1$,I,L,                   ;COMPARE RESULTS
                                IFERROR         ,<<MDR><ADD><PRK>>,
                                ENDLOOP         I                               ;END TEST LOOP
                        
                                SUBTEST                                         ;SUBTEST #2
                        
                        ;/////////////////////////////////////////////////////////////////////////////
                        ;+
```

```
                        008E  7116 ;SUBTEST #2
                        008E  7117 ;
                        008E  7118 ;       TEST XB1
                        008E  7119 ;-
                        008E  7120         INITIALIZE                                  ;INIT CPU
                        0090  7121         LOADDCS         4$,,15,1                    ;SOURCE XB1 TO RDM MICRO WORD
                        0097  7122         LOOP            I,1,6                       ;CREATE A TEST LOOP
                        009E  7123         LDFIELD         1$,I,L,3$,31,62,LON         ;TEST PATTERN TO MICRO WORD
                        00AF  7124         LOADDCS         3$,,06,1                    ;TEST PATTERN TO DCS
                        00B6  7125         LDFIELD         2$,I,L,3$,31,62,LON         ;TEST PATTERN TO MICRO WORD
                        00C7  7126         LOADDCS         3$,,10,1                    ;TEST PATTERN TO DCS
                        00CE  7127         ERRLOOP                                     ;ERROR LOOP
                        00D0  7128         FETCH           0                           ;START DCS PROGRAM
                        00D5  7129         BRSTCLK         <^X10>                      ;END DCS PROGRAM
                        00D9  7130         CMPREG          WHREG,2$,I,L,               ;COMPARE RESULTS
                        00E2  7131         IFERROR         ,<<MDR><ADD><PRK>>,
                        00EA  7132         ENDLOOP         I                           ;END TEST LOOP
                        00ED  7133         SKIP                                        ;GO TO NEXT TEST
                        00F4  7134
                        00F4  7135 BEGIN_TEST_DATA
                        00F4
                        00F4       ;-------------------------------------------------------------------------
                        00F4  7136
AAAAAAAA                00F4  7137 1$:     .LONG    ^XAAAAAAAA                         ;TEST PATTERNS FOR XB0
55555555                00F8  7138         .LONG    ^X55555555
33333333                00FC  7139         .LONG    ^X33333333
0F0F0F0F                0100  7140         .LONG    ^X0F0F0F0F
00FF00FF                0104  7141         .LONG    ^X00FF00FF
0000FFFF                0108  7142         .LONG    ^X0000FFFF
                        010C  7143
0000FFFF                010C  7144 2$:     .LONG    ^X0000FFFF                         ;TEST PATTERNS FOR XB1
00FF00FF                0110  7145         .LONG    ^X00FF00FF
0F0F0F0F                0114  7146         .LONG    ^X0F0F0F0F
33333333                0118  7147         .LONG    ^X33333333
55555555                011C  7148         .LONG    ^X55555555
AAAAAAAA                0120  7149         .LONG    ^XAAAAAAAA
                        0124  7150 3$:
U 06, 7700,7800,7FFF,FFFF,8470,1807     0124  7151 ;% 06    LONLIT_[0]                 ;TEST PATTERN TO LONGLIT REG
                        012F  7155
                        012F  7156 4$:
U 0F, 6701,4817,5924,0202,0470,1810     012F  7157 ;% 0F    RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I]  ;SOURCE XB1 TO RDM
                        013A  7161
                        013A  7162 5$:
U 0F, 7300,C800,0364,0300,0470,1810     013A  7163 ;% 0F    NOP,STOP.CLOCK             ;DCS ADD OF SUBTESTS #1
                     01 0145  7167 6$:     .BYTE    ^X1                                ;TEST MSRC XB
                        0146  7168         VBUSDATA         <^X10>,1
                        0147  7169
                        0147  7170 BEGIN_CPU_DATA
                        0002  7171
                        0002  7172 DCS_DATA
                        0001  7173
U 00, 7700,C800,0364,0300,0470,1801     0001  7174 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802     000C  7178 ;% 01    MEMSCAR_R[TEMP0]           ;PHY/VIR REG ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803     0017  7182 ;% 02    MEMSCR_R[TEMP1]            ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804     0022  7186 ;% 03    MEMSCAR_R[TEMP2]           ;REF CACHE ADD TO MEMSCAR
```

```
U 04. 7700,C800,5BE4,030C,6070,1805  002D  7190 ;% 04   MEMSCR_R[TEMP3]                                  ;TURN CMI OFF
U 05. 7700,0800,5BE4,0310,4A70,1806  0038  7194 ;% 05   VA_R[TEMP4]                                      ;0'S TO VA REGISTER
U 06. 7700,7800,7FFF,FFFF,8470,1807  0043  7198 ;% 06   LONLIT_[0]                                      ;TEST PATTERN TO LONGLIT
U 07. 7701,8800,5BE4,02D4,5590,1808  004E  7202 ;% 07   WRITE.LONG R[LONLIT]                            ;TEST PATTERN TO CACHE ADD 0
U 08. 7700,4800,5BE4,03D4,5470,1809  0059  7206 ;% 08   WB_R[LONLIT],WCTRL/WDR_WB.UR                    ;HOLD DATA FOR EXTRA CYCLE
U 09. 7700,C800,0364,0300,4470,180A  0064  7210 ;% 09   VA_VA+4                                        ;INCREMENT VA BY 4
U 0A. 7700,7800,7FFF,FFFF,8470,180B  006F  7214 ;% 0A   LONLIT_[0]                                      ;TEST PATTERN TO LONGLIT
U 0B. 7701,0800,5BE4,02D4,5590,180C  007A  7218 ;% 0B   WRITE.LONG R[LONLIT]                            ;TEST PATTERN TO CACHE ADD 4
U 0C. 7700,C800,5BE4,03D4,5470,180D  0085  7222 ;% 0C   WB_R[LONLIT],WCTRL/WDR_WB.UR                    ;HOLD DATA FOR EXTRA CYCLE
U 0D. 7700,0800,5BE4,0310,4870,180E  0090  7226 ;% 0D   PC_R[TEMP4]                                     ;0'S TO PC
U 0E. 6601,C817,5924,0202,0470,180F  009B  7230 ;% 0E   RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I],STROBE.V.BUS ;READ XB0
U 0F. 7300,C800,0364,0300,0470,1810  00A6  7234 ;% 0F   NOP,STOP.CLOCK                                  ;STOP CPU CLOCK
U 10. 7300,4800,0364,0300,0470,1811  00B1  7238 ;% 10   NOP,STOP.CLOCK                                  ;STOP CPU CLOCK
                                     00BC  7242
                                     00BC  7243 RTEMP_DATA
                                     0001  7244
                           00000000  0001  7245          .LONG   ^X00000000                          ;PHY/VIR REG ADDRESS
                           00000000  0005  7246          .LONG   ^X00000000                          ;SET MME OFF
                           0E000000  0009  7247          .LONG   ^X0E000000                          ;REF CACHE ONLY ADDRESS
                           01000000  000D  7248          .LONG   ^X01000000                          ;TURN CMI OFF
                           00000000  0011  7249          .LONG   ^X00000000                          ;ADDRESS FOR VA & PC
                                     0015  7250
                                     0015  7251 END_CPU_DATA
                                     0147  7252
                                     0147  7253 END_TEST_DATA
                                     0147
                                     0147       ;-------------------------------------------------------------------------
                                     0147  7254
                                     0147  7255 ENDTEST
```

```
001E                    .SBTTL  "TEST   051     PREFETCH INCREMENTER TEST
001E    7257 ;*****************************************************************************
001E    7258 ;++
001E    7259 ;
001E    7260 ; FUNCTIONAL DESCRIPTION
001E    7261 ;
001E    7262 ;       This test will examine the combinatorial logic that implements the
001E    7263 ;       PC+4 increment used in instruction prefetch. The instruction prefetch
001E    7264 ;       operation must be used to implement this test because the PC+4
001E    7265 ;       increment logic cannot be sourced directly onto the MBUS. Therefore
001E    7266 ;       it is necessary to initiate prefetch by loading the PC with an address
001E    7267 ;       and then comparing the data written into XB0 and XB1 with the expected
001E    7268 ;       data stored in the RDM memory. If the two are different an error has
001E    7269 ;       occurred.
001E    7270 ;
001E    7271 ; TEST ALGORITHM
001E    7272 ;
001E    7273 ;       1.  Initialize the CPU to clear the CACHE and TB.
001E    7274 ;       2.  Disable the CMI.
001E    7275 ;       3.  Enable memory management, TB and CACHE.
001E    7276 ;       4.  Load the cache with the valid PC data code at addresses specified
001E    7277 ;           by the VA.
001E    7278 ;       5.  Load the cache with the valid PC+4 data code at addresses
001E    7279 ;           specified by the +4 increments of the VA.
001E    7280 ;       6.  Initiate instruction prefetch by loading the PC with an address.
001E    7281 ;       7.  Wait for prefetch to finish.
001E    7282 ;       8.  Read XB0 to the RDM and check for the PC valid data code.
001E    7283 ;           If not present, signal an error.
001E    7284 ;       9.  Increment the PC by four.
001E    7285 ;       10. Read XB1 to the RDM and check for the PC+4 valid data code.
001E    7286 ;           If not present, signal an error.
001E    7287 ;       11. Continue steps 6-10 for all 32 address test patterns.
001E    7288 ;       12. Test completed.
001E    7289 ;
001E    7290 ; DATA PATTERNS
001E    7291 ;
001E    7292 ;       Thirty-one address patterns are required to fully test the increment
001E    7293 ;       logic. They are listed below:
001E    7294 ;
001E    7295 ;               LOOP COUNT          PC  ADDRESS             PC+4  ADDRESS
001E    7296 ;
001E    7297 ;       1           1           F F F F F F F 8         F F F F F F F C
001E    7298 ;       2           2           F F F F F F F 4         F F F F F F F 8
001E    7299 ;       3           3           F F F F F F E C         F F F F F F F 0
001E    7300 ;       4           4           F F F F F F D C         F F F F F F E 0
001E    7301 ;       5           5           F F F F F F B C         F F F F F F C 0
001E    7302 ;       6           6           F F F F F F 7 C         F F F F F F 8 0
001E    7303 ;       7           7           F F F F F E F C         F F F F F F 0 0
001E    7304 ;       8           8           F F F F F D F C         F F F F F E 0 0
001E    7305 ;       9           9           F F F F F B F C         F F F F F C 0 0
001E    7306 ;       10          A           F F F F F 7 F C         F F F F F 8 0 0
001E    7307 ;       11          B           F F F F E F F C         F F F F F 0 0 0
001E    7308 ;       12          C           F F F F D F F C         F F F F E 0 0 0
001E    7309 ;       13          D           F F F F B F F C         F F F F C 0 0 0
001E    7310 ;       14          E           F F F F 7 F F C         F F F F 8 0 0 0
```

```
001E ;       15          F          F F F E F F F C        F F F F 0 0 0 0
001E ;       16          10         F F F D F F F C        F F F E 0 0 0 0
001E ;       17          11         F F F B F F F C        F F F C 0 0 0 0
001E ;       18          12         F F F 7 F F F C        F F F 8 0 0 0 0
001E ;       19          13         F F E F F F F C        F F F 0 0 0 0 0
001E ;       20          14         F F D F F F F C        F F E 0 0 0 0 0
001E ;       21          15         F F B F F F F C        F F C 0 0 0 0 0
001E ;       22          16         F F 7 F F F F C        F F 8 0 0 0 0 0
001E ;       23          17         F E F F F F F C        F F 0 0 0 0 0 0
001E ;       24          18         F D F F F F F C        F E 0 0 0 0 0 0
001E ;       25          19         F B F F F F F C        F C 0 0 0 0 0 0
001E ;       26          1A         F 7 F F F F F C        F 8 0 0 0 0 0 0
001E ;       27          1B         E F F F F F F C        F 0 0 0 0 0 0 0
001E ;       28          1C         D F F F F F F C        E 0 0 0 0 0 0 0
001E ;       29          1D         B F F F F F F C        C 0 0 0 0 0 0 0
001E ;       30          1E         7 F F F F F F C        8 0 0 0 0 0 0 0
001E ;       31          1F         F F F F F F F C        0 0 0 0 0 0 0 0
001E ;
001E ;       Cache to XB valid data codes:
001E ;
001E ;               PC valid data code      XB0 = A A A A A A A A
001E ;               PC+4 valid data code    XB1 = 5 5 5 5 5 5 5 5
001E ;
001E ; LOGIC DESCRIPTION
001E ;
001E ;       This test consists of two parts. The first part examines the low order
001E ;       two bits of address to make sure they are not stuck at one or zero.
001E ;       The second part will test the carry logic within the incrementer by
001E ;       floating a zero across all bits in the PC while rippling a carry into
001E ;       the zero from the incrementer.
001E ;
001E ; ASSUMPTIONS
001E ;
001E ;       This test assumes that the DPM, WBUS and MBUS are functional. Since
001E ;       this test requires the use of the instuction prefetch mechanism, all
001E ;       tests on the VA, PC, the two execution buffers, cache, translation
001E ;       buffer and WDR must also have been successfully completed.
001E ;
001E ; ERROR DESCRIPTION
001E ;
001E ;       Below is a table of possible error symptoms and their most likely
001E ;       causes:
001E ;
001E ;               Symptom                         Probable Cause
001E ;       +-----------------------+-----------------------------------------+
001E ;       | An error is called out| A defective PRK gate array is causing   |
001E ;       | regardless of whether | the incrementer or PC from being sent   |
001E ;       | the expected data was | out on the address lines.               |
001E ;       | AAAAAAAA or 55555555. |                                         |
001E ;       +-----------------------+-----------------------------------------+
001E ;       | An error is called out| A problem with either the logic used    |
001E ;       | when the expected data| to load the PC or the PC register       |
001E ;       | was AAAAAAAA.         | itself. The PC register diagnostic      |
001E ;       |                       | should be run to isolate the problem.   |
001E ;       +-----------------------+-----------------------------------------+
```

```
001E  7366 ;       | An error is called out | The ADD gate array used to generate
001E  7367 ;       | with the expected data | bits <07:00> of the memory address.
001E  7368 ;       | being 55555555 and the |
001E  7369 ;       | loop count was in the  | Printset: MIC-03  E9
001E  7370 ;       | range of 1 to 6 hex.   |
001E  7371 ;       +------------------------+-----------------------------------------+
001E  7372 ;       | An error is called out | The ADD gate array used to generate
001E  7373 ;       | with the expected data | bits <15:08> of the memory address.
001E  7374 ;       | being 55555555 and the |
001E  7375 ;       | loop count was in the  | Printset: MIC-03  E10
001E  7376 ;       | range of 7 to E hex.   |
001E  7377 ;       +------------------------+-----------------------------------------+
001E  7378 ;       | An error is called out | The ADD gate array used to generate
001E  7379 ;       | with the expected data | bits <23:16> of the memory address.
001E  7380 ;       | being 55555555 and the |
001E  7381 ;       | loop count was in the  | Printset: MIC-03  E11
001E  7382 ;       | range of 0F to 16 hex. |
001E  7383 ;       +------------------------+-----------------------------------------+
001E  7384 ;       | An error is called out | The ADD gate array used o generate
001E  7385 ;       | with the expected data | bits <31:24> of the memory address.
001E  7386 ;       | being 55555555 and the |
001E  7387 ;       | loop count was in the  | Printset: MIC-03  E12
001E  7388 ;       | range of 17 to 1E hex. |
001E  7389 ;       +------------------------+-----------------------------------------+
001E  7390 ;       | An error is called out | A problem with the carry out from the
001E  7391 ;       | when the loop count    | incrementer not being passed onto the
001E  7392 ;       | was equal to 1F hex.   | next higher order gate array. The
001E  7393 ;       |                        | carry out - carry in connection
001E  7394 ;       |                        | between the gate arrays should be
001E  7395 ;       |                        | checked.
001E  7396 ;       +------------------------+-----------------------------------------+
001E  7397 ;
001E  7398 ;       On error this test will call out the ADD and PRK gate array on the
001E  7399 ;       MIC module. The following information will also by typed out:
001E  7400 ;
001E  7401 ;       Expected XB data:
001E  7402 ;       Received XB data:
001E  7403 ;       Loop Count:
001E  7404 ;
001E  7405 ;--
001E  7406 ;*******************************************************************************
001E  7407 ;///////////////////////////////////////////////////////////////////////////////
001E  7408 ;+
001E  7409 ; Memory control register assignments
001E  7410 ;
001E  7411 ;       Disables the CMI and enables memory management, cache and TB.
001E  7412 ;-
001E  7413         INITIALIZE
0020  7414         LOADDCS         6$,,0,18
0027  7415         FETCH           0
002C  7416         BRSTCLK         <^X11>
0030  7417 ;
0030  7418 ; This code invalidates the Translation Buffers.
0030  7419 ;
0030  7420         LOADDCS         7$,,0,4
```

```
                0037  7421          FETCH           0
                003C  7422          BRSTCLK         3
                0040  7423          LOADDCS         8$,,0,4
                0047  7424          LOOP            I,1,128
                004E  7425          FETCH           0
                0053  7426          BRSTCLK         3
                0057  7427          ENDLOOP         I
                005A  7428 ;
                005A  7429 ; This code is used to invalidate all of the Cache.
                005A  7430 ;
                005A  7431          LOADDCS         7$,,0,4
                0061  7432          FETCH           0
                0066  7433          BRSTCLK         3
                006A  7434          LOADDCS         9$,,0,4
                0071  7435          LOOP            I,1,1024
                0078  7436          FETCH           0
                007D  7437          BRSTCLK         3
                0081  7438          ENDLOOP         I
                0084  7439 ;
                0084  7440 ; This code examines the prefetch incrementer.
                0084  7441 ;
                0084  7442          LOADDCS         10$,,0,<^X1F>
                008B  7443          LOOP            I,1,31
                0092  7444          LOADREG         WDREG,1$,I,LONG
                009A  7445          EXPUTRAP
                009C  7446          ERRLOOP
                009E  7447          FETCH           0
                00A3  7448          BRSTCLK         <^X16>
                00A7  7449          CMPREGMSK       CSTRP,4$,,5$,,WORD
                00B3  7450          IFERROR         ,<<ADD>,<PRK>>
                00BA  7451          CMPREG          WHREG,2$,,LONG
                00C3  7452          IFERROR         ,<<ADD>,<PRK>>
                00CA  7453          FETCH           <^X14>
                00CF  7454          BRSTCLK         <^X16>
                00D3  7455          CMPREGMSK       CSTRP,4$,,5$,,WORD
                00DF  7456          IFERROR         ,<<ADD>,<PRK>>
                00E6  7457          CMPREG          WHREG,3$,,LONG
                00EF  7458          IFERROR         ,<<ADD>,<PRK>>
                00F6  7459          FETCH           <^X17>
                00FB  7460          BRSTCLK         <^X1E>
                00FF  7461          ENDLOOP         I
                0102  7462          SKIP
                0109  7463
                0109  7464 BEGIN_TEST_DATA
                0109
                0109               ;-------------------------------------------------------------------------
                0109  7465
FFFFFFF8        0109  7466 1$:      .LONG    ^X FFFFFFF8                ; PC addresses
FFFFFFF4        010D  7467          .LONG    ^X FFFFFFF4
FFFFFFEC        0111  7468          .LONG    ^X FFFFFFEC
FFFFFFDC        0115  7469          .LONG    ^X FFFFFFDC
FFFFFFBC        0119  7470          .LONG    ^X FFFFFFBC
FFFFFF7C        011D  7471          .LONG    ^X FFFFFF7C
FFFFFEFC        0121  7472          .LONG    ^X FFFFFEFC
FFFFFDFC        0125  7473          .LONG    ^X FFFFFDFC
```

```
                FFFFFBFC  0129  7474          .LONG   ^X FFFFFBFC
                FFFFF7FC  012D  7475          .LONG   ^X FFFFF7FC
                FFFFEFFC  0131  7476          .LONG   ^X FFFFEFFC
                FFFFDFFC  0135  7477          .LONG   ^X FFFFDFFC
                FFFFBFFC  0139  7478          .LONG   ^X FFFFBFFC
                FFFF7FFC  013D  7479          .LONG   ^X FFFF7FFC
                FFFEFFFC  0141  7480          .LONG   ^X FFFEFFFC
                FFFDFFFC  0145  7481          .LONG   ^X FFFDFFFC
                FFFBFFFC  0149  7482          .LONG   ^X FFFBFFFC
                FFF7FFFC  014D  7483          .LONG   ^X FFF7FFFC
                FFEFFFFC  0151  7484          .LONG   ^X FFEFFFFC
                FFDFFFFC  0155  7485          .LONG   ^X FFDFFFFC
                FFBFFFFC  0159  7486          .LONG   ^X FFBFFFFC
                FF7FFFFC  015D  7487          .LONG   ^X FF7FFFFC
                FEFFFFFC  0161  7488          .LONG   ^X FEFFFFFC
                FDFFFFFC  0165  7489          .LONG   ^X FDFFFFFC
                FBFFFFFC  0169  7490          .LONG   ^X FBFFFFFC
                F7FFFFFC  016D  7491          .LONG   ^X F7FFFFFC
                EFFFFFFC  0171  7492          .LONG   ^X EFFFFFFC
                DFFFFFFC  0175  7493          .LONG   ^X DFFFFFFC
                BFFFFFFC  0179  7494          .LONG   ^X BFFFFFFC
                7FFFFFFC  017D  7495          .LONG   ^X 7FFFFFFC
                FFFFFFFC  0181  7496          .LONG   ^X FFFFFFFC
                          0185  7497
                AAAAAAAA  0185  7498 2$:      .LONG   ^X AAAAAAAA            ; PC address valid data code
                55555555  0189  7499 3$:      .LONG   ^X 55555555            ; PC+4 address valid data code
                          018D  7500
                     1816 018D  7501 4$:      .WORD   ^X 1816                ; EXPECTED CS ADDRESS
                     3FFF 018F  7502 5$:      .WORD   ^X 3FFF                ; CS ADDRESS MASK
                          0191  7503
                          0191  7504 6$:
U 00, 7700,C800,0364,0300,0470,1801   0191  7505 ;% 00    NOP
U 01, 7700,3800,78FF,FFFF,8470,1802   019C  7509 ;% 01    LONLIT_[0E000000]      ; Address of CMI enable/disable MEMSCR
U 02, 7700,4800,5BE4,03D4,6870,1803   01A7  7513 ;% 02    MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804   01B2  7517 ;% 03    LONLIT_[01000000]      ; Disables CMI
U 04, 7700,C800,5BE4,03D4,6070,1805   01BD  7521 ;% 04    MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806   01C8  7525 ;% 05    LONLIT_[0]             ; Address of MM enable/disable MEMSCR
U 06, 7700,C800,5BE4,03D4,6870,1807   01D3  7529 ;% 06    MEMSCAR_R[LONLIT]
U 07, 7700,3800,7F7F,FFFF,8470,1808   01DE  7533 ;% 07    LONLIT_[01000000]      ; Enables memory management
U 08, 7700,C800,5BE4,03D4,6070,1809   01E9  7537 ;% 08    MEMSCR_R[LONLIT]
U 09, 7700,F800,7E7F,FFFF,8470,180A   01F4  7541 ;% 09    LONLIT_[03000000]      ; Address of TB enable/disable MEMSCR
U 0A, 7700,C800,5BE4,03D4,6870,180B   01FF  7545 ;% 0A    MEMSCAR_R[LONLIT]
U 0B, 7700,F800,7AFF,FFFF,8470,180C   020A  7549 ;% 0B    LONLIT_[0A000000]      ; Enables TB group 0 only
U 0C, 7700,4800,5BE4,03D4,6070,180D   0215  7553 ;% 0C    MEMSCR_R[LONLIT]
U 0D, 7700,7800,7CFF,FFFF,8470,180E   0220  7557 ;% 0D    LONLIT_[06000000]      ; Address of CACHE enable/disable MEMSCR
U 0E, 7700,4800,5BE4,03D4,6870,180F   022B  7561 ;% 0E    MEMSCAR_R[LONLIT]
U 0F, 7700,7800,7FFF,FFFF,8470,1810   0236  7565 ;% 0F    LONLIT_[0]             ; Enables CACHE
U 10, 7700,C800,5BE4,03D4,6070,1811   0241  7569 ;% 10    MEMSCR_R[LONLIT]
U 11, 7300,4800,0364,0300,0470,1812   024C  7573 ;% 11    STOP.CLOCK
                          0257  7577
                          0257  7578 7$:
U 00, 7700,C800,0364,0300,0470,1801   0257  7579 ;% 00    NOP
U 01, 7700,3800,7FFF,FDFF,8470,1802   0262  7583 ;% 01    LONLIT_[400]
U 02, 7700,0800,5BE6,03D8,4A70,1803   026D  7587 ;% 02    D_VA_0
U 03, 7300,C800,0364,0300,0470,1804   0278  7591 ;% 03    STOP.CLOCK
```

```
                                0283  7595
                                0283  7596 8$:
U 00, 7700,C800,0364,0300,0470,1801  0283  7597 ;% 00   NOP
U 01, 7700,C800,0224,03D8,5360,1802  028E  7601 ;% 01   CLRTB.VA_D                ; WCTRL/CLRTB.VA_WB,WB_D
U 02, 7700,4800,0212,03D4,0470,1803  0299  7605 ;% 02   D_D+R[LONLIT]
U 03, 7300,C800,0364,0300,0470,1804  02A4  7609 ;% 03   STOP.CLOCK
                                02AF  7613
                                02AF  7614 9$:
U 00, 7700,C800,0364,0300,0470,1801  02AF  7615 ;% 00   NOP
U 01, 7700,081B,0024,03D8,5A70,1802  02BA  7619 ;% 01   CLRCH.VA_VA               ; WCTRL/CLRCH.VA_WB,WB_VA
U 02, 7700,C800,0364,0300,4470,1803  02C5  7623 ;% 02   VA_VA+4
U 03, 7300,C800,0364,0300,0470,1804  02D0  7627 ;% 03   STOP.CLOCK
                                02DB  7631
                                02DB  7632 10$:
U 00, 7700,C800,0364,0300,0470,1801  02DB  7633 ;% 00   NOP
U 01, 5700,C800,0364,0300,4A70,1802  02E6  7637 ;% 01   VA_RDM
U 02, 7700,B800,7FFF,FF5B,8470,1803  02F1  7641 ;% 02   LONLIT_[00000148]
U 03, 7700,481B,5BE4,03D4,5070,1804  02FC  7645 ;% 03   TB_R[LONLIT]
U 04, 7700,F800,2AAA,AAAA,8470,1805  0307  7649 ;% 04   LONLIT_[0AAAAAAAA]
U 05, 7701,0800,5BE4,02D4,5590,1806  0312  7653 ;% 05   WRITE.LONG R[LONLIT]
U 06, 7700,C800,5BE4,03D4,5470,1807  031D  7657 ;% 06   WDR_UNROT(R[LONLIT])
U 07, 7700,4800,0364,0300,4470,1808  0328  7661 ;% 07   VA_VA+4
U 08, 7700,B800,7FFF,FF5B,8470,1809  0333  7665 ;% 08   LONLIT_[00000148]
U 09, 7700,C81B,5BE4,03D4,5070,180A  033E  7669 ;% 09   TB_R[LONLIT]
U 0A, 7700,B800,5555,5555,0470,180B  0349  7673 ;% 0A   LONLIT_[55555555]
U 0B, 7701,0800,5BE4,02D4,5590,180C  0354  7677 ;% 0B   WRITE.LONG R[LONLIT]
U 0C, 7700,C800,5BE4,03D4,5470,180D  035F  7681 ;% 0C   WDR_UNROT(R[LONLIT])
U 0D, 7700,4800,0364,0300,4470,180E  036A  7685 ;% 0D   VA_VA+4
U 0E, 7700,B800,7FFF,FF5B,8470,180F  0375  7689 ;% 0E   LONLIT_[00000148]
U 0F, 7700,481B,5BE4,03D4,5070,1810  0380  7693 ;% 0F   TB_R[LONLIT]
U 10, 7700,3800,0000,0000,0470,1811  038B  7697 ;% 10   LONLIT_[0FFFFFFFF]
U 11, 7701,0800,5BE4,02D4,5590,1812  0396  7701 ;% 11   WRITE.LONG R[LONLIT]
U 12, 7700,C800,5BE4,03D4,5470,1813  03A1  7705 ;% 12   WDR_UNROT(R[LONLIT])
U 13, 5700,C800,0364,0300,4870,1814  03AC  7709 ;% 13   PC_RDM
U 14, 7700,C800,0364,0300,0470,1815  03B7  7713 ;% 14   NOP
U 15, 6701,4817,0024,02DA,0470,1816  03C2  7717 ;% 15   PC_PC+4 RDM_XB
U 16, 7300,4800,0364,0300,0470,1817  03CD  7721 ;% 16   STOP.CLOCK
U 17, 7700,4800,0364,0300,0470,1818  03D8  7725 ;% 17   NOP
U 18, 5700,C800,0364,0300,4A70,1819  03E3  7729 ;% 18   VA_RDM
U 19, 7701,8800,5BE4,02D8,5590,181A  03EE  7733 ;% 19   WRITE.LONG R[ZERO]
U 1A, 7700,4800,5BE4,03D8,5470,181B  03F9  7737 ;% 1A   WDR_UNROT(R[ZERO])
U 1B, 7700,4800,0364,0300,4470,181C  0404  7741 ;% 1B   VA_VA+4
U 1C, 7701,0800,5BE4,02D8,5590,181D  040F  7745 ;% 1C   WRITE.LONG R[ZERO]
U 1D, 7700,4800,5BE4,03D8,5470,181E  041A  7749 ;% 1D   WDR_UNROT(R[ZERO])
U 1E, 7300,C800,0364,0300,0470,181F  0425  7753 ;% 1E   STOP.CLOCK
                                0430  7757
                                0430  7758 END_TEST_DATA
                                0430
                                0430       ;-------------------------------------------------------------------------------
                                0430  7759
                                0430  7760 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0033        .SBTTL  "TEST   052     Test PCBACK While PC Increment Logic is Active
0033  7762 ;*******************************************************************************
0033  7763 ;++
0033  7764 ;
0033  7765 ; FUNCTIONAL DESCRIPTION
0033  7766 ;
0033  7767 ;       This test will check the side-affect of the PCBACK register while
0033  7768 ;       incrementing through the execution buffer.  The purpose of this test is
0033  7769 ;       to simulate the case where the CPU is incrementing through the
0033  7770 ;       execution buffer during instruction execution and, during this
0033  7771 ;       incrementation, there is a fault.  After the machine handles this fault
0033  7772 ;       the instruction should be restartable by taking the contents of the
0033  7773 ;       PCBACK register minus 2 for restart PC.  This test will check the
0033  7774 ;       integrity of the PCBACK regsiter to make sure, after performing the
0033  7775 ;       incrementation, it contains the correct value.
0033  7776 ;
0033  7777 ; TEST ALGORITHM
0033  7778 ;
0033  7779 ;       1.  Initialize the CPU to clear the CACHE and TB.
0033  7780 ;       2.  Disable the CMI.
0033  7781 ;       3.  Enable memory management, TB and CACHE.
0033  7782 ;       4.  Load the cache with the valid PC data code at addresses specified
0033  7783 ;           by the VA.
0033  7784 ;       5.  Load the cache with the valid PC+4 data code at addresses
0033  7785 ;           specified by the +4 increments of the VA.
0033  7786 ;       6.  Initiate instruction prefetch by loading the PC with an address.
0033  7787 ;       7.  Disable Ctrl P interrupts so there will be no micro trap when doing
0033  7788 ;           doing IRD1TEST.
0033  7789 ;       8.  Do IRD1TEST which causes the PCBACK & PC to get loaded with PC + 2.
0033  7790 ;       9.  Perform PC_PC+1 MB_XB which is equivalent to sequencing through the
0033  7791 ;           XB with ISIZE set to Byte.
0033  7792 ;       10. Read the PCBACK register.
0033  7793 ;       11. Store the updated PC (+1) in R[TEMP0].  This is used in the error
0033  7794 ;           callout.
0033  7795 ;       12. Check the WHREG (stored PCBACK) for valid data.
0033  7796 ;           If not valid then flag an error.  Note that the lower two bits
0033  7797 ;           (bits 1:0) are not masked when doing the comparison.  This is
0033  7798 ;           because the table used for comparison contains minus two of the
0033  7799 ;           PCBACK register.
0033  7800 ;       13. Continue steps 8-12 for 8 times.
0033  7801 ;       11. Continue steps 4-13 for all 32 address test patterns.
0033  7802 ;       12. Test completed.
0033  7803 ;
0033  7804 ; DATA PATTERNS
0033  7805 ;
0033  7806 ;       In floating a zero through the address Thirty-one patterns are
0033  7807 ;       required.  They are listed below:
0033  7808 ;
0033  7809 ;               LOOP COUNT          PC  ADDRESS
0033  7810 ;
0033  7811 ;       1           1           F F F F F F F 8
0033  7812 ;       2           2           F F F F F F F 4
0033  7813 ;       3           3           F F F F F F E C
0033  7814 ;       4           4           F F F F F F D C
0033  7815 ;       5           5           F F F F F F B C
```

```
0033  7816 ;        6           6           F F F F F F 7 C
0033  7817 ;        7           7           F F F F F E F C
0033  7818 ;        8           8           F F F F F D F C
0033  7819 ;        9           9           F F F F F B F C
0033  7820 ;        10          A           F F F F F 7 F C
0033  7821 ;        11          B           F F F F E F F C
0033  7822 ;        12          C           F F F F D F F C
0033  7823 ;        13          D           F F F F B F F C
0033  7824 ;        14          E           F F F F 7 F F C
0033  7825 ;        15          F           F F F E F F F C
0033  7826 ;        16          10          F F F D F F F C
0033  7827 ;        17          11          F F F B F F F C
0033  7828 ;        18          12          F F F 7 F F F C
0033  7829 ;        19          13          F F E F F F F C
0033  7830 ;        20          14          F F D F F F F C
0033  7831 ;        21          15          F F B F F F F C
0033  7832 ;        22          16          F F 7 F F F F C
0033  7833 ;        23          17          F E F F F F F C
0033  7834 ;        24          18          F D F F F F F C
0033  7835 ;        25          19          F B F F F F F C
0033  7836 ;        26          1A          F 7 F F F F F C
0033  7837 ;        27          1B          E F F F F F F C
0033  7838 ;        28          1C          D F F F F F F C
0033  7839 ;        29          1D          B F F F F F F C
0033  7840 ;        30          1E          7 F F F F F F C
0033  7841 ;        31          1F          F F F F F F F C
0033  7842 ;
0033  7843 ;        Cache to XB valid data codes:
0033  7844 ;
0033  7845 ;                PC valid data code      XB0 = A A A A A A A A
0033  7846 ;                PC+4 valid data code    XB1 = 5 5 5 5 5 5 5 5
0033  7847 ;
0033  7848 ; LOGIC DESCRIPTION
0033  7849 ;
0033  7850 ;        This test checks the side-affect, during the PC increment, on the
0033  7851 ;        PCBACK register.  This is done by floating a zero through the PC and,
0033  7852 ;        during each pc value, doing an IRD1TEST then incrementing through the
0033  7853 ;        execution buffer, checking the value of PCBACK after every increment.
0033  7854 ;
0033  7855 ; ASSUMPTIONS
0033  7856 ;
0033  7857 ;        This test assumes that the DPM, WBUS and MBUS are functional. Since
0033  7858 ;        this test requires the use of the instuction prefetch mechanism, all
0033  7859 ;        tests on the VA, PC, the two execution buffers, cache, translation
0033  7860 ;        buffer and WDR must also have been successfully completed.
0033  7861 ;
0033  7862 ; ERROR DESCRIPTION
0033  7863 ;
0033  7864 ;        First IFERROR:
0033  7865 ;
0033  7866 ;        Expected CS Address
0033  7867 ;        Received CS Address
0033  7868 ;        Loop I Count
0033  7869 ;        Loop J Count
0033  7870 ;
```

```
0033  7871 ;       Second IFERROR:
0033  7872 ;
0033  7873 ;       Expected PCBACK Register (bits <31:2>)
0033  7874 ;       Received PCBACK Register (bits <31:2>)
0033  7875 ;       Loop I Count
0033  7876 ;       Loop J Count
0033  7877 ;
0033  7878 ;--
0033  7879 ;*******************************************************************************
0033  7880 ;///////////////////////////////////////////////////////////////////////////////
0033  7881 ;+
0033  7882 ; Memory control register assignments
0033  7883 ;
0033  7884 ;       Disables the CMI and enables memory management, cache and TB.
0033  7885 ;-
0033  7886         INITIALIZE                                      ; Init the CPU
0035  7887         LOADDCS         6$,,0,18                        ; Load test code
003C  7888         FETCH           0                               ; start dcs
0041  7889         BRSTCLK         <^X11>                          ; end dcs
0045  7890 ;
0045  7891 ; This code invalidates the Translation Buffers.
0045  7892 ;
0045  7893         LOADDCS         7$,,0,4                         ; load set-up code
004C  7894         FETCH           0                               ; start dcs
0051  7895         BRSTCLK         3                               ; end dcs
0055  7896         LOADDCS         8$,,0,4                         ; load TB invalidate code
005C  7897         LOOP            I,1,128                         ; loop 128 times
0063  7898         FETCH           0                               ; start dcs
0068  7899         BRSTCLK         3                               ; end dcs
006C  7900         ENDLOOP         I                               ; continue loop until end
006F  7901 ;
006F  7902 ; This code is used to invalidate all of the Cache.
006F  7903 ;
006F  7904         LOADDCS         7$,,0,4                         ; load set-up code
0076  7905         FETCH           0                               ; start dcs
007B  7906         BRSTCLK         3                               ; end dcs
007F  7907         LOADDCS         9$,,0,4                         ; load cache invalidate code
0086  7908         LOOP            I,1,1024                        ; loop 1024 times
008D  7909         FETCH           0                               ; start dcs
0092  7910         BRSTCLK         3                               ; end dcs
0096  7911         ENDLOOP         I                               ; continue loop until end
0099  7912 ;
0099  7913 ; This code examines the side-affect on PCBACK
0099  7914 ;
0099  7915         LOADDCS         10$,,0,<^X24>                   ; Load test code
00A0  7916         LOOP            I,1,31                          ; set up loop I for 31 address
00A7  7917         LOADREG         WDREG,1$,I,LONG                 ; load current test address
00AF  7918         EXPUTRAP                                        ; expect micro trap
00B1  7919         ERRLOOP                                         ; Set up error scope loop
00B3  7920         FETCH           0                               ; start dcs
00B8  7921         BRSTCLK         <^X1B>                          ; end dcs
00BC  7922         LOOP            J,1,8                           ; set up loop J for 8 inc's
00C3  7923         CMPREGMSK       CSTRP,4$,,5$,,WORD              ; stop at correct CS address ?
00CF  7924         IFERROR         ,<<ADD>,<PRK>>                  ; if not flag error
00D6  7925         CMPREGMSK       WHREG,1$,I,3$,,LONG             ; PCBACK register correct ?
```

```
                    00E2  7926          IFERROR         1,<<ADD><PRK>>,         ; if not flag error
                    00E9  7927          FETCH           <^X17>                  ; start dcs
                    00EE  7928          BRSTCLK         <^X1B>                  ; end dcs
                    00F2  7929          ENDLOOP         J                       ; continue loop until end
                    00F5  7930          FETCH           <^X1C>                  ; start dcs (clean-up)
                    00FA  7931          BRSTCLK         <^X23>                  ; end dcs
                    00FE  7932          ENDLOOP         I                       ; continue loop until end
                    0101  7933          SKIP                                    ; go to next test
                    0108  7934
                    0108  7935 BEGIN_TEST_DATA
                    0108
                    0108          ;-------------------------------------------------------------------------
                    0108  7936
FFFFFFF8            0108  7937 1$:      .LONG   ^X FFFFFFF8             ; PC addresses
FFFFFFF4            010C  7938          .LONG   ^X FFFFFFF4             ; This table is also used when
FFFFFFEC            0110  7939          .LONG   ^X FFFFFFEC             ; comparing for correct PCBACK value.
FFFFFFDC            0114  7940          .LONG   ^X FFFFFFDC
FFFFFFBC            0118  7941          .LONG   ^X FFFFFFBC
FFFFFF7C            011C  7942          .LONG   ^X FFFFFF7C
FFFFFEFC            0120  7943          .LONG   ^X FFFFFEFC
FFFFFDFC            0124  7944          .LONG   ^X FFFFFDFC
FFFFFBFC            0128  7945          .LONG   ^X FFFFFBFC
FFFFF7FC            012C  7946          .LONG   ^X FFFFF7FC
FFFFEFFC            0130  7947          .LONG   ^X FFFFEFFC
FFFFDFFC            0134  7948          .LONG   ^X FFFFDFFC
FFFFBFFC            0138  7949          .LONG   ^X FFFFBFFC
FFFF7FFC            013C  7950          .LONG   ^X FFFF7FFC
FFFEFFFC            0140  7951          .LONG   ^X FFFEFFFC
FFFDFFFC            0144  7952          .LONG   ^X FFFDFFFC
FFFBFFFC            0148  7953          .LONG   ^X FFFBFFFC
FFF7FFFC            014C  7954          .LONG   ^X FFF7FFFC
FFEFFFFC            0150  7955          .LONG   ^X FFEFFFFC
FFDFFFFC            0154  7956          .LONG   ^X FFDFFFFC
FFBFFFFC            0158  7957          .LONG   ^X FFBFFFFC
FF7FFFFC            015C  7958          .LONG   ^X FF7FFFFC
FEFFFFFC            0160  7959          .LONG   ^X FEFFFFFC
FDFFFFFC            0164  7960          .LONG   ^X FDFFFFFC
FBFFFFFC            0168  7961          .LONG   ^X FBFFFFFC
F7FFFFFC            016C  7962          .LONG   ^X F7FFFFFC
EFFFFFFC            0170  7963          .LONG   ^X EFFFFFFC
DFFFFFFC            0174  7964          .LONG   ^X DFFFFFFC
BFFFFFFC            0178  7965          .LONG   ^X BFFFFFFC
7FFFFFFC            017C  7966          .LONG   ^X 7FFFFFFC
FFFFFFFC            0180  7967          .LONG   ^X FFFFFFFC
                    0184  7968
                    0184  7969 3$:
FFFFFFFC            0184  7970          .LONG   ^X FFFFFFFC
                    0188  7971
    181B            0188  7972 4$:      .WORD   ^X 181B                 ; EXPECTED CS ADDRESS
    3FFF            018A  7973 5$:      .WORD   ^X 3FFF                 ; CS ADDRESS MASK
                    018C  7974
                    018C  7975 6$:
U 00, 7700,C800,0364,0300,0470,1801   018C  7976 ;% 00    NOP
U 01, 7700,3800,78FF,FFFF,8470,1802   0197  7980 ;% 01    LONLIT_[0E000000]       ; Address of CMI enable/disable MEMSCR
U 02, 7700,4800,5BE4,03D4,6870,1803   01A2  7984 ;% 02    MEMSCAR_R[LONLIT]
```

```
U 03, 7700,3800,7F7F,FFFF,8470,1804  01AD  7988 ;% 03   LONLIT_[01000000]          ; Disables CMI
U 04, 7700,C800,5BE4,03D4,6070,1805  01B8  7992 ;% 04   MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806  01C3  7996 ;% 05   LONLIT_[0]                 ; Address of MM enable/disable MEMSCR
U 06, 7700,C800,5BE4,03D4,6870,1807  01CE  8000 ;% 06   MEMSCAR_R[LONLIT]
U 07, 7700,3800,7F7F,FFFF,8470,1808  01D9  8004 ;% 07   LONLIT_[01000000]          ; Enables memory management
U 08, 7700,C800,5BE4,03D4,6070,1809  01E4  8008 ;% 08   MEMSCR_R[LONLIT]
U 09, 7700,F800,7E7F,FFFF,8470,180A  01EF  8012 ;% 09   LONLIT_[03000000]          ; Address of TB enable/disable MEMSCR
U 0A, 7700,C800,5BE4,03D4,6870,180B  01FA  8016 ;% 0A   MEMSCAR_R[LONLIT]
U 0B, 7700,F800,7AFF,FFFF,8470,180C  0205  8020 ;% 0B   LONLIT_[0A000000]          ; Enables TB group 0 only
U 0C, 7700,4800,5BE4,03D4,6070,180D  0210  8024 ;% 0C   MEMSCR_R[LONLIT]
U 0D, 7700,7800,7CFF,FFFF,8470,180E  021B  8028 ;% 0D   LONLIT_[06000000]          ; Address of CACHE enable/disable MEMSCR
U 0E, 7700,4800,5BE4,03D4,6870,180F  0226  8032 ;% 0E   MEMSCAR_R[LONLIT]
U 0F, 7700,7800,7FFF,FFFF,8470,1810  0231  8036 ;% 0F   LONLIT_[0]                 ; Enables CACHE
U 10, 7700,C800,5BE4,03D4,6070,1811  023C  8040 ;% 10   MEMSCR_R[LONLIT]
U 11, 7300,4800,0364,0300,0470,1812  0247  8044 ;% 11   STOP.CLOCK
                                     0252  8048
                                     0252  8049 7$:
U 00, 7700,C800,0364,0300,0470,1801  0252  8050 ;% 00   NOP
U 01, 7700,3800,7FFF,FDFF,8470,1802  025D  8054 ;% 01   LONLIT_[400]
U 02, 7700,0800,5BE6,03D8,4A70,1803  0268  8058 ;% 02   D_VA_0
U 03, 7300,C800,0364,0300,0470,1804  0273  8062 ;% 03   STOP.CLOCK
                                     027E  8066
                                     027E  8067 8$:
U 00, 7700,C800,0364,0300,0470,1801  027E  8068 ;% 00   NOP
U 01, 7700,C800,0224,03D8,5360,1802  0289  8072 ;% 01   CLRTB.VA_D               ; WCTRL/CLRTB.VA_WB,WB_D
U 02, 7700,4800,0212,03D4,0470,1803  0294  8076 ;% 02   D_D+R[LONLIT]
U 03, 7300,C800,0364,0300,0470,1804  029F  8080 ;% 03   STOP.CLOCK
                                     02AA  8084
                                     02AA  8085 9$:
U 00, 7700,C800,0364,0300,0470,1801  02AA  8086 ;% 00   NOP
U 01, 7700,081B,0024,03D8,5A70,1802  02B5  8090 ;% 01   CLRCH.VA_VA              ; WCTRL/CLRCH.VA_WB,WB_VA
U 02, 7700,C800,0364,0300,4470,1803  02C0  8094 ;% 02   VA_VA+4
U 03, 7300,C800,0364,0300,0470,1804  02CB  8098 ;% 03   STOP.CLOCK
                                     02D6  8102
                                     02D6  8103 10$:
U 00, 7700,C800,0364,0300,0470,1801  02D6  8104 ;% 00   NOP                       ; Allow time for clock to start
U 01, 5700,C800,0364,0300,4A70,1802  02E1  8108 ;% 01   VA_RDM                     ; Set-up current test address
U 02, 7700,B800,7FFF,FF5B,8470,1803  02EC  8112 ;% 02   LONLIT_[00000148]         ; Lonlit reg gets PTE for TB validation
U 03, 7700,481B,5BE4,03D4,5070,1804  02F7  8116 ;% 03   TB_R[LONLIT]             ; Validate TB by VA
U 04, 7700,F800,2AAA,AAAA,8470,1805  0302  8120 ;% 04   LONLIT_[0AAAAAAAA]       ; Lonlit reg gets data for cache
                                     030D  8124                                 ; validation
U 05, 7701,0800,5BE4,02D4,5590,1806  030D  8125 ;% 05   WRITE.LONG R[LONLIT]     ; First write the data then
U 06, 7700,C800,5BE4,03D4,5470,1807  0318  8129 ;% 06   WDR_UNROT(R[LONLIT])    ; read it back to validate cache
U 07, 7700,4800,0364,0300,4470,1808  0323  8133 ;% 07   VA_VA+4                   ; Update the VA by 4 for next TB/cache
                                     032E  8137                                 ; validation
U 08, 7700,B800,7FFF,FF5B,8470,1809  032E  8138 ;% 08   LONLIT_[00000148]         ; Lonlit reg gets PTE for TB validation
U 09, 7700,C81B,5BE4,03D4,5070,180A  0339  8142 ;% 09   TB_R[LONLIT]             ; Validate TB by VA
U 0A, 7700,B800,5555,5555,0470,180B  0344  8146 ;% 0A   LONLIT_[55555555]       ; Lonlit reg gets data for cache
                                     034F  8150                                 ; validation
U 0B, 7701,0800,5BE4,02D4,5590,180C  034F  8151 ;% 0B   WRITE.LONG R[LONLIT]     ; First write the data
U 0C, 7700,C800,5BE4,03D4,5470,180D  035A  8155 ;% 0C   WDR_UNROT(R[LONLIT])    ; read it back to validate cache
U 0D, 7700,4800,0364,0300,4470,180E  0365  8159 ;% 0D   VA_VA+4                   ; Update the VA by 4 for next TB/cache
                                     0370  8163                                 ; validation
U 0E, 7700,B800,7FFF,FF5B,8470,180F  0370  8164 ;% 0E   LONLIT_[00000148]         ; Lonlit reg gets PTE for TB validation
U 0F, 7700,481B,5BE4,03D4,5070,1810  037B  8168 ;% 0F   TB_R[LONLIT]             ; Validate TB by VA
```

```
U 10. 7700.3800.0000.0000.0470.1811  0386  8172 ;% 10   LONLIT_[0FFFFFFFF]                ; Lonlit reg gets data for cache
                                     0391  8176                                          ; validation
U 11. 7701.0800.5BE4.02D4.5590.1812  0391  8177 ;% 11   WRITE.LONG R[LONLIT]              ; First write the data then
U 12. 7700.C800.5BE4.03D4.5470.1813  039C  8181 ;% 12   WDR_UNROT(R[LONLIT])            ; read it back to validate cache
U 13. 5700.C800.0364.0300.4870.1814  03A7  8185 ;% 13   PC_RDM                          ; Load PC to cause prefetch
U 14. 7700.9800.D370.0360.2470.1815  03B2  8189 ;% 14   CRAR_ZLIT16[0C0]               ; Point to Console command & status reg
U 15. 7700.5800.D370.0320.2C70.1816  03BD  8193 ;% 15   CONREGS_ZLIT16[040]           ; Disable Ctrl P interrupt
U 16. 7700.4800.0364.1700.0470.1817  03C8  8197 ;% 16   IRD1TEST                       ; Perform IRD1TEST to load PCBACK
U 17. 7700.4800.0364.0300.0470.1818  03D3  8201 ;% 17   NOP                           ; Let everything fall into place
U 18. 7701.4817.0364.0002.0470.1819  03DE  8205 ;% 18   PC_PC+1 MB_XB                 ; Increment the PC and source XB
U 19. 6700.C819.0024.03D8.0470.181A  03E9  8209 ;% 19   RDM_PCBACK                    ; Read the PCBACK to RDM for comparison
U 1A. 7700.085A.5924.0300.0470.181B  03F4  8213 ;% 1A   R[TEMP0]_M[PC]               ; Store PC+1 in R[TEMP0] for error call
U 1B. 7300.C800.0364.0300.0470.181C  03FF  8217 ;% 1B   STOP.CLOCK                    ; Return to RDM pseudo code
U 1C. 7700.4800.0364.0300.0470.181D  040A  8221 ;% 1C   NOP                           ; Allow time for clock to start
U 1D. 5700.4800.0364.0300.4A70.181E  0415  8225 ;% 1D   VA_RDM                        ; Re-initialize the VA
U 1E. 7701.8800.5BE4.02D8.5590.181F  0420  8229 ;% 1E   WRITE.LONG R[ZERO]            ; Zero cache location by VA
U 1F. 7700.C800.5BE4.03D8.5470.1820  042B  8233 ;% 1F   WDR_UNROT(R[ZERO])          ; ...
U 20. 7700.C800.0364.0300.4470.1821  0436  8237 ;% 20   VA_VA+4                       ; Update VA by 4
U 21. 7701.0800.5BE4.02D8.5590.1822  0441  8241 ;% 21   WRITE.LONG R[ZERO]            ; Zero cache location by VA
U 22. 7700.C800.5BE4.03D8.5470.1823  044C  8245 ;% 22   WDR_UNROT(R[ZERO])          ; ...
U 23. 7300.4800.0364.0300.0470.1824  0457  8249 ;% 23   STOP.CLOCK                    ; Return to RDM pseudo code
                                     0462  8253
                                     0462  8254
                                     0462  8255 END_TEST_DATA
                                     0462
                                     0462       ;-------------------------------------------------------------------------
                                     0462  8256
                                     0462  8257 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0027          .SBTTL  "TEST   053     TEST EXECUTION BUFFER ROTATE LOGIC
0027  8259 ;*****************************************************************************
0027  8260 ;++
0027  8261 ;
0027  8262 ; FUNCTIONAL DESCRIPTION:
0027  8263 ;
0027  8264 ;       THIS TEST CONSISTS OF 2 SUBTESTS TO CHECK THE OPERATION OF THE ROTATE
0027  8265 ;       LOGIC ON THE OUTPUT OF XB0 AND XB1.
0027  8266 ;
0027  8267 ;
0027  8268 ; TEST ALGORITHM:
0027  8269 ;
0027  8270 ;       1.  CREATE A TEST LOOP
0027  8271 ;       2.  LOAD DCS WITH MICRO WORDS FOR THIS TEST
0027  8272 ;       3.  EXECUTE DCS PROGRAM
0027  8273 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
0027  8274 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
0027  8275 ;               2.  SET MEMORY MANAGEMENT OFF
0027  8276 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
0027  8277 ;               4.  SET CMI OFF
0027  8278 ;               5.  LOAD VIRTUAL ADDRESS REGISTER WITH 0'S
0027  8279 ;               6.  PUT TEST PATTERN #1 INTO LONGLIT REGISTER
0027  8280 ;               7.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 0
0027  8281 ;               8.  INCREMENT VIRTUAL ADDRESS REGISTER BY 4
0027  8282 ;               9.  PUT TEST PATTERN #2 INTO LONGLIT REGISTER
0027  8283 ;              10.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 4
0027  8284 ;              11.  INCREMENT VIRTUAL ADDRESS REGISTER BY 4
0027  8285 ;              12.  PUT TEST PATTERN #3 INTO LONGLIT REGISTER
0027  8286 ;              13.  WRITE TEST PATTERN IN LONGLIT TO CACHE ADDRESS 8
0027  8287 ;              14.  LOAD PROGRAM COUNTER WITH A PREFETCH ADDRESS
0027  8288 ;              12.  READ DESIRED TEST PATTERN FROM XB'S TO RDM
0027  8289 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0027  8290 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ITERATIONS
0027  8291 ;
0027  8292 ;
0027  8293 ; TEST PATTERNS:
0027  8294 ;
0027  8295 ;       ITERATION       PATTERN 3       PATTERN 2       PATTERN 1
0027  8296 ;
0027  8297 ;           1           00000000        FFFFFF00        000000FF
0027  8298 ;           2           00000000        000000FF        FFFFFF00
0027  8299 ;           3           000000FF        FFFF0000        0000FF00
0027  8300 ;           4           00000000        0000FFFF        FFFF0000
0027  8301 ;           5           0000FFFF        FF000000        00FF0000
0027  8302 ;           6           00000000        00FFFFFF        FF000000
0027  8303 ;           7           00FFFFFF        00000000        FF000000
0027  8304 ;           8           00000000        FFFFFFFF        00000000
0027  8305 ;
0027  8306 ;
0027  8307 ; LOGIC DESCRIPTION:
0027  8308 ;
0027  8309 ;       THIS TEST IS DIVIDED INTO 2 SUBTESTS, EACH OF WHICH IS ALMOST
0027  8310 ;       IDENTICAL.  SUBTEST 1 LOOKS AT THE OUTPUT OF XB0 AND SUBTEST 2
0027  8311 ;       LOOKS AT THE OUTPUT OF XB1.  TEST PATTERNS AND THEIR COMPLIMENTS
0027  8312 ;       ARE SHIFTED FROM ALL BYTE POSITIONS OF XB0 AND XB1 VIA THE EXECUTION
```

```
                        ;       BUFFER ROTATOR TO THE WBUS.  THE "MDR" GATE ARRAY IS YOUR BEST FIRST
                        ;       CHOICE TO SWAP IN THE EVENT OF ERRORS.  SECOND AND THIRD CHOICE SHOULD
                        ;       BE "ADD", WHICH CONTROLS THE ADDRESSING, AND "PRK", WHICH CONTROLS THE
                        ;       PREFETCH MECHANISM.
                        ;
                        ; ASSUMPTIONS:
                        ;
                        ;       THIS TEST ASSUMES THE WBUS, CACHE, VA, PC, AND MBUS ARE ALL
                        ;       OPERATIONAL.
                        ;
                        ; ERROR DESCRIPTION:
                        ;
                        ;       EXPECTED XB DATA
                        ;       RECEIVED XB DATA
                        ;       LOOP COUNT
                        ;
                        ;--
                        ;*******************************************************************************

                                SUBTEST                                         ;SUBTEST #1

                        ;///////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       TEST EXECUTION BUFFER ROTATE LOGIC USING XB0
                        ;-
                                INITIALIZE                                      ;INIT CPU
                                LOADDCS         10$,,19,1                       ;MODIFY DCS ADD 19
                                LOOP            I,1,8                           ;CREATE A TEST LOOP
                                LDFIELD         2$,I,L,1$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                                LOADDCS         1$,,06,1                        ;TEST PATTERN #1 TO DCS
                                LDFIELD         3$,I,L,1$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                                LOADDCS         1$,,10,1                        ;TEST PATTERN #2 TO DCS
                                LDFIELD         4$,I,L,1$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                                LOADDCS         1$,,14,1                        ;TEST PATTERN #3 TO DCS
                                LDFIELD         5$,I,B,6$,34,39                 ;RTEMP FOR PC
                                LOADDCS         6$,,17,1                        ;STARTING PC TO DCS
                                ERRLOOP                                         ;ERROR LOOP
                                FETCH           0                               ;START DCS PROGRAM
                                BRSTCLK         <^X13>                          ;END DCS PROGRAM
                                CMPREG          WHREG,7$,I,L,                   ;COMPARE RESULTS
                                IFERROR         ,<<MDR><ADD><PRK>>,
                                ENDLOOP         I                               ;END TEST LOOP

                                SUBTEST                                         ;SUBTEST #2

                        ;///////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #2
                        ;
                        ;       TEST EXECUTION BUFFER ROTATE LOGIC USING XB1
                        ;-
```

```
                    00BC  8364          INITIALIZE                              ;INIT CPU
                    00BE  8365          LOADDCS         8$,,19,1                ;SOURCE XB1 TO RDM MICRO WORD
                    00C5  8366          LOOP            I,1,8                   ;CREATE A TEST LOOP
                    00CC  8367          LDFIELD         2$,I,L,1$,31,62,LON     ;TEST PATTERN TO MICRO WORD
                    00DD  8368          LOADDCS         1$,,06,1                ;TEST PATTERN #1 TO DCS
                    00E4  8369          LDFIELD         3$,I,L,1$,31,62,LON     ;TEST PATTERN TO MICRO WORD
                    00F5  8370          LOADDCS         1$,,10,1                ;TEST PATTERN #2 TO DCS
                    00FC  8371          LDFIELD         4$,I,L,1$,31,62,LON     ;TEST PATTERN TO MICRO WORD
                    010D  8372          LOADDCS         1$,,14,1                ;TEST PATTERN #3 TO DCS
                    0114  8373          LDFIELD         5$,I,B,6$,34,39         ;RTEMP FOR PC
                    0125  8374          LOADDCS         6$,,17,1                ;STARTING PC TO DCS
                    012C  8375          ERRLOOP                                 ;ERROR LOOP
                    012E  8376          FETCH           0                       ;START DCS PROGRAM
                    0133  8377          BRSTCLK         <^X14>                  ;END DCS PROGRAM
                    0137  8378          CMPREG          WHREG,9$,I,L,           ;COMPARE RESULTS
                    0140  8379          IFERROR         ,<<MDR><ADD><PRK>>,
                    0148  8380          ENDLOOP         I                       ;END TEST LOOP
                    014B  8381          SKIP
                    0152  8382
                    0152  8383 BEGIN_TEST_DATA
                    0152
                    0152          ;-------------------------------------------------------------------------
                    0152  8384
                    0152  8385 1$:
U 06, 7700,7800,7FFF,FFFF,8470,1807 0152  8386 ;% 06   LONLIT_[0]                 ;TEST PATTERNS TO LONGLIT REG
                    015D  8390
                000000FF  015D  8391 2$:     .LONG   ^X000000FF                  ;TEST PATTERN #1
                FFFFFF00  0161  8392         .LONG   ^XFFFFFF00
                0000FF00  0165  8393         .LONG   ^X0000FF00
                FFFF0000  0169  8394         .LONG   ^XFFFF0000
                00FF0000  016D  8395         .LONG   ^X00FF0000
                FF000000  0171  8396         .LONG   ^XFF000000
                FF000000  0175  8397         .LONG   ^XFF000000
                00000000  0179  8398         .LONG   ^X00000000
                          017D  8399
                FFFFFF00  017D  8400 3$:     .LONG   ^XFFFFFF00                  ;TEST PATTERN #2
                000000FF  0181  8401         .LONG   ^X000000FF
                FFFF0000  0185  8402         .LONG   ^XFFFF0000
                0000FFFF  0189  8403         .LONG   ^X0000FFFF
                FF000000  018D  8404         .LONG   ^XFF000000
                00FFFFFF  0191  8405         .LONG   ^X00FFFFFF
                00000000  0195  8406         .LONG   ^X00000000
                FFFFFFFF  0199  8407         .LONG   ^XFFFFFFFF
                          019D  8408
                00000000  019D  8409 4$:     .LONG   ^X00000000                  ;TEST PATTERN #3
                00000000  01A1  8410         .LONG   ^X00000000
                000000FF  01A5  8411         .LONG   ^X000000FF
                00000000  01A9  8412         .LONG   ^X00000000
                0000FFFF  01AD  8413         .LONG   ^X0000FFFF
                00000000  01B1  8414         .LONG   ^X00000000
                00FFFFFF  01B5  8415         .LONG   ^X00FFFFFF
                00000000  01B9  8416         .LONG   ^X00000000
                          01BD  8417
                      04  01BD  8418 5$:     .BYTE   ^X04                        ;STARTING PREFETCH ADDRESS
                      04  01BE  8419         .BYTE   ^X04
```

```
                          05  01BF  8420          .BYTE   ^X05
                          05  01C0  8421          .BYTE   ^X05
                          06  01C1  8422          .BYTE   ^X06
                          06  01C2  8423          .BYTE   ^X06
                          07  01C3  8424          .BYTE   ^X07
                          07  01C4  8425          .BYTE   ^X07
                              01C5  8426 6$:
U 11, 7700,8800,5BE4,0310,4870,1812   01C5  8427 ;% 11   PC_R[TEMP4]                          ;PC GETS PREFETCH ADDRESS
                              01D0  8431
                      000000FF        01D0  8432 7$:     .LONG   ^X000000FF                  ;RESULTS SUBTEST #1
                      FFFFFF00        01D4  8433         .LONG   ^XFFFFFF00
                      000000FF        01D8  8434         .LONG   ^X000000FF
                      FFFFFF00        01DC  8435         .LONG   ^XFFFFFF00
                      000000FF        01E0  8436         .LONG   ^X000000FF
                      FFFFFF00        01E4  8437         .LONG   ^XFFFFFF00
                      000000FF        01E8  8438         .LONG   ^X000000FF
                      FFFFFF00        01EC  8439         .LONG   ^XFFFFFF00
                              01F0  8440 8$:
U 13, 6701,C817,5924,0202,0470,1814   01F0  8441 ;% 13   RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I]  ;SOURCE XB1 TO RDM
                              01FB  8445
                      FFFFFF00        01FB  8446 9$:     .LONG   ^XFFFFFF00                  ;RESULTS SUBTEST #2
                      000000FF        01FF  8447         .LONG   ^X000000FF
                      FFFFFF00        0203  8448         .LONG   ^XFFFFFF00
                      000000FF        0207  8449         .LONG   ^X000000FF
                      FFFFFF00        020B  8450         .LONG   ^XFFFFFF00
                      000000FF        020F  8451         .LONG   ^X000000FF
                      FFFFFF00        0213  8452         .LONG   ^XFFFFFF00
                      000000FF        0217  8453         .LONG   ^X000000FF
                              021B  8454
                              021B  8455 10$:
U 13, 7300,4800,0364,0300,0470,1814   021B  8456 ;% 13   NOP,STOP.CLOCK                       ;DCS ADD 19 SUBTEST #1
                              0226  8460
                              0226  8461 BEGIN_CPU_DATA
                              0002  8462
                              0002  8463 DCS_DATA
                              0001  8464
U 00, 7700,C800,0364,0300,0470,1801   0001  8465 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C  8469 ;% 01   MEMSCAR_R[TEMP0]                    ;PHY/VIR REG ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803   0017  8473 ;% 02   MEMSCR_R[TEMP1]                     ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804   0022  8477 ;% 03   MEMSCAR_R[TEMP2]                   ;REF CACHE ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805   002D  8481 ;% 04   MEMSCR_R[TEMP3]                    ;TURN CMI OFF
U 05, 7700,0800,5BE4,0310,4A70,1806   0038  8485 ;% 05   VA_R[TEMP4]                        ;0'S TO VA REGISTER
U 06, 7700,7800,7FFF,FFFF,8470,1807   0043  8489 ;% 06   LONLIT_[0]                        ;TEST PATTERN #1 TO LONGLIT
U 07, 7701,8800,5BE4,02D4,5590,1808   004E  8493 ;% 07   WRITE.LONG R[LONLIT]               ;TEST PATTERN TO CACHE ADD 0
U 08, 7700,4800,5BE4,03D4,5470,1809   0059  8497 ;% 08   WB_R[LONLIT],WCTRL/WDR_WB.UR       ;HOLD DATA FOR EXTRA CYCLE
U 09, 7700,C800,0364,0300,4470,180A   0064  8501 ;% 09   VA_VA+4                            ;INCREMENT VA BY 4
U 0A, 7700,7800,7FFF,FFFF,8470,180B   006F  8505 ;% 0A   LONLIT_[0]                        ;TEST PATTERN #2 TO LONGLIT
U 0B, 7701,0800,5BE4,02D4,5590,180C   007A  8509 ;% 0B   WRITE.LONG R[LONLIT]               ;TEST PATTERN TO CACHE ADD 4
U 0C, 7700,C800,5BE4,03D4,5470,180D   0085  8513 ;% 0C   WB_R[LONLIT],WCTRL/WDR_WB.UR       ;HOLD DATA FOR EXTRA CYCLE
U 0D, 7700,4800,0364,0300,4470,180E   0090  8517 ;% 0D   VA_VA+4                            ;INCREMENT VA BY 4
U 0E, 7700,F800,7FFF,FFFF,8470,180F   009B  8521 ;% 0E   LONLIT_[0]                        ;TEST PATTERN #3 TO LONGLIT
U 0F, 7701,8800,5BE4,02D4,5590,1810   00A6  8525 ;% 0F   WRITE.LONG R[LONLIT]               ;TEST PATTERN TO CACHE ADD 8
U 10, 7700,4800,5BE4,03D4,5470,1811   00B1  8529 ;% 10   WB_R[LONLIT],WCTRL/WDR_WB.UR       ;HOLD DATA FOR EXTRA CYCLE
U 11, 7700,8800,5BE4,0310,4870,1812   00BC  8533 ;% 11   PC_R[TEMP4]                        ;STARTING ADD FOR PREFETCH
U 12, 6701,4817,5924,0202,0470,1813   00C7  8537 ;% 12   RDM_WB,ISIZE[LONG],WB_M[XB.PC_PC+I]  ;READ XB0 TO RDM
```

```
U 13, 7300,4800,0364,0300,0470,1814  00D2  8541 ;% 13   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
U 14, 7300,C800,0364,0300,0470,1815  00DD  8545 ;% 14   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
                                     00E8  8549
                                     00E8  8550 RTEMP_DATA
                                     0001  8551
                           00000000  0001  8552         .LONG   ^X00000000              ;PHY/VIR REG ADDRESS
                           00000000  0005  8553         .LONG   ^X00000000              ;SET MME OFF
                           0E000000  0009  8554         .LONG   ^X0E000000              ;REF CACHE ONLY ADDRESS
                           01000000  000D  8555         .LONG   ^X01000000              ;TURN CMI OFF
                           00000000  0011  8556         .LONG   ^X00000000              ;ADDRESS FOR VA & PC
                           00000001  0015  8557         .LONG   ^X00000001              ;ADDRESS FOR PC
                           00000002  0019  8558         .LONG   ^X00000002              ;ADDRESS FOR PC
                           00000003  001D  8559         .LONG   ^X00000003              ;ADDRESS FOR PC
                                     0021  8560
                                     0021  8561 END_CPU_DATA
                                     0226  8562
                                     0226  8563 END_TEST_DATA
                                     0226
                                     0226       ;-----------------------------------------------------------------------------
                                     0226  8564
                                     0226  8565 ENDTEST
```

```
                        .SBTTL  "TEST   054     TEST FOR XB0 MACHINE CHECK WITH TB MULTIPLE HIT
                    ;*******************************************************************************
                    ;++
                    ;
                    ; FUNCTIONAL DESCRIPTION:
                    ;
                    ;       THIS TEST INSURES THAT CPU CAN DETECT A TB MULTIPLE HIT WHEN XB0 IS
                    ;       LOADED BY A PREFETCH OPERATION.
                    ;
                    ;
                    ; TEST ALGORITHM:
                    ;
                    ;       1.  EXECUTE DCS PROGRAM
                    ;               a.  SET MEMORY MANAGEMENT ON
                    ;               b.  LOAD VA REGISTER WITH ZERO
                    ;               c.  WRITE AAAAAAAA TO CACHE ADDRESS 0
                    ;               d.  INCREMENT VA BY 4
                    ;               e.  WRITE 55555555 TO CACHE ADDRESS 4
                    ;               f.  SET BIT 8 ON THE WBUS WHILE DOING A CLRTB FUNCTION.
                    ;                   (THIS WILL CAUSE A MULTIPLE HIT)
                    ;               g.  LOAD PC WITH ZERO TO CAUSE A PREFETCH
                    ;               h.  TRY TO READ XB0 TO RTEMP0; SHOULD MACHINE CHECK
                    ;       2.  TEST MICRO VECTOR LINES ON VBUS FOR CORRECT VECTOR
                    ;       3.  IF NO ERROR TEST CS ADDRESS FOR MACHINE CHECK
                    ;       4.  IF NO ERROR EXECUTE DCS PROGRAM
                    ;               a.  READ MACHINE CHECK ERROR SUMMARY REGISTER TO WHREG
                    ;               b.  CLEAR TB MULTIPLE HIT
                    ;               c.  CLEAR MACHINE CHECK ERROR SUMMARY REGISTER
                    ;       5.  TEST WHREG FOR MACHINE CHECK ADDRESS
                    ;       6.  IF NO ERROR GO TO NEXT TEST
                    ;
                    ;
                    ; LOGIC DESCRIPTION:
                    ;
                    ;       THIS TEST MOSTLY INVOLVES LOGIC IN THE "UTR" GATE ARRAY.  THE MACHINE
                    ;       CHECK LOGIC IN UTR SHOULD DETECT A MULTIPLE TB HIT AND ASSERT A MACHINE
                    ;       CHECK MICROTRAP.  THE ONLY TRICKY PART OF THIS TEST IS WHERE BIT 8 (THE
                    ;       VALID BIT) IS SET ON THE WBUS DURING A CLEAR TB MICRO ORDER.  THIS
                    ;       RESULTS IN VALIDATING RATHER THAN CLEARING BOTH TB GROUPS.
                    ;
                    ;
                    ; ASSUMPTIONS:
                    ;
                    ;       THIS TEST ASSUMES THE TB DATA INTEGRITY TESTS HAVE RUN SUCCESSFULLY
                    ;
                    ;
                    ; ERROR DESCRIPTION:
                    ;
                    ;       NOTE: THERE ARE THREE IFERROR STATEMENTS IN THIS TEST.  SO YOU WILL
                    ;       HAVE TO CHECK THE FAILING PC TO DETERMINE WHICH ERROR PRINTOUT IS
                    ;       VALID.
                    ;
                    ;       FIRST IFERROR:
                    ;       EXPECTED VBUS DATA (MICRO VECTOR LINES)
                    ;       RECEIVED VBUS DATA
```

```
                                0034  ;
                                0034  ;       SECOND IFERROR:
                                0034  ;       EXPECTED CS ADDRESS
                                0034  ;       RECEIVED CS ADDRESS
                                0034  ;       CONTENTS OF RTEMP0 (DATA READ FROM CACHE)
                                0034  ;
                                0034  ;       THIRD IFERROR:
                                0034  ;       EXPECTED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                                0034  ;       RECEIVED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                                0034  ;       CONTENTS OF RTEMP0 (DATA READ FROM CACHE)
                                0034  ;
                                0034  ;--
                                0034  ;*******************************************************************************
                                0034
                                0034
                                0034
                                0034          INITIALIZE                                  ;INIT THE CPU
                                0036          EXPUTRAP                                    ;EXPECT A MACHINE CHECK
                                0038          ERRLOOP                                     ;ERROR LOOP POINT
                                003A          FETCH           0                           ;START DCS PROGRAM
                                003F          BRSTCLK         <^X0D>                      ;END DCS PROGRAM
                                0043          CMPVBUS         5$                          ;TEST MICRO VECTOR LINES
                                0048          IFERROR         ,<<UTR>>,                   ;WRONG VECTOR
                                004E          CMPREGMSK       CSTRP,3$,,4$,,W,            ;TEST THE CS ADDRESS
                                005A          IFERROR         1,,<<DPM>>                  ;NOT A MACHINE CHECK
                                0060          FETCH           <^X0E>                      ;START DCS PROGRAM
                                0065          BRSTCLK         <^X14>                      ;END DCS PROGRAM
                                0069          CMPREGMSK       WHREG,1$,,2$,,L,            ;TEST FOR TB ERROR
                                0075          IFERROR         1,<<UTR>>,                  ;INCORRECT ERROR
                                007B          SKIP                                        ;GO TO NEXT TEST
                                0082
                                0082
                                0082
                                0082
                                0082  BEGIN_TEST_DATA
                                0082
                                0082          ;-----------------------------------------------------------------------
                                0082
                        05000000  0082  1$:     .LONG   ^X05000000                          ;MACHINE CHECK REG DATA
                        0F000000  0086  2$:     .LONG   ^X0F000000                          ;MASK FOR CMPREGMSK PSEUDO OP
                            0028  008A  3$:     .WORD   ^X0028                              ;MACHINE CHECK U-TRAP
                            3FFF  008C  4$:     .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                              04  008E  5$:     .BYTE   ^X4                                 ;VBUS MICRO VECTOR LINES
                                008F          VBUSDATA        <^X08>,1
                                0090          VBUSDATA        <^X09>,0
                                0091          VBUSDATA        <^X0A>,0
                                0092          VBUSDATA        <^X0B>,0
                                0093
                                0093
                                0093
                                0093  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
```

```
U 01. 7700.8800.5BE4.0304.6870.1802  000C  8677 ;% 01   MEMSCAR_R[TEMP1]                          ;SET MME ON
U 02. 7700.8800.5BE4.0308.6070.1803  0017  8681 ;% 02   MEMSCR_R[TEMP2]
U 03. 7700.C800.5BE4.03D8.4A70.1804  0022  8685 ;% 03   VA_R[ZERO]                               ;LOAD VA REGISTER
U 04. 7700.C800.5BE4.030C.5C80.1805  002D  8689 ;% 04   WRITE.PHY R[TEMP3]                       ;VALIDATE THE CACHE
U 05. 7700.4800.5BE4.030C.5470.1806  0038  8693 ;% 05   WDR_UNROT(R[TEMP3])
U 06. 7700.4800.0364.0300.4470.1807  0043  8697 ;% 06   VA_VA+4                                 ;INCREMENT VA BY 4
U 07. 7700.0800.5BE4.0310.5C80.1808  004E  8701 ;% 07   WRITE.PHY R[TEMP4]                       ;VALIDATE THE CACHE
U 08. 7700.0800.5BE4.0310.5470.1809  0059  3705 ;% 08   WDR_UNROT(R[TEMP4])
U 09. 7700.4800.5BE4.0314.0470.180A  0064  8709 ;% 09   WB_R[TEMP5]
U 0A. 7700.C800.5BE4.0314.5360.180B  006F  8713 ;% 0A   CLRTB.VA_WB,WB_R[TEMP5]                 ;MAKE TB A MULTIPLE HIT
U 0B. 7700.C800.5BE4.03D8.4670.180C  007A  8717 ;% 0B   PC_R[ZERO]                             ;CAUSE PREFETCH
U 0C. 7601.0857.5924.0202.0470.180D  0085  8721 ;% 0C   R[TEMP0]_XB PC_PC+4,STROBE.V.BUS       ;READ XB0 TO RTEMP0
U 0D. 7300.C800.0364.0300.0470.180E  0090  8725 ;% 0D   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
U 0E. 7700.4800.0364.0300.0470.180F  009B  8729 ;% 0E   NOP
U 0F. 7700.C800.5BE4.0318.6870.1810  00A6  8733 ;% 0F   MEMSCAR_R[TEMP6]                        ;READ MACHINE CHECK REGISTER
U 10. 6700.4800.0364.0300.6470.1811  00B1  8737 ;% 10   RDM_WB,WCTRL/MEMSCR
U 11. 7700.4800.5BE4.03D8.4A70.1812  00BC  8741 ;% 11   VA_R[ZERO]
U 12. 7700.C800.5BE4.03D8.5360.1813  00C7  8745 ;% 12   CLRTB.VA_WB,WB_R[ZERO]                  ;FLUSH TB
U 13. 7700.C800.5BE4.03D8.6070.1814  00D2  8749 ;% 13   MEMSCR_R[ZERO]                          ;CLEAR MACHINE CHECK REGISTER
U 14. 7300.C800.0364.0300.0470.1815  00DD  8753 ;% 14   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                     00E8  8757
                                     00E8  8758 RTEMP_DATA
                                     0001  8759
                          00000000   0001  8760         .LONG   ^X00000000                    ;DATA READ FROM XB
                          00000000   0005  8761         .LONG   ^X00000000                    ;PHY/VIR REGISTER ADDRESS
                          01000000   0009  8762         .LONG   ^X01000000                    ;SET MME ON
                          AAAAAAAA   000D  8763         .LONG   ^XAAAAAAAA                    ;CACHE DATA
                          55555555   0011  8764         .LONG   ^X55555555                    ;CACHE DATA
                          00000348   0015  8765         .LONG   ^X00000348                    ;USE FOR MULTIPLE HIT
                          08000000   0019  8766         .LONG   ^X08000000                    ;MACHINE CHECK REG ADDRESS
                                     001D  8767
                                     001D  8768 END_CPU_DATA
                                     0093  8769
                                     0093  8770
                                     0093  8771
                                     0093  8772 END_TEST_DATA
                                     0093
                                     0093       ;-------------------------------------------------------------------------------
                                     0093  8773
                                     0093  8774 ENDTEST
```

```
0034          .SBTTL  "TEST   055     TEST FOR XB1 MACHINE CHECK WITH TB MULTIPLE HIT
0034   8776 ;*****************************************************************************
0034   8777 ;++
0034   8778 ;
0034   8779 ; FUNCTIONAL DESCRIPTION:
0034   8780 ;
0034   8781 ;       THIS TEST INSURES THAT CPU CAN DETECT A TB MULTIPLE HIT WHEN XB1 IS
0034   8782 ;       LOADED BY A PREFETCH OPERATION.
0034   8783 ;
0034   8784 ;
0034   8785 ; TEST ALGORITHM:
0034   8786 ;
0034   8787 ;       1.  EXECUTE DCS PROGRAM
0034   8788 ;               a.  SET MEMORY MANAGEMENT ON
0034   8789 ;               b.  LOAD VA WITH ADDRESS 000001FC
0034   8790 ;               c.  VALIDATE THIS LOCATION IN THE TB
0034   8791 ;               d.  VALIDATE THE CACHE AT THIS LOCATION
0034   8792 ;               e.  INCREMENT VA BY 4
0034   8793 ;               f.  VALIDATE THIS LOCATION IN CACHE
0034   8794 ;               g.  MAKE THIS LOCATION A MULTIPLE HIT
0034   8795 ;               h.  LOAD PC WITH 000001FC TO CAUSE PREFETCH
0034   8796 ;               i.  READ XB0 TO RTEMP 0
0034   8797 ;               j.  READ XB1 TO RTEMP 0.  THIS SHOULD CAUSE A MACHINE CHECK
0034   8798 ;       2.  TEST MICROVECTOR LINES ON VBUS FOR A MACHINE CHECK
0034   8799 ;       3.  IF NO ERROR TEST THE CS ADDRESS FOR A MACHINE CHECK
0034   8800 ;       4.  IF NO ERROR EXECUTE DCS PROGRAM
0034   8801 ;               a.  READ MACHINE CHECK ERROR SUMMARY REGISTER TO WHREG
0034   8802 ;               b.  CLEAR THE MULTIPLE HIT IN THE TB
0034   8803 ;               c.  CLEAR THE MACHINE CHECK ERROR SUMMARY REGISTER
0034   8804 ;       5.  TEST THE WHREG FOR TB ERROR
0034   8805 ;       6.  IF NO ERROR GO TO NEXT TEST
0034   8806 ;
0034   8807 ;
0034   8808 ; LOGIC DESCRIPTION:
0034   8809 ;
0034   8810 ;       THIS TEST MOSTLY INVOLVES LOGIC IN THE "UTR" GATE ARRAY.  THE MACHINE
0034   8811 ;       CHECK LOGIC IN UTR SHOULD DETECT A MULTIPLE TB HIT AND ASSERT A MACHINE
0034   8812 ;       CHECK MICROTRAP.  THE ONLY TRICKY PART OF THIS TEST IS WHERE BIT 8 (THE
0034   8813 ;       VALID BIT) IS SET ON THE WBUS DURING A CLEAR TB MICRO ORDER.  THIS
0034   8814 ;       RESULTS IN VALIDATING RATHER THAN CLEARING BOTH TB GROUPS.
0034   8815 ;
0034   8816 ;
0034   8817 ; ASSUMPTIONS:
0034   8818 ;
0034   8819 ;       THIS TEST ASSUMES THE TB DATA INTEGRITY TESTS HAVE RUN SUCCESSFULLY
0034   8820 ;
0034   8821 ;
0034   8822 ; ERROR DESCRIPTION:
0034   8823 ;
0034   8824 ;       NOTE: THERE ARE THREE IFERROR STATEMENTS IN THIS TEST.  SO YOU WILL
0034   8825 ;       HAVE TO CHECK THE FAILING PC TO DETERMINE WHICH ERROR PRINTOUT IS
0034   8826 ;       VALID.
0034   8827 ;
0034   8828 ;       FIRST IFERROR:
0034   8829 ;       EXPECTED VBUS DATA (MICRO VECTOR LINES)
```

```
                     0034  8830 ;       RECEIVED VBUS DATA
                     0034  8831 ;
                     0034  8832 ;       SECOND IFERROR:
                     0034  8833 ;       EXPECTED CS ADDRESS
                     0034  8834 ;       RECEIVED CS ADDRESS
                     0034  8835 ;       CONTENTS OF RTEMP0 (DATA READ FROM CACHE)
                     0034  8836 ;
                     0034  8837 ;       THIRD IFERROR:
                     0034  8838 ;       EXPECTED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                     0034  8839 ;       RECEIVED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                     0034  8840 ;       CONTENTS OF RTEMP0 (DATA READ FROM CACHE)
                     0034  8841 ;
                     0034  8842 ;--
                     0034  8843 ;*******************************************************************************
                     0034  8844
                     0034  8845
                     0034  8846
                     0034  8847         INITIALIZE                                  ;INIT THE CPU
                     0036  8848         EXPUTRAP                                    ;EXPECT A MACHINE CHECK
                     0038  8849         ERRLOOP                                     ;ERROR LOOP POINT
                     003A  8850         FETCH           0                           ;START DCS PROGRAM
                     003F  8851         BRSTCLK         <^X0F>                      ;END DCS PROGRAM
                     0043  8852         CMPVBUS         5$                          ;TEST MICRO VECTOR LINES
                     0048  8853         IFERROR         ,<<UTR>>,                   ;WRONG VECTOR
                     004E  8854         CMPREGMSK       CSTRP,3$,,4$,,W,            ;TEST THE CS ADDRESS
                     005A  8855         IFERROR         1,,<<DPM>>                  ;NOT A MACHINE CHECK
                     0060  8856         FETCH           <^X10>                      ;START DCS PROGRAM
                     0065  8857         BRSTCLK         <^X16>                      ;END DCS PROGRAM
                     0069  8858         CMPREGMSK       WHREG,1$,,2$,,L,            ;TEST FOR TB ERROR
                     0075  8859         IFERROR         1,<<UTR>>,                  ;INCORRECT ERROR
                     007B  8860         SKIP                                        ;GO TO NEXT TEST
                     0082  8861
                     0082  8862
                     0082  8863
                     0082  8864
                     0082  8865 BEGIN_TEST_DATA
                     0082
                     0082          ;-------------------------------------------------------------------------
                     0082  8866
05000000             0082  8867 1$:      .LONG    ^X05000000                         ;MACHINE CHECK REG DATA
0F000000             0086  8868 2$:      .LONG    ^X0F000000                         ;MASK FOR CMPREGMSK PSEUDO OP
    0028             008A  8869 3$:      .WORD    ^X0028                             ;MACHINE CHECK U-TRAP
    3FFF             008C  8870 4$:      .WORD    ^X3FFF                             ;MASK FOR CMPREGMSK PSEUDO OP
      04             008E  8871 5$:      .BYTE    ^X4                                ;VBUS MICRO VECTOR LINES
                     008F  8872          VBUSDATA         <^X08>,1
                     0090  8873          VBUSDATA         <^X09>,0
                     0091  8874          VBUSDATA         <^X0A>,0
                     0092  8875          VBUSDATA         <^X0B>,0
                     0093  8876
                     0093  8877
                     0093  8878
                     0093  8879 BEGIN_CPU_DATA
                     0002  8880
                     0002  8881 DCS_DATA
                     0001  8882
```

```
U 00. 7700.C800.0364.0300.0470.1801  0001  8883 ;* 00   NOP
U 01. 7700.8800.5BE4.0304.6870.1802  000C  8887 ;* 01   MEMSCAR_R[TEMP1]                          ;SET MME ON
U 02. 7700.8800.5BE4.0308.6070.1803  0017  8891 ;* 02   MEMSCR_R[TEMP2]
U 03. 7700.C800.5BE4.030C.4A70.1804  0022  8895 ;* 03   VA_R[TEMP3]                              ;LOAD VA REGISTER
U 04. 7700.C81B.5BE4.0324.5070.1805  002D  8899 ;* 04   TB_R[TEMP9]                              ;VALIDATE THE TB
U 05. 7700.8800.5BE4.0310.5C80.1806  0038  8903 ;* 05   WRITE.PHY R[TEMP4]                       ;VALIDATE THE CACHE
U 06. 7700.8800.5BE4.0310.5470.1807  0043  8907 ;* 06   WDR_UNROT(R[TEMP4])
U 07. 7700.4800.0364.0300.4470.1808  004E  8911 ;* 07   VA_VA+4                                 ;INCREMENT TB BY 4
U 08. 7700.C800.5BE4.0314.5C80.1809  0059  8915 ;* 08   WRITE.PHY R[TEMP5]                       ;VALIDATE THE CACHE
U 09. 7700.4800.5BE4.0314.5470.180A  0064  8919 ;* 09   WDR_UNROT(R[TEMP5])
U 0A. 7700.C800.5BE4.0318.0470.180B  006F  8923 ;* 0A   WB_R[TEMP6]
U 0B. 7700.4800.5BE4.0318.5360.180C  007A  8927 ;* 0B   CLRTB.VA_WB,WB_R[TEMP6]                 ;MAKE TB A MULTIPLE HIT
U 0C. 7700.4800.5BE4.030C.4870.180D  0085  8931 ;* 0C   PC_R[TEMP3]                             ;CAUSE PREFETCH
U 0D. 7701.0857.5924.0202.0470.180E  0090  8935 ;* 0D   R[TEMP0]_XB PC_PC+4                     ;READ XB0 TO RTEMP0
U 0E. 7601.8857.5924.0202.0470.180F  009B  8939 ;* 0E   R[TEMP0]_XB PC_PC+4,STROBE.V.BUS       ;READ XB1 TO RTEMP0
U 0F. 7300.C800.0364.0300.0470.1810  00A6  8943 ;* 0F   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
U 10. 7700.4800.0364.0300.0470.1811  00B1  8947 ;* 10   NOP
U 11. 7700.0800.5BE4.031C.6870.1812  00BC  8951 ;* 11   MEMSCAR_R[TEMP7]                        ;READ MACHINE CHECK REGISTER
U 12. 6700.C800.0364.0300.6470.1813  00C7  8955 ;* 12   RDM_WB,WCTRL/MEMSCR
U 13. 7700.0800.5BE4.0320.4A70.1814  00D2  8959 ;* 13   VA_R[TEMP8]
U 14. 7700.C800.5BE4.03D8.5360.1815  00DD  8963 ;* 14   CLRTB.VA_WB,WB_R[ZERO]                   ;FLUSH TB
U 15. 7700.4800.5BE4.03D8.6070.1816  00E8  8967 ;* 15   MEMSCR_R[ZERO]                          ;CLEAR MACHINE CHECK REGISTER
U 16. 7300.4800.0364.0300.0470.1817  00F3  8971 ;* 16   NOP,STOP.CLOCK                         ;STOP THE CPU CLOCK
                                     00FE  8975
                                     00FE  8976 RTEMP_DATA
                                     0001  8977
                     00000000        0001  8978          .LONG   ^X00000000                  ;DATA READ FROM XB
                     00000000        0005  8979          .LONG   ^X00000000                  ;PHY/VIR REGISTER ADDRESS
                     01000000        0009  8980          .LONG   ^X01000000                  ;SET MME ON
                     000001FC        000D  8981          .LONG   ^X000001FC                  ;VA ADDRESS
                     AAAAAAAA        0011  8982          .LONG   ^XAAAAAAAA                  ;CACHE DATA
                     55555555        0015  8983          .LONG   ^X55555555                  ;CACHE DATA
                     00000348        0019  8984          .LONG   ^X00000348                  ;USE FOR MULTIPLE HIT
                     08000000        001D  8985          .LONG   ^X08000000                  ;MACHINE CHECK REG ADDRESS
                     00000200        0021  8986          .LONG   ^X00000200                  ;ADDRESS WITH MULTIPLE HIT
                     00000148        0025  8987          .LONG   ^X00000148                  ;VALIDATE TB
                                     0029  8988
                                     0029  8989 END_CPU_DATA
                                     0093  8990
                                     0093  8991
                                     0093  8992
                                     0093  8993 END_TEST_DATA
                                     0093
                                     0093          ;-----------------------------------------------------------------------------
                                     0093  8994
                                     0093  8995 ENDTEST
```

```
0038        .SBTTL  "TEST   056     TEST XB0 FOR MACHINE CHECK WITH TB TAG PARITY ERROR
0038  8997 ;*******************************************************************************
0038  8998 ;++
0038  8999 ;
0038  9000 ; FUNCTIONAL DESCRIPTION:
0038  9001 ;
0038  9002 ;       THIS TEST VERIFIES THAT CPU CAN DETECT AND REPORT A MACHINE CHECK
0038  9003 ;       WHEN XB0 ENCOUNTERS A TB TAG PARITY ERROR.
0038  9004 ;
0038  9005 ;
0038  9006 ; TEST ALGORITHM:
0038  9007 ;
0038  9008 ;       1.  MODIFY DCS U-CODE TO DISABLE ONE TB GROUP
0038  9009 ;       2.  EXECUTE DCS PROGRAM
0038  9010 ;               a.  TURN MEMORY MANAGEMENT ON
0038  9011 ;               b.  DISABLE ONE TB GROUP
0038  9012 ;               c.  LOAD ZEROS INTO THE VA REGISTER
0038  9013 ;               d.  WRITE CACHE LOCATION ZERO
0038  9014 ;               e.  INCREMENT VA BY 4
0038  9015 ;               f.  WRITE CACHE LOCATION 4
0038  9016 ;               g.  PUT A PARITY ERROR IN THE TB TAG FIELD
0038  9017 ;               h.  LOAD THE PC WITH ZEROS TO FORCE A PREFETCH
0038  9018 ;               i.  TRY TO READ XB0 TO RTEMP6.  THIS SHOULD MACHINE CHECK
0038  9019 ;       3.  TEST THE MICRO VECTOR LINES VIA THE VBUS FOR A MACHINE CHECK
0038  9020 ;       4.  IF NO ERROR TEST THE CS ADDRESS FOR 28
0038  9021 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
0038  9022 ;               a.  READ THE MACHINE CHECK ERROR SUMMARY REGISTER TO THE WH REG
0038  9023 ;               b.  CLEAR THE PARITY ERROR OUT OF THE TB
0038  9024 ;               c.  CLEAR THE MACHINE CHECK ERROR SUMMARY REGISTER
0038  9025 ;       6.  TEST THE MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
0038  9026 ;       7.  IF NO ERROR CONTINUE
0038  9027 ;
0038  9028 ;
0038  9029 ; TEST PATTERNS:
0038  9030 ;
0038  9031 ;       NONE
0038  9032 ;
0038  9033 ;
0038  9034 ; LOGIC DESCRIPTION:
0038  9035 ;
0038  9036 ;       THIS TEST FORCES A PARITY ERROR INTO THE TB TAG FIELD OF A LOCATION
0038  9037 ;       TO BE USED BY THE PREFETCH MECHANISM OF CPU.  THE UTR GATE ARRAY
0038  9038 ;       SHOULD DETECT THE ERROR AND REPORT IT VIA A MACHINE CHECK WHEN XB0
0038  9039 ;       IS SOURCED.
0038  9040 ;
0038  9041 ;
0038  9042 ; ASSUMPTIONS:
0038  9043 ;
0038  9044 ;       THIS TEST ASSUMES THE XB AND TB DATA INTEGRITY TESTS HAVE BEEN RUN.
0038  9045 ;
0038  9046 ;
0038  9047 ; ERROR DESCRIPTION:
0038  9048 ;
0038  9049 ;       NOTE:  THERE ARE THREE IFERROR PSEUDO OPS IN EACH SUBTEST.  CHECK THE
0038  9050 ;       FAILING PC TO FIND THE CORRECT PRINTOUT.
```

```
                     9051 ;
                     9052 ;       FIRST IFERROR:
                     9053 ;               EXPECTED VBUS DATA (MICRO VECTOR LINES)
                     9054 ;               RECEIVED VBUS DATA
                     9055 ;                       BITS <7:0> = VBUS BIT POSITION
                     9056 ;                       BITS <8>   = BIT VALUE
                     9057 ;
                     9058 ;       SECOND IFERROR:
                     9059 ;               EXPECTED CS ADDRESS
                     9060 ;               RECEIVED CS ADDRESS
                     9061 ;
                     9062 ;       THIRD IFERROR:
                     9063 ;               EXPECTED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                     9064 ;               RECEIVED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                     9065 ;
                     9066 ;--
                     9067 ;*****************************************************************************
                     9068 ;/////////////////////////////////////////////////////////////////////////////
                     9069 ;+
                     9070 ;SUBTEST #1
                     9071 ;
                     9072 ;       TEST XB0 WITH TB GROUP 0
                     9073 ;-
0038                 9074         SUBTEST                                    ;SUBTEST #1
003B                 
003B                 ;/////////////////////////////////////////////////////////////////////////////
003B                 9075         INITIALIZE                                 ;INIT THE CPU
003D                 9076         EXPUTRAP                                   ;EXPECT A MACHINE CHECK
003F                 9077         LOADDCS         1$,,4,1                    ;MODIFY DCS ADDRESS 4
0046                 9078         ERRLOOP                                    ;ERROR LOOP POINT
0048                 9079         FETCH           0                          ;START DCS PROGRAM
004D                 9080         BRSTCLK         <^X0E>                     ;END DCS PROGRAM
0051                 9081         CMPVBUS         3$                         ;TEST MICRO VECTOR LINES
0056                 9082         IFERROR         ,<<UTR>>,                  ;WRONG VECTOR
005C                 9083         CMPREGMSK       CSTRP,4$,,5$,,W,           ;TEST THE CS ADDRESS
0068                 9084         IFERROR         ,,<<DPM>>                  ;NOT A MACHINE CHECK
006E                 9085         FETCH           <^X0F>                     ;START DCS PROGRAM
0073                 9086         BRSTCLK         <^X14>                     ;END DCS PROGRAM
0077                 9087         CMPREGMSK       WHREG,6$,,7$,,L,           ;TEST FOR TB ERROR WITH XB ON
0083                 9088         IFERROR         ,<<UTR>>,                  ;INCORRECT ERROR
0089                 9089
0089                 9090
0089                 9091
0089                 9092 ;/////////////////////////////////////////////////////////////////////////////
0089                 9093 ;+
0089                 9094 ;SUBTEST #2
0089                 9095 ;
0089                 9096 ;       TEST XB0 WITH TB GROUP 1
0089                 9097 ;-
0089                 9098         SUBTEST                                    ;SUBTEST #2
008C                 
008C                 ;/////////////////////////////////////////////////////////////////////////////
008C                 9099         INITIALIZE                                 ;INIT THE CPU
008E                 9100         EXPUTRAP                                   ;EXPECT A MACHINE CHECK
0090                 9101         LOADDCS         2$,,4,1                    ;MODIFY DCS ADDRESS 4
```

```
                     0097  9102          ERRLOOP                                         ;ERROR LOOP POINT
                     0099  9103          FETCH           0                               ;START DCS PROGRAM
                     009E  9104          BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                     00A2  9105          CMPVBUS         3$                              ;TEST MICRO VECTOR LINES
                     00A7  9106          IFERROR         ,<<UTR>>,                       ;WRONG VECTOR
                     00AD  9107          CMPREGMSK       CSTRP,4$,,5$,,W,                ;TEST THE CS ADDRESS
                     00B9  9108          IFERROR         ,,<<DPM>>                       ;NOT A MACHINE CHECK
                     00BF  9109          FETCH           <^X0F>                          ;START DCS PROGRAM
                     00C4  9110          BRSTCLK         <^X14>                          ;END DCS PROGRAM
                     00C8  9111          CMPREGMSK       WHREG,6$,,7$,,L,                ;TEST FOR TB ERROR WITH XB ON
                     00D4  9112          IFERROR         ,<<UTR>>,                       ;INCORRECT ERROR
                     00DA  9113          SKIP                                            ;GO TO NEXT TEST
                     00E1  9114
                     00E1  9115
                     00E1  9116
                     00E1  9117 BEGIN_TEST_DATA
                     00E1
                     00E1          ;-------------------------------------------------------------------------
                     00E1  9118
                     00E1  9119 1$:
U 04, 7700,C800,5BE4,030C,6070,1805  00E1  9120 ;% 04    MEMSCR_R[TEMP3]                 ;U-CODE SUBTEST #1
                     00EC  9124 2$:
U 04, 7700,8800,5BE4,0310,6070,1805  00EC  9125 ;% 04    MEMSCR_R[TEMP4]                 ;U-CODE SUBTEST #2
                  04 00F7  9129 3$:      .BYTE   ^X4                                     ;VBUS MICRO VECTOR LINES
                     00F8  9130          VBUSDATA        <^X08>,1
                     00F9  9131          VBUSDATA        <^X09>,0
                     00FA  9132          VBUSDATA        <^X0A>,0
                     00FB  9133          VBUSDATA        <^X0B>,0
                0028 00FC  9134 4$:      .WORD   ^X0028                                  ;MACHINE CHECK U-TRAP
                3FFF 00FE  9135 5$:      .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO OP
            05000000 0100  9136 6$:      .LONG   ^X05000000                              ;MACHINE CHECK REG DATA
            0F000000 0104  9137 7$:      .LONG   ^X0F000000                              ;MASK FOR CMPREGMSK PSEUDO OP
                     0108  9138
                     0108  9139
                     0108  9140 BEGIN_CPU_DATA
                     0002  9141
                     0002  9142 DCS_DATA
                     0001  9143
U 00, 7700,C800,0364,0300,0470,1801  0001  9144 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  9148 ;% 01    MEMSCAR_R[TEMP0]                ;SET MME ON
U 02, 7700,8800,5BE4,0304,6070,1803  0017  9152 ;% 02    MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,6870,1804  0022  9156 ;% 03    MEMSCAR_R[TEMP2]                ;DISABLE ONE TB GROUP
U 04, 7700,C800,5BE4,030C,6070,1805  002D  9160 ;% 04    MEMSCR_R[TEMP3]
U 05, 7700,4800,5BE4,03D8,4A70,1806  0038  9164 ;% 05    VA_R[ZERO]                      ;PUT 0 IN VA REGISTER
U 06, 7700,4800,5BE4,03D8,5C80,1807  0043  9168 ;% 06    WRITE.PHY R[ZERO]               ;VALIDATE CACHE LOCATION 0
U 07, 7700,C800,5BE4,03D8,5470,1808  004E  9172 ;% 07    WDR_UNROT(R[ZERO])              ;HOLD DATA FOR EXTRA CYCLE
U 08, 7700,C800,0364,0300,4470,1809  0059  9176 ;% 08    VA_VA+4                         ;INCREMENT VA BY 4
U 09, 7700,C800,5BE4,03D8,5C80,180A  0064  9180 ;% 09    WRITE.PHY R[ZERO]               ;VALIDATE CACHE LOCATION 4
U 0A, 7700,C800,5BE4,03D8,5470,180B  006F  9184 ;% 0A    WDR_UNROT(R[ZERO])              ;HOLD DATA FOR EXTRA CYCLE
U 0B, 7700,4F1B,5BE4,03D8,5070,180C  007A  9188 ;% 0B    TB_R[ZERO],MISC/FORCE.TB        ;PUT PARITY ERROR IN TB
U 0C, 7700,4800,5BE4,03D8,4870,180D  0085  9192 ;% 0C    PC_R[ZERO]                      ;FORCE PREFETCH
U 0D, 7601,0857,5924,021A,0470,180E  0090  9196 ;% 0D    R[TEMP6]_XB PC_PC+4,STROBE.V.BUS
U 0E, 7300,4800,0364,0300,0470,180F  009B  9200 ;% 0E    NOP,STOP.CLOCK                  ;STOP CPU CLOCK
U 0F, 7700,C800,0364,0300,0470,1810  00A6  9204 ;% 0F    NOP
U 10, 7700,4800,5BE4,0314,6870,1811  00B1  9208 ;% 10    MEMSCAR_R[TEMP5]                ;READ MACHINE CHECK REGISTER
```

```
U 11, 6700,4800,0364,0300,6470,1812  00BC  9212 ;% 11   RDM_WB,WCTRL/MEMSCR
U 12, 7700,481B,5BE4,03D8,5070,1813  00C7  9216 ;% 12   TB_R[ZERO]                          ;CLEAR PARITY ERROR
U 13, 7700,C800,5BE4,03D8,6070,1814  00D2  9220 ;% 13   MEMSCR_R[ZERO]                     ;CLEAR MACHINE CHECK
U 14, 7300,C800,0364,0300,0470,1815  00DD  9224 ;% 14   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
                                     00E8  9228
                                     00E8  9229 RTEMP_DATA
                                     0001  9230
                           00000000  0001  9231          .LONG   ^X00000000              ;PHY/VIR REGISTER ADDRESS
                           01000000  0005  9232          .LONG   ^X01000000              ;SET MME ON
                           03000000  0009  9233          .LONG   ^X03000000              ;TB GROUP DISABLE REG ADDRESS
                           0A000000  000D  9234          .LONG   ^X0A000000              ;DISABLE GROUP 1
                           0D000000  0011  9235          .LONG   ^X0D000000              ;DISABLE GROUP 0
                           08000000  0015  9236          .LONG   ^X08000000              ;MACHINE CHECK REGISTER ADDRESS
                                     0019  9237
                                     0019  9238 END_CPU_DATA
                                     0108  9239
                                     0108  9240
                                     0108  9241 END_TEST_DATA
                                     0108
                                     0108       ;-----------------------------------------------------------------------
                                     0108  9242
                                     0108  9243 ENDTEST
```

```
                        .SBTTL  "TEST   057     TEST XB1 FOR MACHINE CHECK WITH TB TAG PARITY ERROR
0038   9245 ;*******************************************************************************
0038   9246 ;++
0038   9247 ;
0038   9248 ; FUNCTIONAL DESCRIPTION:
0038   9249 ;
0038   9250 ;       THIS TEST VERIFIES THAT CPU CAN DETECT AND REPORT A MACHINE CHECK
0038   9251 ;       WHEN XB1 ENCOUNTERS A TB TAG PARITY ERROR.
0038   9252 ;
0038   9253 ;
0038   9254 ; TEST ALGORITHM:
0038   9255 ;
0038   9256 ;       1.  MODIFY DCS U-CODE TO DISABLE ONE TB GROUP
0038   9257 ;       2.  EXECUTE DCS PROGRAM
0038   9258 ;               a.  TURN MEMORY MANAGEMENT ON
0038   9259 ;               b.  DISABLE ONE TB GROUP
0038   9260 ;               c.  LOAD 1FC INTO THE VA REGISTER
0038   9261 ;               d.  WRITE CACHE LOCATION 1FC
0038   9262 ;               e.  VALIDATE TB FOR XB0 PREFETCH
0038   9263 ;               f.  INCREMENT VA BY 4
0038   9264 ;               g.  WRITE CACHE LOCATION 200
0038   9265 ;               h.  PUT A PARITY ERROR IN THE TB TAG FIELD
0038   9266 ;               i.  LOAD THE PC WITH 1FC TO FORCE A PREFETCH
0038   9267 ;               j.  TRY TO READ XB1 TO RTEMP8.  THIS SHOULD MACHINE CHECK
0038   9268 ;       3.  TEST THE MICRO VECTOR LINES VIA THE VBUS FOR A MACHINE CHECK
0038   9269 ;       4.  IF NO ERROR TEST THE CS ADDRESS FOR 28
0038   9270 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
0038   9271 ;               a.  READ THE MACHINE CHECK ERROR SUMMARY REGISTER TO THE WH REG
0038   9272 ;               b.  CLEAR THE PARITY ERROR OUT OF THE TB
0038   9273 ;               c.  CLEAR THE MACHINE CHECK ERROR SUMMARY REGISTER
0038   9274 ;       6.  TEST THE MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
0038   9275 ;       7.  IF NO ERROR CONTINUE
0038   9276 ;
0038   9277 ;
0038   9278 ; TEST PATTERNS:
0038   9279 ;
0038   9280 ;       NONE
0038   9281 ;
0038   9282 ;
0038   9283 ; LOGIC DESCRIPTION:
0038   9284 ;
0038   9285 ;       THIS TEST FORCES A PARITY ERROR INTO THE TB TAG FIELD OF A LOCATION
0038   9286 ;       TO BE USED BY THE PREFETCH MECHANISM OF CPU.  THE UTR GATE ARRAY
0038   9287 ;       SHOULD DETECT THE ERROR AND REPORT IT VIA A MACHINE CHECK WHEN XB1
0038   9288 ;       IS SOURCED.
0038   9289 ;
0038   9290 ;
0038   9291 ; ASSUMPTIONS:
0038   9292 ;
0038   9293 ;       THIS TEST ASSUMES THE XB AND TB DATA INTEGRITY TESTS HAVE BEEN RUN.
0038   9294 ;
0038   9295 ;
0038   9296 ; ERROR DESCRIPTION:
0038   9297 ;
0038   9298 ;       NOTE:  THERE ARE THREE IFERROR PSEUDO OPS IN EACH SUBTEST.  CHECK THE
```

```
                        9299 ;       FAILING PC TO FIND THE CORRECT PRINTOUT.
                        9300 ;
                        9301 ;       FIRST IFERROR:
                        9302 ;               EXPECTED VBUS DATA (MICRO VECTOR LINES)
                        9303 ;               RECEIVED VBUS DATA
                        9304 ;                       BITS <7:0> = VBUS BIT POSITION
                        9305 ;                       BITS <8>   = BIT VALUE
                        9306 ;
                        9307 ;       SECOND IFERROR:
                        9308 ;               EXPECTED CS ADDRESS
                        9309 ;               RECEIVED CS ADDRESS
                        9310 ;
                        9311 ;       THIRD IFERROR:
                        9312 ;               EXPECTED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                        9313 ;               RECEIVED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                        9314 ;
                        9315 ;--
                        9316 ;*****************************************************************************
                        9317 ;/////////////////////////////////////////////////////////////////////////////
                        9318 ;+
                        9319 ;SUBTEST #1
                        9320 ;
                        9321 ;       TEST XB1 WITH TB GROUP 0
                        9322 ;-
0038                    9323         SUBTEST                                    ;SUBTEST #1
003B                    9324 
003B                    9325 ;/////////////////////////////////////////////////////////////////////////////
003B                    9326         INITIALIZE                                 ;INIT THE CPU
003D                    9327         EXPUTRAP                                   ;EXPECT A MACHINE CHECK
003F                    9328         LOADDCS         1$,,4,1                    ;MODIFY DCS ADDRESS 4
0046                    9329         ERRLOOP                                    ;ERROR LOOP POINT
0048                    9330         FETCH           0                          ;START DCS PROGRAM
004D                    9331         BRSTCLK         <^X10>                     ;END DCS PROGRAM
0051                    9332         CMPVBUS         3$                         ;TEST MICRO VECTOR LINES
0056                    9333         IFERROR         ,<<UTR>>,                  ;WRONG VECTOR
005C                    9334         CMPREGMSK       CSTRP,4$,,5$,,W,           ;TEST THE CS ADDRESS
0068                    9335         IFERROR         ,,<<DPM>>                  ;NOT A MACHINE CHECK
006E                    9336         FETCH           <^X11>                     ;START DCS PROGRAM
0073                    9337         BRSTCLK         <^X19>                     ;END DCS PROGRAM
0077                    9338         CMPREGMSK       WHREG,6$,,7$,,L,           ;TEST FOR TB ERROR WITH XB ON
0083                    9339         IFERROR         ,<<UTR>>,                  ;INCORRECT ERROR
0089                    9340 
0089                    9341 ;/////////////////////////////////////////////////////////////////////////////
0089                    9342 ;+
0089                    9343 ;SUBTEST #2
0089                    9344 ;
0089                    9345 ;       TEST XB1 WITH TB GROUP 1
0089                    9346 ;-
0089                    9347         SUBTEST                                    ;SUBTEST #2
008C                    9348 
008C                    9349 ;/////////////////////////////////////////////////////////////////////////////
008C                    9350         INITIALIZE                                 ;INIT THE CPU
008E                    9351         EXPUTRAP                                   ;EXPECT A MACHINE CHECK
```

```
                    0090  9350          LOADDCS         2$,,4,1                 ;MODIFY DCS ADDRESS 4
                    0097  9351          ERRLOOP                                 ;ERROR LOOP POINT
                    0099  9352          FETCH           0                       ;START DCS PROGRAM
                    009E  9353          BRSTCLK         <^X10>                  ;END DCS PROGRAM
                    00A2  9354          CMPVBUS         3$                      ;TEST MICRO VECTOR LINES
                    00A7  9355          IFERROR         ,<<UTR>>,               ;WRONG VECTOR
                    00AD  9356          CMPREGMSK       CSTRP,4$,,5$,,W,        ;TEST THE CS ADDRESS
                    00B9  9357          IFERROR         ,,<<DPM>>               ;NOT A MACHINE CHECK
                    00BF  9358          FETCH           <^X11>                  ;START DCS PROGRAM
                    00C4  9359          BRSTCLK         <^X19>                  ;END DCS PROGRAM
                    00C8  9360          CMPREGMSK       WHREG,6$,,7$,,L,        ;TEST FOR TB ERROR WITH XB ON
                    00D4  9361          IFERROR         ,<<UTR>>,               ;INCORRECT ERROR
                    00DA  9362          SKIP                                    ;GO TO NEXT TEST
                    00E1  9363
                    00E1  9364
                    00E1  9365
                    00E1  9366 BEGIN_TEST_DATA
                    00E1
                    00E1          ;-------------------------------------------------------------------------
                    00E1  9367
                    00E1  9368 1$:
U 04, 7700,C800,5BE4,030C,6070,1805 00E1  9369 ;% 04    MEMSCR_R[TEMP3]                 ;U-CODE SUBTEST #1
                    00EC  9373 2$:
U 04, 7700,8800,5BE4,0310,6070,1805 00EC  9374 ;% 04    MEMSCR_R[TEMP4]                 ;U-CODE SUBTEST #2
                      04  00F7  9378 3$:      .BYTE   ^X4                     ;VBUS MICRO VECTOR LINES
                    00F8  9379          VBUSDATA        <^X08>,1
                    00F9  9380          VBUSDATA        <^X09>,0
                    00FA  9381          VBUSDATA        <^X0A>,0
                    00FB  9382          VBUSDATA        <^X0B>,0
                    0028  00FC  9383 4$:      .WORD   ^X0028                  ;MACHINE CHECK U-TRAP
                    3FFF  00FE  9384 5$:      .WORD   ^X3FFF                  ;MASK FOR CMPREGMSK PSEUDO OP
                05000000  0100  9385 6$:      .LONG   ^X05000000              ;MACHINE CHECK REG DATA
                0F000000  0104  9386 7$:      .LONG   ^X0F000000              ;MASK FOR CMPREGMSK PSEUDO OP
                    0108  9387
                    0108  9388
                    0108  9389 BEGIN_CPU_DATA
                    0002  9390
                    0002  9391 DCS_DATA
                    0001  9392
U 00, 7700,C800,0364,0300,0470,1801 0001  9393 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  9397 ;% 01    MEMSCAR_R[TEMP0]                ;SET MME ON
U 02, 7700,8800,5BE4,0304,6070,1803 0017  9401 ;% 02    MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,6870,1804 0022  9405 ;% 03    MEMSCAR_R[TEMP2]                ;DISABLE ONE TB GROUP
U 04, 7700,C800,5BE4,030C,6070,1805 002D  9409 ;% 04    MEMSCR_R[TEMP3]
U 05, 7700,4800,5BE4,0314,4A70,1806 0038  9413 ;% 05    VA_R[TEMP5]                     ;1FC TO VA REGISTER
U 06, 7700,4800,5BE4,03D8,5C80,1807 0043  9417 ;% 06    WRITE.PHY R[ZERO]               ;VALIDATE CACHE LOCATION 1FC
U 07, 7700,C800,5BE4,03D8,5470,1808 004E  9421 ;% 07    WDR_UNROT(R[ZERO])              ;HOLD DATA FOR EXTRA CYCLE
U 08, 7700,C81B,5BE4,0318,5070,1809 0059  9425 ;% 08    TB_R[TEMP6]                     ;VALIDATE TB FOR XB0 PREFETCH
U 09, 7700,C800,0364,0300,4470,180A 0064  9429 ;% 09    VA_VA+4                         ;INCREMENT VA BY 4
U 0A, 7700,4800,5BE4,03D8,5C80,180B 006F  9433 ;% 0A    WRITE.PHY R[ZERO]               ;VALIDATE CACHE LOCATION 200
U 0B, 7700,4800,5BE4,03D8,5470,180C 007A  9437 ;% 0B    WDR_UNROT(R[ZERO])              ;HOLD DATA FOR EXTRA CYCLE
U 0C, 7700,CF1B,5BE4,03D8,5070,180D 0085  9441 ;% 0C    TB_R[ZERO],MISC/FORCE.TB        ;PUT PARITY ERROR IN TB
U 0D, 7700,4800,5BE4,0314,4870,180E 0090  9445 ;% 0D    PC_R[TEMP5]                     ;FORCE PREFETCH
U 0E, 7701,C857,5924,0222,0470,180F 009B  9449 ;% 0E    R[TEMP8]_XB PC_PC+4             ;READ XB0 TO RTEMP8
U 0F, 7601,4857,5924,0222,0470,1810 00A6  9453 ;% 0F    R[TEMP8]_XB PC_PC+4,STROBE.V.BUS ;READ XB1 TO RTEMP8
```

```
U 10, 7300,4800,0364,0300,0470,1811  00B1  9457 ;% 10   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
U 11, 7700,4800,0364,0300,0470,1812  00BC  9461 ;% 11   NOP
U 12, 7700,8800,5BE4,031C,6870,1813  00C7  9465 ;% 12   MEMSCAR_R[TEMP7]                 ;READ MACHINE CHECK REGISTER
U 13, 6700,4800,0364,0300,6470,1814  00D2  9469 ;% 13   RDM_WB,WCTRL/MEMSCR
U 14, 7700,4800,5BE4,03D8,6070,1815  00DD  9473 ;% 14   MEMSCR_R[ZERO]                   ;CLEAR MACHINE CHECK
U 15, 7700,C800,5BE4,0314,4A70,1816  00E8  9477 ;% 15   VA_R[TEMP5]                     ;1FC TO VA REGISTER
U 16, 7700,C81B,5BE4,03D8,5070,1817  00F3  9481 ;% 16   TB_R[ZERO]                     ;INVALIDATE TB
U 17, 7700,C800,0364,0300,4470,1818  00FE  9485 ;% 17   VA_VA+4                        ;INCREMENT VA TO 200
U 18, 7700,481B,5BE4,03D8,5070,1819  0109  9489 ;% 18   TB_R[ZERO]                     ;INVALIDATE TB
U 19, 7300,C800,0364,0300,0470,181A  0114  9493 ;% 19   NOP,STOP.CLOCK                  ;STOP CPU CLOCK
                                     011F  9497
                                     011F  9498 RTEMP_DATA
                                     0001  9499
                           00000000  0001  9500          .LONG   ^X00000000           ;PHY/VIR REGISTER ADDRESS
                           01000000  0005  9501          .LONG   ^X01000000           ;SET MME ON
                           03000000  0009  9502          .LONG   ^X03000000           ;TB GROUP DISABLE REG ADDRESS
                           0A000000  000D  9503          .LONG   ^X0A000000           ;DISABLE GROUP 1
                           0D000000  0011  9504          .LONG   ^X0D000000           ;DISABLE GROUP 0
                           000001FC  0015  9505          .LONG   ^X000001FC           ;ADDRESS FOR VA AND PC
                           00000148  0019  9506          .LONG   ^X00000148           ;VALIDATE TB
                           08000000  001D  9507          .LONG   ^X08000000           ;MACHINE CHECK REGISTER ADDRESS
                                     0021  9508
                                     0021  9509 END_CPU_DATA
                                     0108  9510
                                     0108  9511
                                     0108  9512 END_TEST_DATA
                                     0108
                                     0108       ;-----------------------------------------------------------------------------
                                     0108  9513
                                     0108  9514 ENDTEST
```

```
0039        .SBTTL  "TEST   058     TEST XB0 FOR MACHINE CHECK WITH TB DATA PARITY ERROR
0039  9516 ;*******************************************************************************
0039  9517 ;++
0039  9518 ;
0039  9519 ; FUNCTIONAL DESCRIPTION:
0039  9520 ;
0039  9521 ;       THIS TEST VERIFIES THAT CPU CAN DETECT AND REPORT A MACHINE CHECK
0039  9522 ;       WHEN XB0 ENCOUNTERS A TB DATA PARITY ERROR.
0039  9523 ;
0039  9524 ;
0039  9525 ; TEST ALGORITHM:
0039  9526 ;
0039  9527 ;       1.  EXECUTE DCS PROGRAM
0039  9528 ;               a.  SET MEMORY MANAGEMENT ON
0039  9529 ;               b.  LOAD VA REGISTER WITH ZEROS
0039  9530 ;               c.  VALIDATE THE TB AT LOCATION ZERO
0039  9531 ;               d.  LOAD THE PC WITH ZEROS TO FORCE PREFETCH
0039  9532 ;               e.  MSRC XB0 WHILE ASSERTING A DATA PARITY ERROR
0039  9533 ;       2.  TEST THE VBUS FOR THE CORRECT MICRO VECTOR LINES
0039  9534 ;       3.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT MICROTRAP
0039  9535 ;       4.  IF NO ERROR EXECUTE DCS PROGRAM
0039  9536 ;               a.  READ THE MACHINE CHECK REGISTER TO THE RDM WH REGISTER
0039  9537 ;               b.  CLEAR THE MACHINE CHECK REGISTER
0039  9538 ;       5.  TEST THE WH REGISTER FOR THE CORRECT MACHINE CHECK
0039  9539 ;       6.  IF NO ERROR GO TO NEXT TEST
0039  9540 ;
0039  9541 ;
0039  9542 ; TEST PATTERNS:
0039  9543 ;
0039  9544 ;       NONE
0039  9545 ;
0039  9546 ;
0039  9547 ; LOGIC DESCRIPTION:
0039  9548 ;
0039  9549 ;       THIS TEST FORCES A PARITY ERROR INTO THE TB DATA FIELD OF A LOCATION
0039  9550 ;       BEING USED BY THE PREFETCH MECHANISM OF CPU.  THE UTR GATE ARRAY
0039  9551 ;       SHOULD DETECT THE ERROR AND REPORT IT VIA A MACHINE CHECK WHEN XB0
0039  9552 ;       IS SOURCED.
0039  9553 ;
0039  9554 ;
0039  9555 ; ASSUMPTIONS:
0039  9556 ;
0039  9557 ;       THIS TEST ASSUMES THE XB AND TB DATA INTEGRITY TESTS HAVE BEEN RUN.
0039  9558 ;
0039  9559 ;
0039  9560 ; ERROR DESCRIPTION:
0039  9561 ;
0039  9562 ;       NOTE:  THERE ARE THREE IFERROR PSEUDO OPS.  CHECK THE FAILING PC TO
0039  9563 ;       FIND THE CORRECT PRINTOUT.
0039  9564 ;
0039  9565 ;       FIRST IFERROR:
0039  9566 ;               EXPECTED VBUS DATA (MICRO VECTOR LINES)
0039  9567 ;               RECEIVED VBUS DATA
0039  9568 ;                       BITS <7:0> = VBUS BIT POSITION
0039  9569 ;                       BITS <8>   = BIT VALUE
```

```
                        0039  9570 ;
                        0039  9571 ;       SECOND IFERROR:
                        0039  9572 ;               EXPECTED CS ADDRESS
                        0039  9573 ;               RECEIVED CS ADDRESS
                        0039  9574 ;
                        0039  9575 ;       THIRD IFERROR:
                        0039  9576 ;               EXPECTED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                        0039  9577 ;               RECEIVED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                        0039  9578 ;
                        0039  9579 ;--
                        0039  9580 ;*****************************************************************************
                        0039  9581
                        0039  9582
                        0039  9583
                        0039  9584          INITIALIZE                              ;INIT THE CPU
                        003B  9585          EXPUTRAP                                ;EXPECT A MACHINE CHECK
                        003D  9586          ERRLOOP                                 ;ERROR LOOP POINT
                        003F  9587          FETCH           0                       ;START DCS PROGRAM
                        0044  9588          BRSTCLK         7                       ;END DCS PROGRAM
                        0048  9589          CMPVBUS         3$                      ;TEST MICRO VECTOR LINES
                        004D  9590          IFERROR         ,<<UTR>>,               ;WRONG VECTOR
                        0053  9591          CMPREGMSK       CSTRP,4$,,5$,,W,        ;TEST THE CS ADDRESS
                        005F  9592          IFERROR         ,,<<DPM>>               ;NOT A MACHINE CHECK
                        0065  9593          FETCH           8                       ;START DCS PROGRAM
                        006A  9594          BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                        006E  9595          CMPREGMSK       WHREG,6$,,7$,,L,        ;TEST FOR TB ERROR WITH XB ON
                        007A  9596          IFERROR         ,<<UTR>>,               ;INCORRECT ERROR
                        0080  9597          SKIP                                    ;GO TO NEXT TEST
                        0087  9598
                        0087  9599
                        0087  9600
                        0087  9601
                        0087  9602 BEGIN_TEST_DATA
                        0087
                        0087          ;-----------------------------------------------------------------------
                        0087  9603
                     04 0087  9604 3$:       .BYTE   ^X4                             ;VBUS MICRO VECTOR LINES
                        0088  9605          VBUSDATA        <^X08>,1
                        0089  9606          VBUSDATA        <^X09>,0
                        008A  9607          VBUSDATA        <^X0A>,0
                        008B  9608          VBUSDATA        <^X0B>,0
                   0028 008C  9609 4$:       .WORD   ^X0028                          ;MACHINE CHECK U-TRAP
                   3FFF 008E  9610 5$:       .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
               05000000 0090  9611 6$:       .LONG   ^X05000000                      ;MACHINE CHECK REG DATA
               0F000000 0094  9612 7$:       .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                        0098  9613
                        0098  9614
                        0098  9615 BEGIN_CPU_DATA
                        0002  9616
                        0002  9617 DCS_DATA
                        0001  9618
U 00, 7700,C800,0364,0300,0470,1801     0001  9619 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802     000C  9623 ;% 01    MEMSCAR_R[TEMP0]            ;SET MME ON
U 02, 7700,8800,5BE4,0304,6070,1803     0017  9627 ;% 02    MEMSCR_R[TEMP1]
U 03, 7700,C800,5BE4,03D8,4A70,1804     0022  9631 ;% 03    VA_R[ZERO]                  ;PUT 0 IN VA REGISTER
```

```
U 04. 7700.881B.5BE4.0308.5070.1805  002D  9635 ;% 04   TB_R[TEMP2]                                        ;VALIDATE THE TB
U 05. 7700.C800.5BE4.03D8.4870.1806  0038  9639 ;% 05   PC_R[ZERO]                                         ;FORCE PREFETCH
U 06. 7601.CF57.5924.0212.0470.1807  0043  9643 ;% 06   R[TEMP4]_XB PC_PC+4,MISC/FORCE.TB,STROBE.V.BUS
U 07. 7300.C800.0364.0300.0470.1808  004E  9647 ;% 07   NOP,STOP.CLOCK                                    ;STOP CPU CLOCK
U 08. 7700.4800.0364.0300.0470.1809  0059  9651 ;% 08   NOP
U 09. 7700.4800.5BE4.030C.6870.180A  0064  9655 ;% 09   MEMSCAR_R[TEMP3]                                  ;READ MACHINE CHECK REGISTER
U 0A. 6700.C800.0364.0300.6470.180B  006F  9659 ;% 0A   RDM_WB,WCTRL/MEMSCR
U 0B. 7700.C800.5BE4.03D8.6070.180C  007A  9663 ;% 0B   MEMSCR_R[ZERO]                                   ;CLEAR MACHINE CHECK
U 0C. 7300.C800.0364.0300.0470.180D  0085  9667 ;% 0C   NOP,STOP.CLOCK                                    ;STOP CPU CLOCK
                                     0090  9671
                                     0090  9672 RTEMP_DATA
                                     0001  9673
                          00000000   0001  9674         .LONG   ^X00000000                              ;PHY/VIR REGISTER ADDRESS
                          01000000   0005  9675         .LONG   ^X01000000                              ;SET MME ON
                          00000148   0009  9676         .LONG   ^X00000148                              ;VALIDATE THE TB
                          08000000   000D  9677         .LONG   ^X08000000                              ;MACHINE CHECK REGISTER ADDRESS
                                     0011  9678
                                     0011  9679 CACHE_DATA
                                     0001  9680
                          00000000   0001  9681         .LONG   ^X00000000                              ;VALIDATE THE CACHE 0 AND 4
                          00000000   0005  9682         .LONG   ^X00000000
                                     0009  9683
                                     0009  9684 END_CPU_DATA
                                     0098  9685
                                     0098  9686
                                     0098  9687 END_TEST_DATA
                                     0098
                                     0098         ;-----------------------------------------------------------------------------
                                     0098  9688
                                     0098  9689 ENDTEST
```

```
0039        .SBTTL  "TEST   059     TEST XB1 FOR MACHINE CHECK WITH TB DATA PARITY ERROR
0039  9691 ;*******************************************************************************
0039  9692 ;++
0039  9693 ;
0039  9694 ; FUNCTIONAL DESCRIPTION:
0039  9695 ;
0039  9696 ;       THIS TEST VERIFIES THAT CPU CAN DETECT AND REPORT A MACHINE CHECK
0039  9697 ;       WHEN XB1 ENCOUNTERS A TB DATA PARITY ERROR.
0039  9698 ;
0039  9699 ;
0039  9700 ; TEST ALGORITHM:
0039  9701 ;
0039  9702 ;       1.  EXECUTE DCS PROGRAM
0039  9703 ;               a.  TURN MEMORY MANAGEMENT ON
0039  9704 ;               b.  LOAD VA REGISTER WITH ZEROS
0039  9705 ;               c.  VALIDATE THE TB AT LOCATION ZERO
0039  9706 ;               d.  LOAD THE PC WITH ZEROS TO CAUSE PREFETCH
0039  9707 ;               e.  MSRC XB0
0039  9708 ;               f.  PERFORM NOP TO GIVE XB0 TIME TO START A PREFETCH
0039  9709 ;               g.  MSRC XB1
0039  9710 ;               h.  PERFORM NOP TO GIVE XB1 TIME TO START A PREFETCH
0039  9711 ;               i.  MSRC XB0 WHILE ASSERTING TB DATA PARITY ERROR.  THIS SHOULD
0039  9712 ;                   CATCH XB1 DOING A PREFETCH.
0039  9713 ;               j.  MSRC XB1.  THIS SHOULD MACHINE CHECK DUE TO PARITY ERROR
0039  9714 ;       2.  TEST VBUS DATA FOR CORRECT MICRO VECTOR
0039  9715 ;       3.  IF NO ERROR TEST CS ADDRESS FOR CORRECT MICROTRAP
0039  9716 ;       4.  IF NO ERROR EXECUTE DCS PROGRAM
0039  9717 ;               a.  READ MACHINE CHECK REGISTER TO RDM WH REGISTER
0039  9718 ;               b.  CLEAR MACHINE CHECK REGISTER
0039  9719 ;       5.  TEST WH REGISTER FOR CORRECT MACHINE CHECK
0039  9720 ;       6.  IF NO ERROR GO TO NEXT TEST
0039  9721 ;
0039  9722 ;
0039  9723 ; TEST PATTERNS:
0039  9724 ;
0039  9725 ;       NONE
0039  9726 ;
0039  9727 ;
0039  9728 ; LOGIC DESCRIPTION:
0039  9729 ;
0039  9730 ;       THIS TEST FORCES A PARITY ERROR INTO THE TB DATA FIELD OF A LOCATION
0039  9731 ;       BEING USED BY THE PREFETCH MECHANISM OF CPU.  THE UTR GATE ARRAY
0039  9732 ;       SHOULD DETECT THE ERROR AND REPORT IT VIA A MACHINE CHECK WHEN XB1
0039  9733 ;       IS SOURCED.
0039  9734 ;
0039  9735 ;
0039  9736 ; ASSUMPTIONS:
0039  9737 ;
0039  9738 ;       THIS TEST ASSUMES THE XB AND TB DATA INTEGRITY TESTS HAVE BEEN RUN.
0039  9739 ;
0039  9740 ;
0039  9741 ; ERROR DESCRIPTION:
0039  9742 ;
0039  9743 ;       NOTE:  THERE ARE THREE IFERROR PSEUDO OPS IN EACH SUBTEST.  CHECK THE
0039  9744 ;       FAILING PC TO FIND THE CORRECT PRINTOUT.
```

```
                     0039  9745 ;
                     0039  9746 ;       FIRST IFERROR:
                     0039  9747 ;               EXPECTED VBUS DATA (MICRO VECTOR LINES)
                     0039  9748 ;               RECEIVED VBUS DATA
                     0039  9749 ;                       BITS <7:0> = VBUS BIT POSITION
                     0039  9750 ;                       BITS <8>   = BIT VALUE
                     0039  9751 ;
                     0039  9752 ;       SECOND IFERROR:
                     0039  9753 ;               EXPECTED CS ADDRESS
                     0039  9754 ;               RECEIVED CS ADDRESS
                     0039  9755 ;
                     0039  9756 ;       THIRD IFERROR:
                     0039  9757 ;               EXPECTED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                     0039  9758 ;               RECEIVED MACHINE CHECK ERROR SUMMARY REGISTER CONTENTS
                     0039  9759 ;
                     0039  9760 ;--
                     0039  9761 ;*****************************************************************************
                     0039  9762
                     0039  9763
                     0039  9764
                     0039  9765         INITIALIZE                              ;INIT THE CPU
                     003B  9766         EXPUTRAP                                ;EXPECT A MACHINE CHECK
                     003D  9767         ERRLOOP                                 ;ERROR LOOP POINT
                     003F  9768         FETCH           0                       ;START DCS PROGRAM
                     0044  9769         BRSTCLK         <^X0C>                  ;END DCS PROGRAM
                     0048  9770         CMPVBUS         3$                      ;TEST MICRO VECTOR LINES
                     004D  9771         IFERROR         ,<<UTR>>,               ;WRONG VECTOR
                     0053  9772         CMPREGMSK       CSTRP,4$,,5$,,W,        ;TEST THE CS ADDRESS
                     005F  9773         IFERROR         ,,<<DPM>>               ;NOT A MACHINE CHECK
                     0065  9774         FETCH           <^X0D>                  ;START DCS PROGRAM
                     006A  9775         BRSTCLK         <^X11>                  ;END DCS PROGRAM
                     006E  9776         CMPREGMSK       WHREG,6$,,7$,,L,        ;TEST FOR TB ERROR WITH XB ON
                     007A  9777         IFERROR         ,<<UTR>>,               ;INCORRECT ERROR
                     0080  9778         SKIP                                    ;GO TO NEXT TEST
                     0087  9779
                     0087  9780 BEGIN_TEST_DATA
                     0087
                     0087         ;-----------------------------------------------------------------------
                     0087  9781
                  04 0087  9782 3$:      .BYTE   ^X4                             ;VBUS MICRO VECTOR LINES
                     0088  9783         VBUSDATA        <^X08>,1
                     0089  9784         VBUSDATA        <^X09>,0
                     008A  9785         VBUSDATA        <^X0A>,0
                     008B  9786         VBUSDATA        <^X0B>,0
                0028 008C  9787 4$:     .WORD   ^X0028                          ;MACHINE CHECK U-TRAP
                3FFF 008E  9788 5$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
            05000000 0090  9789 6$:     .LONG   ^X05000000                      ;MACHINE CHECK REG DATA
            0F000000 0094  9790 7$:     .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                     0098  9791
                     0098  9792
                     0098  9793 BEGIN_CPU_DATA
                     0002  9794
                     0002  9795 DCS_DATA
                     0001  9796
U 00, 7700,C800,0364,0300,0470,1801  0001  9797 ;* 00    NOP
```

```
U 01. 7700.C800.5BE4.0300.6870.1802  000C  9801 ;% 01   MEMSCAR_R[TEMP0]                          ;SET MME ON
U 02. 7700.8800.5BE4.0304.6070.1803  0017  9805 ;% 02   MEMSCR_R[TEMP1]
U 03. 7700.C800.5BE4.03D8.4A70.1804  0022  9809 ;% 03   VA_R[ZERO]                               ;PUT 0 IN VA REGISTER
U 04. 7700.881B.5BE4.0308.5070.1805  002D  9813 ;% 04   TB_R[TEMP2]                             ;VALIDATE THE TB
U 05. 7700.C800.5BE4.03D8.4870.1806  0038  9817 ;% 05   PC_R[ZERO]                             ;FORCE PREFETCH
U 06. 7701.4857.5924.0212.0470.1807  0043  9821 ;% 06   R[TEMP4]_XB PC_PC+4                    ;SOURCE XB0
U 07. 7700.C800.0364.0300.0470.1808  004E  9825 ;% 07   NOP                                    ;ALLOW PREFETCH OF XB0
U 08. 7701.C857.5924.0212.0470.1809  0059  9829 ;% 08   R[TEMP4]_XB PC_PC+4                    ;SOURCE XB1
U 09. 7700.4800.0364.0300.0470.180A  0064  9833 ;% 09   NOP                                    ;ALLOW PREFETCH OF XB1
U 0A. 7701.CF57.5924.0212.0470.180B  006F  9837 ;% 0A   R[TEMP4]_XB PC_PC+4,MISC/FORCE.TB      ;SOURCE XB0,CAUSE PE ON XB1
U 0B. 7601.C857.5924.0212.0470.180C  007A  9841 ;% 0B   R[TEMP4]_XB PC_PC+4,STROBE.V.BUS      ;SOURCE XB1
U 0C. 7300.C800.0364.0300.0470.180D  0085  9845 ;% 0C   NOP,STOP.CLOCK                         ;STOP CPU CLOCK
U 0D. 7700.C800.0364.0300.0470.180E  0090  9849 ;% 0D   NOP
U 0E. 7700.4800.5BE4.030C.6870.180F  009B  9853 ;% 0E   MEMSCAR_R[TEMP3]                       ;READ MACHINE CHECK REGISTER
U 0F. 6700.C800.0364.0300.6470.1810  00A6  9857 ;% 0F   RDM_WB,WCTRL/MEMSCR
U 10. 7700.C800.5BE4.03D8.6070.1811  00B1  9861 ;% 10   MEMSCR_R[ZERO]                         ;CLEAR MACHINE CHECK
U 11. 7300.4800.0364.0300.0470.1812  00BC  9865 ;% 11   NOP,STOP.CLOCK                         ;STOP CPU CLOCK
                                     00C7  9869
                                     00C7  9870 RTEMP_DATA
                                     0001  9871
                     00000000        0001  9872          .LONG   ^X00000000                  ;PHY/VIR REGISTER ADDRESS
                     01000000        0005  9873          .LONG   ^X01000000                  ;SET MME ON
                     00000148        0009  9874          .LONG   ^X00000148                  ;VALIDATE THE TB
                     08000000        000D  9875          .LONG   ^X08000000                  ;MACHINE CHECK REGISTER ADDRESS
                                     0011  9876
                                     0011  9877 CACHE_DATA
                                     0001  9878
                     00000000        0001  9879          .LONG   ^X00000000                  ;VALIDATE ADDREESSES 0 TO 10
                     00000000        0005  9880          .LONG   ^X00000000
                     00000000        0009  9881          .LONG   ^X00000000
                     00000000        000D  9882          .LONG   ^X00000000
                     00000000        0011  9883          .LONG   ^X00000000
                                     0015  9884
                                     0015  9885 END_CPU_DATA
                                     0098  9886
                                     0098  9887
                                     0098  9888 END_TEST_DATA
                                     0098
                                     0098          ;-----------------------------------------------------------------------------
                                     0098  9889
                                     0098  9890 ENDTEST
```

```
0027        .SBTTL  "TEST   05A     TEST FOR MICRO-TRAP ON XB0 TB MISS
0027  9892 ;*******************************************************************************
0027  9893 ;++
0027  9894 ;
0027  9895 ; FUNCTIONAL DESCRIPTION:
0027  9896 ;
0027  9897 ;       THIS TEST WILL VERIFY THAT CPU CAN DETECT AND REPORT A TB MISS WHEN
0027  9898 ;       XB0 IS TRYING TO PREFETCH INSTRUCTIONS.
0027  9899 ;
0027  9900 ;
0027  9901 ; TEST ALGORITHM:
0027  9902 ;
0027  9903 ;       1.  MODIFY MICROCODE FOR SPECIFIC SUBTEST
0027  9904 ;       2.  EXECUTE DCS PROGRAM
0027  9905 ;               a.  TURN MEMORY MANAGEMENT ON
0027  9906 ;               b.  PUT ZEROS IN THE VA REGISTER
0027  9907 ;               c.  MAKE TB LOCATION ZERO A MISS
0027  9908 ;               d.  LOAD PC WITH ZEROS TO FORCE A PREFETCH
0027  9909 ;               e.  TRY TO MSRC XB0.  THIS SHOULD MICRO TRAP.
0027  9910 ;                   (SUBTEST #2 WILL TRY TO PERFORM AN IRD1TST ON XB0)
0027  9911 ;       3.  TEST VBUS FOR CORRECT MICRO VECTOR LINES AND GEN DEST INH
0027  9912 ;       4.  IF NO ERROR TEST CS ADDRESS FOR CORRECT TRAP
0027  9913 ;       5.  IF NO ERROR GO TO NEXT TEST
0027  9914 ;
0027  9915 ;
0027  9916 ; TEST PATTERNS:
0027  9917 ;
0027  9918 ;       NONE
0027  9919 ;
0027  9920 ;
0027  9921 ; LOGIC DESCRIPTION:
0027  9922 ;
0027  9923 ;       THE UTR GATE ARRAY SHOULD LATCH THE FACT THAT XB0 ENCOUNTERED A TB
0027  9924 ;       MISS WHEN IT TRIED TO PREFETCH INSTRUCTIONS.  IT SHOULD REPORT THE
0027  9925 ;       MISS, VIA A MICROTRAP, WHEN XB0 IS SOURCED.
0027  9926 ;
0027  9927 ;
0027  9928 ; ASSUMPTIONS:
0027  9929 ;
0027  9930 ;       THIS TEST ASSUMES THE DATA INTEGRITY TESTS FOR THE XB AND TB HAVE
0027  9931 ;       BEEN RUN.
0027  9932 ;
0027  9933 ;
0027  9934 ; ERROR DESCRIPTION:
0027  9935 ;
0027  9936 ;       NOTE: THERE ARE TWO IFERROR STATEMENTS IN THIS TEST.  SO VERIFY THE
0027  9937 ;       FAILING PC TO DETERMINE THE CORRECT PRINT OUT.
0027  9938 ;
0027  9939 ;       FIRST IFERROR:
0027  9940 ;       EXPECTED VBUS DATA (MICRO VECTOR LINES)
0027  9941 ;       RECEIVED VBUS DATA
0027  9942 ;               (BITS <7:0> = VBUS BIT POSITION)
0027  9943 ;               (BITS <8> = BIT VALUE)
0027  9944 ;
0027  9945 ;       SECOND IFERROR:
```

```
                        9946 ;       EXPECTED CS ADDRESS
                        9947 ;       RECEIVED CS ADDRESS
                        9948 ;
                        9949 ;
                        9950 ;**************************************************************************************
                        9951 ;//////////////////////////////////////////////////////////////////////////////////////
                        9952 ;+
                        9953 ;SUBTEST #1
                        9954 ;
                        9955 ;       TEST XB0 MISS WITH MSRC MICRO ORDER
                        9956 ;-
0027                    9957         SUBTEST                                        ;SUBTEST #1
002A                    9958
002A                    9959 ;//////////////////////////////////////////////////////////////////////////////////////
002A                    9960         INITIALIZE                                     ;INIT THE CPU
002C                    9961         LOADDCS         4$,,06,1                       ;MODIFY DCS U-CODE
0033                    9962         EXPUTRAP                                       ;EXPECT A MICRO TRAP
0035                    9963         ERRLOOP                                        ;ERROR LOOP POINT
0037                    9964         FETCH           0                              ;START DCS PROGRAM
003C                    9965         BRSTCLK         7                              ;END DCS PROGRAM
0040                    9966         CMPVBUS         1$                             ;TEST MICRO VECTOR LINES
0045                    9967         IFERROR         ,<<UTR>>,                      ;WRONG VECTOR
004B                    9968         CMPREGMSK       CSTRP,2$,,3$,,W,               ;TEST CS ADDRESS
0057                    9969         IFERROR         ,,<<DPM>>                      ;WRONG CS ADDRESS
005D                    9970
005D                    9971 ;//////////////////////////////////////////////////////////////////////////////////////
005D                    9972 ;+
005D                    9973 ;SUBTEST #2
005D                    9974 ;
005D                    9975 ;       TEST XB0 MISS WITH IRD1TST MICRO ORDER
005D                    9976 ;-
0060                    9977         SUBTEST                                        ;SUBTEST #2
0060                    9978
0060                    9979 ;//////////////////////////////////////////////////////////////////////////////////////
0062                    9980         INITIALIZE                                     ;INIT THE CPU
0064                    9981         EXPUTRAP                                       ;EXPECT A MICRO TRAP
006B                    9982         LOADDCS         5$,,06,1                       ;MODIFY DCS ADDRESS 06
006D                    9983         ERRLOOP                                        ;ERROR LOOP POIN
0072                    9984         FETCH           0                              ;START DCS PROGRAM
0076                    9985         BRSTCLK         7                              ;END DCS PROGRAM
007B                    9986         CMPVBUS         6$                             ;TEST MICOR VECTOR LINES
0081                    9987         IFERROR         ,<<UTR>>,                      ;WRONG VECTOR LINES
008D                    9988         CMPREGMSK       CSTRP,7$,,3$,,W,               ;TEST FOR CORRECT CS ADDRESS
0093                    9989         IFERROR         ,,<<DPM>>                      ;WRONG CS ADDRESS
009A                    9990         SKIP                                           ;GO TO NEXT TEST
009A                    9991
009A                    9992 BEGIN_TEST_DATA
009A                    9993
009A                    9994 ;-------------------------------------------------------------------------
009A                    9995
05  009A                9996 1$:     .BYTE   ^X5                                    ;MICRO VECTOR LINES
009B                    9997         VBUSDATA        <^X08>,0
009C                    9998         VBUSDATA        <^X09>,0
```

```
                                009D  9995          VBUSDATA        <^X0A>,1
                                009E  9996          VBUSDATA        <^X0B>,0
                                009F  9997          VBUSDATA        <^X0C>,0                ;GEN DEST INH L
                          0022  00A0  9998 2$:      .WORD   ^X0022                          ;XB MISS U-TRAP
                          3FFF  00A2  9999 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                00A4 10000 4$:
U 06, 7601,0857,5924,020E,0470,1807   00A4 10001 ;* 06   R[TEMP3]_XB PC_PC+4,STROBE.V.BUS      ;U-CODE SUBTEST #1
                                00AF 10005 5$:
U 06, 7600,C800,0364,1700,0470,1807   00AF 10006 ;* 06   BUT/IRD1TST,STROBE.V.BUS             ;U-CODE SUBTEST #2
                             05   00BA 10010 6$:      .BYTE   ^X5                             ;MICRO VECTOR LINES
                                00BB 10011          VBUSDATA        <^X08>,1
                                00BC 10012          VBUSDATA        <^X09>,0
                                00BD 10013          VBUSDATA        <^X0A>,0
                                00BE 10014          VBUSDATA        <^X0B>,1
                                00BF 10015          VBUSDATA        <^X0C>,1                ;GEN DEST INH L
                          0029  00C0 10016 7$:      .WORD   ^X0029                          ;BUT XB MISS U-TRAP
                                00C2 10017
                                00C2 10018
                                00C2 10019
                                00C2 10020 BEGIN_CPU_DATA
                                0002 10021
                                0002 10022 DCS_DATA
                                0001 10023
U 00, 7700,C800,0364,0300,0470,1801   0001 10024 ;* 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C 10028 ;* 01   MEMSCAR_R[TEMP0]                    ;TURN MME ON
U 02, 7700,8800,5BE4,0304,6070,1803   0017 10032 ;* 02   MEMSCR_R[TEMP1]
U 03, 7700,C800,5BE4,03D8,4A70,1804   0022 10036 ;* 03   VA_R[ZERO]                         ;0'S TO VA REGISTER
U 04, 7700,881B,5BE4,0308,5070,1805   002D 10040 ;* 04   TB_R[TEMP2]                       ;MAKE TB A MISS
U 05, 7700,C800,5BE4,03D8,4870,1806   0038 10044 ;* 05   PC_R[ZERO]                         ;FORCE PREFETCH
U 06, 7601,0857,5924,020E,0470,1807   0043 10048 ;* 06   R[TEMP3]_XB PC_PC+4,STROBE.V.BUS    ;CAUSE U-TRAP
U 07, 7300,C800,0364,0300,0470,1808   004E 10052 ;* 07   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
                                0059 10056
                                0059 10057 RTEMP_DATA
                                0001 10058
                    00000000    0001 10059          .LONG   ^X00000000                      ;PHY/VIR REGISTER ADDRESS
                    01000000    0005 10060          .LONG   ^X01000000                      ;TURN MME ON
                    00000048    0009 10061          .LONG   ^X00000048                      ;TB MISS
                                000D 10062
                                000D 10063 CACHE_DATA
                                0001 10064
                    00000000    0001 10065          .LONG   ^X00000000                      ;VALIDATE CACHE FOR PREFETCH
                    00000000    0005 10066          .LONG   ^X00000000
                                0009 10067
                                0009 10068 END_CPU_DATA
                                00C2 10069
                                00C2 10070
                                00C2 10071 END_TEST_DATA
                                00C2
                                00C2          ;-----------------------------------------------------------------------
                                00C2 10072 ENDTEST
```

```
                        0027        .SBTTL  "TEST   05B     TEST FOR MICRO-TRAP ON XB1 TB MISS
                        0027 10074 ;*******************************************************************************
                        0027 10075 ;++
                        0027 10076 ;
                        0027 10077 ; FUNCTIONAL DESCRIPTION:
                        0027 10078 ;
                        0027 10079 ;       THIS TEST VERIFIES THAT CPU CAN DETECT AND REPORT A TB MISS WHEN
                        0027 10080 ;       XB1 IS PREFETCHING INSTRUCTIONS.
                        0027 10081 ;
                        0027 10082 ;
                        0027 10083 ; TEST ALGORITHM:
                        0027 10084 ;
                        0027 10085 ;       1.  LOAD MICROCODE FOR SPECIFIC SUBTEST
                        0027 10086 ;       2.  EXECUTE DCS PROGRAM
                        0027 10087 ;               a.  TURN MEMORY MANAGEMENT ON
                        0027 10088 ;               b.  LOAD VA WITH 1FC
                        0027 10089 ;               c.  VALIDATE TB AND CACHE AT THIS LOCATION
                        0027 10090 ;               d.  INCREMENT VA TO 200
                        0027 10091 ;               e.  MAKE TB A MISS AT THIS LOCATION
                        0027 10092 ;               f.  VALIDATE CACHE AT THIS LOCATION
                        0027 10093 ;               g.  LOAD PC WITH 1FC TO CAUSE A PREFETCH
                        0027 10094 ;               h.  MSRC XB0
                        0027 10095 ;               i.  MSRC XB1. THIS SHOULD MICROTRAP
                        0027 10096 ;                   (SUBTEST 2 WILL USE IRD1TST TO TEST FOR MICROTRAP)
                        0027 10097 ;       3.  TEST VBUS FOR CORRECT MICRO VECTOR LINES AND GEN DEST INH
                        0027 10098 ;       4.  IF NO ERROR TEST CS ADDRESS FOR CORRECT MICROTRAP
                        0027 10099 ;       5.  IF NO ERROR GO TO NEXT TEST
                        0027 10100 ;
                        0027 10101 ;
                        0027 10102 ; TEST PATTERNS:
                        0027 10103 ;
                        0027 10104 ;       NONE
                        0027 10105 ;
                        0027 10106 ;
                        0027 10107 ; LOGIC DESCRIPTION:
                        0027 10108 ;
                        0027 10109 ;       THE UTR GATE ARRAY SHOULD LATCH THE FACT THAT XB1 EXPERIENCED A
                        0027 10110 ;       TB MISS DURING PREFETCH.  IT SHOULD REPORT THIS, VIA A MICROTRAP,
                        0027 10111 ;       WHEN XB1 IS SOURCED.
                        0027 10112 ;
                        0027 10113 ;
                        0027 10114 ; ASSUMPTIONS:
                        0027 10115 ;
                        0027 10116 ;       THIS TEST ASSUMES THE TB AND XB DATA INTEGRITY TESTS HAVE RUN.
                        0027 10117 ;
                        0027 10118 ;
                        0027 10119 ; ERROR DESCRIPTION:
                        0027 10120 ;
                        0027 10121 ;       NOTE: THERE ARE TWO IFERROR PSEUDO OPS IN THIS TEST.  SO CHECK THE
                        0027 10122 ;       FAILING PC TO GET THE CORRECT PRINT OUT.
                        0027 10123 ;
                        0027 10124 ;       FIRST IFERROR:
                        0027 10125 ;       EXPECTED VBUS DATA (MICRO VECTOR LINES)
                        0027 10126 ;       RECEIVED VBUS DATA
                        0027 10127 ;               (BITS <7:0> = VBUS BIT POSITION)
```

```
0027 10128 ;                (BITS <8> = BIT VALUE)
0027 10129 ;
0027 10130 ;        SECOND IFERROR:
0027 10131 ;        EXPECTED CS ADDRESS
0027 10132 ;        RECEIVED CS ADDRESS
0027 10133 ;
0027 10134 ;--
0027 10135 ;*****************************************************************************
0027 10136 ;/////////////////////////////////////////////////////////////////////////////
0027 10137 ;+
0027 10138 ;SUBTEST #1
0027 10139 ;
0027 10140 ;        TEST XB1 MISS WITH MSRC MICRO ORDER
0027 10141 ;-
0027 10142          SUBTEST                                     ;SUBTEST #1
002A
002A        ;/////////////////////////////////////////////////////////////////////////
002A 10143          INITIALIZE                                  ;INIT THE CPU
002C 10144          EXPUTRAP                                    ;EXPECT A MICRO TRAP
002E 10145          LOADDCS         4$,,<^X0D>,1                ;MODIFY U-CODE
0035 10146          ERRLOOP                                     ;ERROR LOOP POINT
0037 10147          FETCH           0                           ;START DCS PROGRAM
003C 10148          BRSTCLK         <^X0E>                      ;END DCS PROGRAM
0040 10149          CMPVBUS         1$                          ;TEST MICRO VECTOR LINES
0045 10150          IFERROR         ,<<UTR>>,                   ;WRONG VECTOR
004B 10151          CMPREGMSK       CSTRP,2$,,3$,,W,            ;TEST CS ADDRESS
0057 10152          IFERROR         ,,<<DPM>>                   ;WRONG CS ADDRESS
005D 10153
005D 10154
005D 10155
005D 10156 ;/////////////////////////////////////////////////////////////////////////////
005D 10157 ;+
005D 10158 ;SUBTEST #2
005D 10159 ;
005D 10160 ;        TEST XB1 MISS WITH IRD1TST MICRO ORDER
005D 10161 ;-
005D 10162          SUBTEST                                     ;SUBTEST #2
0060
0060        ;/////////////////////////////////////////////////////////////////////////
0060 10163          INITIALIZE                                  ;INIT THE CPU
0062 10164          EXPUTRAP                                    ;EXPECT A MICRO TRAP
0064 10165          LOADDCS         5$,,<^X0D>,1                ;MODIFY U-CODE
006B 10166          ERRLOOP                                     ;ERROR LOOP POINT
006D 10167          FETCH           0                           ;START DCS PROGRAM
0072 10168          BRSTCLK         <^X0E>                      ;END DCS PROGRAM
0076 10169          CMPVBUS         6$                          ;TEST MICRO VECTOR LINES
007B 10170          IFERROR         ,<<UTR>>,                   ;WRONG VECTOR
0081 10171          CMPREGMSK       CSTRP,7$,,3$,,W,            ;TEST CS ADDRESS
008D 10172          IFERROR         ,,<<DPM>>                   ;INCORRECT ADDRESS
0093 10173          SKIP                                        ;GO TO NEXT TEST
009A 10174
009A 10175
009A 10176 BEGIN_TEST_DATA
009A
009A        ;-----------------------------------------------------------------------
```

```
                        009A 10177
                    05  009A 10178 1$:     .BYTE   ^X5                                     ;MICRO VECTOR LINES
                        009B 10179          VBUSDATA        <^X08>,0
                        009C 10180          VBUSDATA        <^X09>,0
                        009D 10181          VBUSDATA        <^X0A>,1
                        009E 10182          VBUSDATA        <^X0B>,0
                        009F 10183          VBUSDATA        <^X0C>,0                        ;GEN DEST INH
                  0022  00A0 10184 2$:      .WORD   ^X0022                                  ;XB MISS U-TRAP
                  3FFF  00A2 10185 3$:      .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO OP
                        00A4 10186 4$:
U 0D, 7601,0857,5924,0216,0470,180E     00A4 10187 ;* 0D    R[TEMP5]_XB PC_PC+4,STROBE.V.BUS     ;U-CODE SUBTEST #1
                        00AF 10191 5$:
U 0D, 7600,C800,0364,1700,0470,180E     00AF 10192 ;* 0D    BUT/IRD1TST,STROBE.V.BUS             ;U-CODE SUBTEST #2
                    05  00BA 10196 6$:     .BYTE   ^X5                                     ;MICRO VECTOR LINES
                        00BB 10197          VBUSDATA        <^X08>,1
                        00BC 10198          VBUSDATA        <^X09>,0
                        00BD 10199          VBUSDATA        <^X0A>,0
                        00BE 10200          VBUSDATA        <^X0B>,1
                        00BF 10201          VBUSDATA        <^X0C>,1                        ;GEN DEST INH
                  0029  00C0 10202 7$:      .WORD   ^X0029                                  ;BUT XB MISS U-TRAP
                        00C2 10203
                        00C2 10204
                        00C2 10205
                        00C2 10206 BEGIN_CPU_DATA
                        0002 10207
                        0002 10208 DCS_DATA
                        0001 10209
U 00, 7700,C800,0364,0300,0470,1801     0001 10210 ;* 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802     000C 10214 ;* 01    MEMSCAR_R[TEMP0]                    ;TURN MME ON
U 02, 7700,8800,5BE4,0304,6070,1803     0017 10218 ;* 02    MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,4A70,1804     0022 10222 ;* 03    VA_R[TEMP2]                         ;1FC TO VA REGISTER
U 04, 7700,C81B,5BE4,030C,5070,1805     002D 10226 ;* 04    TB_R[TEMP3]                         ;VALIDATE TB
U 05, 7700,C800,5BE4,03D8,5C80,1806     0038 10230 ;* 05    WRITE.PHY R[ZERO]                   ;VALIDATE CACHE
U 06, 7700,C800,5BE4,03D8,5470,1807     0043 10234 ;* 06    WDR_UNROT(R[ZERO])                  ;HOLD DATA FOR EXTRA CYCLE
U 07, 7700,4800,0364,0300,4470,1808     004E 10238 ;* 07    VA_VA+4                             ;INCREMENT VA TO 200
U 08, 7700,881B,5BE4,0310,5070,1809     0059 10242 ;* 08    TB_R[TEMP4]                         ;MAKE TB A MISS
U 09, 7700,C800,5BE4,03D8,5C80,180A     0064 10246 ;* 09    WRITE.PHY R[ZERO]                   ;VALIDATE CACHE
U 0A, 7700,C800,5BE4,03D8,5470,180B     006F 10250 ;* 0A    WDR_UNROT(R[ZERO])                  ;HOLD DATA FOR EXTRA CYCLE
U 0B, 7700,8800,5BE4,0308,4870,180C     007A 10254 ;* 0B    PC_R[TEMP2]                         ;FORCE PREFETCH
U 0C, 7701,0857,5924,0216,0470,180D     0085 10258 ;* 0C    R[TEMP5]_XB PC_PC+4                 ;READ XB0
U 0D, 7601,0857,5924,0216,0470,180E     0090 10262 ;* 0D    R[TEMP5]_XB PC_PC+4,STROBE.V.BUS    ;READ XB1/GET U-TRAP
U 0E, 7300,4800,0364,0300,0470,180F     009B 10266 ;* 0E    NOP,STOP.CLOCK                      ;STOP CPU CLOCK
                        00A6 10270
                        00A6 10271 RTEMP_DATA
                        0001 10272
            00000000    0001 10273          .LONG   ^X00000000                              ;PHY/VIR REGISTER ADDRESS
            01000000    0005 10274          .LONG   ^X01000000                              ;TURN MME ON
            000001FC    0009 10275          .LONG   ^X000001FC                              ;ADDRESS FOR VA
            00000148    000D 10276          .LONG   ^X00000148                              ;VALIDATE TB
            00000048    0011 10277          .LONG   ^X00000048                              ;TB MISS
                        0015 10278
                        0015 10279 END_CPU_DATA
                        00C2 10280
                        00C2 10281
                        00C2 10282 END_TEST_DATA
```

```
                    00C2
                    00C2        ;-----------------------------------------------------------------------------
                    00C2 10283 ENDTEST
```

```
                        0034        .SBTTL  "TEST   05C     TEST XB0 FOR MICROTRAP WITH TB ACCESS VIOLATION
                        0034 10285 ;*******************************************************************************
                        0034 10286 ;++
                        0034 10287 ;
                        0034 10288 ; FUNCTIONAL DESCRIPTION:
                        0034 10289 ;
                        0034 10290 ;       THIS TEST CHECKS THAT CPU CAN DETECT AN ACCESS VIOLATION DURING
                        0034 10291 ;       INSTRUCTION PREFETCH AND REPORT THE SAME VIA A MICROTRAP.
                        0034 10292 ;
                        0034 10293 ; TEST ALGORITHM:
                        0034 10294 ;
                        0034 10295 ;       1.  MODIFY MICROCODE FOR SPECIFIC SUBTEST
                        0034 10296 ;       2.  EXECUTE DCS PROGRAM
                        0034 10297 ;               a.  TURN MEMORY MANAGEMENT ON
                        0034 10298 ;               b.  PUT ZEROS INTO THE VA REGISTER
                        0034 10299 ;               c.  LOAD TB WITH AN ACCESS VIOLATION
                        0034 10300 ;               d.  LOAD PC WITH ZEROS TO FORCE A PREFETCH
                        0034 10301 ;               e.  SOURCE XB0; THIS SHOULD MICROTRAP
                        0034 10302 ;       3.  TEST VBUS FOR CORRECT MICRO VECTOR LINES
                        0034 10303 ;       4.  IF NO ERROR TEST CS ADDRESS FOR PROPER MICROTRAP
                        0034 10304 ;       5.  IF NO ERROR GO TO NEXT TEST
                        0034 10305 ;
                        0034 10306 ;
                        0034 10307 ; TEST PATTERNS:
                        0034 10308 ;
                        0034 10309 ;       NONE
                        0034 10310 ;
                        0034 10311 ;
                        0034 10312 ; LOGIC DESCRIPTION:
                        0034 10313 ;
                        0034 10314 ;       THE UTR GATE ARRAY SHOULD DETECT THE FACT THAT XB0 EXPERIENCES AN
                        0034 10315 ;       ACCESS VIOLATION WHILE PREFETCHING INSTRUCTIONS.  THE ERROR CONDITION
                        0034 10316 ;       SHOULD BE REPORTED VIA A MICROTRAP WHEN XB0 IS SOURCED.
                        0034 10317 ;
                        0034 10318 ;
                        0034 10319 ; ASSUMPTIONS:
                        0034 10320 ;
                        0034 10321 ;       THIS TEST ASSUMES THE TB AND XB DATA INTEGRITY TESTS HAVE BEEN RUN
                        0034 10322 ;
                        0034 10323 ;
                        0034 10324 ; ERROR DESCRIPTION:
                        0034 10325 ;
                        0034 10326 ;       NOTE: THERE ARE TWO IFERROR PSEUDO OPS IN EACH SUBTEST.  YOU WILL
                        0034 10327 ;       HAVE TO CHECK THE FAILING PC TO DETERMINE THE CORRECT PRINTOUT.
                        0034 10328 ;
                        0034 10329 ;       FIRST IFERROR:
                        0034 10330 ;       EXPECTED VBUS DATA (MICRO VECTOR LINES)
                        0034 10331 ;       RECEIVED VBUS DATA
                        0034 10332 ;               (BITS <7:0> = VBUS BIT POSITION)
                        0034 10333 ;               (BITS <8> = BIT VALUE)
                        0034 10334 ;
                        0034 10335 ;       SECOND IFERROR:
                        0034 10336 ;       EXPECTED CS ADDRESS
                        0034 10337 ;       RECEIVED CS ADDRESS
                        0034 10338 ;
```

```
                0034 10339 ;--
                0034 10340 ;*******************************************************************************
                0034 10341 ;///////////////////////////////////////////////////////////////////////////////
                0034 10342 ;+
                0034 10343 ;SUBTEST #1
                0034 10344 ;
                0034 10345 ;       TEST FOR MICROTRAP USING MSRC XB0
                0034 10346 ;-
                0034 10347         SUBTEST                                         ;SUBTEST #1
                0037
                0037       ;////////////////////////////////////////////////////////////////////////
                0037 10348         INITIALIZE                                      ;INIT THE CPU
                0039 10349         EXPUTRAP                                        ;EXPECT A MICROTRAP
                003B 10350         LOADDCS         1$,,06,1                        ;MODIFY U-CODE FOR THIS SUBTEST
                0042 10351         ERRLOOP                                         ;ERROR LOOP POINT
                0044 10352         FETCH           0                               ;START DCS PROGRAM
                0049 10353         BRSTCLK         7                               ;END DCS PROGRAM
                004D 10354         CMPVBUS         2$                              ;TEST MICRO VECTOR LINES
                0052 10355         IFERROR         ,<<UTR>>,                       ;WRONG VECTOR
                0058 10356         CMPREGMSK       CSTRP,3$,,4$,,W,                ;TEST CS ADDRESS
                0064 10357         IFERROR         ,,<<DPM>>                       ;WRONG CS ADDRESS
                006A 10358
                006A 10359
                006A 10360
                006A 10361 ;*******************************************************************************
                006A 10362 ;+
                006A 10363 ;SUBTEST #2
                006A 10364 ;
                006A 10365 ;       TEST FOR MICROTRAP USING IRD1TST ON XB0
                006A 10366 ;-
                006A 10367         SUBTEST                                         ;SUBTEST #2
                006D
                006D       ;////////////////////////////////////////////////////////////////////////
                006D 10368         INITIALIZE                                      ;INIT THE CPU
                006F 10369         EXPUTRAP                                        ;EXPECT A U-TRAP
                0071 10370         LOADDCS         5$,,06,1                        ;MODIFY U-CODE FOR THIS SUBTEST
                0078 10371         ERRLOOP                                         ;ERROR LOOP POINT
                007A 10372         FETCH           0                               ;START DCS PROGRAM
                007F 10373         BRSTCLK         7                               ;END DCS PROGRAM
                0083 10374         CMPVBUS         6$                              ;TEST MICRO VECTOR LINES
                0088 10375         IFERROR         ,<<UTR>>,                       ;WRONG VECTOR
                008E 10376         CMPREGMSK       CSTRP,7$,,4$,,W,                ;TEST CS ADDRESS
                009A 10377         IFERROR         ,,<<DPM>>                       ;WRONG CS ADDRESS
                00A0 10378         SKIP
                00A7 10379
                00A7 10380
                00A7 10381
                00A7 10382 BEGIN_TEST_DATA
                00A7
                00A7       ;-------------------------------------------------------------------------
                00A7 10383
                00A7 10384 1$:
U 06, 7601,0857,5924,020E,0470,1807
                00A7 10385 ;% 06   R[TEMP3]_XB PC_PC+4,STROBE.V.BUS                ;U-CODE SUBTEST #1
                   04
                00B2 10389 2$:     .BYTE   ^X4                                     ;MICRO VECTOR LINES
                00B3 10390         VBUSDATA        <^X08>,0
```

```
                                00B4 10391          VBUSDATA        <^X09>,0
                                00B5 10392          VBUSDATA        <^X0A>,1
                                00B6 10393          VBUSDATA        <^X0B>,1
                          0023  00B7 10394 3$:      .WORD   ^X0023                          ;MSRC ACV MICROTRAP
                          3FFF  00B9 10395 4$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                                00BB 10396 5$:
U 06, 7600,C800,0364,1700,0470,1807 00BB 10397 ;* 06   BUT/IRD1TST,STROBE.V.BUS            ;U-CODE SUBTEST #2
                            04  00C6 10401 6$:      .BYTE   ^X4                             ;MICRO VECTOR LINES
                                00C7 10402          VBUSDATA        <^X08>,1
                                00C8 10403          VBUSDATA        <^X09>,1
                                00C9 10404          VBUSDATA        <^X0A>,0
                                00CA 10405          VBUSDATA        <^X0B>,1
                          002D  00CB 10406 7$:      .WORD   ^X002D                          ;IRD1TST ACV MICROTRAP
                                00CD 10407
                                00CD 10408
                                00CD 10409
                                00CD 10410 BEGIN_CPU_DATA
                                0002 10411
                                0002 10412 DCS_DATA
                                0001 10413
U 00, 7700,C800,0364,0300,0470,1801 0001 10414 ;* 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C 10418 ;* 01   MEMSCAR_R[TEMP0]                    ;SET MME OFF
U 02, 7700,8800,5BE4,0304,6070,1803 0017 10422 ;* 02   MEMSCR_R[TEMP1]
U 03, 7700,C800,5BE4,03D8,4A70,1804 0022 10426 ;* 03   VA_R[ZERO]                         ;ZEROS TO VA REGISTER
U 04, 7700,881B,5BE4,0308,5070,1805 002D 10430 ;* 04   TB_R[TEMP2]                       ;PUT ACCESS VIOLATION IN TB
U 05, 7700,C800,5BE4,03D8,4870,1806 0038 10434 ;* 05   PC_R[ZERO]                        ;FORCE PREFETCH
U 06, 7601,0857,5924,020E,0470,1807 0043 10438 ;* 06   R[TEMP3]_XB PC_PC+4,STROBE.V.BUS  ;CAUSE MICROTRAP
U 07, 7300,C800,0364,0300,0470,1808 004E 10442 ;* 07   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
                                0059 10446
                                0059 10447 RTEMP_DATA
                                0001 10448
                    00000000    0001 10449          .LONG   ^X00000000                  ;PHY/VIR REG ADDRESS
                    01000000    0005 10450          .LONG   ^X01000000                  ;SET MME ON
                    00000108    0009 10451          .LONG   ^X00000108                  ;ACCESS VIOLATION FOR TB
                                000D 10452
                                000D 10453 CACHE_DATA
                                0001 10454
                    00000000    0001 10455          .LONG   ^X00000000                  ;VALIDATE CACHE 0 & 4
                    00000000    0005 10456          .LONG   ^X00000000
                                0009 10457
                                0009 10458 END_CPU_DATA
                                00CD 10459
                                00CD 10460
                                00CD 10461 END_TEST_DATA
                                00CD
                                00CD        ;-----------------------------------------------------------------------------
                                00CD 10462
                                00CD 10463 ENDTEST
```

```
0034        .SBTTL  "TEST   05D     TEST XB1 FOR MICROTRAP WITH TB ACCESS VIOALTION
0034 10465 ;*******************************************************************************
0034 10466 ;++
0034 10467 ;
0034 10468 ; FUNCTIONAL DESCRIPTION:
0034 10469 ;
0034 10470 ;       THIS TEST CHECKS THAT CPU CAN DETECT A TB ACCESS VIOLATION DURING
0034 10471 ;       INSTRUCTION PREFETCH AND REPORT SAME VIA A MICROTRAP.
0034 10472 ;
0034 10473 ;
0034 10474 ; TEST ALGORITHM:
0034 10475 ;
0034 10476 ;       1.  MODIFY DCS MICROCODE FOR SPECIFIC SUBTEST
0034 10477 ;       2.  EXECUTE DCS PROGRAM
0034 10478 ;               a.  TURN MEMORY MANAGEMENT ON
0034 10479 ;               b.  PUT 1FC INTO VA REGISTER
0034 10480 ;               c.  VALIDATE THIS LOCATION IN THE CACHE AND TB
0034 10481 ;               d.  INCREMENT VA TO 200
0034 10482 ;               e.  INVALIDATE THE TB AT THIS LOCATION
0034 10483 ;               f.  VALIDATE THE CACHE
0034 10484 ;               g.  LOAD THE PC WITH 1FC TO FORCE PREFETCH
0034 10485 ;               h.  SOURCE XB0
0034 10486 ;               i.  SOURCE XB1; THIS SHOULD MICROTRAP
0034 10487 ;       3.  TEST THE VBUS FOR THE CORRECT MICRO VECTOR LINES
0034 10488 ;       4.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT TRAP
0034 10489 ;       5.  IF NO ERROR GO TO NEXT TEST
0034 10490 ;
0034 10491 ;
0034 10492 ; TEST PATTERNS:
0034 10493 ;
0034 10494 ;       NONE
0034 10495 ;
0034 10496 ;
0034 10497 ; LOGIC DESCRIPTION:
0034 10498 ;
0034 10499 ;       THE UTR GATE ARRAY SHOULD DETECT THE FACT THAT A TB ACCESS VIOLATION
0034 10500 ;       HAS OCCURRED WHEN XB1 IS PREFETCHING INSTRUCTIONS.  IT SHOULD REPORT
0034 10501 ;       THIS VIA A MICROTRAP WHEN XB1 IS SOURCED.
0034 10502 ;
0034 10503 ;
0034 10504 ; ERROR DESCRIPTION:
0034 10505 ;
0034 10506 ;       NOTE:  THERE ARE TWO IFERROR PSEUDO OPS IN EACH SUBTEST.  SO YOU WILL
0034 10507 ;       HAVE TO CHECK THE FAILING PC TO DETERMINE THE CORRECT PRINTOUT.
0034 10508 ;
0034 10509 ;       FIRST IFERROR:
0034 10510 ;       EXPECTED VBUS DATA (MICRO VECTOR LINES)
0034 10511 ;       RECEIVED VBUS DATA
0034 10512 ;               (BITS <7:0> = VBUS BIT POSITION)
0034 10513 ;               (BITS <8> = BIT VALUE)
0034 10514 ;
0034 10515 ;       SECOND IFERROR:
0034 10516 ;       EXPECTED CS ADDRESS
0034 10517 ;       RECEIVED CS ADDRESS
0034 10518 ;
```

```
                     0034 10519 ;--
                     0034 10520 ;*******************************************************************************
                     0034 10521 ;///////////////////////////////////////////////////////////////////////////////
                     0034 10522 ;+
                     0034 10523 ;SUBTEST #1
                     0034 10524 ;
                     0034 10525 ;       TEST ACCESS VIOLATION USING MSRC XB1
                     0034 10526 ;-
                     0034 10527         SUBTEST                                         ;SUBTEST #1
                     0037
                     0037       ;///////////////////////////////////////////////////////////////////////////////
                     0037 10528         INITIALIZE                                      ;INIT THE CPU
                     0039 10529         EXPUTRAP                                        ;EXPECT A MICROTRAP
                     003B 10530         LOADDCS         1$,,<^X0D>,1                    ;MODIFY U-CODE FOR THIS SUBTEST
                     0042 10531         ERRLOOP                                         ;ERROR LOOP POINT
                     0044 10532         FETCH           0                               ;START DCS PROGRAM
                     0049 10533         BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                     004D 10534         CMPVBUS         2$                              ;TEST MICRO VECTOR LINES
                     0052 10535         IFERROR         ,<<UTR>>,                       ;INCORRECT VECTOR
                     0058 10536         CMPREGMSK       CSTRP,3$,,4$,,W,                ;TEST CS ADDRESS
                     0064 10537         IFERROR         ,,<<DPM>>                       ;WRONG CS ADDRESS
                     006A 10538
                     006A 10539
                     006A 10540 ;///////////////////////////////////////////////////////////////////////////////
                     006A 10541 ;+
                     006A 10542 ;SUBTEST #2
                     006A 10543 ;
                     006A 10544 ;       TEST FOR MICROTRAP USING IRD1TST ON XB1
                     006A 10545 ;-
                     006A 10546         SUBTEST                                         ;SUBTEST #2
                     006D
                     006D       ;///////////////////////////////////////////////////////////////////////////////
                     006D 10547         INITIALIZE                                      ;INIT THE CPU
                     006F 10548         EXPUTRAP                                        ;EXPECT A MICRO TRAP
                     0071 10549         LOADDCS         5$,,<^X0D>,1                    ;MODIFY U-CODE FOR THIS SUBTEST
                     0078 10550         ERRLOOP                                         ;ERROR LOOP POINT
                     007A 10551         FETCH           0                               ;START DCS PROGRAM
                     007F 10552         BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                     0083 10553         CMPVBUS         6$                              ;TEST MICRO VECTOR LINES
                     0088 10554         IFERROR         ,<<UTR>>,                       ;INCORRECT VECTOR
                     008E 10555         CMPREGMSK       CSTRP,7$,,4$,,W,                ;TEST CS ADDRESS
                     009A 10556         IFERROR         ,,<<DPM>>                       ;INCORRECT ADDRESS
                     00A0 10557         SKIP                                            ;GO TO NEXT TEST
                     00A7 10558
                     00A7 10559
                     00A7 10560
                     00A7 10561 BEGIN_TEST_DATA
                     00A7
                     00A7       ;-------------------------------------------------------------------------
                     00A7 10562
                     00A7 10563 1$:
U 0D, 7601,0857,5924,0216,0470,180E
                     00A7 10564 ;% 0D    R[TEMP5]_XB PC_PC+4,STROBE.V.BUS                ;U-CODE SUBTEST #1
                  04 00B2 10568 2$:      .BYTE   ^X4                                     ;MICRO VECTOR LINES
                     00B3 10569         VBUSDATA        <^X08>,0
                     00B4 10570         VBUSDATA        <^X09>,0
```

```
                                00B5 10571          VBUSDATA        <^X0A>,1
                                00B6 10572          VBUSDATA        <^X0B>,1
                          0023  00B7 10573 3$:       .WORD   ^X0023                  ;MSRC ACV MICROTRAP
                          3FFF  00B9 10574 4$:       .WORD   ^X3FFF                  ;MASK FOR CMPRGEMSK PSEUDO OP
                                00BB 10575 5$:
U 0D, 7600,C800,0364,1700,0470,180E  00BB 10576 ;% 0D   BUT/IRD1TST,STROBE.V.BUS    ;U-CODE SUBTEST #2
                             04 00C6 10580 6$:      .BYTE   ^X4                     ;MICRO VECTOR LINES
                                00C7 10581          VBUSDATA        <^X08>,1
                                00C8 10582          VBUSDATA        <^X09>,1
                                00C9 10583          VBUSDATA        <^X0A>,0
                                00CA 10584          VBUSDATA        <^X0B>,1
                          002D  00CB 10585 7$:      .WORD   ^X002D                  ;IRD1TST ACV MICROTRAP
                                00CD 10586
                                00CD 10587
                                00CD 10588 BEGIN_CPU_DATA
                                0002 10589
                                0002 10590 DCS_DATA
                                0001 10591
U 00, 7700,C800,0364,0300,0470,1801  0001 10592 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C 10596 ;% 01   MEMSCAR_R[TEMP0]            ;SET MME ON
U 02, 7700,8800,5BE4,0304,6070,1803  0017 10600 ;% 02   MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,4A70,1804  0022 10604 ;% 03   VA_R[TEMP2]                 ;1FC TO VA REGISTER
U 04, 7700,C81B,5BE4,030C,5070,1805  002D 10608 ;% 04   TB_R[TEMP3]                 ;VALIDATE THE TB
U 05, 7700,C800,5BE4,03D8,5C80,1806  0038 10612 ;% 05   WRITE.PHY R[ZERO]          ;VALIDATE THE CACHE
U 06, 7700,C800,5BE4,03D8,5470,1807  0043 10616 ;% 06   WDR_UNROT(R[ZERO])        ;HOLD DATA FOR EXTRA CYCLE
U 07, 7700,4800,0364,0300,4470,1808  004E 10620 ;% 07   VA_VA+4                    ;INCREMENT VA TO 200
U 08, 7700,881B,5BE4,0310,5070,1809  0059 10624 ;% 08   TB_R[TEMP4]                 ;ACCESS VIOLATION TO TB
U 09, 7700,C800,5BE4,03D8,5C80,180A  0064 10628 ;% 09   WRITE.PHY R[ZERO]          ;VALIDATE THE CACHE
U 0A, 7700,C800,5BE4,03D8,5470,180B  006F 10632 ;% 0A   WDR_UNROT(R[ZERO])        ;HOLD DATA FOR EXTRA CYCLE
U 0B, 7700,8800,5BE4,0308,4870,180C  007A 10636 ;% 0B   PC_R[TEMP2]               ;FORCE PREFETCH AT 1FC & 200
U 0C, 7701,0857,5924,0216,0470,180D  0085 10640 ;% 0C   R[TEMP5]_XB PC_PC+4       ;MSRC XB0
U 0D, 7601,0857,5924,0216,0470,180E  0090 10644 ;% 0D   R[TEMP5]_XB PC_PC+4,STROBE.V.BUS  ;SHOULD U-TRAP
U 0E, 7300,4800,0364,0300,0470,180F  009B 10648 ;% 0E   NOP,STOP.CLOCK             ;STOP THE CPU CLOCK
                                00A6 10652
                                00A6 10653 RTEMP_DATA
                                0001 10654
                    00000000    0001 10655          .LONG   ^X00000000          ;PHY/VIR REGISTER ADDRESS
                    01000000    0005 10656          .LONG   ^X01000000          ;SET MME ON
                    000001FC    0009 10657          .LONG   ^X000001FC          ;ADDRESS FOR VA AND PC
                    00000148    000D 10658          .LONG   ^X00000148          ;VALIDATE TB
                    00000108    0011 10659          .LONG   ^X00000108          ;ACCESS VIOLATION
                                0015 10660
                                0015 10661 END_CPU_DATA
                                00CD 10662
                                00CD 10663
                                00CD 10664 END_TEST_DATA
                                00CD
                                00CD        ;-------------------------------------------------------------------------
                                00CD 10665 ENDTEST
```

```
                        .SBTTL  "TEST   05E     TEST D CLOCK INHIBIT ON MICROTRAP
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST WILL CHECK THAT D CLOCK IS INHIBITED WHEN A MICRO TRAP
                ;       OCCURS.
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  EXECUTE DCS PROGRAM
                ;               a.  ZERO RTEMP3
                ;               b.  CAUSE PREFETCH BY ZEROING THE PC REGISTER
                ;               c.  ENABLE MEMORY MANAGEMENT
                ;               d.  ZERO THE VA REGISTER
                ;               e.  MAKE MME LOCATION ZERO A MISS
                ;               f.  RELOAD PC REGISTER TO CAUSE A SECOND PREFETCH
                ;               g.  TRY TO WRITE THE XB INTO RTEMP3
                ;       2.  TEST THE VBUS FOR GEN DEST INH
                ;       3.  IF NO ERROR EXECUTE DCS PROGRAM
                ;               a.  READ RTEMP3 TO THE RDM
                ;       4.  TEST RTEMP3 FOR ZEROS
                ;       5.  IF NO ERROR GO TO NEXT TEST
                ;
                ; TEST PATTERNS:
                ;
                ;       NONE
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       THIS TEST VERIFIES THAT D CLOCK IS BLOCKED WHEN A MICROTRAP TAKES
                ;       PLACE.  THE UTR GATE ARRAY ON MIC SIGNALS THE SAC GATE ARRAY ON
                ;       DPM THAT A MICRO TRAP HAS OCCURRED VIA THE GEN DEST INH SIGNAL.
                ;
                ;
                ; ASSUMPTIONS:
                ;
                ;       THIS TEST ASSUMES THAT THE XB MICRO TRAP TESTS HAVE BEEN RUN.
                ;
                ;
                ; ERROR DESCRIPTION:
                ;
                ;       FIRST IFERROR:
                ;               EXPECTED VBUS BITS
                ;               RECEIVED VBUS BITS
                ;                       (BITS <7:0> = VBUS BIT POSITION)
                ;                       (BITS <8> = BIT STATE)
                ;
                ;       SECOND IFERROR:
                ;               EXPECTED RTEMP3
                ;               RECEIVED RTEMP3
```

```
                        0026 10721 ;
                        0026 10722 ;--
                        0026 10723 ;*****************************************************************************
                        0026 10724
                        0026 10725
                        0026 10726         INITIALIZE                                  ;INIT THE CPU
                        0028 10727         EXPUTRAP                                    ;EXPECT A U-TRAP
                        002A 10728         ERRLOOP                                     ;ERROR LOOP POINT
                        002C 10729         FETCH           0                           ;START DCS PROGRAM
                        0031 10730         BRSTCLK         9                           ;END DCS PROGRAM
                        0035 10731         CMPVBUS         1$                          ;TEST GEN DEST INH ON VBUS
                        003A 10732         IFERROR         ,<UTR>,                     ;GEN DEST INH MISSING
                        0040 10733         FETCH           <^X0A>                      ;START DCS PROGRAM
                        0045 10734         BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                        0049 10735         CMPREG          WHREG,2$,,L,                ;TEST FOR RTEMP3 DATA
                        0052 10736         IFERROR         ,<SAC>,<DPM>                ;D CLOCK NOT INHIBITED
                        0059 10737         SKIP                                        ;GO TO NEXT TEST
                        0060 10738
                        0060 10739
                        0060 10740 BEGIN_TEST_DATA
                        0060
                        0060               ;-------------------------------------------------------------------
                        0060 10741
                     01 0060 10742 1$:     .BYTE   ^X01
                        0061 10743         VBUSDATA        <^X0C>,0                    ;GEN DEST INH ON VBUS
               00000000 0062 10744 2$:     .LONG   ^X00000000                          ;EXPECTED CONTENTS OF RTEMP3
                        0066 10745
                        0066 10746
                        0066 10747
                        0066 10748 BEGIN_CPU_DATA
                        0002 10749
                        0002 10750 DCS_DATA
                        0001 10751
U 00, 7700,C800,0364,0300,0470,1801     0001 10752 ;% 00    NOP
U 01, 7700,C840,5B70,030C,0470,1802     000C 10756 ;% 01    R[TEMP3]_0                  ;ZERO RTEMP3
U 02, 7700,C800,5BE4,03D8,4870,1803     0017 10760 ;% 02    PC_R[ZERO]                  ;CAUSE XB'S TO PREFETCH
U 03, 7700,C800,5BE4,0300,6870,1804     0022 10764 ;% 03    MEMSCAR_R[TEMP0]            ;TURN MME ON
U 04, 7700,8800,5BE4,0304,6070,1805     002D 10768 ;% 04    MEMSCR_R[TEMP1]             ;...
U 05, 7700,4800,5BE4,03D8,4A70,1806     0038 10772 ;% 05    VA_R[ZERO]                  ;ZERO THE VA REGISTER
U 06, 7700,081B,5BE4,0308,5070,1807     0043 10776 ;% 06    TB_R[TEMP2]                 ;MAKE TB A MISS
U 07, 7700,4800,5BE4,03D8,4870,1808     004E 10780 ;% 07    PC_R[ZERO]                  ;FORCE PREFETCH
U 08, 7601,8857,5924,020E,0470,1809     0059 10784 ;% 08    R[TEMP3]_XB PC_PC+4,STROBE.V.BUS ;CAUSE U-TRAP
U 09, 7300,4800,0364,0300,0470,180A     0064 10788 ;% 09    NOP,STOP.CLOCK              ;STOP THE CLOCK
U 0A, 7700,C800,0364,0300,0470,180B     006F 10792 ;% 0A    NOP
U 0B, 6700,4800,5BE4,030C,0470,180C     007A 10796 ;% 0B    RDM_WB,WB_R[TEMP3]          ;READ RTEMP3 TO RDM
U 0C, 7300,C800,0364,0300,0470,180D     0085 10800 ;% 0C    NOP,STOP.CLOCK              ;STOP THE CPU CLOCK
                        0090 10804
                        0090 10805 RTEMP_DATA
                        0001 10806
                  00000000 0001 10807         .LONG   ^X00000000                      ;PHY/VIR REGISTER ADDRESS
                  01000000 0005 10808         .LONG   ^X01000000                      ;TURN MME ON
                  00000048 0009 10809         .LONG   ^X00000048                      ;TB MISS
                  00000000 000D 10810         .LONG   ^X00000000                      ;RTEMP3
                        0011 10811
                        0011 10812 CACHE_DATA
```

```
                     0001 10813
FFFFFFFF             0001 10814          .LONG   ^XFFFFFFFF                      ;VALIDATE CACHE LOCATION 0
FFFFFFFF             0005 10815          .LONG   ^XFFFFFFFF                      ;AND 4
                     0009 10816
                     0009 10817 END_CPU_DATA
                     0066 10818
                     0066 10819
                     0066 10820 END_TEST_DATA
                     0066
                     0066        ;-----------------------------------------------------------------------
                     0066 10821 ENDTEST
```

```
                0000 10823 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN             = 00000000
$TEST_NUMBER     = 0000005F
ACV              = 0000000F
AC_LOW           = 00000080
ADD              = 00000019
ADDR_MATCH_HI    = 00007405
ADDR_MATCH_LO    = 00007404
ADK              = 0000000B
ALK              = 00000000
ALP              = 00000016
ARG_CNT          = 00000053 R      ED
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_HEAD       = 00000000 R      F0
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
CML              = 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               = 0000000E
D1               = 00000005
D2               = 00000000
DATA_FLAG        = 00000000
DATA_LENGTH      = 00000002
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      EE
DCS_LENGTH       = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START        = 00001800
DC_LOW           = 00000040
DIAGNOSE_FLAG    = 00000004
DPM              = 00000000
DUM1             = FFFFFFFE
ENDTEST_LOC      = 0000006B R      ED
END_SLICE        = 00000025
END_TEST_49        0000006C R      02
END_TEST_50        00000130 R      09
END_TEST_51        00000130 R      0F
END_TEST_52        00000113 R      15
END_TEST_53        00000113 R      1B
END_TEST_54        00000120 R      21
END_TEST_55        00000120 R      27
END_TEST_56        00000113 R      2D
END_TEST_57        00000113 R      33
END_TEST_58        00000171 R      39
END_TEST_59        00000171 R      3B
END_TEST_60        000001B5 R      3D
END_TEST_61        000001B5 R      3F
END_TEST_62        000001D9 R      41
END_TEST_63        000001D9 R      43
END_TEST_64        0000028A R      45
END_TEST_65        0000028A R      4B
END_TEST_66        000002F1 R      51
END_TEST_67        000002F1 R      57
END_TEST_68        000002B0 R      5D
END_TEST_69        000002B0 R      63
END_TEST_70        000001B3 R      69
END_TEST_71        000000A3 R      6F
END_TEST_72        000000E6 R      71
END_TEST_73        000000EA R      77
END_TEST_74        000000B4 R      7D
END_TEST_75        000000B8 R      83
END_TEST_76        000000B8 R      89
END_TEST_77        000000B5 R      8F
END_TEST_78        0000008F R      95
END_TEST_79        000001BD R      9B
END_TEST_80        00000147 R      A1
END_TEST_81        00000430 R      A7
END_TEST_82        00000462 R      A9
END_TEST_83        00000226 R      AB
END_TEST_84        00000093 R      B1
END_TEST_85        00000093 R      B7
END_TEST_86        00000108 R      BD
END_TEST_87        00000108 R      C3
END_TEST_88        00000098 R      C9
END_TEST_89        00000098 R      CF
END_TEST_90        000000C2 R      D5
END_TEST_91        000000C2 R      DB
END_TEST_92        000000CD R      E1
END_TEST_93        000000CD R      E7
END_TEST_94        00000066 R      ED
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          MAX_ARGUMENTS   = 00000010          OP_PATTERN_GEN = 00000010
FRONT2            = 00007421          MA_KEY          = 00000038          OP_PRINT_N     = 00000036
FRONT_PNL_1       = 00007420          MC0             = 00000003          OP_PRINT_S     = 00000034
FRONT_PNL_2       = 00007421          MC1             = 00000009          OP_SAVE_INDEX  = 0000002C
FRONT_PNL_LOCK    = 00000080          MC2             = 0000000B          OP_SKIP        = 0000001C
HALT_FLAG         = 00000001          MDL             = 0000001B          OP_SUB_TEST    = 0000001E
HALT_KEY          = 00000008          MDR             = 00000018          OVERLAY_HEAD   = 00000000 R     F2
HALT_ON_MATCH     = 00000001          MDREG           = 00007414          PARITY_ERROR   = 00000020
HEAD              = FFFF7C00          MD_KEY          = 00000034          PAR_CHK_ENABL  = 00000008
HIGH              = 00000001          MD_REG_BYTE_0   = 00007414          PAR_STOP_ENABL = 00000010
H_FROM_WBUS       = 00000010          MD_REG_BYTE_1   = 00007415          PASS_KEY       = 00000002
IB_FLAG           = 00000020          MD_REG_BYTE_2   = 00007416          PB_INIT        = 00000040
IB_KEY            = 00000012          MD_REG_BYTE_3   = 00007417          PB_KEY         = 00000036
INSTR_FLAG        = 00000004          MEC             = 0000001A          PC_KEY         = 00000030
INSTR_KEY         = 00000018          MHREG           = 00007418          PHB            = 00000007
INT               = 00000010          MH_REG_BYTE_0   = 00007418          PRK            = 0000000C
IRD               = 00000005          MH_REG_BYTE_1   = 00007419          PS_KEY         = 0000003E
I_INDEX_FLAG      = 00000000          MH_REG_BYTE_2   = 0000741A          QA_FLAG        = 00000040
I_INIT            = 00000000          MH_REG_BYTE_3   = 0000741B          QA_KEY         = 00000014
J_INDEX_FLAG      = 00000000          MIC             = 00000001          QV_FLAG        = 00000002
J_INIT            = 00000000          MICROWORD       = 0000000A          QV_KEY         = 00000028
K_INDEX_FLAG      = 00000000          MICRO_ADDR_INH  = 00000080          RDCTRL         = 0000741E
K_INIT            = 00000000          MONITOR_SIZE    = 00002C5E          RDM            = 00000004
L                 = 00000004          MSQ             = 00000008          RDM_FLAG       = 00000004
LIST_END          = 00000059 R    ED  MT_KEY          = 0000002E          RDM_KEY        = 0000002A
LIST_HEAD         = 00000054 R    ED  N               = 00000001          RDM_LAST       = 00000008
LIST_LENGTH       = 00000005          NER_FLAG        = 00000004          RD_CTRL_REG    = 0000741E
LITERAL           = 00000000          NER_KEY         = 0000000C          REMOTE         = 00000001
LONG              = 00000004          NOERROR         = 00000001          RTEMP_FLAG     = 00000000
LOOP_ERROR_FLAG   = 00000020          NOTEQUAL        = 00000003          RTEMP_FLAG1    = 00000000
LOOP_FLAG         = 00000002          ONEQUAL         = 00000002          RTEMP_HEAD     = 00000000 R     EF
LOOP_KEY          = 0000000A          ONERROR         = 00000000          RTEMP_LENGTH   = 00000001
LOST_FLAG         = 00000010          OP_BEGIN_LOOP   = 00000008          RT_KEY         = 0000002C
LOST_KEY          = 0000001A          OP_BURST_CLOCK  = 00000002          SAC            = 00000009
LOW               = 00000000          OP_COMPARE_REG  = 00000004          SA_CLOCK       = 00000080
M                 = 0000000A          OP_COMPARE_REGM = 00000006          SA_FLAG        = 00000010
M$TEST_SIZE       = 000007A2          OP_COMPARE_VB   = 00000020          SA_KEY         = 00000010
M$TEST_SIZE_D     = 000003E2          OP_DUMPLOG      = 00000030          SA_START_STOP  = 00000080
MA0               = 00000008          OP_ENABL_STALL  = 00000038          SBE_FLAG       = 00000020
MA1               = 0000000A          OP_ENABL_TRAP   = 0000002E          SBE_KEY        = 00000042
MA2               = 0000000C          OP_END_FILE     = 00000024          SECTION_FLAG   = 00000080
MAD               = 00000013          OP_END_LOOP     = 0000000A          SF_KEY         = 00000040
MAIN32            = 0000741D          OP_END_TEST     = 0000000C          SOMM_FLAG      = 00000001
MAINT_A_TO_CMI    = 00000080          OP_ERRLOG       = 00000032          SOMM_KEY       = 00000006
MAINT_CMI_TO_MH   = 00000020          OP_ERROR_LOOP   = 0000000E          SPA            = 00000002
MAINT_DCS_ENABL   = 00000004          OP_FETCH        = 00000022          SRK            = 00000001
MAINT_MD_TO_CMI   = 00000040          OP_IF_ERROR     = 00000012          SRM            = 00000017
MAINT_REG         = 0000741D          OP_INC_INDEX    = 0000002A          SR_KEY         = 0000003C
MAINT_STB_INH     = 00000002          OP_INIT         = 00000014          STACK_SIZE     = 00000400
MAINT_TRAP_OFF    = 00000001          OP_LOAD_DCS     = 00000000          STATUS         = 00007406
MAINT_WB_TO_WH    = 00000008          OP_LOAD_FIELD   = 00000026          STATUS_REG     = 00007406
MAINT_WD_TO_WB    = 00000010          OP_LOAD_INDEX   = 00000028          STEP_KEY       = 0000001E
MAP               = 00000012          OP_LOAD_REG     = 00000016          STOP_CLOCK     = 00000004
MAREG             = 00007410          OP_MASK         = 00000018          STROBE_CMI     = 00000040
MASTER_HALT_EN    = 00000008          OP_NEW_TEST     = 0000001A          ST_KEY         = 0000001E
```

```
TEMP            = 00000002 R     EC
TEMP1           = 00000054
TEMP2           = 000000AA
TEMP3           = 00000002
TEMP4           = 00000002 R     EC
TEST            = 00000004
TEST_FLAG       = 00000000
TEST_HEAD       = 00000023 R     ED
TEST_KEY        = 00000000
TEST_LENGTH     = 00000001
TEST_SECT_HEAD  = 00000000 R     ED
THIS_TEST       = 0000005E
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 synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| X_0049_D | 00000002 ( 2.) | 01 ( 1.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0049_T | 00000080 ( 128.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0049_GDCS | 00000182 ( 386.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0049_ERTEMP | 00000001 ( 1.) | 04 ( 4.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0049_FCACHE | 00000001 ( 1.) | 05 ( 5.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0049_HFILL | 0000007A ( 122.) | 06 ( 6.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| DIRECTORY | 0000002E ( 46.) | 07 ( 7.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0050_D | 00000002 ( 2.) | 08 ( 8.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0050_T | 00000180 ( 384.) | 09 ( 9.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0050_GDCS | 00000090 ( 144.) | 0A ( 10.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0050_ERTEMP | 00000001 ( 1.) | 0B ( 11.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0050_FCACHE | 00000001 ( 1.) | 0C ( 12.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0050_HFILL | 0000006C ( 108.) | 0D ( 13.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0051_D | 00000002 ( 2.) | 0E ( 14.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0051_T | 00000180 ( 384.) | 0F ( 15.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0051_GDCS | 00000090 ( 144.) | 10 ( 16.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0051_ERTEMP | 00000001 ( 1.) | 11 ( 17.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0051_FCACHE | 00000001 ( 1.) | 12 ( 18.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0051_HFILL | 0000006C ( 108.) | 13 ( 19.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0052_D | 00000002 ( 2.) | 14 ( 20.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0052_T | 00000180 ( 384.) | 15 ( 21.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0052_GDCS | 0000009B ( 155.) | 16 ( 22.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0052_ERTEMP | 00000001 ( 1.) | 17 ( 23.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0052_FCACHE | 00000001 ( 1.) | 18 ( 24.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0052_HFILL | 00000061 ( 97.) | 19 ( 25.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0053_D | 00000002 ( 2.) | 1A ( 26.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0053_T | 00000180 ( 384.) | 1B ( 27.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0053_GDCS | 0000009B ( 155.) | 1C ( 28.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0053_ERTEMP | 00000001 ( 1.) | 1D ( 29.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0053_FCACHE | 00000001 ( 1.) | 1E ( 30.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0053_HFILL | 00000061 ( 97.) | 1F ( 31.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0054_D | 00000002 ( 2.) | 20 ( 32.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0054_T | 00000180 ( 384.) | 21 ( 33.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0054_GDCS | 00000090 ( 144.) | 22 ( 34.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0054_ERTEMP | 00000001 ( 1.) | 23 ( 35.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0054_FCACHE | 00000001 ( 1.) | 24 ( 36.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0054_HFILL | 0000006C ( 108.) | 25 ( 37.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0055_D | 00000002 ( 2.) | 26 ( 38.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0055_T | 00000180 ( 384.) | 27 ( 39.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0055_GDCS | 00000090 ( 144.) | 28 ( 40.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0055_ERTEMP | 00000001 ( 1.) | 29 ( 41.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0055_FCACHE | 00000001 ( 1.) | 2A ( 42.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0055_HFILL | 0000006C ( 108.) | 2B ( 43.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0056_D | 00000002 ( 2.) | 2C ( 44.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0056_T | 00000180 ( 384.) | 2D ( 45.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0056_GDCS | 0000009B ( 155.) | 2E ( 46.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0056_ERTEMP | 00000001 ( 1.) | 2F ( 47.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0056_FCACHE | 00000001 | ( 1.) | 30 | ( 48.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0056_HFILL | 00000061 | ( 97.) | 31 | ( 49.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0057_D | 00000002 | ( 2.) | 32 | ( 50.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0057_T | 00000180 | ( 384.) | 33 | ( 51.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0057_GDCS | 0000009B | ( 155.) | 34 | ( 52.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0057_ERTEMP | 00000001 | ( 1.) | 35 | ( 53.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0057_FCACHE | 00000001 | ( 1.) | 36 | ( 54.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0057_HFILL | 00000061 | ( 97.) | 37 | ( 55.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0058_D | 00000002 | ( 2.) | 38 | ( 56.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0058_T | 0000017E | ( 382.) | 39 | ( 57.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0059_D | 00000002 | ( 2.) | 3A | ( 58.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0059_T | 0000017E | ( 382.) | 3B | ( 59.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0060_D | 00000002 | ( 2.) | 3C | ( 60.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0060_T | 000001FE | ( 510.) | 3D | ( 61.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0061_D | 00000002 | ( 2.) | 3E | ( 62.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0061_T | 000001FE | ( 510.) | 3F | ( 63.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0062_D | 00000002 | ( 2.) | 40 | ( 64.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0062_T | 000001FE | ( 510.) | 41 | ( 65.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0063_D | 00000002 | ( 2.) | 42 | ( 66.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0063_T | 000001FE | ( 510.) | 43 | ( 67.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0064_D | 00000002 | ( 2.) | 44 | ( 68.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0064_T | 00000300 | ( 768.) | 45 | ( 69.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0064_ERTEMP | 00000011 | ( 17.) | 46 | ( 70.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0064_FCACHE | 00000001 | ( 1.) | 47 | ( 71.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0064_GDCS | 00000001 | ( 1.) | 48 | ( 72.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0064_HFILL | 0000006B | ( 107.) | 49 | ( 73.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0065_D | 00000002 | ( 2.) | 4A | ( 74.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0065_T | 00000300 | ( 768.) | 4B | ( 75.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0065_ERTEMP | 00000011 | ( 17.) | 4C | ( 76.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0065_FCACHE | 00000001 | ( 1.) | 4D | ( 77.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0065_GDCS | 00000001 | ( 1.) | 4E | ( 78.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0065_HFILL | 0000006B | ( 107.) | 4F | ( 79.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0066_D | 00000002 | ( 2.) | 50 | ( 80.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0066_T | 00000300 | ( 768.) | 51 | ( 81.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0066_ERTEMP | 0000001D | ( 29.) | 52 | ( 82.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0066_FCACHE | 00000001 | ( 1.) | 53 | ( 83.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0066_GDCS | 00000001 | ( 1.) | 54 | ( 84.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0066_HFILL | 0000005F | ( 95.) | 55 | ( 85.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0067_D | 00000002 | ( 2.) | 56 | ( 86.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0067_T | 00000300 | ( 768.) | 57 | ( 87.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0067_ERTEMP | 0000001D | ( 29.) | 58 | ( 88.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0067_FCACHE | 00000001 | ( 1.) | 59 | ( 89.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0067_GDCS | 00000001 | ( 1.) | 5A | ( 90.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0067_HFILL | 0000005F | ( 95.) | 5B | ( 91.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0068_D | 00000002 | ( 2.) | 5C | ( 92.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0068_T | 00000300 | ( 768.) | 5D | ( 93.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0068_ERTEMP | 0000001D | ( 29.) | 5E | ( 94.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0068_FCACHE | 00000001 | ( 1.) | 5F | ( 95.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0068_GDCS | 00000001 | ( 1.) | 60 | ( 96.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0068_HFILL | 0000005F | ( 95.) | 61 | ( 97.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0069_D | 00000002 | ( 2.) | 62 | ( 98.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0069_T | 00000300 | ( 768.) | 63 | ( 99.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0069_ERTEMP | 0000001D | ( 29.) | 64 | (100.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0069_FCACHE | 00000001 | ( 1.) | 65 | (101.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0069_GDCS | 00000001 | ( 1.) | 66 | (102.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0069_HFILL | 0000005F | ( 95.) | 67 (103.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0070_D | 00000002 | ( 2.) | 68 (104.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0070_T | 00000200 | ( 512.) | 69 (105.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0070_ERTEMP | 00000009 | ( 9.) | 6A (106.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0070_FCACHE | 00000001 | ( 1.) | 6B (107.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0070_GDCS | 00000001 | ( 1.) | 6C (108.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0070_HFILL | 00000073 | ( 115.) | 6D (109.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0071_D | 00000002 | ( 2.) | 6E (110.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0071_T | 000000FE | ( 254.) | 6F (111.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0072_D | 00000002 | ( 2.) | 70 (112.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0072_T | 00000100 | ( 256.) | 71 (113.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0072_GDCS | 00000114 | ( 276.) | 72 (114.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0072_ERTEMP | 0000001D | ( 29.) | 73 (115.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0072_FCACHE | 00000001 | ( 1.) | 74 (116.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0072_HFILL | 0000004C | ( 76.) | 75 (117.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0073_D | 00000002 | ( 2.) | 76 (118.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0073_T | 00000100 | ( 256.) | 77 (119.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0073_GDCS | 00000114 | ( 276.) | 78 (120.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0073_ERTEMP | 0000001D | ( 29.) | 79 (121.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0073_FCACHE | 00000001 | ( 1.) | 7A (122.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0073_HFILL | 0000004C | ( 76.) | 7B (123.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0074_D | 00000002 | ( 2.) | 7C (124.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0074_T | 00000100 | ( 256.) | 7D (125.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0074_GDCS | 000000BC | ( 188.) | 7E (126.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0074_ERTEMP | 00000019 | ( 25.) | 7F (127.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0074_FCACHE | 00000001 | ( 1.) | 80 (128.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0074_HFILL | 00000028 | ( 40.) | 81 (129.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0075_D | 00000002 | ( 2.) | 82 (130.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0075_T | 00000100 | ( 256.) | 83 (131.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0075_GDCS | 000000F3 | ( 243.) | 84 (132.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0075_ERTEMP | 00000019 | ( 25.) | 85 (133.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0075_FCACHE | 00000001 | ( 1.) | 86 (134.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0075_HFILL | 00000071 | ( 113.) | 87 (135.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0076_D | 00000002 | ( 2.) | 88 (136.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0076_T | 00000100 | ( 256.) | 89 (137.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0076_GDCS | 000000F3 | ( 243.) | 8A (138.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0076_ERTEMP | 00000019 | ( 25.) | 8B (139.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0076_FCACHE | 00000001 | ( 1.) | 8C (140.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0076_HFILL | 00000071 | ( 113.) | 8D (141.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0077_D | 00000002 | ( 2.) | 8E (142.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0077_T | 00000100 | ( 256.) | 8F (143.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0077_GDCS | 000000BC | ( 188.) | 90 (144.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0077_ERTEMP | 0000001D | ( 29.) | 91 (145.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0077_FCACHE | 00000001 | ( 1.) | 92 (146.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0077_HFILL | 00000024 | ( 36.) | 93 (147.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0078_D | 00000002 | ( 2.) | 94 (148.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0078_T | 00000100 | ( 256.) | 95 (149.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0078_GDCS | 00000090 | ( 144.) | 96 (150.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0078_ERTEMP | 00000011 | ( 17.) | 97 (151.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0078_FCACHE | 00000001 | ( 1.) | 98 (152.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0078_HFILL | 0000005C | ( 92.) | 99 (153.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0079_D | 00000002 | ( 2.) | 9A (154.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0079_T | 00000200 | ( 512.) | 9B (155.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0079_FCACHE | 00000005 | ( 5.) | 9C (156.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |
| X_0079_ERTEMP | 00000001 | ( 1.) | 9D (157.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC | BYTE |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0079_GDCS | 00000001 | ( 1.) | 9E (158.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0079_HFILL | 00000077 | ( 119.) | 9F (159.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0080_D | 00000002 | ( 2.) | A0 (160.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0080_T | 00000180 | ( 384.) | A1 (161.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0080_GDCS | 000000BC | ( 188.) | A2 (162.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0080_ERTEMP | 00000015 | ( 21.) | A3 (163.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0080_FCACHE | 00000001 | ( 1.) | A4 (164.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0080_HFILL | 0000002C | ( 44.) | A5 (165.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0081_D | 00000002 | ( 2.) | A6 (166.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0081_T | 0000047E | ( 1150.) | A7 (167.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0082_D | 00000002 | ( 2.) | A8 (168.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0082_T | 0000047E | ( 1150.) | A9 (169.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0083_D | 00000002 | ( 2.) | AA (170.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0083_T | 00000280 | ( 640.) | AB (171.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0083_GDCS | 000000E8 | ( 232.) | AC (172.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0083_ERTEMP | 00000021 | ( 33.) | AD (173.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0083_FCACHE | 00000001 | ( 1.) | AE (174.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0083_HFILL | 00000074 | ( 116.) | AF (175.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0084_D | 00000002 | ( 2.) | B0 (176.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0084_T | 00000100 | ( 256.) | B1 (177.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0084_GDCS | 000000E8 | ( 232.) | B2 (178.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0084_ERTEMP | 0000001D | ( 29.) | B3 (179.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0084_FCACHE | 00000001 | ( 1.) | B4 (180.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0084_HFILL | 00000078 | ( 120.) | B5 (181.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0085_D | 00000002 | ( 2.) | B6 (182.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0085_T | 00000100 | ( 256.) | B7 (183.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0085_GDCS | 000000FE | ( 254.) | B8 (184.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0085_ERTEMP | 00000029 | ( 41.) | B9 (185.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0085_FCACHE | 00000001 | ( 1.) | BA (186.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0085_HFILL | 00000056 | ( 86.) | BB (187.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0086_D | 00000002 | ( 2.) | BC (188.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0086_T | 00000180 | ( 384.) | BD (189.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0086_GDCS | 000000E8 | ( 232.) | BE (190.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0086_ERTEMP | 00000019 | ( 25.) | BF (191.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0086_FCACHE | 00000001 | ( 1.) | C0 (192.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0086_HFILL | 0000007C | ( 124.) | C1 (193.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0087_D | 00000002 | ( 2.) | C2 (194.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0087_T | 00000180 | ( 384.) | C3 (195.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0087_GDCS | 0000011F | ( 287.) | C4 (196.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0087_ERTEMP | 00000021 | ( 33.) | C5 (197.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0087_FCACHE | 00000001 | ( 1.) | C6 (198.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0087_HFILL | 0000003D | ( 61.) | C7 (199.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0088_D | 00000002 | ( 2.) | C8 (200.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0088_T | 00000100 | ( 256.) | C9 (201.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0088_GDCS | 00000090 | ( 144.) | CA (202.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0088_ERTEMP | 00000011 | ( 17.) | CB (203.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0088_FCACHE | 00000009 | ( 9.) | CC (204.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0088_HFILL | 00000054 | ( 84.) | CD (205.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0089_D | 00000002 | ( 2.) | CE (206.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0089_T | 00000100 | ( 256.) | CF (207.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0089_GDCS | 000000C7 | ( 199.) | D0 (208.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0089_ERTEMP | 00000011 | ( 17.) | D1 (209.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0089_FCACHE | 00000015 | ( 21.) | D2 (210.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0089_HFILL | 00000011 | ( 17.) | D3 (211.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0090_D | 00000002 | ( 2.) | D4 (212.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |

```
X_0090_T                00000100  (  256.)  D5 (213.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0090_GDCS             00000059  (   89.)  D6 (214.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0090_ERTEMP           0000000D  (   13.)  D7 (215.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0090_FCACHE           00000009  (    9.)  D8 (216.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0090_HFILL            0000000F  (   15.)  D9 (217.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0091_D                00000002  (    2.)  DA (218.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0091_T                00000100  (  256.)  DB (219.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0091_GDCS             000000A6  (  166.)  DC (220.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0091_ERTEMP           00000015  (   21.)  DD (221.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0091_FCACHE           00000001  (    1.)  DE (222.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0091_HFILL            00000042  (   66.)  DF (223.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0092_D                00000002  (    2.)  E0 (224.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0092_T                00000100  (  256.)  E1 (225.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0092_GDCS             00000059  (   89.)  E2 (226.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0092_ERTEMP           0000000D  (   13.)  E3 (227.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0092_FCACHE           00000009  (    9.)  E4 (228.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0092_HFILL            0000000F  (   15.)  E5 (229.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0093_D                00000002  (    2.)  E6 (230.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0093_T                00000100  (  256.)  E7 (231.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0093_GDCS             000000A6  (  166.)  E8 (232.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0093_ERTEMP           00000015  (   21.)  E9 (233.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0093_FCACHE           00000001  (    1.)  EA (234.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0093_HFILL            00000042  (   66.)  EB (235.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0094_D                00000002  (    2.)  EC (236.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0094_T                00000080  (  128.)  ED (237.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0094_GDCS             00000090  (  144.)  EE (238.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0094_ERTEMP           00000011  (   17.)  EF (239.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0094_FCACHE           00000009  (    9.)  F0 (240.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0094_HFILL            00000054  (   84.)  F1 (241.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
X_0095_D                00000000  (    0.)  F2 (242.)  NOPIC  USR  CON  REL  LCL NOSHR  EXE  RD  WRT NOVEC BYTE
```

## Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 27 | 00:00:00.17 | 00:00:00.45 |
| Command processing | 860 | 00:00:00.98 | 00:00:02.48 |
| Pass 1 | 4696 | 00:02:38.28 | 00:05:28.11 |
| Symbol table sort | 0 | 00:00:00.78 | 00:00:00.78 |
| Pass 2 | 34 | 00:00:43.17 | 00:00:58.78 |
| Symbol table output | 0 | 00:00:00.23 | 00:00:00.34 |
| Psect synopsis output | 0 | 00:00:00.90 | 00:00:01.21 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 5617 | 00:03:24.51 | 00:06:32.15 |

```
The working set limit was 7950 pages.
2021950 bytes (3950 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 376 non-local and 279 local symbols.
11541 source lines were read in Pass 1, producing 528 object records in Pass 2.
102 pages of virtual memory were used to define 47 macros.
```

### Macro library statistics

| Macro library name | Macros defined |
|---|---|
| DISK$UCODPACK00:[VAX750]COMETMAC.MLB;1 | 47 |
| SYS$COMMON:[SYSLIB]STARLET.MLB;2 | 0 |
| TOTALS (all libraries) | 47 |

2065 GETS were required to define 47 macros.

There were 0 errors, 2 warnings and 0 information messages, on lines:
 7760 (1)  8257 (1)

MACRO UCOD$0:[VAX750.MIC]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.MIC]ECKAC2/LIS=ECKAC2/OBJ=ECKAC2

(1)  24  "MIC Revision History"
(1)  1  ""
(1)  3  "DIAGNOSTIC MICROCODE MACROS"
(1)  98  "  RDM Control File Macros"
(1)  109  "  Diagnostic Macros"
(1)  194  ""
(1)  195  "MODULE GATE ARRAY MAPS"
(1)  197  "  L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1)  254  "  L0002 GATE ARRAY LOCATION"
(1)  311  "  L0003 GATE ARRAY LOCATION"
(1)  369  "  L0004 GATE ARRAY LOCATION"
(1)  427  "  L0011 & L0016 GATE ARRAY LOCATION"
(1)  485  ""
(1)  490  "EQUATED SYMBOLS
(1)  493  ""
(1)  2  "TEST  05F  TEST XB0 FOR MICROTRAP WITH CACHE PARITY ERROR
(1)  188  "TEST  060  TEST XB1 FOR MICROTRAP WITH CACHE PARITY ERROR
(1)  409  "TEST  061  TEST IR AND OSR REGISTERS
(1)  692  "TEST  062  COMPATIBILITY MODE ADDRESS TEST
(1)  853  "TEST  063  MEMORY ADDRESS REGISTER TEST
(1)  1100  "TEST  064  PAGE BOUNDARY TEST FOR BYTE/WORD/LONG DTYPE
(1)  1447  "TEST  065  UNALIGNED DATA READ FOR BYTE/WORD/LONG DTYPE
(1)  1692  "TEST  066  UNALIGNED DATA WRITE FOR BYTE/WORD/LONG DTYPE
(1)  1937  "TEST  067  TEST PTE ACCESS CHECK, READ MICRO ORDER
(1)  2134  "TEST  068  TEST PTE ACCESS CHECK, READ, KERNEL MODE
(1)  2286  "TEST  069  TEST PTE ACCESS CHECK, WRITE MICRO ORDER
(1)  2502  "TEST  06A  TEST PROBE ACCESS, READ, MODE SPECIFIED MICRO ORDER
(1)  2888  "TEST  06B  TEST PROBE ACCESS, READ AND WRITE MICRO ORDERS
(1)  3348  "TEST  06C  TEST PROBE ACCESS, WRITE, MODE SPECIFIED MICRO ORDER
(1)  3736  "TEST  06D  TEST BUS FIELD NOP, READ, AND READ.PHY MICRO ORDERS
(1)  3925  "TEST  06E  TEST BUS FIELD READ.LNG MICRO ORDER
(1)  4099  "TEST  06F  TEST BUS FIELD READ.LNG.MOD MICRO ORDER
(1)  4334  "TEST  070  TEST BUS FIELD READ.MOD MICRO ORDER
(1)  4564  "TEST  071  TEST BUS FIELD READ.NT MICRO ORDER
(1)  4758  "TEST  072  TEST BUS FIELD READ.SEC MICRO ORDER
(1)  4946  "TEST  073  TEST SECOND MDR REGISTER
(1)  5081  "TEST  074  TEST CMI DATA INTEGRITY
(1)  5235  "TEST  075  TEST CMI CONTROL AND STATUS
(1)  5389  "TEST  076  TEST CPU CMI DRIVERS
(1)  5518  "TEST  077  TEST WRITE VECTOR FROM RDM TO CPU
(1)  5713  "TEST  078  TEST RDM TO MAIN MEMORY WRITE/READ
(1)  5816  "TEST  079  TEST CPU TO MAIN MEMORY WRITE/READ
(1)  5993  "TEST  07A  TEST WRITE BUS ERROR INTERRUPT
(1)  6234  "TEST  07B  TEST MAIN MEMORY CSR0 AND CSR1
(1)  6424  "TEST  07C  TEST FOR CONTROLLER TYPE AND ARRAY CONFIGURATION-PART 1
(1)  6790  "TEST  07D  TEST FOR CONTROLLER TYPE AND ARRAY CONFIGURATION-PART 2
(1)  7148  "TEST  07E  TEST DATA INTEGRITY FOR ADDRESS TEST
(1)  7340  "TEST  07F  MAIN MEMORY DUAL ADDRESSING TEST
(1)  7604  "TEST  080  ECC SINGLE BIT ERROR TEST
(1)  7947  "TEST  081  ECC DOUBLE BIT ERROR TEST

```
0000     1          .TITLE  ECKAC VAX-11/750 MIC MICRO DIAGNOSTICS
0000     2          .IDENT  /V08.00/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1982
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  "MIC Revision History"
                        25 ;++
                        26 ; MIC MICRO-DIAGNOSTIC PACKAGE
                        27 ;
                        28 ; REVISION HISTORY
                        29 ;
                        30 ; 00-01 Bill Landry and Dave Spain      4-FEB-1980
                        31 ;       Started file.
                        32 ;
                        33 ; 00-02 Bill Landry and Dave Spain      5-FEB-1980
                        34 ;       Fixed numerous tests to invalidate both groups of TB before using them.
                        35 ;
                        36 ; 00-03 Bill Landry                     6-FEB-1980
                        37 ;       Fixed my typing errors in test 4C.
                        38 ;
                        39 ; 00-04 Bill Landry and Dave Spain      7-FEB-1980
                        40 ;       Skipped subtest 7 - 10 in test 4C. Also added cache data to PC
                        41 ;       increment and ADD adder tests.
                        42 ;
                        43 ; 00-05 Bill Landry     21-FEB-1980
                        44 ;       Fixed CMPREGMSK psuedo-ops in TB tests.
                        45 ;
                        46 ; 00-06 Bill Landry and Dave Spain      27-FEB-1980
                        47 ;       Changed a CLRTB.VA_VA macro in test 4C to CLRTB.VA_R[]. Also added
                        48 ;       cache validates to test 12 to prevent random cache parity errors.
                        49 ;
                        50 ; 00-07 Bill Landry     3-MAR-1980
                        51 ;       Added changes for revised SKIP pseudo op.
                        52 ;
                        53 ; 00-08 Bill Landry     13-MAR-1980
                        54 ;       Fixed tests that didnot use a CMPREGMSK pseudo op for testing the
                        55 ;       CSTRP register.
                        56 ; 00-09 Bill Landry     25-MAR-1980
                        57 ;       Added cache tests.
                        58 ;
                        59 ; 01-10 Dave Spain       4-APR-1980
                        60 ;       Added IRD tests to official release version.
                        61 ;
                        62 ; 01-11 Bill Landry     4-APR-1980
                        63 ;       Moved XB, PREFETCH and IRD tests to ECKAB.SRC so manufacturing can
                        64 ;       run one tape for DPM.
                        65 ;
                        66 ; 00-12 Bill Landry     4-APR-1980
                        67 ;       Let's save version 1.0 for our first SDC release
                        68 ;
                        69 ; 00-13 Bill Landry     8-APR-1980
                        70 ;       Revised cache tests to survive parity error traps.
                        71 ;
                        72 ; 00-14 Bill Landry and Dave Spain      15-APR-1980
                        73 ;       Reassembled using new COMETDEF.
                        74 ;
                        75 ; 00-15 Dave Spain      18-APR-1980
                        76 ;       Changed tests 4F and 50 to call out the ACV and UTR gate arrays instead
                        77 ;       of the MIC module.
                        78 ;
```

```
0000     79 ; 00-16 Bill Landry     1-MAY-1980
0000     80 ;       Revised TB tests to loop on error nicely.
0000     81 ;       Deleted redundent TB tests.
0000     82 ;
0000     83 ; 00-17 Bill Landry     14-MAY-1980
0000     84 ;       Added TB parity error tests
0000     85 ;
0000     86 ; 00-18 Bill Landry     10-JUN-1980
0000     87 ;       Added RDM tests to beginning of tape.
0000     88 ;
0000     89 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     90 ;       Added diagnostic macro file to title section and submitted to
0000     91 ;       RELEASE ENGINEERING.
0000     92 ;
0000     93 ; 01-01 Dave Spain      19-JUN-1980
0000     94 ;       Fixed Prefetch Incrementer test to clear out cache data after
0000     95 ;       every prefetch so that old data from cache will not cause test
0000     96 ;       to mistake bad addresses for good ones.
0000     97 ;
0000     98 ; 01-02 Bill Landry     26-JUN-1980
0000     99 ;       Added tests that check XB trap conditions
0000    100 ;
0000    101 ; 01-03 Bill Landry     30-JUN-1980
0000    102 ;       Added tests that check XB access violations
0000    103 ;
0000    104 ; 01-04 Dave Spain      11-JUL-1980
0000    105 ;       Changed location of BEGINSA pseudo-op to fix signature analysis
0000    106 ;       problem in PC_PC+WBUS test.
0000    107 ;
0000    108 ; 01-05 Bill Landry     8-AUG-1980
0000    109 ;       Added CMI tests in honor of my vacation.
0000    110 ;
0000    111 ; 01-06 Bill Landry     25-AUG-1980
0000    112 ;       Remodeled RDM to CMI tests.
0000    113 ;
0000    114 ; 01-07 Bill Landry     23-SEP-1980
0000    115 ;       Added byte mask testing and improved testing of first memory array.
0000    116 ;
0000    117 ; 02-00 Bill Landry     29-SEP-1980
0000    118 ;       Second release to SDC
0000    119 ;
0000    120 ; 02-01 Bill Landry     5-NOV-1980
0000    121 ;       Added miscellaneous enhancements
0000    122 ;
0000    123 ; 03-00 Bill Landry     17-NOV-1980
0000    124 ;       Third release to SDC
0000    125 ;
0000    126 ; 03-01 Bill Landry     24-NOV-1980
0000    127 ;       Added tests for cache parity logic
0000    128 ;
0000    129 ; 03-02 Bill Landry     30-DEC-1980
0000    130 ;       Separated RDM and MIC tests into two files
0000    131 ;
0000    132 ; 04-00 Bill Landry     05-JAN-1981
0000    133 ;       Submitted to RELEASE ENGINEERING
```

```
0000    134 ;
0000    135 ; 04-01 Bill Landry     14-JAN-1981
0000    136 ;       Revised ECC test to fix bug caused by ECO to INT PEND logic.
0000    137 ;
0000    138 ; 04-02 Bill Landry     27-FEB-1981
0000    139 ;       Udated MDR test to include branching logic.
0000    140 ;       Fixed PC_* tests to fix bug found by field service.
0000    141 ;
0000    142 ; 05-00 Bill Landry     2-MAR-1981
0000    143 ;       Added WRITE BUS ERROR INTERRUPT test
0000    144 ;       Submitted to RELEASE ENGINEERING
0000    145 ;
0000    146 ; 05-01 Bill Landry     18-MAR-1981
0000    147 ;       Made minor changes to various tests as suggested by BT manufacturing.
0000    148 ;
0000    149 ; 05-02 Bill Landry     8-JUN-1981
0000    150 ;       Fixed bug found by Paul Magnet in TB TAG PARITY TESTS.
0000    151 ;
0000    152 ; 05-03 Bill Landry     20-AUG-1981
0000    153 ;       Fixed bugs in memory tests caused by eco to interrupt pending logic.
0000    154 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    155 ;       logic.
0000    156 ;       Replaced all references to COMET with the word CPU.
0000    157 ;       Fixed bug in CACHE TAG MEMORY DUAL ADDRESSING TEST.
0000    158 ;       Submitted to RELEASE ENGINEERING.
0000    159 ;
0000    160 ; 05-04 Bill Landry     4-NOV-1981
0000    161 ;       Moved WCS test from DPM tape to this tape
0000    162 ;       Submitted to RELEASE ENGINEERING.
0000    163 ;
0000    164 ; 05-05 Bill Landry     10-NOV-1981
0000    165 ;       Changed all references of CLRTB.VA_WB to CLRTB.VA_VA.  This fixes a
0000    166 ;       problem caused by plugging in the FPA.
0000    167 ;       Re-released to SDC. Stopped release of 5.4
0000    168 ;
0000    169 ; 05-06 Bill Landry     6-JAN-1982
0000    170 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    171 ;       logic (same as 5.3 above).
0000    172 ;
0000    173 ; 05-07 Bill Landry     23-FEB-1982
0000    174 ;       Fixed more bugs in single bit error tests found by PAUL NATUSH.
0000    175 ;       Same as revision 5.3.
0000    176 ; 05-08 Dave Shull      9-Mar-1982
0000    177 ;       Changed the the IFERROR callouts so when L0011 (MC0) is encountered
0000    178 ;       L0016 (MC1) will be included, when M8728 (MA0) is encountered M8750
0000    179 ;       (MA1) will be included, and when the MAP gate array is encountered
0000    180 ;       MAD gate array will be included.
0000    181 ; 05-09 Dave Shull      10-Mar-1982
0000    182 ;       Added a test to check MS750 controller CSR2 and determine the
0000    183 ;       controller type (ie., L0011 or L0016) and to also print a configuration
0000    184 ;       map of the array's that are installed.
0000    185 ; 06-00 Dave Shull      22-mar-1982
0000    186 ;       Changed the 4 existing memory tests to check for array 0 type then
0000    187 ;       adjust addresses accordingly and test all of the first array versus
0000    188 ;       only testing the first 256kb.
```

```
0000    189 ;       Added 4 new tests that will check to see if array 1 is present and if
0000    190 ;       it is then determine the size and test accordingly.  It looks at both
0000    191 ;       array 0 and array 1, array 0 to determine the starting address of array
0000    192 ;       1 and array 1 to determine the starting and ending address to be tested
0000    193 ; 06-01 Dave Shull      26-Mar-1982
0000    194 ;       Changed READ.SEC MICRO ORDER test so the test could ran in an infinite
0000    195 ;       loop by itself.  Added code to reload Cache everytime the test is
0000    196 ;       restarted in any of the test that are using cache data that could get
0000    197 ;       blown away.
0000    198 ; 06-02 Dave Shull      05-Apr-1982
0000    199 ;       Removed all of the BEGINSA and ENDSA pseudo's
0000    200 ;       Submitted to RELEASE ENGINEERING
0000    201 ; 07-00 Dave Shull      10-May-1982
0000    202 ;       Added new test ( name <Test PCBACK While PC Increment Logic is Active>)
0000    203 ;       to check for side-affects on PCBACK register while incrementin the PC
0000    204 ;       through the execution buffer.
0000    205 ; 07-01 Dave Shull      14-May-1982
0000    206 ;       Changed all CLRTB.VA_VA macro's to prevent translation buffer
0000    207 ;       invalidate failures when manufacturing runs the micro's with an
0000    208 ;       external clock set to fast margins.
0000    209 ; 07-02 Dave Shull      17-May-1982
0000    210 ;       Added coverage in the Read Lock Function Test for the CML gate array.
0000    211 ; 07-03 Dave Shull      14-Jun-1982
0000    212 ;       Changed CMK callouts to CML for new gate array.
0000    213 ; 08-00 Joe Zagarella   11-Feb-1986
0000    214 ;       Enhanced the following tests to be compatible with the new memory
0000    215 ;       controller(L0022) and array card(M7199):
0000    216 ;       Test write bus error interrupt
0000    217 ;       Test for controller type and array configuration
0000    218 ;       Test data integrity for address test
0000    219 ;       Main memory dual addressing test
0000    220 ;       Array 0 memory test (part1 thru part4)
0000    221 ;       Array 1 memory test (part1 thru part4)
0000    222 ;
0000    223 ;--
0000      1         .SBTTL  ""
```

```
                        3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
                        4 ;++
                        5 ; Revision History
                        6 ;
                        7 ; 28    Dave Spain      22-Jun-82
                        8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
                        9 ;       Decided not to use them.
                       10 ;
                       11 ; 27    Dave Spain      15-Jun-82
                       12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
                       13 ;
                       14 ; 26    Dave Spain      19-May-82
                       15 ;       Added M[]_FPA macro.
                       16 ;
                       17 ; 25    Rich Lewis      23-Feb-82
                       18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
                       19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
                       20 ;
                       21 ; 24    Dave Spain      02-Feb-82
                       22 ;       Added FPA_R[]+M[] macro.
                       23 ;
                       24 ; 23    Bill Landry     22-OCT-81
                       25 ;       Added PSL_RDM macro
                       26 ;
                       27 ; 22    Dave Spain       5 Oct 81
                       28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
                       29 ;
                       30 ; 21    Bill Landry     29 Sept 81
                       31 ;       Added FPA macro, FPA_M[] FPA_R[].
                       32 ;
                       33 ; 20    Dave Spain      14 Aug 81
                       34 ;       Added the TESTFPA [] macro.
                       35 ;
                       36 ; 19    Dave Spain      18 July 81
                       37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
                       38 ;
                       39 ; 18    Dave Spain      24 Sept 80
                       40 ;       Added RDM_Q macro. This macro uses the ALU.
                       41 ;
                       42 ; 17    Bill Landry     23 Sept 80
                       43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
                       44 ;
                       45 ; 16    Dave Spain      27 August 80
                       46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
                       47 ;
                       48 ; 15    Bill Landry     4 June 80
                       49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
                       50 ;       satisfy validity checks.
                       51 ;
                       52 ; 14    Dave Spain      23 May 80
                       53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
                       54 ;       Bill.
                       55 ;
                       56 ; 13    Dave Spain      14 Apr 80
                       57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

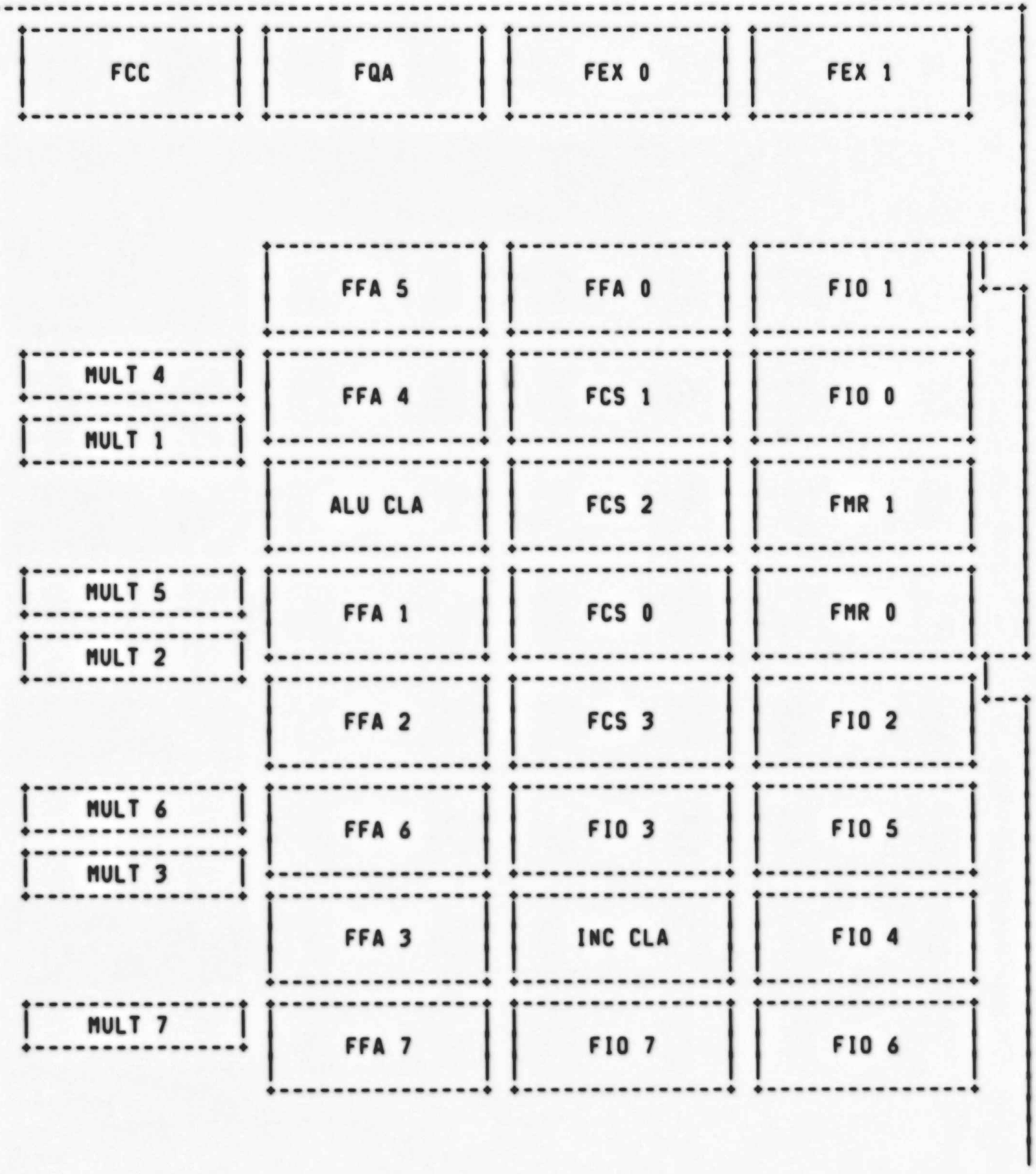
```
0000     58 ;       the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain      27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry     19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry     19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry     18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry     07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain      06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain      03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain      22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain      30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain      30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain      29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                          "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                        "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                     "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
                0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_ -
cont> WB"
                0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   115 ;                                        BUS/PRB.RD.PTE"
                0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   118 ;                                        BUS/PRB.RD.PTE"
                0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_ -
cont> WB"
                0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
                0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
                0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
                0000   124 ;                                        WCTRL/VA_WB"
                0000   125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
                0000   126 ;FPA_M[] FPA_R[]                       "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX -
cont> /R.S,
                0000   127 ;                                        ROT/ZERO,ALU/OR"
                0000   128 ;FPA_RDM                                "WB_RDM,FPA/FPA_DATA.WBUS"
                0000   129 ;FPA_R[]                                "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_D -
cont> ATA.WBUS"
                0000   130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FP -
cont> A/FPA_DATA.WBUS"
                0000   131 ;FPA_WB                         "FPA/FPA_DATA.WBUS"
                0000   132 ;MEMSCAR_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
                0000   133 ;M[]_FPA                                "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
                0000   134 ;M[]_LONLIT                     "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
                0000   135 ;M[]_R[].OR.NIB0(M)             "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
                0000   136 ;                                        MUX/R.S,DQ1/NOP"
                0000   137 ;M[]_RDM                                "WB_RDM,SPW/MLONG,MSRC/a1"
                0000   138 ;PC_PC+RDM                      "WCTRL/PC_PC+WB,WB_RDM"
                0000   139 ;PC_PC+4 MB_XB                  "MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
                0000   140 ;PC_PC+4 RDM_XB                 "RSRC/ZERO,MSRC/XB,PC_PC+I,ISTRM/ISIZE_DSIZE,
                0000   141 ;                                        MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
                0000   142 ;PC_RDM                         "WB_RDM,WCTRL/PC_WB"
                0000   143 ;PSL_RDM                                "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
                0000   144 ;PSL_WB                         "CCPSL/PSL_WB.CCBR_ALUS"
                0000   145 ;Q_D WX_R                       "ALPCTL/WX_R.Q_D"
                0000   146 ;Q_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
                0000   147 ;Q_M[].RL.PTE                   "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
                0000   148 ;Q_R[].AND.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
                0000   149 ;Q_R[].ASL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
                0000   150 ;Q_R[].OR.Q                     "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
                0000   151 ;Q_R[].RL.P                     "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
                0000   152 ;RDM_ALUF                       "ALPCTL/WB_ALUF,RDM_WB"
                0000   153 ;RDM_FPA                                "FPA/WBUS_FPA,RDM_WB"
                0000   154 ;RDM_LONLIT                     "WB_LONLIT,RDM_WB"
                0000   155 ;RDM_LONLIT.Q_M                 "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
                0000   156 ;RDM_M[]                                "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
                0000   157 ;RDM_M[].OR.R[]                 "WB_M[a1].OR.R[a2],RDM_WB"
                0000   158 ;RDM_M[]-R[]                    "WB_M[a1]-R[a2],RDM_WB"
                0000   159 ;RDM_M[].XZ.OR.R[]              "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                0000   160 ;                                RDM_WB"
                0000   161 ;RDM_PC                         "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
                0000   162 ;RDM_PCBACK                     "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
                0000   163 ;RDM_Q                          "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
                0000   164 ;RDM_Q Q_D                      "ALPCTL/WX_Q.Q_D,RDM_WB"
```

```
0000   165 ;RDM_R[]                          "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                  "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                  "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

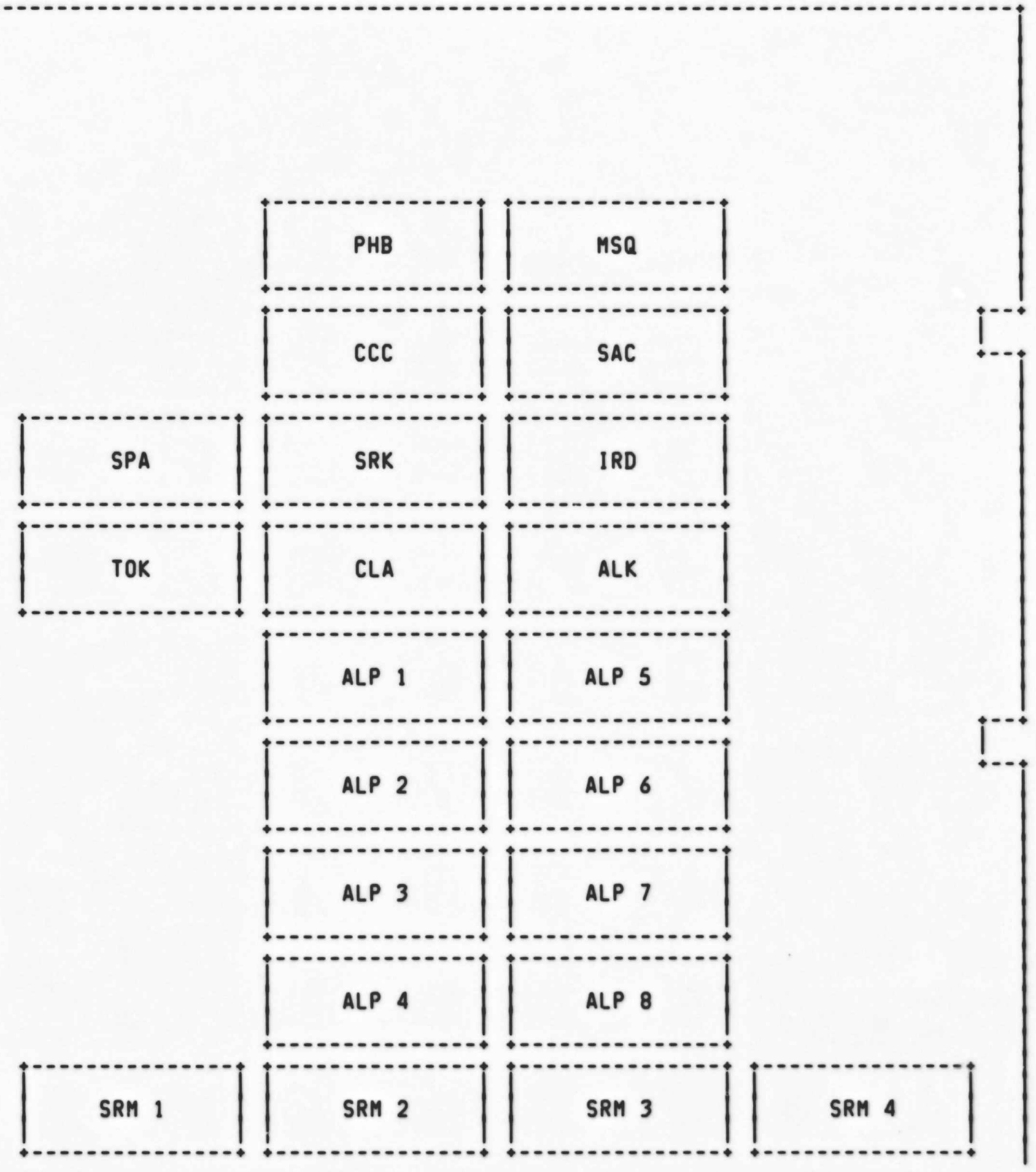
```
                     0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
                     0000   169 ;R[]_RDM                                "WB_RDM,SPW/RLONG,RSRC/a1"
                     0000   170 ;R[].DT_Q                        "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
                     0000   171 ;R[]_VA                         "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
                     0000   172 ;R[]_WB                         "RSRC/a1,SPW/RLONG"
                     0000   173 ;TB_RDM                         "WCTRL/TB_WB,WB_RDM,MSRC/VA"
                     0000   174 ;TB_WB                          "WCTRL/TB_WB,MSRC/VA"
                     0000   175 ;TESTFPA []                     "TESTFPA/a1"
                     0000   176 ;VA_PC+LONLIT+I.PC_PC+I         "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/ -
cont> OR,
                     0000   177 ;                                       ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
                     0000   178 ;                                       SIZE[LONG]"
                     0000   179 ;VA_Q.OR.R[]                    "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
                     0000   180 ;VA_RDM                         "WB_RDM,WCTRL/VA_WB"
                     0000   181 ;VA_R[].RL.P                    "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
                     0000   182 ;VA_VA-ZLIT0[]                  "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
                     0000   183 ;                                       MUX/M.S,ALU/A-B-CI"
                     0000   184 ;VA_WB                          "WCTRL/VA_WB"
                     0000   185 ;WB_FPA CCR                     "FPA/WBUS_FPA.CC"
                     0000   186 ;WB_FPA TCR                     "WCTRL/FPTCR"
                     0000   187 ;WB_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                     0000   188 ;WB_LONLIT.Q_M                  "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                     0000   189 ;WB_M[]+R[]                     "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
                     0000   190 ;WB_R[].Q_D                     "RSRC/a1,ALPCTL/WX_R.Q_D"
                     0000   191 ;WDR_R[]                                "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/Z -
cont> ERO"
                     0000   192 ;WRITE.LONG R[]                 "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
                     0000   193 ;                                       MUX/R.S,ALU/OR,SIZE[LONG]"
                     0000   194         .SBTTL  ""
                     0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                        .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```



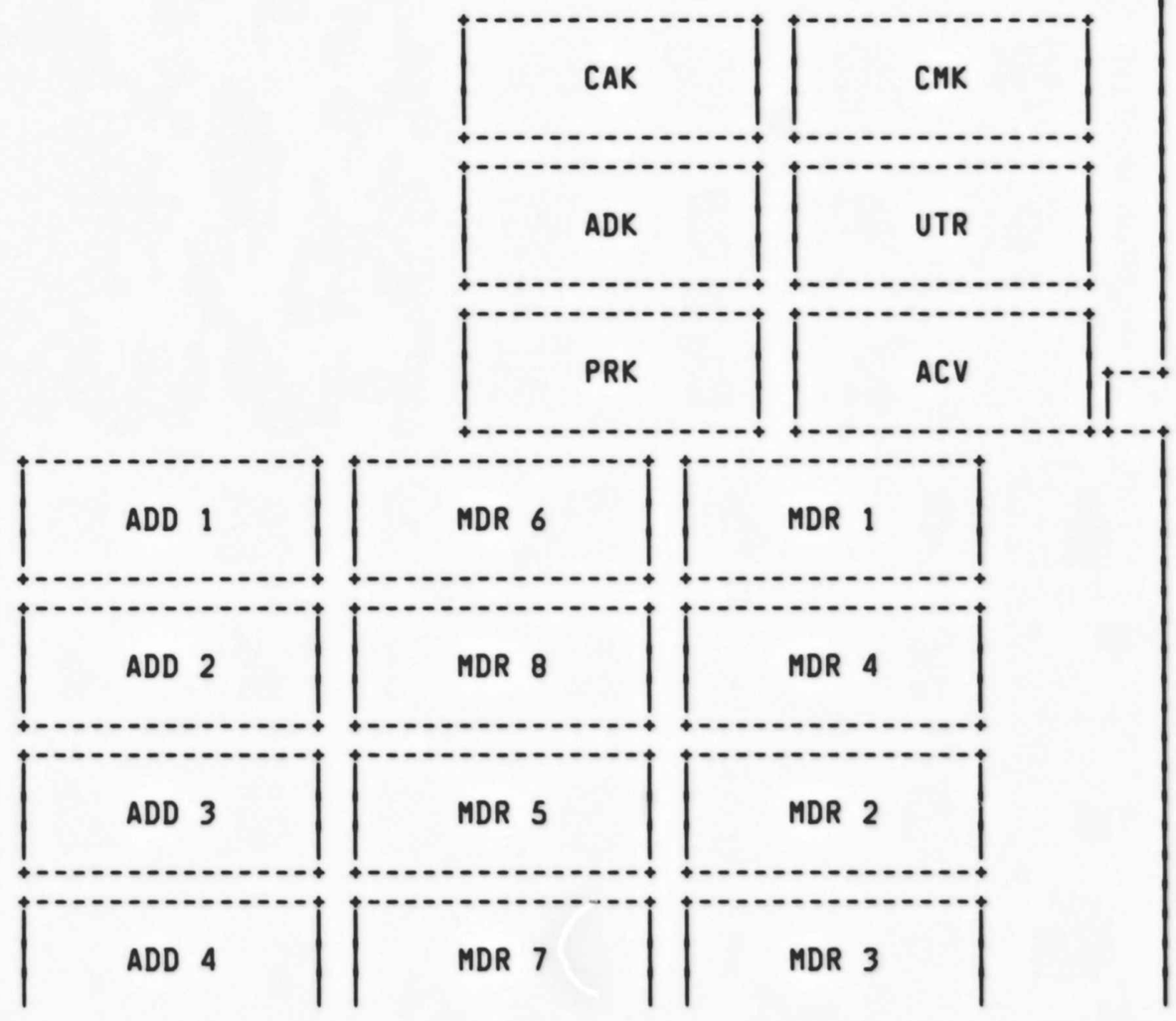
```
                0000   252 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0002 GATE ARRAY LOCATION"
```



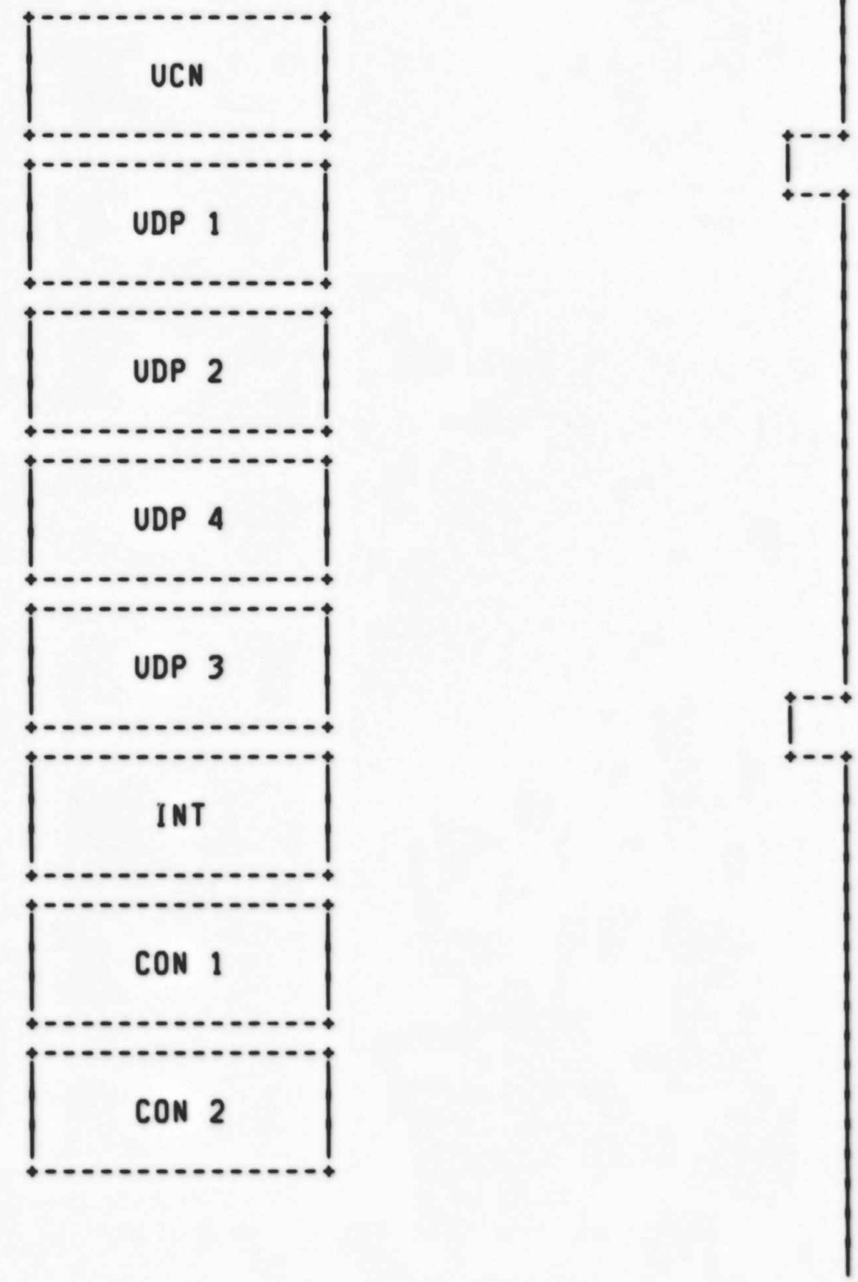
```
0000   309 ;+-----------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0003 GATE ARRAY LOCATION"
```



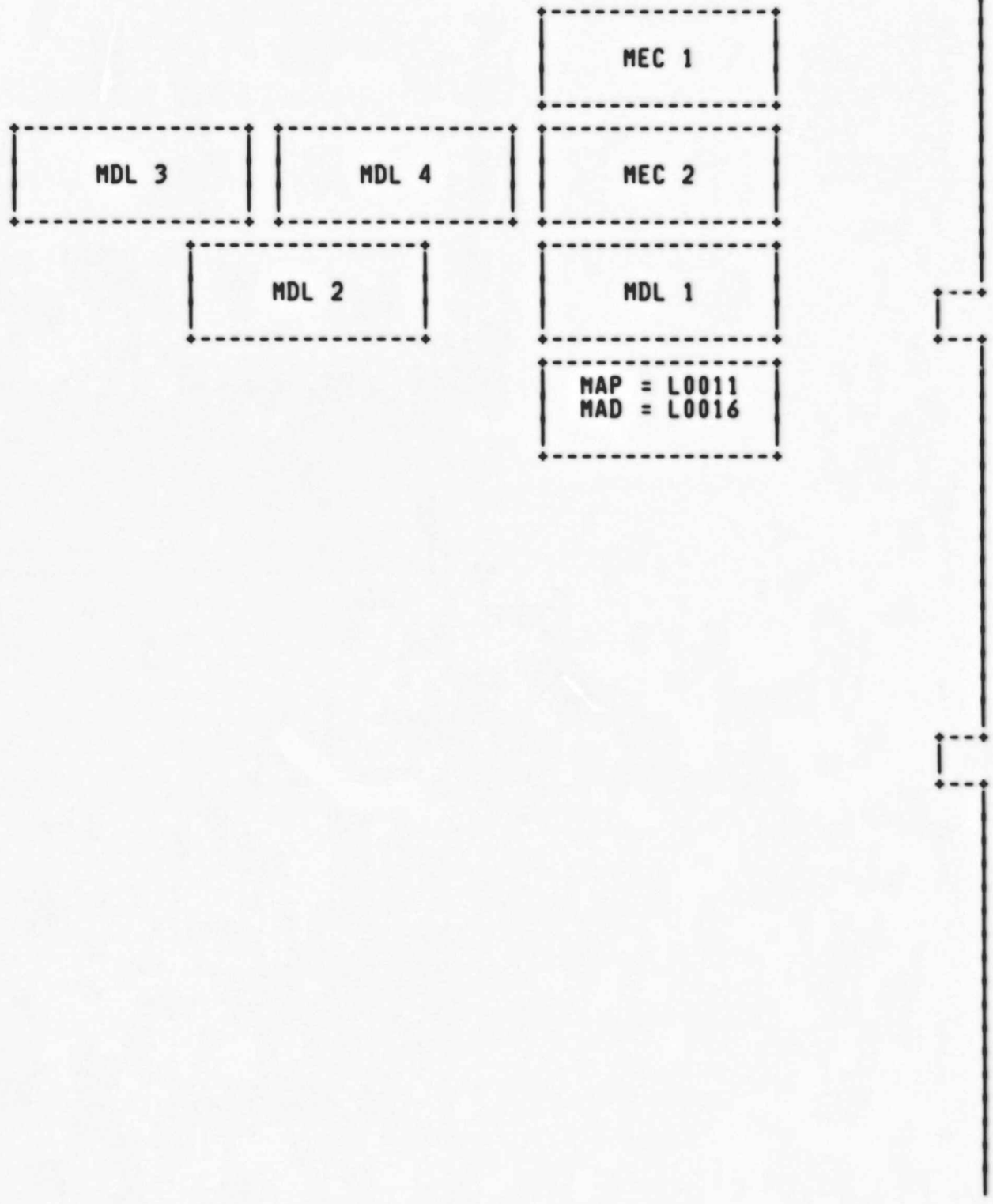
```
0000   366 ;|              +---------------+ +---------------+ +---------------+          |
0000   367 ;+-------------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0004 GATE ARRAY LOCATION"
```



```
0000   424 ;|                                                                                |
0000   425 ;+--------------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
```



```
0000   482 ;|                                                                    |
0000   483 ;+-------------------------------------------------------------------+
```

```
                        0000    485          .SBTTL  ""
                        0000    486          .LIST   MC
                        0000    487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                        0000    488          .LIST   ME
                        0000    489          .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
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                          ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_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    491   .NLIST  ME
          0000    492   .LIST   MC
          0000    493   .SBTTL  ""
          0000    494   .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
0033        .SBTTL  "TEST   05F     TEST XB0 FOR MICROTRAP WITH CACHE PARITY ERROR
0033      3 ;*******************************************************************************
0033      4 ;++
0033      5 ;
0033      6 ; FUNCTIONAL DESCRIPTION:
0033      7 ;
0033      8 ;       THIS TEST WILL VERIFY THAT A MACHINE CHECK OCCURS WHEN XB0 ENCOUNTERS
0033      9 ;       A CACHE PARITY ERROR DURING PREFETCH.
0033     10 ;
0033     11 ;
0033     12 ; TEST ALGORITHM:
0033     13 ;
0033     14 ;       1.  EXECUTE DCS PROGRAM
0033     15 ;               a.  WRITE ZERO INTO THE VA REGISTER
0033     16 ;               b.  WRITE CACHE ADDRESS ZERO WITH A PARTIY ERROR
0033     17 ;               c.  INCREMENT VA TO ADDRESS 4
0033     18 ;               d.  VALIDATE CACHE ADDRESS 4
0033     19 ;               e.  LOAD PC WITH ZERO TO CAUSE PREFETCH
0033     20 ;               f.  SOURCE XB0 TO ALLOW MACHINE CHECK TO OCCUR
0033     21 ;       2.  TEST THE CS ADDRESS FOR MACHINE CHECK
0033     22 ;       3.  IF NO ERROR EXECUTE DCS PROGRAM
0033     23 ;               a.  READ THE BUS ERROR REGISTER TO THE RDM
0033     24 ;       4.  TEST THE BUS ERROR REGISTER FOR UNCORRECTABLE AND LOST ERROR
0033     25 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
0033     26 ;               a.  READ THE MACHINE CHECK REGISTER TO THE RDM
0033     27 ;               b.  CLEAR THE MACHIHE CHECK REGISTER
0033     28 ;       6.  TEST THE MACHINE CHECK REGISTER FOR BUS AND XB ERROR
0033     29 ;       7.  IF NO ERROR GO TO NEXT TEST
0033     30 ;
0033     31 ;
0033     32 ; TEST PATTERNS:
0033     33 ;
0033     34 ;       NONE
0033     35 ;
0033     36 ;
0033     37 ; LOGIC DESCRIPTION:
0033     38 ;
0033     39 ;       THIS TEST VERFIES THAT THE UTR GATE ARRAY CAN LATCH THE FACT THAT A
0033     40 ;       PARITY ERROR OCCURED DURING PREFETCH AND REPORT SUCH WHEN THE XB IS
0033     41 ;       SOURCED.  THE BUS ERROR AND MACHINE CHECK REGISTERS ARE ALSO CHECKED
0033     42 ;       FOR CORRECT DATA.
0033     43 ;
0033     44 ;
0033     45 ; ASSUMPTIONS:
0033     46 ;
0033     47 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0033     48 ;
0033     49 ;
0033     50 ; ERROR DESCRIPTION:
0033     51 ;
0033     52 ;       FIRST IFERROR:
0033     53 ;               EXPECTED CS ADDRESS
0033     54 ;               RECEIVED CS ADDRESS
0033     55 ;
0033     56 ;       SECOND IFERROR:
```

```
                                  0033   ;                EXPECTED BUS ERROR REGISTER CONTENTS
                                  0033   ;                RECEIVED BUS ERROR REGISTER CONTENTS
                                  0033   ;
                                  0033   ;        THIRD IFERROR:
                                  0033   ;                EXPECTED MACHINE CHECK REGISTER CONTENTS
                                  0033   ;                RECEIVED BUS ERROR REGISTE CONTENS
                                  0033   ;
                                  0033   ;--
                                  0033   ;*****************************************************************************
                                  0033
                                  0033
                                  0033           INITIALIZE                                  ;INIT THE CPU
                                  0035           EXPUTRAP                                    ;EXPECT MACHINE CHECK
                                  0037           ERRLOOP                                     ;ERROR LOOP POINT
                                  0039           FETCH          0                            ;START DCS PROGRAM
                                  003E           BRSTCLK        9                            ;END DCS PROGRAM
                                  0042           CMPREGMSK      CSTRP,1$,,2$,,W,             ;TEST FOR MACHINE CHECK
                                  004E           IFERROR        ,<UTR>,                      ;NO MACHINE CHECK
                                  0054           FETCH          <^X0A>                       ;START DCS PROGRAM
                                  0059           BRSTCLK        <^X0D>                       ;END DCS PROGRAM
                                  005D           CMPREGMSK      WHREG,3$,,4$,,L,             ;TEST BUS ERROR REGISTER
                                  0069           IFERROR        ,<UTR>,                      ;REGISTER NOT SET CORRECTLY
                                  006F           FETCH          <^X0E>                       ;START DCS PROGRAM
                                  0074           BRSTCLK        <^X12>                       ;END DCS PROGRAM
                                  0078           CMPREGMSK      WHREG,5$,,4$,,L,             ;TEST MACHINE CHECK REGISTER
                                  0084           IFERROR        ,<UTR>,                      ;REGISTER NOT SET CORRECTLY
                                  008A           SKIP                                        ;GO TO NEXT TEST
                                  0091
                                  0091
                                  0091   BEGIN_TEST_DATA
                                  0091
                                  0091       ;-----------------------------------------------------------------------
                                  0091
                           0028   0091   1$:      .WORD   ^X0028                              ;MACHINE CHECK VECTOR
                           3FFF   0093   2$:      .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                       06000000   0095   3$:      .LONG   ^X06000000                          ;CONTENTS OF BUS ERROR REGISTER
                       0F000000   0099   4$:      .LONG   ^X0F000000                          ;MASK FOR CMPREGMSK PSEUDO OP
                       09000000   009D   5$:      .LONG   ^X09000000                          ;CONTENTS OF MACHINE CHECK REG
                                  00A1
                                  00A1
                                  00A1
                                  00A1   BEGIN_CPU_DATA
                                  0002
                                  0002   DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001   ;% 00   NOP
U 01, 7700,C800,5BE4,03D8,4A70,1802  000C   ;% 01   VA_R[ZERO]                                 ;ZERO THE VA REGISTER
U 02, 7700,4F80,5BE4,03D8,5480,1803  0017   ;% 02   WRITE.PHY,WCTRL/WDR_WB.UR,WB_R[ZERO],MISC/FORCE.CACHE
                                  0022                                                      ;WRITE PARITY ERROR TO ADD 0
U 03, 7700,CF80,5BE4,03D8,5470,1804  0022   ;% 03   WCTRL/WDR_WB.UR,WB_R[ZERO],MISC/FORCE.CACHE
U 04, 7700,C800,0364,0300,4470,1805  002D   ;% 04   VA_VA+4                                     ;INCREMENT VA TO 4
U 05, 7700,4800,5BE4,03D8,5480,1806  0038   ;% 05   WRITE.PHY,WCTRL/WDR_WB.UR,WB_R[ZERO]        ;WRITE ZERO TO ADDRESS 4
U 06, 7700,C800,5BE4,03D8,5470,1807  0043   ;% 06   WCTRL/WDR_WB.UR,WB_R[ZERO]                  ;...
U 07, 7700,4800,5BE4,03D8,4870,1808  004E   ;% 07   PC_R[ZERO]                                 ;CAUSE PREFETCH
U 08, 7701,C857,5924,020A,0470,1809  0059   ;% 08   R[TEMP2]_XB PC_PC+4                         ;CAUSE MACHINE CHECK
```

```
U 09. 7300,4800,0364,0300,0470,180A  0064   137 ;% 09   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
U 0A. 7700,C800,0364,0300,0470,180B  006F   141 ;% 0A   NOP
U 0B. 7700,4800,5BE4,0300,6870,180C  007A   145 ;% 0B   MEMSCAR_R[TEMP0]                       ;READ BUS ERROR REG TO RDM
U 0C. 6700,C800,0364,0300,6470,180D  0085   149 ;% 0C   RDM_WB,WCTRL/MEMSCR                    ;...
U 0D. 7300,C800,0364,0300,0470,180E  0090   153 ;% 0D   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
U 0E. 7700,4800,0364,0300,0470,180F  009B   157 ;% 0E   NOP
U 0F. 7700,8800,5BE4,0304,6870,1810  00A6   161 ;% 0F   MEMSCAR_R[TEMP1]                       ;READ MACHINE CHECK REG TO RDM
U 10. 6700,4800,0364,0300,6470,1811  00B1   165 ;% 10   RDM_WB,WCTRL/MEMSCR                    ;...
U 11. 7700,8800,5B70,0300,6070,1812  00BC   169 ;% 11   MEMSCR_0                              ;CLEAR MACHINE CHECK REGISTER
U 12. 7300,C800,0364,0300,0470,1813  00C7   173 ;% 12   NOP,STOP.CLOCK                          ;STOP THE CPU CLOCK
                                     00D2   177
                                     00D2   178 RTEMP_DATA
                                     0001   179
                          09000000   0001   180         .LONG   ^X09000000                    ;ADDRESS OF BUS ERROR REGISTER
                          08000000   0005   181         .LONG   ^X08000000                    ;ADDRESS OF MACHINE CHECK REG
                                     0009   182
                                     0009   183 END_CPU_DATA
                                     00A1   184
                                     00A1   185
                                     00A1   186 END_TEST_DATA
                                     00A1
                                     00A1        ;-----------------------------------------------------------------------
                                     00A1   187 ENDTEST
```

```
0033          .SBTTL  "TEST   060     TEST XB1 FOR MICROTRAP WITH CACHE PARITY ERROR
0033    189 ;*****************************************************************************
0033    190 ;++
0033    191 ;
0033    192 ; FUNCTIONAL DESCRIPTION:
0033    193 ;
0033    194 ;       THIS TEST WILL VERIFY THAT A MACHINE CHECK OCCURS WHEN XB1 ENCOUNTERS
0033    195 ;       A PARITY ERROR ON PREFETCH.
0033    196 ;
0033    197 ;
0033    198 ; TEST ALGORITHM:
0033    199 ;
0033    200 ;       1.  EXECUTE DCS PROGRAM
0033    201 ;               a.  ZERO RTEMP4
0033    202 ;               b.  WRITE ONES TO ADDRESS FFC
0033    203 ;               c.  INCREMENT VA TO 1000
0033    204 ;               d.  WRITE ZEROS TO ADDRESS 1000 WHILE FORCING A PARITY ERROR
0033    205 ;               e.  LOAD PC WITH FFC TO CAUSE PREFETCH
0033    206 ;               f.  SOURCE XB0 TO RTEMP4
0033    207 ;               g.  SOURCE XB1 TO RTEMP4
0033    208 ;       2.  TEST CS ADDRESS FOR MACHINE CHECK VECTOR
0033    209 ;       3.  IF NO ERROR EXECUTE DCS PROGRAM
0033    210 ;               a.  READ RTEMP4 TO RDM
0033    211 ;       4.  TEST RTEMP 4 FOR ALL ONES
0033    212 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
0033    213 ;               a.  READ THE BUS ERROR REGISTER TO THE RDM
0033    214 ;       6.  TEST BUS ERROR REGISTER FOR UNCORRECTABLE AND LOST ERROR
0033    215 ;       7.  IF NO ERROR EXECUTE DCS PROGRAM
0033    216 ;               a.  READ MACHINE CHECK REGISTER TO RDM
0033    217 ;               b.  CLEAR THE MACHINE CHECK REGISTER
0033    218 ;       8.  TEST MACHINE CHECK REGISTER FOR BUS ERROR AND XB
0033    219 ;       9.  IF NO ERROR GO TO NEXT TEST
0033    220 ;
0033    221 ;
0033    222 ; TEST PATTERNS:
0033    223 ;
0033    224 ;       NONE
0033    225 ;
0033    226 ;
0033    227 ; LOGIC DESCRIPTION:
0033    228 ;
0033    229 ;       THIS TEST INSURES THAT THE UTR CAN LATCH THE FACT THAT XB1 HAS
0033    230 ;       EXPERICENCED A CACHE PARITY ERROR DURING PREFETCH.  THE UTR GATE
0033    231 ;       ARRAY MUST CAUSE A MACHINE CHECK WHEN XB1 IS SOURCED.  ANY ERRORS
0033    232 ;       SHOULD BE CONFINED TO THE GATE ARRAY.
0033    233 ;
0033    234 ;
0033    235 ; ASSUMPTIONS:
0033    236 ;
0033    237 ;       THIS TEST ASSUMES THE PREFETCH LOGIC IS OPERATIONAL.
0033    238 ;
0033    239 ;
0033    240 ; ERROR DESCRIPTION:
0033    241 ;
0033    242 ;       FIRST IFERROR:
```

```
                 0033   ;                EXPECTED CS ADDRESS
                 0033   ;                RECEIVED CS ADDRESS
                 0033   ;
                 0033   ;        SECOND IFERROR:
                 0033   ;                EXPECTED CONTENTS OF RTEMP4
                 0033   ;                RECEIVED CONTENTS OF RTEMP4
                 0033   ;
                 0033   ;        THIRD IFERROR:
                 0033   ;                EXPECTED CONTENTS OF BUS ERROR REGISTER
                 0033   ;                RECEIVED CONTENTS OF BUS ERROR REGISTER
                 0033   ;
                 0033   ;        FORTH IFERROR:
                 0033   ;                EXPECTED CONTENTS OF MACHINE CHECK REGISTER
                 0033   ;                RECEIVED CONTENTS OF MACHINE CHECK REGISTER
                 0033   ;
                 0033   ;--
                 0033   ;*******************************************************************************
                 0033
                 0033
                 0033           INITIALIZE                                  ;INIT THE CPU
                 0035           EXPUTRAP                                    ;EXPECT MACHINE CHECK
                 0037           ERRLOOP                                     ;ERROR LOOP POINT
                 0039           FETCH           0                           ;START DCS PROGRAM
                 003E           BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                 0042           CMPREGMSK       CSTRP,1$,,2$,,W,            ;TEST FOR MACHINE CHECK
                 004E           IFERROR         ,<UTR>,                     ;NO MACHINE CHECK
                 0054           FETCH           <^X0C>                      ;START DCS PROGRAM
                 0059           BRSTCLK         <^X0E>                      ;END DCS PROGRAM
                 005D           CMPREG          WHREG,6$,,L,                ;TEST RTEMP4 FOR CORRECT DATA
                 0066           IFERROR         ,<UTR>                      ;INCORRECT U-TRAP
                 006C           FETCH           <^X0F>                      ;START DCS PROGRAM
                 0071           BRSTCLK         <^X12>                      ;END DCS PROGRAM
                 0075           CMPREGMSK       WHREG,3$,,4$,,L,            ;TEST BUS ERROR REGISTER
                 0081           IFERROR         ,<UTR>,                     ;REGISTER NOT SET CORRECTLY
                 0087           FETCH           <^X13>                      ;START DCS PROGRAM
                 008C           BRSTCLK         <^X17>                      ;END DCS PROGRAM
                 0090           CMPREGMSK       WHREG,5$,,4$,,L,            ;TEST MACHINE CHECK REGISTER
                 009C           IFERROR         ,<UTR>,                     ;REGISTER NOT SET CORRECTLY
                 00A2           SKIP                                        ;GO TO NEXT TEST
                 00A9
                 00A9
                 00A9   BEGIN_TEST_DATA
                 00A9
                 00A9   ;-----------------------------------------------------------------------
                 00A9
          0028   00A9   1$:     .WORD   ^X0028                              ;MACHINE CHECK VECTOR
          3FFF   00AB   2$:     .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
      06000000   00AD   3$:     .LONG   ^X06000000                          ;CONTENTS OF BUS ERROR REGISTER
      0F000000   00B1   4$:     .LONG   ^X0F000000                          ;MASK FOR CMPREGMSK PSEUDO OP
      09000000   00B5   5$:     .LONG   ^X09000000                          ;CONTENTS OF MACHINE CHECK REG
      FFFFFFFF   00B9   6$:     .LONG   ^XFFFFFFFF                          ;CONTENTS OF RTEMP4
                 00BD
                 00BD
                 00BD   BEGIN_CPU_DATA
                 0002
```

```
                                0002   296 DCS_DATA
                                0001   297
U 00, 7700,C800,0364,0300,0470,1801   0001   298 ;% 00   NOP
U 01, 7700,8840,5B70,0310,0470,1802   000C   302 ;% 01   R[TEMP4]_0                                   ;ZERO RTEMP4
U 02, 7700,0800,5BE4,0308,4A70,1803   0017   306 ;% 02   VA_R[TEMP2]                                  ;LOAD FFC INTO VA REGISTER
U 03, 7700,C800,5BE4,030C,5480,1804   0022   310 ;% 03   WRITE.PHY,WCTRL/WDR_WB.UR,WB_R[TEMP3]       ;WRITE ADDRESS FFC
U 04, 7700,4800,5BE4,030C,5470,1805   002D   314 ;% 04   WCTRL/WDR_WB.UR,WB_R[TEMP3]                  ;...
U 05, 7700,C800,0364,0300,4470,1806   0038   318 ;% 05   VA_VA+4                                     ;INCREMENT VA TO 1000
U 06, 7700,CF80,5BE4,03D8,5480,1807   0043   322 ;% 06   WRITE.PHY,WCTRL/WDR_WB.UR,WB_R[ZERO],MISC/FORCE.CACHE
                                      004E   326                                                    ;WRITE PARITY ERROR TO 1000
U 07, 7700,CF80,5BE4,03D8,5470,1808   004E   327 ;% 07   WCTRL/WDR_WB.UR,WB_R[ZERO],MISC/FORCE.CACHE
U 08, 7700,8800,5BE4,0308,4870,1809   0059   331 ;% 08   PC_R[TEMP2]                                  ;CAUSE PREFETCH
U 09, 7701,C857,5924,0212,0470,180A   0064   335 ;% 09   R[TEMP4]_XB PC_PC+4                          ;SOURCE XB0
U 0A, 7701,4857,5924,0212,0470,180B   006F   339 ;% 0A   R[TEMP4]_XB PC_PC+4                          ;CAUSE MACHINE CHECK
U 0B, 7300,4800,0364,0300,0470,180C   007A   343 ;% 0B   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
U 0C, 7700,C800,0364,0300,0470,180D   0085   347 ;% 0C   NOP
U 0D, 6700,8800,5BE4,0310,0470,180E   0090   351 ;% 0D   RDM_WB,WB_R[TEMP4]                          ;READ RTEMP4 TO RDM
U 0E, 7300,4800,0364,0300,0470,180F   009B   355 ;% 0E   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
U 0F, 7700,C800,0364,0300,0470,1810   00A6   359 ;% 0F   NOP
U 10, 7700,4800,5BE4,0300,6870,1811   00B1   363 ;% 10   MEMSCAR_R[TEMP0]                             ;READ BUS ERROR REG TO RDM
U 11, 6700,4800,0364,0300,6470,1812   00BC   367 ;% 11   RDM_WB,WCTRL/MEMSCR                          ;...
U 12, 7300,C800,0364,0300,0470,1813   00C7   371 ;% 12   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
U 13, 7700,4800,0364,0300,0470,1814   00D2   375 ;% 13   NOP
U 14, 7700,8800,5BE4,0304,6870,1815   00DD   379 ;% 14   MEMSCAR_R[TEMP1]                             ;READ MACHINE CHECK REG TO RDM
U 15, 6700,C800,0364,0300,6470,1816   00E8   383 ;% 15   RDM_WB,WCTRL/MEMSCR                          ;...
U 16, 7700,8800,5B70,0300,6070,1817   00F3   387 ;% 16   MEMSCR_0                                    ;CLEAR MACHINE CHECK REGISTER
U 17, 7300,4800,0364,0300,0470,1818   00FE   391 ;% 17   NOP,STOP.CLOCK                               ;STOP THE CPU CLOCK
                                      0109   395
                                0109   396 RTEMP_DATA
                                0001   397
                    09000000    0001   398         .LONG   ^X09000000                               ;ADDRESS OF BUS ERROR REGISTER
                    08000000    0005   399         .LONG   ^X08000000                               ;ADDRESS OF MACHINE CHECK REG
                    00000FFC    0009   400         .LONG   ^X00000FFC                               ;ADDRESS FOR VA
                    FFFFFFFF    000D   401         .LONG   ^XFFFFFFFF                               ;CACHE DATA
                    00000000    0011   402         .LONG   ^X00000000                               ;TEST LOCATION
                                0015   403
                                0015   404 END_CPU_DATA
                                00BD   405
                                00BD   406
                                00BD   407 END_TEST_DATA
                                00BD
                                00BD       ;-----------------------------------------------------------------------------
                                00BD   408 ENDTEST
```

```
                        .SBTTL  "TEST   061     TEST IR AND OSR REGISTERS
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST WILL CHECK THE DATA INTEGRITY OF THE INSTRUCTION REGISTER
                ;       (IR) AND OPERAND SPECIFIER REGISTER (OSR) BY WRITING AND READING TEST
                ;       PATTERNS TO BOTH REGISTERS.  THE REGISTERS WILL BE WRITTEN USING THE
                ;       BUT FIELD MICRO-ORDER "IRD1TST".  THEY WILL BE READ BACK TO THE MBUS
                ;       USING "WCTRL/MDR_IR" AND "CCPSL/MDR_OSR.CCBR_BRATST".
                ;
                ; TEST ALGORITHM:
                ;
                ;       SUBTEST #1 & 2:
                ;
                ;       1.  MODIFY DCS MICRO WORDS FOR THIS SUBTEST
                ;       2.  CREATE A TEST LOOP
                ;       3.  LOAD LONLIT WITH TEST PATTERN
                ;       4.  EXECUTE DCS PROGRAM
                ;               1.  DISABLE ^P INTERRUPTS
                ;               2.  SET PSL TO NATIVE MODE AND IPL = 1F
                ;               3.  LOAD 0'S INTO VIRTUAL ADDRESS REGISTER
                ;               4.  WRITE TEST PATTERN INTO LONGLIT REGISTER
                ;               5.  MOVE TEST PATTERN FROM LONLGLIT REGISTER TO CACHE
                ;                   ADDRESS 0
                ;               6.  LOAD 0'S INTO PROGRAM COUNTER
                ;               7.  MOVE TEST PATTERN INTO IR AND OSR REGISTERS
                ;               8.  READ IR OR OSR REGISTER TO MDR REGISTER
                ;               9.  READ MDR REGISTER TO RDM
                ;       5.  TEST VBUS FOR LD OSR (SUBTEST 2 ONLY)
                ;       6.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                ;       7.  IF NO ERROR GO TO STEP 3 FOR REMAINING TEST PATTERNS
                ;
                ;       SUBTEST #3 & 4:
                ;
                ;       1.  CHANGE DCS ADDRESS 03 TO SPECIFY COMPATABILITY MODE
                ;       2.  EXECUTE SAME DCS PROGRAM AS IN SUBTEST #1 & 2 WITH ONLY
                ;           ONE PASS USING TEST PATTERN #5
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       ITERATION       TEST PATTERN
                ;           1               AAAA
                ;           2               5555
                ;           3               3333
                ;           4               0F0F
                ;           5               00FF
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
```

```
                        001E    464 ;       THIS TEST CHECKS THE LOADING AND READING OF THE INSTRUCTION REGISTER
                        001E    465 ;       AND OPERAND SPECIFIER REGISTER.  BOTH OF THESE REGISTERS ARE LOCATED
                        001E    466 ;       IN THE "IRD" GATE ARRAY ON THE DPM MODULE.  THIS SHOULD BE THE MOST
                        001E    467 ;       LIKELY ARRAY TO FAIL.  OTHER GATE ARRAYS TO WATCH ARE:  "MDR" WHICH
                        001E    468 ;       CONTAINS THE DATA PATH TO AND FROM THE IRD; "ADK" WHICH CONTROLS
                        001E    469 ;       THE DBUS SELECT WITHIN THE MDR; AND "CAK" WHICH CONTROLS THE DBUS
                        001E    470 ;       ROTATOR.  IF REPLACING THESE ARRAYS DOES NOT FIX YOUR PROBLEM THEN
                        001E    471 ;       YOUR ONLY HOPE IS PRAYER.
                        001E    472 ;       TO HELP ISOLATE ANY PROBLEMS THE LD OSR SIGNAL IS TESTED ON THE
                        001E    473 ;       VBUS IN SUBTEST 2.
                        001E    474 ;
                        001E    475 ; ASSUMPTIONS:
                        001E    476 ;
                        001E    477 ;       THIS TEST ASSUMES THE WBUS, CACHE, VA, PC, AND MBUS ARE OPERATIONAL.
                        001E    478 ;
                        001E    479 ;
                        001E    480 ; ERROR DESCRIPTION:
                        001E    481 ;
                        001E    482 ;       SUBTEST 2 ONLY:
                        001E    483 ;               FIRST IFERROR:
                        001E    484 ;                       EXPECTED VBUS DATA (LD OSR)
                        001E    485 ;                       RECEIVED VBUS DATA
                        001E    486 ;                       LOOP COUNT I
                        001E    487 ;
                        001E    488 ;               SECOND IFERROR:
                        001E    489 ;                       EXPECTED OSR DATA
                        001E    490 ;                       RECEIVED OSR DATA
                        001E    491 ;                       LOOP COUNT I
                        001E    492 ;
                        001E    493 ;       ALL OTHER SUBTESTS:
                        001E    494 ;               EXPECTED IR/OSR DATA
                        001E    495 ;               RECEIVED IR/OSR DATA
                        001E    496 ;               LOOP COUNT I (SUBTEST 1)
                        001E    497 ;
                        001E    498 ;--
                        001E    499 ;*****************************************************************************
                        001E    500
                        001E    501         SUBTEST                                         ;SUBTEST #1
                        0021
                        0021        ;/////////////////////////////////////////////////////////////////////////////
                        0021    502 ;+
                        0021    503 ;SUBTEST #1
                        0021    504 ;
                        0021    505 ;       TEST INSTRUCTION REGISTER
                        0021    506 ;-
                        0021    507         INITIALIZE                                      ;INIT CPU
                        0023    508         LOADDCS         10$,,03,1                       ;SET NATIVE MODE MICRO WORD
                        002A    509         LOOP            I,1,5                           ;CREATE A TEST LOOP
                        0031    510         LDFIELD         2$,I,W,1$,31,62,LON             ;TEST PATTERN TO MICRO WORD
                        0042    511         LOADDCS         1$,,05,1                        ;TEST PATTERN TO DCS
                        0049    512         ERRLOOP                                         ;ERROR LOOP
                        004B    513         FETCH           0                               ;START DCS PROGRAM
                        0050    514         BRSTCLK         <^X0C>                          ;END DCS PROGRAM
                        0054    515         CMPREG          WHREG,3$,I,L,                   ;COMPARE RESULTS
                        005D    516         IFERROR         ,<<IRD><MDR><ADK><CAK>>,<DPM>
```

```
0067            ENDLOOP         I                       ;END TEST LOOP
006A
006A            SUBTEST                                 ;SUBTEST #2
006D
006D        ;////////////////////////////////////////////////////////////////////////////
006D        ;+
006D        ;SUBTEST #2
006D        ;
006D        ;       TEST OPERAND SPECIFIER REGISTER
006D        ;-
006D            INITIALIZE                              ;INIT CPU
006F            LOADDCS         4$,,<^X0A>,1            ;READ OSR MICRO WORD
0076            LOOP            I,1,5                   ;CREATE A TEST LOOP
007D            LDFIELD         2$,I,W,1$,31,62,LON     ;TEST PATTERN TO MICRO WORD
008E            LOADDCS         1$,,05,1                ;TEST PATTERN TO DCS
0095            ERRLOOP                                 ;ERROR LOOP
0097            FETCH           0                       ;START DCS PROGRAM
009C            BRSTCLK         <^X0C>                  ;END DCS PROGRAM
00A0            CMPVBUS         11$                     ;TEST LD OSR SIGNAL
00A5            IFERROR         ,<MSQ>,<DPM>            ;SIGNAL NOT PRESENT
00AC            CMPREG          WHREG,5$,I,L,           ;COMPARE RESULTS
00B5            IFERROR         ,<<IRD><MDR><ADK><CAK>>,<<DPM>>
00BF            ENDLOOP         I                       ;END TEST LOOP
00C2
00C2            SUBTEST                                 ;SUBTEST #3
00C5
00C5        ;////////////////////////////////////////////////////////////////////////////
00C5        ;+
00C5        ;SUBTEST #3
00C5        ;
00C5        ;       TEST OPERAND SPECIFIER REGISTER IN COMPATABILITY MODE
00C5        ;-
00C5            INITIALIZE                              ;INIT CPU
00C7            LOADDCS         8$,,03,1                ;SET CM BIT MICRO WORD
00CE            ERRLOOP                                 ;ERROR LOOP
00D0            FETCH           0                       ;START DCS PROGRAM
00D5            BRSTCLK         <^X0C>                  ;END DCS PROGRAM
00D9            CMPREG          WHREG,6$,,L,            ;COMPARE RESULTS
00E2            IFERROR         ,<<IRD><MDR><ADK><CAK>>,<DPM>
00EC
00EC            SUBTEST                                 ;SUBTEST #4
00EF
00EF        ;////////////////////////////////////////////////////////////////////////////
00EF        ;+
00EF        ;SUBTEST #4
00EF        ;
00EF        ;       TEST INSTRUCTION REGISTER IN COMPATABILITY MODE
00EF        ;-
00EF            INITIALIZE                              ;INIT CPU
00F1            LOADDCS         9$,,<^X0A>,1            ;READ IR MICRO WORD
00F8            ERRLOOP                                 ;ERROR LOOP
00FA            FETCH           0                       ;START DCS PROGRAM
00FF            BRSTCLK         <^X0C>                  ;END DCS PROGRAM
0103            CMPREG          WHREG,7$,,L,            ;COMPARE RESULTS
010C            IFERROR         ,<<IRD><MDR><ADK><CAK>>,<DPM>
```

```
                                  0116          SKIP                                    ;GO TO NEXT TEST
                                  011D
                                  011D  BEGIN_TEST_DATA
                                  011D
                                  011D  ;-----------------------------------------------------------------------
                                  011D
                                  011D  1$:
U 05, 7700,F800,7FFF,FFFF,8470,1806 011D  ;% 05   LONLIT_[0]                              ;TEST PATTERN TO LONGLIT
                                  0128
                            AAAA  0128  2$:     .WORD   ^XAAAA                          ;TEST PATTERNS
                            5555  012A          .WORD   ^X5555
                            3333  012C          .WORD   ^X3333
                            0F0F  012E          .WORD   ^X0F0F
                            00FF  0130          .WORD   ^X00FF
                                  0132
                        000000AA  0132  3$:     .LONG   ^X000000AA                      ;RESULTS SUBTEST #1
                        00000055  0136          .LONG   ^X00000055
                        00000033  013A          .LONG   ^X00000033
                        0000000F  013E          .LONG   ^X0000000F
                        000000FF  0142          .LONG   ^X000000FF
                                  0146  4$:
U 0A, 7700,C800,0364,0300,5E70,180B 0146  ;% 0A   CCPSL/MDR_OSR.CCBR_BRATST               ;READ OSR TO MDR REG
                                  0151
                        000000AA  0151  5$:     .LONG   ^X000000AA                      ;RESULTS SUBTEST #2
                        00000055  0155          .LONG   ^X00000055
                        00000033  0159          .LONG   ^X00000033
                        0000000F  015D          .LONG   ^X0000000F
                        00000000  0161          .LONG   ^X00000000
                                  0165
                        000000FF  0165  6$:     .LONG   ^X000000FF                      ;RESULTS SUBTEST #3
                                  0169
                        00000000  0169  7$:     .LONG   ^X00000000                      ;RESULTS SUBTEST #4
                                  016D
                                  016D  8$:
U 03, 7700,0800,5BF4,030C,0070,1804 016D  ;% 03   PSL_R[TEMP3]                            ;SET CM BIT IN PSL
                                  0178
                                  0178  9$:
U 0A, 7700,4800,0364,0300,5670,180B 0178  ;% 0A   WCTRL/MDR_IR                            ;READ IR TO MDR REGISTER
                                  0183
                                  0183  10$:
U 03, 7700,4800,5BF4,0308,0070,1804 0183  ;% 03   PSL_R[TEMP2]                            ;SET NATIVE MODE IN PSL
                                  018E
                              01  018E  11$:    .BYTE   ^X1                             ;LD OSR SIGNAL
                                  018F          VBUSDATA        <^X1E>,0
                                  0190
                                  0190  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801 0001  ;% 00   NOP
U 01, 7700,4800,5BE4,0300,2470,1802 000C  ;% 01   WCTRL/LOADCRAR,WB_R[TEMP0]              ;DISABLE ^P INTERRUPT
U 02, 7700,0800,5BE4,0304,2C70,1803 0017  ;% 02   WCTRL/CONWRITE,WB_R[TEMP1]              ;...
U 03, 7700,4800,5BF4,0308,0070,1804 0022  ;% 03   PSL_R[TEMP2]                            ;SET NATIVE MODE
U 04, 7700,4800,5BE4,03D8,4A70,1805 002D  ;% 04   VA_R[ZERO]                              ;0'S TO VA REGIATER
U 05, 7700,F800,7FFF,FFFF,8470,1806 0038  ;% 05   LONLIT_[0]                              ;TEST PATTERN TO LONGLIT REG
```

```
U 06. 7701.8800.5BE4.02D4.5590.1807  0043   652 ;% 06   WRITE.LONG R[LONLIT]                          ;TEST PATTERN TO CACHE ADD 0
U 07. 7700.C800.5BE4.03D4.5470.1808  004E   656 ;% 07   WB_R[LONLIT],WCTRL/WDR_WB.UR                ;HOLD DATA FOR EXTRA CYCLE
U 08. 7700.C800.5BE4.03D8.4870.1809  0059   660 ;% 08   PC_R[ZERO]                                  ;0'S TO PROGRAM COUNTER
U 09. 7600.4800.0364.1700.0470.180A  0064   664 ;% 09   BUT/IRD1TST,STROBE.V.BUS                   ;LOAD IR AND OSR REGISTERS
U 0A. 7700.4800.0364.0300.5670.180B  006F   668 ;% 0A   WCTRL/MDR_IR                               ;READ IR TO MDR REGISTER
U 0B. 6700.0812.5924.0300.0470.180C  007A   672 ;% 0B   RDM_WB,WB_M[MDR]                           ;RDM GETS MDR REGISTER
U 0C. 7300.C800.0364.0300.0470.180D  0085   676 ;% 0C   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
                                     0090   680
                                     0090   681 RTEMP_DATA
                                     0001   682
                           00C00000  0001   683         .LONG   ^X00C00000                       ;DISABLE ^P INTERRUPT
                           00400000  0005   684         .LONG   ^X00400000                       ;...
                           001F0000  0009   685         .LONG   ^X001F0000                       ;SET NATIVE MODE
                           801F0000  000D   686         .LONG   ^X801F0000                       ;SET CM BIT IN PSL
                                     0011   687
                                     0011   688 END_CPU_DATA
                                     0190   689
                                     0190   690 END_TEST_DATA
                                     0190
                                     0190       ;-----------------------------------------------------------------------------
                                     0190   691 ENDTEST
```

```
0024          .SBTTL  "TEST   062     COMPATIBILITY MODE ADDRESS TEST
0024    693 ;*****************************************************************************
0024    694 ;++
0024    695 ;
0024    696 ; FUNCTIONAL DESCRIPTION
0024    697 ;
0024    698 ;       This test checks the MA output multiplexer on the two high order ADD
0024    699 ;       gate arrays to insure that their outputs remain zero when the computer
0024    700 ;       is generating addresses in compatibility mode. This test is
0024    701 ;       implemented by using the VA register to insure that the upper order
0024    702 ;       sixteen bits of address remain zero.
0024    703 ;
0024    704 ; TEST ALGORITHM
0024    705 ;
0024    706 ;       1. Initialize the CPU.
0024    707 ;       2. Load the longlit required to place the CPU into compatibility mode
0024    708 ;          into the RDM D register.
0024    709 ;       3. Execute the following micro-instructions to activate compatibility
0024    710 ;          mode:
0024    711 ;                1. NOP
0024    712 ;                2. WB_RDM,CCPSL/PSL_WB.CCBR_ALUS
0024    713 ;                3. NOP,STOP.CLOCK
0024    714 ;       4. Load the test data pattern into the RDM D register.
0024    715 ;       5. Execute the micro-instuctions:
0024    716 ;                4. Load the VA with the pattern from the RDM D register.
0024    717 ;                5. Read the VA back to the RDM H register.
0024    718 ;       6. Compare the H register withthe correct pattern from the RDM.
0024    719 ;       7. Signal an error if the two are different.
0024    720 ;       8. Test completed.
0024    721 ;
0024    722 ; DATA PATTERNS
0024    723 ;
0024    724 ;       The following data pattern is sufficient to test compatibility mode
0024    725 ;       addresses.
0024    726 ;
0024    727 ;               INPUT                           RESULT
0024    728 ;
0024    729 ;          F F F F F F F F                 0 0 0 0 F F F F
0024    730 ;
0024    731 ; LOGIC DESCRIPTION
0024    732 ;
0024    733 ;       This test insures that the MA output multiplexer on the ADD gate
0024    734 ;       array is disabled for the two high order ADD bit slices when the
0024    735 ;       CPU is in compatibility mode.
0024    736 ;
0024    737 ; ASSUMPTIONS
0024    738 ;
0024    739 ;       This test assumes that the DPM and WBUS are functional, that the CPU
0024    740 ;       can be placed into compatibility mode without error, and that the VA
0024    741 ;       register on the ADD gate arrays has been successfully tested in native
0024    742 ;       mode prior to running this test.
0024    743 ;
0024    744 ; ERROR DESCRIPTION
0024    745 ;
0024    746 ;       If the comparison between the expected and received data indicates
```

```
                                0024   747 ;       that the problem is within a particular byte of the address, the ADD
                                0024   748 ;       gate array associated with that byte should be replaced. If more than
                                0024   749 ;       one byte of the address is bad, ADD chips should be replaced until
                                0024   750 ;       either the problem goes away or all four chips have been replaced
                                0024   751 ;       with no effect. If the problem persists after all four ADD chips have
                                0024   752 ;       been replaced, or data other that 0 or F hex appears in the received
                                0024   753 ;       data, then the PRK gate array is suspect and should be replaced.
                                0024   754 ;
                                0024   755 ;       On error this test will call out the ADD and PRK gate arrays on the
                                0024   756 ;       MIC module. The following information will also by typed out:
                                0024   757 ;
                                0024   758 ;       Expected Data:
                                0024   759 ;       Received Data:
                                0024   760 ;
                                0024   761 ;--
                                0024   762 ;*******************************************************************************
                                0024   763 ;///////////////////////////////////////////////////////////////////////////////
                                0024   764 ;
                                0024   765 ; This code puts the CPU into compatibility mode.
                                0024   766 ;
                                0024   767         INITIALIZE
                                0026   768         LOADDCS         3$,,0,4
                                002D   769         FETCH           0
                                0032   770         BRSTCLK         3
                                0036   771 ;
                                0036   772 ; This code checks to see that the ADD chip is generating compatiblity mode
                                0036   773 ; addresses.
                                0036   774 ;
                                0036   775         LOADDCS         4$,,0,4
                                003D   776         LOADREG         WDREG,1$,,LONG
                                0045   777         ERRLOOP
                                0047   778         FETCH           0
                                004C   779         BRSTCLK         3
                                0050   780         CMPREG          WHREG,2$,,LONG
                                0059   781         IFERROR         ,<<ADD>,<PRK>>,
                                0060   782 ;
                                0060   783 ; This code returns the CPU to native mode.
                                0060   784 ;
                                0060   785         LOADDCS         5$,,0,4
                                0067   786         FETCH           0
                                006C   787         BRSTCLK         3
                                0070   788         SKIP
                                0077   789
                                0077   790 BEGIN_TEST_DATA
                                0077
                                0077       ;-----------------------------------------------------------------------
                                0077   791
                    FFFFFFFF    0077   792 1$:     .LONG   ^X FFFFFFFF     ; Input VA data
                                007B   793
                    0000FFFF    007B   794 2$:     .LONG   ^X 0000FFFF     ; Correct VA output data
                                007F   795
                                007F   796 3$:
U 00, 7700,C800,0364,0300,0470,1801 007F   797 ;% 00   NOP                             ; Places CPU into comp.mode
U 01, 7700,3800,3FFF,FFFF,8470,1802 008A   801 ;% 01   LONLIT_[80000000]
U 02, 7700,C800,5BE4,03D4,0070,1803 0095   805 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS
```

```
U 03. 7300.C800.0364.0300.0470.1804  00A0   809 ;% 03   STOP.CLOCK
                                     00AB   813
                                     00AB   814 4$:
U 00. 7700.C800.0364.0300.0470.1801  00AB   815 ;% 00   NOP                                  ; Examines generated addresses
U 01. 5700.C800.0364.0300.4A70.1802  00B6   819 ;% 01   VA_RDM
U 02. 6700.081B.0024.03D8.0470.1803  00C1   823 ;% 02   RDM_VA
U 03. 7300.C800.0364.0300.0470.1804  00CC   827 ;% 03   STOP.CLOCK
                                     00D7   831
                                     00D7   832 5$:
U 00. 7700.C800.0364.0300.0470.1801  00D7   833 ;% 00   NOP                                  ; Places CPU into native mode
U 01. 7700.7800.7FFF.FFFF.8470.1802  00E2   837 ;% 01   LONLIT_[0]
U 02. 7700.C800.5BE4.03D4.0070.1803  00ED   841 ;% 02   WB_R[LONLIT],CCPSL/PSL_WB.CCBR_ALUS
U 03. 7300.C800.0364.0300.0470.1804  00F8   845 ;% 03   STOP.CLOCK
                                     0103   849
                                     0103   850 END_TEST_DATA
                                     0103
                                     0103       ;-------------------------------------------------------------------------
                                     0103   851
                                     0103   852 ENDTEST
```

```
0021          .SBTTL  "TEST   063     MEMORY ADDRESS REGISTER TEST
0021    854 ;*****************************************************************************
0021    855 ;++
0021    856 ;
0021    857 ; FUNCTIONAL DESCRIPTION
0021    858 ;
0021    859 ;       This test examines the memory address register (MA).
0021    860 ;
0021    861 ;       The MA is a latch on the outputs of the address chips which holds an
0021    862 ;       address from the time a memory reference is initiated until it has
0021    863 ;       been tested for micro-traps and latched in the MDR chips. Therefore,
0021    864 ;       the VA may be changed in the same micro-step which specifies a memory
0021    865 ;       bus function.
0021    866 ;
0021    867 ;
0021    868 ; TEST ALGORITHM
0021    869 ;
0021    870 ;       1.  LOAD FIRST SET OF MICROCODE INTO DCS
0021    871 ;       2.  EXECUTE DCS PROGRAM
0021    872 ;               a.  DISABLE THE CMI
0021    873 ;               b.  ENABLE MEMORY MANAGEMENT
0021    874 ;               c.  ENABLE TRANSLATION BUFFER
0021    875 ;               d.  ENABLE CACHE
0021    876 ;               e.  WRITE PTE TO TRANSLATION BUFFER
0021    877 ;       3.  LOAD SECOND SET OF MICROCODE INTO DCS
0021    878 ;       4.  CREATE A TEST LOOP OF 2
0021    879 ;       5.  EXECUTE DCS PROGRAM
0021    880 ;               a.  LOAD VA WITH 55555555
0021    881 ;               b.  WRITE ZEROS TO THIS LOCATION IN CACHE
0021    882 ;               c.  WRITE 5'S TO THIS LOCATION WHILE CHANGING THE VA
0021    883 ;               d.  SAVE VA IN RTEMP2
0021    884 ;               e.  RELOAD VA WITH 55555555
0021    885 ;               f.  READ THIS LOCATION BACK TO THE RDM
0021    886 ;       6.  TEST THE DATA READ
0021    887 ;       7.  IF CORRECT CHANGE MICRO CODE TO READ RTEMP2 ON SECOND LOOP
0021    888 ;       8.  REEXECUTE THE MICRO CODE
0021    889 ;       9.  TEST RTEMP2 FOR CORRECT VA ADDRESS
0021    890 ;      10.  GO TO THE NEXT TEST
0021    891 ;
0021    892 ; TEST DATA
0021    893 ;
0021    894 ;       The following data is used to implement this test:
0021    895 ;
0021    896 ;       1. Test Address and Memory Data    =  5 5 5 5 5 5 5 5
0021    897 ;       2. Complement of the Test Address  =  A A A A A A A A
0021    898 ;
0021    899 ; LOGIC DESCRIPTION
0021    900 ;
0021    901 ;       TBS
0021    902 ;
0021    903 ; ASSUMPTIONS
0021    904 ;
0021    905 ;       This test assumes that the WBUS, MBUS and the RTEMP dual port
0021    906 ;       temporaries on the DPM have been fully tested before this test is run
0021    907 ;
```

```
                        0021    908 ; ERROR DESCRIPTION
                        0021    909 ;
                        0021    910 ;       On error this test will type out the ADD and PRK gate arrays on the
                        0021    911 ;       MIC module. The following information will also be typed out:
                        0021    912 ;
                        0021    913 ;       Expected Data
                        0021    914 ;       Received Data
                        0021    915 ;       Loop Count
                        0021    916 ;
                        0021    917 ;--
                        0021    918 ;*******************************************************************************
                        0021    919 ;///////////////////////////////////////////////////////////////////////////////
                        0021    920
                        0021    921         INITIALIZE                              ; INITIALIZE THE CPU
                        0023    922         LOADDCS         2$,,0,<^X15>            ; MICRO-CODE FOR CMI/MME
                        002A    923         FETCH           0                       ; CMI = OFF MME,TB,CACHE = ON
                        002F    924         BRSTCLK         <^X14>                  ; AND LOAD TEST PTE
                        0033    925         LOADDCS         3$,,0,<^X0D>            ; LOAD TEST U-CODE
                        003A    926         LOOP            I,1,2                   ; LOOP FOR MEMORY AND VA READS
                        0041    927         ERRLOOP                                 ; ERROR LOOP
                        0043    928         FETCH           0                       ; TEST FOR LATCHED MA ALTERING
                        0048    929         BRSTCLK         <^X0C>                  ; VA WHILE WRITING TO CACHE
                        004C    930         CMPREG          WHREG,1$,I,LONG         ; I = 1  MEMORY ADDRESS OK?
                        0055    931         IFERROR         ,<<ADD><PRK>>,          ; I = 2  VA GET CHANGED?
                        005C    932         LOADDCS         4$,,<^X0B>,1            ; U-CODE FOR VA READ
                        0063    933         ENDLOOP         I                       ; CONTINUE LOOP
                        0066    934         SKIP                                    ; END OF TEST
                        006D    935
                        006D    936 BEGIN_TEST_DATA
                        006D            ;-------------------------------------------------------------------------
                        006D    937
                55555555 006D   938 1$:     .LONG   ^X 55555555                     ; FIRST ADDRESS AND MEMORY DATA
                AAAAAAAA 0071   939         .LONG   ^X AAAAAAAA                     ; SECOND ADDRESS
                        0075    940
                        0075    941 2$:
U 00, 7700,C800,0364,0300,0470,1801 0075    942 ;% 00   NOP
U 01, 7700,3800,78FF,FFFF,8470,1802 0080    946 ;% 01   LONLIT_[0E000000]           ; DISABLE CMI
U 02, 7700,4800,5BE4,03D4,6870,1803 008B    950 ;% 02   MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804 0096    954 ;% 03   LONLIT_[01000000]
U 04, 7700,C800,5BE4,03D4,6070,1805 00A1    958 ;% 04   MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806 00AC    962 ;% 05   LONLIT_[0]                  ; ENABLE MM
U 06, 7700,C800,5BE4,03D4,6870,1807 00B7    966 ;% 06   MEMSCAR_R[LONLIT]
U 07, 7700,3800,7F7F,FFFF,8470,1808 00C2    970 ;% 07   LONLIT_[01000000]
U 08, 7700,C800,5BE4,03D4,6070,1809 00CD    974 ;% 08   MEMSCR_R[LONLIT]
U 09, 7700,F800,7E7F,FFFF,8470,180A 00D8    978 ;% 09   LONLIT_[03000000]           ; ENABLE TB
U 0A, 7700,C800,5BE4,03D4,6870,180B 00E3    982 ;% 0A   MEMSCAR_R[LONLIT]
U 0B, 7700,F800,7FFF,FFFF,8470,180C 00EE    986 ;% 0B   LONLIT_[0]
U 0C, 7700,4800,5BE4,03D4,6070,180D 00F9    990 ;% 0C   MEMSCR_R[LONLIT]
U 0D, 7700,7800,7CFF,FFFF,8470,180E 0104    994 ;% 0D   LONLIT_[06000000]           ; ENABLE CACHE
U 0E, 7700,4800,5BE4,03D4,6870,180F 010F    998 ;% 0E   MEMSCAR_R[LONLIT]
U 0F, 7700,7800,7FFF,FFFF,8470,1810 011A   1002 ;% 0F   LONLIT_[0]
U 10, 7700,C800,5BE4,03D4,6070,1811 0125   1006 ;% 10   MEMSCR_R[LONLIT]
U 11, 7700,4800,5BE4,0300,4A70,1812 0130   1010 ;% 11   VA_R[TEMP0]                 ; STORE PTE FOR TEST
U 12, 7700,3800,7FFF,FF5B,8470,1813 013B   1014 ;% 12   LONLIT_[00000148]
```

```
U 13, 7700,C81B,5BE4,03D4,5070,1814  0146  1018 ;% 13   TB_R[LONLIT]
U 14, 7300,C800,0364,0300,0470,1815  0151  1022 ;% 14   STOP.CLOCK
                                     015C  1026
                                     015C  1027 3$:
U 00, 7700,C800,0364,0300,0470,1801  015C  1028 ;% 00   NOP                                 ; WAIT FOR A STABLE CLOCK
U 01, 7700,C800,5BE4,0300,4A70,1802  0167  1032 ;% 01   VA_R[TEMP0]                         ; LOAD THE VA WITH THE FIRST ADDRESS
U 02, 7700,C800,5BE4,03D8,5C70,1803  0172  1036 ;% 02   WDR_R[ZERO]                        ; LOAD WDR WITH ZEROS
U 03, 7701,8800,0364,0200,0590,1804  017D  1040 ;% 03   BUS/WRITE.LNG,SIZE[LONG]           ; CLEAR THE TEST LOCATION
U 04, 7700,C800,5BE4,03D8,5C70,1805  0188  1044 ;% 04   WDR_R[ZERO]                        ; HOLD WDR FOR COMPLETION OF CYCLE
U 05, 7700,C800,5BE4,0300,5C70,1806  0193  1048 ;% 05   WDR_R[TEMP0]                        ; LOAD THE WDR WITH THE MEMORY DATA
U 06, 7701,C800,5BE4,0204,4B90,1807  019E  1052 ;% 06   BUS/WRITE.LNG,SIZE[LONG],VA_R[TEMP1]
                                     01A9  1056                                          ; START MEMORY CYCLE AND CHANGE THE VA
U 07, 7700,4800,5BE4,0300,5C70,1808  01A9  1057 ;% 07   WDR_R[TEMP0]                        ; HOLD WDR FOR COMPLETION OF CYCLE
U 08, 7700,085B,5924,0308,0470,1809  01B4  1061 ;% 08   R[TEMP2]_VA                         ; STORE THE SECOND VA ADDRESS
U 09, 7700,4800,5BE4,0300,4A70,180A  01BF  1065 ;% 09   VA_R[TEMP0]                         ; REASSIGN VA TO FIRST ADDRESS
U 0A, 7700,4800,0364,0300,0510,180B  01CA  1069 ;% 0A   READ.LONG                           ; READ MEMORY
U 0B, 6700,0812,5924,0300,0470,180C  01D5  1073 ;% 0B   RDM_M[MDR]                          ; SEND MEMORY DATA TO THE RDM
U 0C, 7300,C800,0364,0300,0470,180D  01E0  1077 ;% 0C   STOP.CLOCK                          ; STOP
                                     01EB  1081
                                     01EB  1082 4$:
U 0B, 6700,0800,5BE4,0308,0470,180C  01EB  1083 ;% 0B   RDM_R[TEMP2]                        ; SEND SECOND VA ADDRESS TO RDM
                                     01F6  1087
                                     01F6  1088 BEGIN_CPU_DATA
                                     0002  1089
                                     0002  1090 RTEMP_DATA
                                     0001  1091
55555555                             0001  1092         .LONG   ^X 55555555      ; R[TEMP0]
AAAAAAAA                             0005  1093         .LONG   ^X AAAAAAAA      ; R[TEMP1]
                                     0009  1094
                                     0009  1095 END_CPU_DATA
                                     01F6  1096
                                     01F6  1097 END_TEST_DATA
                                     01F6
                                     01F6       ;-------------------------------------------------------------------------
                                     01F6  1098
                                     01F6  1099 ENDTEST
```

```
0030        .SBTTL  "TEST   064     PAGE BOUNDARY TEST FOR BYTE/WORD/LONG DTYPE
0030  1101 ;*******************************************************************************
0030  1102 ;++
0030  1103 ;
0030  1104 ; FUNCTIONAL DESCRIPTION
0030  1105 ;
0030  1106 ;       This test will examine the PAGE BOUNDARY signal being generated by the
0030  1107 ;       ADD gate arrays to insure that it is correct for byte, word, and long-
0030  1108 ;       word data types.
0030  1109 ;
0030  1110 ; TEST ALGORITHM
0030  1111 ;
0030  1112 ;       Three subtests are used to implement this test. Each subtest requires
0030  1113 ;       two data tables, one for VA addresses and the other for data types.
0030  1114 ;       These will be used with a write memory micro-instruction in order to
0030  1115 ;       examine the PAGE BOUNDARY signal.
0030  1116 ;
0030  1117 ;       Subtest 1
0030  1118 ;
0030  1119 ;       This test examines PAGE BOUNDARY active for VA<8:2> = 1 and unaligned
0030  1120 ;       data. This subtest should generate a PAGE BOUNDARY micro-trap.
0030  1121 ;
0030  1122 ;        1. Prepare a loop of 4 iterations indexed by I in order to test all
0030  1123 ;           possible combinations of VA addresses and data types that can
0030  1124 ;           produce a page boundary micro-trap.
0030  1125 ;        2. Load an address into the RDM D register from Table 1 indexed by I.
0030  1126 ;        3. Load and execute the following micro-code:
0030  1127 ;                           00 NOP
0030  1128 ;                           01 VA_RDM
0030  1129 ;                           02 LONLIT_[00000148]
0030  1130 ;                           03 TB_R[LONLIT]
0030  1131 ;                           04 STOP.CLOCK
0030  1132 ;        4. Load the DTYPE field of a WRITE R[TEMP0] micro-instruction with a
0030  1133 ;           data type chosen from Table 2 as indexed by I.
0030  1134 ;        5. Load and exeucte the following micro-code:
0030  1135 ;                           00 NOP
0030  1136 ;                           01 WRITE R[TEMP0],STROBE.V.BUS
0030  1137 ;                           02 STOP.CLOCK
0030  1138 ;        6. Examine the V-BUS for UTRAP-L.
0030  1139 ;        7. Signal an error if UTRAP-L is not present.
0030  1140 ;        8. Examine the V-BUS for a micro-vector of 27 hex.
0030  1141 ;        9. Signal an error if the micro-vector is unequal to 27 hex.
0030  1142 ;       10. Examine the CS-ADDRESS lines for an address of 27 hex.
0030  1143 ;       11. Signal an error if the micro-address does not equal 27 hex.
0030  1144 ;       12. If there were no errors, continue looping through steps 2-11.
0030  1145 ;
0030  1146 ;       Subtest 2
0030  1147 ;
0030  1148 ;       This subtest makes sure PAGE BOUNDARY does not assert when VA<8:2>
0030  1149 ;       are all ones and there in no unaligned data. This subtest should not
0030  1150 ;       generate any micro-traps.
0030  1151 ;
0030  1152 ;       13. Preppare a loop of 8 iterations indexed by I. This loop will test
0030  1153 ;           all possible combinations of VA addresses and data types that do
0030  1154 ;           not produce PAGE BOUNDARY or UNALIGNED DATA.
```

```
0030  1155 ;       14. Load an address into the RDM D register from Table 3 indexed by I.
0030  1156 ;       15. Load and execute the following micro-code:
0030  1157 ;                       00 NOP
0030  1158 ;                       01 VA_RDM
0030  1159 ;                       02 LONLIT [00000148]
0030  1160 ;                       03 TB_R[LONLIT]
0030  1161 ;                       04 STOP.CLOCK
0030  1162 ;       16. Load the DTYPE field of a WRITE R[TEMP0] micro-instruction with a
0030  1163 ;           data type chosen from Table 4 as indexed by I.
0030  1164 ;       17. Load and exeucte the following micro-code:
0030  1165 ;                       00 NOP
0030  1166 ;                       01 WRITE R[TEMP0],STROBE.V.BUS
0030  1167 ;                       02 STOP.CLOCK
0030  1168 ;       18. Examine the V-BUS for UTRAP H.
0030  1169 ;       19. Signal an error if UTRAP is asserted.
0030  1170 ;       20. Compare the CS ADDRESS lines with 1802 hex.
0030  1171 ;       21. Signal an error if the two are not equal.
0030  1172 ;       22. If there were no errors, continue looping through steps 14-22.
0030  1173 ;
0030  1174 ;       Subtest 3
0030  1175 ;
0030  1176 ;       This subtest makes sure PAGE BOUNDARY does not assert when VA<8:2> are
0030  1177 ;       not all ones and unaligned data is being written to cache. This subtest
0030  1178 ;       should generate an UNALIGNED DATA WRITE micro-trap.
0030  1179 ;
0030  1180 ;       23. Prepare another loop of 7 iterations indexed by I. This will test
0030  1181 ;           all possible combinations of VA addresses and data types that do
0030  1182 ;           not produce a page boundary micro-trap but will produce an
0030  1183 ;           unaligned data micro-trap.
0030  1184 ;       24. Reload the micro-code in step 3 to load the VA and validate the TB.
0030  1185 ;       25. Load an address from Table 5 indexed by I into the RDM D register.
0030  1186 ;       26. Execute the micro-code in DCS.
0030  1187 ;       27. Load the DTYPE field of a WRITE R[TEMP0] micro-instruction with a
0030  1188 ;           data type chosen from Table 6 indexed by I.
0030  1189 ;       28. Reload the micro-code from step 5.
0030  1190 ;       29. Execute the micro-code, thereby generating an UNALIGNED DATA micro-
0030  1191 ;           trap.
0030  1192 ;       30. Examine the V-BUS for UTRAP-L.
0030  1193 ;       31. If UTRAP is not asserted signal an error.
0030  1194 ;       32. Examine the V-BUS for a micro-vector of 25 hex.
0030  1195 ;       33. If the micro-vector does not equal 25 hex, signal an error.
0030  1196 ;       34. Compare the CS ADDRESS with 25 hex.
0030  1197 ;       35. If the cs address does not equal 25 hex, signal an error.
0030  1198 ;       36. If there were no errors, continue looping through steps 24-36.
0030  1199 ;       37. End of test.
0030  1200 ;
0030  1201 ; DATA PATTERNS
0030  1202 ;
0030  1203 ;       TBS
0030  1204 ;
0030  1205 ; LOGIC DESCRIPTION
0030  1206 ;
0030  1207 ;       TBS
0030  1208 ;
0030  1209 ; ASSUMPTIONS
```

```
                     0030  1210 ;
                     0030  1211 ;       TBS
                     0030  1212 ;
                     0030  1213 ; ERROR DESCRIPTION
                     0030  1214 ;
                     0030  1215 ;       TBS
                     0030  1216 ;
                     0030  1217 ;--
                     0030  1218 ;*******************************************************************************
                     0030  1219
                     0030  1220         SUBTEST         1
                     0033
                     0033       ;////////////////////////////////////////////////////////////////////////////////
                     0033  1221         INITIALIZE
                     0035  1222         LOADDCS         15$,,0,18
                     003C  1223         FETCH           0
                     0041  1224         BRSTCLK         <^X11>
                     0045  1225         LOADDCS         16$,,0,6
                     004C  1226         LOOP            I,1,4
                     0053  1227         LOADREG         WDREG,1$,I,LONG
                     005B  1228         LDFIELD         2$,I,BYTE,17$,40,41
                     006C  1229         LOADDCS         17$,,4,1
                     0073  1230         EXPUTRAP
                     0075  1231         ERRLOOP
                     0077  1232         FETCH           0
                     007C  1233         BRSTCLK         5
                     0080  1234         CMPVBUS         3$
                     0085  1235         IFERROR         ,<<ADD><UTR>>,<<MIC><DPM>>
                     008E  1236         CMPREGMSK       CSTRP,5$,,6$,,WORD
                     009A  1237         IFERROR         ,<<ADD><UTR>>,
                     00A1  1238         ENDLOOP         I
                     00A4  1239
                     00A4  1240         SUBTEST         2
                     00A7
                     00A7       ;////////////////////////////////////////////////////////////////////////////////
                     00A7  1241         LOOP            I,1,8
                     00AE  1242         LOADREG         WDREG,7$,I,LONG
                     00B6  1243         LDFIELD         8$,I,BYTE,17$,40,41
                     00C7  1244         LOADDCS         17$,,4,1
                     00CE  1245         EXPUTRAP
                     00D0  1246         ERRLOOP
                     00D2  1247         FETCH           0
                     00D7  1248         BRSTCLK         5
                     00DB  1249         CMPVBUS         9$
                     00E0  1250         IFERROR         ,<<ADD><UTR>>,<<MIC><DPM>>
                     00E9  1251         CMPREGMSK       CSTRP,10$,,6$,,WORD
                     00F5  1252         IFERROR         ,<<ADD><UTR>>,
                     00FC  1253         ENDLOOP         I
                     00FF  1254
                     00FF  1255         SUBTEST         3
                     0102
                     0102       ;////////////////////////////////////////////////////////////////////////////////
                     0102  1256         LOOP            I,1,7
                     0109  1257         LOADREG         WDREG,11$,I,LONG
                     0111  1258         LDFIELD         12$,,BYTE,17$,40,41
```

```
                    0122  1259          LOADDCS         17$,,4,1
                    0129  1260          EXPUTRAP
                    012B  1261          ERRLOOP
                    012D  1262          FETCH           0
                    0132  1263          BRSTCLK         5
                    0136  1264          CMPVBUS         13$
                    013B  1265          IFERROR         ,<<ADD><UTR>>,<<MIC><DPM>>
                    0144  1266          CMPREGMSK       CSTRP,14$,,6$,,WORD
                    0150  1267          IFERROR         ,<<ADD><UTR>>,
                    0157  1268          ENDLOOP         1
                    015A  1269          SKIP
                    0161  1270
                    0161  1271 BEGIN_TEST_DATA
                    0161
                    0161          ;-----------------------------------------------------------------------
                    0161  1272
000001FF            0161  1273 1$:      .LONG   ^X 1FF          ; VA ADDRESSES
000001FD            0165  1274          .LONG   ^X 1FD
000001FE            0169  1275          .LONG   ^X 1FE
000001FF            016D  1276          .LONG   ^X 1FF
                    0171  1277
          01        0171  1278 2$:      .BYTE   1
          02        0172  1279          .BYTE   2
          02        0173  1280          .BYTE   2
          02        0174  1281          .BYTE   2
                    0175  1282
          05        0175  1283 3$:      .BYTE   5
                    0176  1284          VBUSDATA        <^X08>,0
                    0177  1285          VBUSDATA        <^X09>,1
                    0178  1286          VBUSDATA        <^X0A>,1
                    0179  1287          VBUSDATA        <^X0B>,1
                    017A  1288          VBUSDATA        <^X0D>,0
                    017B  1289
        0027        017B  1290 5$:      .WORD   ^X 27
                    017D  1291
        3FFF        017D  1292 6$:      .WORD   ^X 3FFF
                    017F  1293
000001FC            017F  1294 7$:      .LONG   ^X 1FC
000001FD            0183  1295          .LONG   ^X 1FD
000001FE            0187  1296          .LONG   ^X 1FE
000001FF            018B  1297          .LONG   ^X 1FF
000001FC            018F  1298          .LONG   ^X 1FC
000001FD            0193  1299          .LONG   ^X 1FD
000001FE            0197  1300          .LONG   ^X 1FE
000001FC            019B  1301          .LONG   ^X 1FC
                    019F  1302
          00        019F  1303 8$:      .BYTE   0
          00        01A0  1304          .BYTE   0
          00        01A1  1305          .BYTE   0
          00        01A2  1306          .BYTE   0
          01        01A3  1307          .BYTE   1
          01        01A4  1308          .BYTE   1
          01        01A5  1309          .BYTE   1
          02        01A6  1310          .BYTE   2
                    01A7  1311
```

```
                      01  01A7  1312 9$:     .BYTE   1
                          01A8  1313         VBUSDATA        <^X0D>,1
                          01A9  1314
                    1805  01A9  1315 10$:    .WORD   ^X 1805
                          01AB  1316
                000001FB  01AB  1317 11$:    .LONG   ^X 1FB
                000001F7  01AF  1318         .LONG   ^X 1F7
                000001EF  01B3  1319         .LONG   ^X 1EF
                000001DF  01B7  1320         .LONG   ^X 1DF
                000001BF  01BB  1321         .LONG   ^X 1BF
                0000017F  01BF  1322         .LONG   ^X 17F
                000000FF  01C3  1323         .LONG   ^X 0FF
                          01C7  1324
                      02  01C7  1325 12$:    .BYTE   2
                          01C8  1326
                      05  01C8  1327 13$:    .BYTE   5
                          01C9  1328         VBUSDATA        <^X08>,0
                          01CA  1329         VBUSDATA        <^X09>,1
                          01CB  1330         VBUSDATA        <^X0A>,0
                          01CC  1331         VBUSDATA        <^X0B>,1
                          01CD  1332         VBUSDATA        <^X0D>,0
                    0025  01CE  1333 14$:    .WORD   ^X 25
                          01D0  1334
                          01D0  1335 15$:
U 00, 7700,C800,0364,0300,0470,1801   01D0  1336 ;% 00   NOP
U 01, 7700,3800,78FF,FFFF,8470,1802   01DB  1340 ;% 01   LONLIT_[0E000000]                ; DISABLE CMI
U 02, 7700,4800,5BE4,03D4,6870,1803   01E6  1344 ;% 02   MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804   01F1  1348 ;% 03   LONLIT_[01000000]
U 04, 7700,C800,5BE4,03D4,6070,1805   01FC  1352 ;% 04   MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806   0207  1356 ;% 05   LONLIT_[0]                     ; ENABLE MEMORY MANAGEMENT
U 06, 7700,C800,5BE4,03D4,6870,1807   0212  1360 ;% 06   MEMSCAR_R[LONLIT]
U 07, 7700,3800,7F7F,FFFF,8470,1808   021D  1364 ;% 07   LONLIT_[01000000]
U 08, 7700,C800,5BE4,03D4,6070,1809   0228  1368 ;% 08   MEMSCR_R[LONLIT]
U 09, 7700,F800,7E7F,FFFF,8470,180A   0233  1372 ;% 09   LONLIT_[03000000]                ; TURN ON TB
U 0A, 7700,C800,5BE4,03D4,6870,180B   023E  1376 ;% 0A   MEMSCAR_R[LONLIT]
U 0B, 7700,F800,7FFF,FFFF,8470,180C   0249  1380 ;% 0B   LONLIT_[0]
U 0C, 7700,4800,5BE4,03D4,6070,180D   0254  1384 ;% 0C   MEMSCR_R[LONLIT]
U 0D, 7700,7800,7CFF,FFFF,8470,180E   025F  1388 ;% 0D   LONLIT_[06000000]                ; TURN ON THE CACHE
U 0E, 7700,4800,5BE4,03D4,6870,180F   026A  1392 ;% 0E   MEMSCAR_R[LONLIT]
U 0F, 7700,7800,7FFF,FFFF,8470,1810   0275  1396 ;% 0F   LONLIT_[0]
U 10, 7700,C800,5BE4,03D4,6070,1811   0280  1400 ;% 10   MEMSCR_R[LONLIT]
U 11, 7300,4800,0364,0300,0470,1812   028B  1404 ;% 11   STOP.CLOCK
                                      0296  1408
                                      0296  1409 16$:
U 00, 7700,C800,0364,0300,0470,1801   0296  1410 ;% 00   NOP
U 01, 5700,C800,0364,0300,4A70,1802   02A1  1414 ;% 01   VA_RDM
U 02, 7700,B800,7FFF,FF5B,8470,1803   02AC  1418 ;% 02   LONLIT_[00000148]
U 03, 7700,481B,5BE4,03D4,5070,1804   02B7  1422 ;% 03   TB_R[LONLIT]
                                      02C2  1426 17$:
U 04, 7601,4800,5BE4,0300,5D80,1805   02C2  1427 ;% 04   WRITE R[TEMP0],VSIZE/1,STROBE.V.BUS
U 05, 7300,4800,0364,0300,0470,1806   02CD  1431 ;% 05   STOP.CLOCK
                                      02D8  1435
                                      02D8  1436 BEGIN_CPU_DATA
                                      0002  1437
                                      0002  1438 RTEMP_DATA
```

```
                    0001  1439
AAAAAAAA            0001  1440          .LONG   ^X AAAAAAAA     ; R[TEMP0]
                    0005  1441
                    0005  1442 END_CPU_DATA
                    02D8  1443
                    02D8  1444 END_TEST_DATA
                    02D8
                    02D8        ;-----------------------------------------------------------------------
                    02D8  1445
                    02D8  1446 ENDTEST
```

```
0031          .SBTTL  "TEST   065     UNALIGNED DATA READ FOR BYTE/WORD/LONG DTYPE
0031   1448 ;*****************************************************************************
0031   1449 ;++
0031   1450 ;
0031   1451 ; FUNCTIONAL DESCRIPTION
0031   1452 ;
0031   1453 ;       This test will examine the UNALIGNED DATA READ micro-trap. Memory read
0031   1454 ;       references for byte, word and longword data types will be used to
0031   1455 ;       implement the functional test.
0031   1456 ;
0031   1457 ; TEST ALGORITHM
0031   1458 ;
0031   1459 ;       This test has been divided up into two subtests.
0031   1460 ;
0031   1461 ;       Subtest 1.
0031   1462 ;
0031   1463 ;       This subtest will try to force an UNALIGNED DATA READ micro-trap using
0031   1464 ;       the proper combinations of VA addresses and data types.
0031   1465 ;
0031   1466 ;        1. Initialize the CPU.
0031   1467 ;        2. Turn off memory management, cache and CMI.
0031   1468 ;        3. Prepare a loop of four iterations indexed by I.
0031   1469 ;        4. Modify the LONLIT field of a micro-instruction with a VA address.
0031   1470 ;        5. Modify the DTYPE filed of a micro-instruction for a test data type.
0031   1471 ;        6. Load the test micro-code into DCS.
0031   1472 ;        7. Execute micro-code to:
0031   1473 ;                1. Load the VA with a test address.
0031   1474 ;                2. Execute a READ with a data type loaded in step 5.
0031   1475 ;        8. Compare the address in the CS ADDRESS TRACE REGISTER with an
0031   1476 ;           expected address of 21 hex.
0031   1477 ;        9. Signal an error if the two are not equal.
0031   1478 ;       10. If there is no error continue looping.
0031   1479 ;
0031   1480 ;       Subtest 2.
0031   1481 ;
0031   1482 ;       This subtest will make sure an UNALIGNED DATA READ micro-trap does not
0031   1483 ;       occur when VA address and data types are longword aligned.
0031   1484 ;
0031   1485 ;       11. Prepare a loop of 8 iterations indexed by I.
0031   1486 ;       12. Modify the LONLIT field of a micro-instruction with a VA address.
0031   1487 ;       13. Modify the DTYPE field of a micro-instruction for a test data type.
0031   1488 ;       14. Load the test micro-code into DCS.
0031   1489 ;       15. Execute micro-code to:
0031   1490 ;                1. Load the VA with a test address.
0031   1491 ;                2. Execute a READ with a data type loaded from step 13.
0031   1492 ;       16. Compare the address in the CS ADDRESS TRACE REGISTER with an
0031   1493 ;           an address of 21 hex.
0031   1494 ;       17. Signal an error if the two are equal.
0031   1495 ;       18. If there is no error continue the loop.
0031   1496 ;       19. End of test.
0031   1497 ;
0031   1498 ; TEST DATA
0031   1499 ;
0031   1500 ;       The data for each subtest is contained in a pair of tables. Each pair
0031   1501 ;       consists of a table of test VA addresses while the other contains data
```

```
0031  1502 ;       types associated with each address.
0031  1503 ;
0031  1504 ;       Subtest 1
0031  1505 ;
0031  1506 ;       The data used in this subtest should cause a micro-trap.
0031  1507 ;
0031  1508 ;                  VA ADDRESS                    DATA TYPE
0031  1509 ;
0031  1510 ;       1.      0 0 0 0 0 0 0 1                     LONG
0031  1511 ;       2.      0 0 0 0 0 0 0 2                     LONG
0031  1512 ;       3.      0 0 0 0 0 0 0 3                     LONG
0031  1513 ;       4.      0 0 0 0 0 0 0 3                     WORD
0031  1514 ;
0031  1515 ;       Subtest 2
0031  1516 ;
0031  1517 ;       The data used in this subtest should never cause a micro-trap.
0031  1518 ;
0031  1519 ;                  VA ADDRESS                    DATA TYPE
0031  1520 ;
0031  1521 ;       1.      0 0 0 0 0 0 0 0                     LONG
0031  1522 ;       2.      0 0 0 0 0 0 0 0                     WORD
0031  1523 ;       3.      0 0 0 0 0 0 0 1                     WORD
0031  1524 ;       4.      0 0 0 0 0 0 0 2                     WORD
0031  1525 ;       5.      0 0 0 0 0 0 0 0                     BYTE
0031  1526 ;       6.      0 0 0 0 0 0 0 1                     BYTE
0031  1527 ;       7.      0 0 0 0 0 0 0 2                     BYTE
0031  1528 ;       8.      0 0 0 0 0 0 0 3                     BYTE
0031  1529 ;
0031  1530 ; LOGIC DESCRIPTION
0031  1531 ;
0031  1532 ;       TBS
0031  1533 ;
0031  1534 ; ASSUMPTIONS
0031  1535 ;
0031  1536 ;       TBS
0031  1537 ;
0031  1538 ; ERROR DESCRIPTION
0031  1539 ;
0031  1540 ;       On error this test will call out the following gate arrays;
0031  1541 ;
0031  1542 ;       The following data will also be typed out:
0031  1543 ;
0031  1544 ;       Expected Data
0031  1545 ;       Received Data
0031  1546 ;       Loop Count
0031  1547 ;
0031  1548 ;--
0031  1549 ;*****************************************************************************
0031  1550
0031  1551         SUBTEST         1
0034
0034       ;////////////////////////////////////////////////////////////////////////////
0034  1552         INITIALIZE
0036  1553         LOADDCS         8$,,0,13
003D  1554         FETCH           0
```

```
                     0042  1555          BRSTCLK         <^X0C>
                     0046  1556          EXPUTRAP
                     0048  1557          LOOP            I,1,4
                     004F  1558          LOADREG         WDREG,1$,I,LONG
                     0057  1559          LDFIELD         2$,I,BYTE,10$,40,41
                     0068  1560          LOADDCS         9$,,0,4
                     006F  1561          ERRLOOP
                     0071  1562          FETCH           0
                     0076  1563          BRSTCLK         3
                     007A  1564          CMPREGMSK       CSTRP,5$,,7$,,WORD
                     0086  1565          IFERROR         ,<<ACV><UTR>>
                     008D  1566          ENDLOOP         I
                     0090  1567
                     0090  1568          SUBTEST         2
                     0093
                     0093              ;////////////////////////////////////////////////////////////////////////////////
                     0093  1569          EXPUTRAP
                     0095  1570          LOOP            I,1,8
                     009C  1571          LOADREG         WDREG,3$,I,LONG
                     00A4  1572          LDFIELD         4$,I,BYTE,10$,40,41
                     00B5  1573          LOADDCS         9$,,0,4
                     00BC  1574          ERRLOOP
                     00BE  1575          FETCH           0
                     00C3  1576          BRSTCLK         3
                     00C7  1577          CMPREGMSK       CSTRP,6$,,7$,,WORD
                     00D3  1578          IFERROR         ,<<ACV><UTR>>
                     00DA  1579          ENDLOOP         I
                     00DD  1580          SKIP
                     00E4  1581
                     00E4  1582 BEGIN_TEST_DATA
                     00E4
                     00E4              ;-------------------------------------------------------------------------------
                     00E4  1583
00000001             00E4  1584 1$:      .LONG   1       ; VA ADDRESSES FOR SUBTEST 1
00000002             00E8  1585          .LONG   2
00000003             00EC  1586          .LONG   3
00000003             00F0  1587          .LONG   3
                     00F4  1588
      02             00F4  1589 2$:      .BYTE   2       ; DATA TYPES FOR SUBTEST 1
      02             00F5  1590          .BYTE   2
      02             00F6  1591          .BYTE   2
      01             00F7  1592          .BYTE   1
                     00F8  1593
00000000             00F8  1594 3$:      .LONG   0       ; VA ADDRESSES FOR SUBTEST 2
00000000             00FC  1595          .LONG   0
00000001             0100  1596          .LONG   1
00000002             0104  1597          .LONG   2
00000000             0108  1598          .LONG   0
00000001             010C  1599          .LONG   1
00000002             0110  1600          .LONG   2
00000003             0114  1601          .LONG   3
                     0118  1602
      02             0118  1603 4$:      .BYTE   2       ; DATA TYPES FOR SUBTEST 2
      01             0119  1604          .BYTE   1
      01             011A  1605          .BYTE   1
```

```
                        01  011B  1606          .BYTE   1
                        00  011C  1607          .BYTE   0
                        00  011D  1608          .BYTE   0
                        00  011E  1609          .BYTE   0
                        00  011F  1610          .BYTE   0
                            0120  1611
                      0021  0120  1612 5$:      .WORD   ^X 0021          ; UNALIGNED DATA READ MICRO-TRAP ADDRESS
                      1803  0122  1613 6$:      .WORD   ^X 1803          ; NO MICRO-TRAP NEXT ADDRESS
                      3FFF  0124  1614 7$:      .WORD   ^X 3FFF          ; MASK FOR ADDRESS COMPARISON
                            0126  1615
                            0126  1616 8$:
U 00, 7700,C800,0364,0300,0470,1801   0126  1617 ;% 00    NOP
U 01, 7700,3800,78FF,FFFF,8470,1802   0131  1621 ;% 01    LONLIT_[0E000000]        ; DISABLE THE CMI
U 02, 7700,4800,5BE4,03D4,6870,1803   013C  1625 ;% 02    MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804   0147  1629 ;% 03    LONLIT_[01000000]
U 04, 7700,C800,5BE4,03D4,6070,1805   0152  1633 ;% 04    MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806   015D  1637 ;% 05    LONLIT_[0]               ; DISABLE MEMORY MANAGEMENT
U 06, 7700,C800,5BE4,03D4,6870,1807   0168  1641 ;% 06    MEMSCAR_R[LONLIT]
U 07, 7700,4800,5BE4,03D4,6070,1808   0173  1645 ;% 07    MEMSCR_R[LONLIT]
U 08, 7700,F800,7CFF,FFFF,8470,1809   017E  1649 ;% 08    LONLIT_[06000000]        ; DISABLE THE CACHE
U 09, 7700,4800,5BE4,03D4,6870,180A   0189  1653 ;% 09    MEMSCAR_R[LONLIT]
U 0A, 7700,3800,7F7F,FFFF,8470,180B   0194  1657 ;% 0A    LONLIT_[01000000]
U 0B, 7700,C800,5BE4,03D4,6070,180C   019F  1661 ;% 0B    MEMSCR_R[LONLIT]
U 0C, 7300,C800,0364,0300,0470,180D   01AA  1665 ;% 0C    STOP.CLOCK
                            01B5  1669
                            01B5  1670 9$:
U 00, 7700,C800,0364,0300,0470,1801   01B5  1671 ;% 00    NOP                      ; PAUSE TO GET CPU UP TO SPEED
U 01, 5700,C800,0364,0300,4A70,1802   01C0  1675 ;% 01    VA_RDM                   ; LOAD A TEST ADDRESS INTO THE VA REGISTER
                            01CB  1679 10$:
U 02, 7701,4800,0364,0300,0500,1803   01CB  1680 ;% 02    READ,VSIZE/1             ; EXECUTE A MEMORY READ
U 03, 7300,C800,0364,0300,0470,1804   01D6  1684 ;% 03    STOP.CLOCK               ; HALT
                            01E1  1688
                            01E1  1689 END_TEST_DATA
                            01E1
                            01E1       ;-------------------------------------------------------------------------
                            01E1  1690
                            01E1  1691 ENDTEST
```

```
0032        .SBTTL  "TEST   066     UNALIGNED DATA WRITE FOR BYTE/WORD/LONG DTYPE
0032  1693 ;*******************************************************************************
0032  1694 ;++
0032  1695 ;
0032  1696 ; FUNCTIONAL DESCRIPTION
0032  1697 ;
0032  1698 ;       This test will examine the UNALIGNED DATA WRITE micro-trap. Memory
0032  1699 ;       write references for byte, word and longword data types will be used to
0032  1700 ;       implement the functional test.
0032  1701 ;
0032  1702 ; TEST ALGORITHM
0032  1703 ;
0032  1704 ;       This test has been divided up into two subtests.
0032  1705 ;
0032  1706 ;       Subtest 1.
0032  1707 ;
0032  1708 ;       This subtest will try to force an UNALIGNED DATA WRITE micro-trap using
0032  1709 ;       the proper combinations of VA addresses and data types.
0032  1710 ;
0032  1711 ;        1. Initialize the CPU.
0032  1712 ;        2. Turn off memory management, cache and CMI.
0032  1713 ;        3. Prepare a loop of four iterations indexed by I.
0032  1714 ;        4. Modify the LONLIT field of a micro-instruction with a VA address.
0032  1715 ;        5. Modify the DTYPE filed of a micro-instruction for a test data type.
0032  1716 ;        6. Load the test micro-code into DCS.
0032  1717 ;        7. Execute micro-code to:
0032  1718 ;                1. Load the VA with a test address.
0032  1719 ;                2. Execute a WRITE with a data type loaded in step 5.
0032  1720 ;        8. Compare the address in the CS ADDRESS TRACE REGISTER with an
0032  1721 ;           expected address of 21 hex.
0032  1722 ;        9. Signal an error if the two are not equal.
0032  1723 ;       10. If there is no error continue looping.
0032  1724 ;
0032  1725 ;       Subtest 2.
0032  1726 ;
0032  1727 ;       This subtest will make sure an UNALIGNED DATA WRITE micro-trap does not
0032  1728 ;       occur when VA address and data types are longword aligned.
0032  1729 ;
0032  1730 ;       11. Prepare a loop of 8 iterations indexed by I.
0032  1731 ;       12. Modify the LONLIT field of a micro-instruction with a VA address.
0032  1732 ;       13. Modify the DTYPE field of a micro-instruction for a test data type.
0032  1733 ;       14. Load the test micro-code into DCS.
0032  1734 ;       15. Execute micro-code to:
0032  1735 ;                1. Load the VA with a test address.
0032  1736 ;                2. Execute a WRITE with a data type loaded from step 13.
0032  1737 ;       16. Compare the address in the CS ADDRESS TRACE REGISTER with an
0032  1738 ;           an address of 21 hex.
0032  1739 ;       17. Signal an error if the two are equal.
0032  1740 ;       18. If there is no error continue the loop.
0032  1741 ;       19. End of test.
0032  1742 ;
0032  1743 ; TEST DATA
0032  1744 ;
0032  1745 ;       The data for each subtest is contained in a pair of tables. Each pair
0032  1746 ;       consists of a table of test VA addresses while the other contains data
```

```
                     1747 ;       types associated with each address.
                     1748 ;
                     1749 ;       Subtest 1
                     1750 ;
                     1751 ;       The data used in this subtest should cause a micro-trap.
                     1752 ;
                     1753 ;                    VA ADDRESS                    DATA TYPE
                     1754 ;
                     1755 ;       1.      0 0 0 0 0 0 0 1                       LONG
                     1756 ;       2.      0 0 0 0 0 0 0 2                       LONG
                     1757 ;       3.      0 0 0 0 0 0 0 3                       LONG
                     1758 ;       4.      0 0 0 0 0 0 0 3                       WORD
                     1759 ;
                     1760 ;       Subtest 2
                     1761 ;
                     1762 ;       The data used in this subtest should never cause a micro-trap.
                     1763 ;
                     1764 ;                    VA ADDRESS                    DATA TYPE
                     1765 ;
                     1766 ;       1.      0 0 0 0 0 0 0 0                       LONG
                     1767 ;       2.      0 0 0 0 0 0 0 0                       WORD
                     1768 ;       3.      0 0 0 0 0 0 0 1                       WORD
                     1769 ;       4.      0 0 0 0 0 0 0 2                       WORD
                     1770 ;       5.      0 0 0 0 0 0 0 0                       BYTE
                     1771 ;       6.      0 0 0 0 0 0 0 1                       BYTE
                     1772 ;       7.      0 0 0 0 0 0 0 2                       BYTE
                     1773 ;       8.      0 0 0 0 0 0 0 3                       BYTE
                     1774 ;
                     1775 ; LOGIC DESCRIPTION
                     1776 ;
                     1777 ;       TBS
                     1778 ;
                     1779 ; ASSUMPTIONS
                     1780 ;
                     1781 ;       TBS
                     1782 ;
                     1783 ; ERROR DESCRIPTION
                     1784 ;
                     1785 ;       On error this test will call out the following gate arrays;
                     1786 ;
                     1787 ;       The following data will also be typed out:
                     1788 ;
                     1789 ;       Expected Data
                     1790 ;       Received Data
                     1791 ;       Loop Count
                     1792 ;
                     1793 ;--
                     1794 ;*****************************************************************************
                     1795
                     1796         SUBTEST         1
                     1797
                     1797        ;/////////////////////////////////////////////////////////////////////////////
                     1797         INITIALIZE
                     1798         LOADDCS         8$,,0,13
                     1799         FETCH           0
```

```
                0043  1800          BRSTCLK         <^X0C>
                0047  1801          EXPUTRAP
                0049  1802          LOOP            I,1,4
                0050  1803          LOADREG         WDREG,1$,I,LONG
                0058  1804          LDFIELD         2$,I,BYTE,10$,40,41
                0069  1805          LOADDCS         9$,,0,4
                0070  1806          ERRLOOP
                0072  1807          FETCH           0
                0077  1808          BRSTCLK         3
                007B  1809          CMPREGMSK       CSTRP,5$,,7$,,WORD
                0087  1810          IFERROR         ,<<ACV><UTR>>
                008E  1811          ENDLOOP         I
                0091  1812
                0091  1813          SUBTEST         2
                0094
                0094          ;////////////////////////////////////////////////////////////////////////////////
                0094  1814          EXPUTRAP
                0096  1815          LOOP            I,1,8
                009D  1816          LOADREG         WDREG,3$,I,LONG
                00A5  1817          LDFIELD         4$,I,BYTE,10$,40,41
                00B6  1818          LOADDCS         9$,,0,4
                00BD  1819          ERRLOOP
                00BF  1820          FETCH           0
                00C4  1821          BRSTCLK         3
                00C8  1822          CMPREGMSK       CSTRP,6$,,7$,,WORD
                00D4  1823          IFERROR         ,<<ACV><UTR>>
                00DB  1824          ENDLOOP         I
                00DE  1825          SKIP
                00E5  1826
                00E5  1827  BEGIN_TEST_DATA
                00E5
                00E5          ;-----------------------------------------------------------------------------
                00E5  1828
00000001        00E5  1829  1$:     .LONG   1       ; VA ADDRESSES FOR SUBTEST 1
00000002        00E9  1830          .LONG   2
00000003        00ED  1831          .LONG   3
00000003        00F1  1832          .LONG   3
                00F5  1833
      02        00F5  1834  2$:     .BYTE   2       ; DATA TYPES FOR SUBTEST 1
      02        00F6  1835          .BYTE   2
      02        00F7  1836          .BYTE   2
      01        00F8  1837          .BYTE   1
                00F9  1838
00000000        00F9  1839  3$:     .LONG   0       ; VA ADDRESSES FOR SUBTEST 2
00000000        00FD  1840          .LONG   0
00000001        0101  1841          .LONG   1
00000002        0105  1842          .LONG   2
00000000        0109  1843          .LONG   0
00000001        010D  1844          .LONG   1
00000002        0111  1845          .LONG   2
00000003        0115  1846          .LONG   3
                0119  1847
      02        0119  1848  4$:     .BYTE   2       ; DATA TYPES FOR SUBTEST 2
      01        011A  1849          .BYTE   1
      01        011B  1850          .BYTE   1
```

```
                                 01  011C  1851          .BYTE   1
                                 00  011D  1852          .BYTE   0
                                 00  011E  1853          .BYTE   0
                                 00  011F  1854          .BYTE   0
                                 00  0120  1855          .BYTE   0
                                     0121  1856
                               0025  0121  1857 5$:      .WORD   ^X 0025          ; UNALIGNED DATA WRITE MICRO-TRAP ADDRESS
                               1803  0123  1858 6$:      .WORD   ^X 1803          ; NO MICRO-TRAP NEXT ADDRESS
                               3FFF  0125  1859 7$:      .WORD   ^X 3FFF          ; MASK FOR ADDRESS COMPARISON
                                     0127  1860
                                     0127  1861 8$:
U 00, 7700,C800,0364,0300,0470,1801  0127  1862 ;% 00    NOP
U 01, 7700,3800,78FF,FFFF,8470,1802  0132  1866 ;% 01    LONLIT_[0E000000]        ; DISABLE THE CMI
U 02, 7700,4800,5BE4,03D4,6870,1803  013D  1870 ;% 02    MEMSCAR_R[LONLIT]
U 03, 7700,3800,7F7F,FFFF,8470,1804  0148  1874 ;% 03    LONLIT_[01000000]
U 04, 7700,C800,5BE4,03D4,6070,1805  0153  1878 ;% 04    MEMSCR_R[LONLIT]
U 05, 7700,F800,7FFF,FFFF,8470,1806  015E  1882 ;% 05    LONLIT_[0]               ; DISABLE MEMORY MANAGEMENT
U 06, 7700,C800,5BE4,03D4,6870,1807  0169  1886 ;% 06    MEMSCAR_R[LONLIT]
U 07, 7700,4800,5BE4,03D4,6070,1808  0174  1890 ;% 07    MEMSCR_R[LONLIT]
U 08, 7700,F800,7CFF,FFFF,8470,1809  017F  1894 ;% 08    LONLIT_[06000000]        ; DISABLE THE CACHE
U 09, 7700,4800,5BE4,03D4,6870,180A  018A  1898 ;% 09    MEMSCAR_R[LONLIT]
U 0A, 7700,3800,7F7F,FFFF,8470,180B  0195  1902 ;% 0A    LONLIT_[01000000]
U 0B, 7700,C800,5BE4,03D4,6070,180C  01A0  1906 ;% 0B    MEMSCR_R[LONLIT]
U 0C, 7300,C800,0364,0300,0470,180D  01AB  1910 ;% 0C    STOP.CLOCK
                                     01B6  1914
                                     01B6  1915 9$:
U 00, 7700,C800,0364,0300,0470,1801  01B6  1916 ;% 00    NOP                      ; PAUSE TO GET CPU UP TO SPEED
U 01, 5700,C800,0364,0300,4A70,1802  01C1  1920 ;% 01    VA_RDM                   ; LOAD A TEST ADDRESS INTO THE VA REGISTER
                                     01CC  1924 10$:
U 02, 7701,4800,0364,0300,5D80,1803  01CC  1925 ;% 02    WRITE,VSIZE/1            ; EXECUTE A MEMORY WRITE
U 03, 7300,C800,0364,0300,0470,1804  01D7  1929 ;% 03    STOP.CLOCK               ; HALT
                                     01E2  1933
                                     01E2  1934 END_TEST_DATA
                                     01E2
                                     01E2       ;-------------------------------------------------------------------------
                                     01E2  1935
                                     01E2  1936 ENDTEST
```

```
                        .SBTTL  "TEST   067     TEST PTE ACCESS CHECK, READ MICRO ORDER
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST WILL CHECK THE "BUS/PRB.RD.PTE" MICRO ORDER FOR CORRECT
                ;       OPERATION.
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  CREATE A TEST LOOP
                ;       2.  SELECT A PTE FROM A TABLE AND LOAD INTO D REGISTER (RDM)
                ;       3.  EXECUTE DCS PROGRAM
                ;               1.  LOAD PHYSICAL/VIRTUAL ADDRESSING REGISTER ADDRESS INTO
                ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
                ;               2.  SET MEMORY MANAGEMENT BIT OFF
                ;               3.  LOAD REFERENCE CACHE ONLY REGISTER ADDRESS INTO MEMSCAR
                ;               4.  SET CMI OFF
                ;               5.  SET CURRENT MODE REGISTER TO EXECUTIVE MODE
                ;               6.  PERFORM "BUS/PRB.RD.PTE" WHILE ENABLING PTE ONTO WBUS
                ;                   FROM RDM
                ;               7.  STROBE VBUS AND CS LINES TO RDM
                ;       4.  COMPARE RESULTS ON VBUS AND CS LINES (EXPECTED AND RECEIVED)
                ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING PTE'S
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       ITERATION       PTE             VBUS      CS LINES
                ;
                ;           1           00000150        0011        1803
                ;           2           000000A8        0001        1801
                ;           3           00000060        0001        1801
                ;           4           00000038        0000        1800
                ;           5           00000148        0111        1807
                ;           6           00000128        0000        1800
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       THIS TEST CHECKS THAT THE "PTE ACCESS CHECK, READ" OPERATES CORRECTLY.
                ;       MOST OF THE LOGIC USED HAS ALL READY BEEN VERIFIED BY PREVIOUS TESTS
                ;       OF THE PROBE MICRO ORDERS.  ONE THING TO LOOKOUT FOR IS THAT THE
                ;       PTE CHECK ALGORITHM IS IMPLEMENTED PROPERLY BY THE ACV GATE ARRAY.
                ;       IF THE VBUS BITS ARE INCORRECT THEN ACV IS YOUR BEST BET TO SWAP
                ;       FIRST.  ANOTHER AREA TO WATCH IS THE TB CONTROL LOGIC WHICH ALLOWS
                ;       THE WBUS TO BE GATED RATHER THAN THE TB DATA.  THE ADK GATE ARRAY
                ;       IS RESPONSIBLE FOR GATING THE WBUS ONTO THE DBUS AND THROUGH THE
                ;       AMUX TO THE TB OUTPUT MULTIPLEXER.  ADK IS A GOOD SECOND CHOICE TO
                ;       SWAP IF ACV DOES NOT FIX YOUR PROBLEM.  THE ONLY OTHER GATE ARRAY
                ;       USED IS UTR WHICH CONTROLS MICRO VECTOR LINES 2 AND 3.
                ;
                ;
```

```
                     002C  1992 ; ERROR DESCRIPTION:
                     002C  1993 ;
                     002C  1994 ;       THIS TEST CAN FAIL IN ONE OF TWO PLACES.  SO YOU'LL HAVE TO
                     002C  1995 ;       CHECK THE PC TO DETERMINE WHICH IFERROR STATEMENT IS DETECTING
                     002C  1996 ;       THE FAILURE.
                     002C  1997 ;
                     002C  1998 ;       VBUS FAILURE:  BIT <7> CONTAINS THE VALUE
                     002C  1999 ;                      BITS <6:0> CONTAIN THE BIT POSITION
                     002C  2000 ;
                     002C  2001 ;              EXPECTED VBUS DATA
                     002C  2002 ;              RECEIVED VBUS DATA
                     002C  2003 ;              LOOP COUNT
                     002C  2004 ;              BALANCE OF VBUS BITS FOLLOWING FIRST FAILURE
                     002C  2005 ;
                     002C  2006 ;       CS ADDRESS FAILURE:
                     002C  2007 ;
                     002C  2008 ;              EXPECTED CS ADDRESS
                     002C  2009 ;              RECEIVED CS ADDRESS
                     002C  2010 ;              LOOP COUNT
                     002C  2011 ;
                     002C  2012 ;--
                     002C  2013 ;*****************************************************************************
                     002C  2014 ;/////////////////////////////////////////////////////////////////////////////
                     002C  2015
                     002C  2016         INITIALIZE                                  ;INIT CPU
                     002E  2017         LOOP            I,1,6                       ;CREATE TEST LOOP
                     0035  2018         LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO D REGISTER
                     003D  2019         ERRLOOP                                     ;ERROR LOOP
                     003F  2020         FETCH           0                           ;START DCS PROGRAM
                     0044  2021         BRSTCLK         7                           ;END DCS PROGRAM
                     0048  2022         CMPVBUS         2$,I                        ;COMPARE VBUS RESULTS
                     004D  2023         IFERROR         ,<<ACV><ADK><UTR>>,         ;POSSIBLE FAILING ARRAY
                     0055  2024         CMPREGMSK       CSTRP,3$,I,4$,,W,           ;COMPARE CS ADDRESS RESULTS
                     0061  2025         IFERROR         ,,<<DPM>>                   ;POSSIBLE FAILING MODULE
                     0067  2026         ENDLOOP         I                           ;END TEST LOOP
                     006A  2027         SKIP
                     0071  2028
                     0071  2029 BEGIN_TEST_DATA
                     0071
                     0071         ;-----------------------------------------------------------------------
                     0071  2030
00000150             0071  2031 1$:      .LONG           ^X00000150                  ;PTE TABLE
000000A8             0075  2032         .LONG           ^X000000A8
00000060             0079  2033         .LONG           ^X00000060
00000038             007D  2034         .LONG           ^X00000038
00000148             0081  2035         .LONG           ^X00000148
00000128             0085  2036         .LONG           ^X00000128
                     0089  2037
      04             0089  2038 2$:      .BYTE           ^X4                         ;VBUS DATA TABLE
                     008A  2039         VBUSDATA        <^X08>,0                    ;ITERATION #1
                     008B  2040         VBUSDATA        <^X09>,0
                     008C  2041         VBUSDATA        <^X0A>,1
                     008D  2042         VBUSDATA        <^X0B>,1
                     008E  2043
      04             008E  2044         .BYTE           ^X4                         ;ITERATION #2
```

```
                                008F  2045          VBUSDATA        <^X08>,0
                                0090  2046          VBUSDATA        <^X09>,0
                                0091  2047          VBUSDATA        <^X0A>,0
                                0092  2048          VBUSDATA        <^X0B>,1
                                0093  2049
                             04 0093  2050          .BYTE           ^X4                     ;ITERATION #3
                                0094  2051          VBUSDATA        <^X08>,0
                                0095  2052          VBUSDATA        <^X09>,0
                                0096  2053          VBUSDATA        <^X0A>,0
                                0097  2054          VBUSDATA        <^X0B>,1
                                0098  2055
                             04 0098  2056          .BYTE           ^X4                     ;ITERATION #4
                                0099  2057          VBUSDATA        <^X08>,0
                                009A  2058          VBUSDATA        <^X09>,0
                                009B  2059          VBUSDATA        <^X0A>,0
                                009C  2060          VBUSDATA        <^X0B>,0
                                009D  2061
                             04 009D  2062          .BYTE           ^X4                     ;ITERATION #5
                                009E  2063          VBUSDATA        <^X08>,0
                                009F  2064          VBUSDATA        <^X09>,1
                                00A0  2065          VBUSDATA        <^X0A>,1
                                00A1  2066          VBUSDATA        <^X0B>,1
                                00A2  2067
                             04 00A2  2068          .BYTE           ^X4                     ;ITERATION #6
                                00A3  2069          VBUSDATA        <^X08>,0
                                00A4  2070          VBUSDATA        <^X09>,0
                                00A5  2071          VBUSDATA        <^X0A>,0
                                00A6  2072          VBUSDATA        <^X0B>,0
                                00A7  2073
                           1803 00A7  2074 3$:      .WORD           ^X1803                  ;CS ADDRESS TABLE
                           1801 00A9  2075          .WORD           ^X1801
                           1801 00AB  2076          .WORD           ^X1801
                           1800 00AD  2077          .WORD           ^X1800
                           1807 00AF  2078          .WORD           ^X1807
                           1800 00B1  2079          .WORD           ^X1800
                                00B3  2080
                           3FFF 00B3  2081 4$:      .WORD           ^X3FFF                  ;MASK FOR CMPREGMSK PSEUDO-OP
                                00B5  2082
                                00B5  2083 BEGIN_CPU_DATA
                                0002  2084
                                0002  2085 DCS_DATA
                                0001  2086
U 00, 7700,C800,0364,0300,0470,1801  0001  2087 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  2091 ;% 01    MEMSCAR_R[TEMP0]                ;PHY/VIR ADDRESS TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  2095 ;% 02    MEMSCR_R[TEMP1]                 ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  2099 ;% 03    MEMSCAR_R[TEMP2]                ;REF CACHE ONLY TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805  002D  2103 ;% 04    MEMSCR_R[TEMP3]                 ;TURN CMI OFF
U 05, 7700,8800,5BE4,0310,6A70,1806  0038  2107 ;% 05    PSL(PREV_CURM ISCURM_R[TEMP4])  ;SET MODE TO EXECUTIVE
U C6, 5600,C800,0364,7B00,0560,9800  0043  2111 ;% 06    STROBE.V.BUS,WB_RDM,PTE CHECK READ?,NEXT/1800
                                     004E  2115                                          ;PERFORM PTE CHECK
U 07, 7B00,C800,0364,0300,0470,1808  004E  2116 ;% 07    NOP,STOP.CLOCK,LATCH.CS.ADDR    ;STOP CPU CLOCK
                                     0059  2120
                                     0059  2121 RTEMP_DATA
                                     0001  2122
                           00000000  0001  2123          .LONG   ^X00000000              ;PHY/VIR REG ADDRESS
```

```
00000000  0005  2124          .LONG   ^X00000000                      ;SET MME OFF
0E000000  0009  2125          .LONG   ^X0E000000                      ;REF CACHE ONLY REG ADDRESS
01000000  000D  2126          .LONG   ^X01000000                      ;SET CMI OFF
01000000  0011  2127          .LONG   ^X01000000                      ;SET EXECUTIVE MODE
          0015  2128
          0015  2129 END_CPU_DATA
          00B5  2130
          00B5  2131 END_TEST_DATA
          00B5
          00B5        ;-------------------------------------------------------------------------
          00B5  2132
          00B5  2133 ENDTEST
```

```
                        .SBTTL  "TEST   068     TEST PTE ACCESS CHECK, READ, KERNEL MODE
                2135 ;*******************************************************************************
                2136 ;++
                2137 ;
                2138 ; FUNCTIONAL DESCRIPTION:
                2139 ;
                2140 ;       THIS TEST VERIFIES THAT THE "BUS/PRB.RD.PTE.K" MICRO ORDER
                2141 ;       FUNCTIONS CORRECTLY.
                2142 ;
                2143 ;
                2144 ; TEST ALGORITHM:
                2145 ;
                2146 ;       1.  CREATE A TEST LOOP
                2147 ;       2.  SELECT A PTE FROM A TABLE AND LOAD INTO D REGISTER (RDM)
                2148 ;       3.  EXECUTE DCS PROGRAM
                2149 ;               1.  LOAD PHYSICAL/VIRTUAL ADDRESSING REGISTER ADDRESS INTO
                2150 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
                2151 ;               2.  SET MEMORY MANAGEMENT BIT OFF
                2152 ;               3.  LOAD REFERENCE CACHE ONLY REGISTER ADDRESS INTO MEMSCAR
                2153 ;               4.  SET CMI OFF
                2154 ;               5.  SET CURRENT MODE REGISTER TO USER MODE
                2155 ;               6.  PERFORM BUS/PRB.RD.PTE.K WHILE ENABLING PTE FROM
                2156 ;                   D REGISTER (RDM) ONTO WBUS
                2157 ;               7.  STROBE VBUS AND CS LINES TO RDM
                2158 ;       4.  COMPARE RESULTS ON VBUS AND CS LINES (EXPECTED AND RECEIVED)
                2159 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING PTE'S
                2160 ;
                2161 ;
                2162 ; TEST PATTERNS:
                2163 ;
                2164 ;       ITERATION           PTE           VBUS       CS ADDRESS
                2165 ;
                2166 ;           1             00000000        0000          1800
                2167 ;           2             00000020        0001          1801
                2168 ;
                2169 ;
                2170 ; LOGIC DESCRIPTION:
                2171 ;
                2172 ;       ALL MOST ALL THE LOGIC INVOLVED IN THIS TEST HAS BEEN CHECKED BY
                2173 ;       PREVIOUS TESTS.  IF ANY ERRORS OCCUR IT SHOULD BE A RESULT OF THE
                2174 ;       ACV GATE ARRAY NOT DECODING THE BUS FIELD PROPERLY.
                2175 ;
                2176 ;
                2177 ; ERROR DESCRIPTION:
                2178 ;
                2179 ;       THIS TEST CAN FAIL IN ONE OF TWO PLACES.  SO YOU'LL HAVE TO
                2180 ;       CHECK THE PC TO DETERMINE WHICH IFERROR STATEMENT IS DETECTING
                2181 ;       THE FAILURE.
                2182 ;
                2183 ;       VBUS FAILURE:  BIT <7> CONTAINS THE VALUE
                2184 ;                      BITS <6:0> CONTAIN THE BIT POSITION
                2185 ;
                2186 ;               EXPECTED VBUS DATA
                2187 ;               RECEIVED VBUS DATA
                2188 ;               LOOP COUNT
```

```
                     002D  2189 ;                BALANCE OF VBUS BITS FOLLOWING FIRST FAILURE
                     002D  2190 ;
                     002D  2191 ;       CS ADDRESS FAILURE:
                     002D  2192 ;
                     002D  2193 ;                EXPECTED CS ADDRESS
                     002D  2194 ;                RECEIVED CS ADDRESS
                     002D  2195 ;                LOOP COUNT
                     002D  2196 ;
                     002D  2197 ;--
                     002D  2198 ;*******************************************************************************
                     002D  2199 ;///////////////////////////////////////////////////////////////////////////////
                     002D  2200
                     002D  2201         INITIALIZE                                  ;INIT CPU
                     002F  2202         LOOP            I,1,2                       ;CREATE A TEST LOOP
                     0036  2203         LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO D REGISTER
                     003E  2204         ERRLOOP                                     ;ERROR LOOP
                     0040  2205         FETCH           0                           ;START DCS PROGRAM
                     0045  2206         BRSTCLK         7                           ;END DCS PROGRAM
                     0049  2207         CMPVBUS         2$,I                        ;COMPARE VBUS RESULTS
                     004E  2208         IFERROR         ,<<ACV><UTR><ADK>>,         ;POSSIBLE FAILING ARRAY
                     0056  2209         CMPREGMSK       CSTRP,3$,I,4$,,W,           ;COMPARE CS ADDRESS RESULTS
                     0062  2210         IFERROR         ,,<<DPM>>                   ;POSSIBLE FAILING MODULE
                     0068  2211         ENDLOOP         I                           ;END TEST LOOP
                     006B  2212         SKIP
                     0072  2213
                     0072  2214 BEGIN_TEST_DATA
                     0072
                     0072         ;-------------------------------------------------------------------------
                     0072  2215
                00000000  0072  2216 1$:      .LONG           ^X00000000              ;PTE TABLE
                00000020  0076  2217          .LONG           ^X00000020
                     007A  2218
                       04  007A  2219 2$:      .BYTE           ^X4                     ;VBUS DATA TABLE
                     007B  2220               VBUSDATA        <^X08>,0
                     007C  2221               VBUSDATA        <^X09>,0
                     007D  2222               VBUSDATA        <^X0A>,0
                     007E  2223               VBUSDATA        <^X0B>,0
                     007F  2224
                       04  007F  2225          .BYTE           ^X4
                     0080  2226               VBUSDATA        <^X08>,0
                     0081  2227               VBUSDATA        <^X09>,0
                     0082  2228               VBUSDATA        <^X0A>,0
                     0083  2229               VBUSDATA        <^X0B>,1
                     0084  2230
                     1800  0084  2231 3$:      .WORD           ^X1800                  ;CS ADDRESS TABLE
                     1801  0086  2232          .WORD           ^X1801
                     0088  2233
                     3FFF  0088  2234 4$:      .WORD           ^X3FFF                  ;MASK FOR CMPREGMSK PSEUDO-OP
                     008A  2235
                     008A  2236 BEGIN_CPU_DATA
                     0002  2237
                     0002  2238 DCS_DATA
                     0001  2239
U 00, 7700,C800,0364,0300,0470,1801  0001  2240 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  2244 ;% 01    MEMSCAR_R[TEMP0]            ;PHY/VIR ADDRESS TO MEMSCAR
```

```
U 02. 7700.8800.5BE4.0304.6070.1803  0017  2248 ;% 02   MEMSCR_R[TEMP1]                              ;SET MME OFF
U 03. 7700.8800.5BE4.0308.6870.1804  0022  2252 ;% 03   MEMSCAR_R[TEMP2]                            ;REF CACHE ONLY TO MEMSCAR
U 04. 7700.C800.5BE4.030C.6070.1805  002D  2256 ;% 04   MEMSCR_R[TEMP3]                             ;SET CMI OFF
U 05. 7700.8800.5BE4.0310.6A70.1806  0038  2260 ;% 05   PSL(PREV_CURM ISCURM_R[TEMP4])             ;SET MODE TO USER
U 06. 5600.4800.0364.7B00.0570.9800  0043  2264 ;% 06   STROBE.V.BUS,WB_RDM,PTE CHECK READ KERNAL?,NEXT/1800
                                     004E  2268                                                  ;PERFORM PTE CHECK
U 07. 7B00.C800.0364.0300.0470.1808  004E  2269 ;% 07   NOP,STOP.CLOCK,LATCH.CS.ADDR              ;STOP CPU CLOCK
                                     0059  2273
                                     0059  2274 RTEMP_DATA
                                     0001  2275
                          00000000   0001  2276          .LONG   ^X00000000                     ;PHY/VIR REG ADDRESS
                          00000000   0005  2277          .LONG   ^X00000000                     ;SET MME OFF
                          0E000000   0009  2278          .LONG   ^X0E000000                     ;REF CACHE ONLY REG ADDRESS
                          03000000   000D  2279          .LONG   ^X03000000                     ;SET USER MODE
                                     0011  2280
                                     0011  2281 END_CPU_DATA
                                     008A  2282
                                     008A  2283 END_TEST_DATA
                                     008A
                                     008A          ;-------------------------------------------------------------------------
                                     008A  2284
                                     008A  2285 ENDTEST
```

```
                        .SBTTL  "TEST   069     TEST PTE ACCESS CHECK, WRITE MICRO ORDER
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST WILL CHECK THE "BUS/PRB.WR.PTE" MICRO ORDER FOR CORRECT
                ;       OPERATION.
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  CREATE A TEST LOOP
                ;       2.  SELECT A PTE FROM A TABLE AND LOAD INTO D REGISTER (RDM)
                ;       3.  EXECUTE DCS PROGRAM
                ;               1.  LOAD PHYSICAL/VIRTUAL ADDRESSING REGISTER ADDRESS INTO
                ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
                ;               2.  SET MEMORY MANAGEMENT BIT OFF
                ;               3.  LOAD REFERENCE CACHE ONLY REGISTER ADDRESS INTO MEMSCAR
                ;               4.  SET CMI OFF
                ;               5.  SET CURRENT MODE REGISTER TO SUPERVISOR MODE
                ;               6.  PERFORM "BUS/PRB.WR.PTE" WHILE ENABLING PTE ONTO WBUS
                ;                   FROM RDM
                ;               7.  STROBE VBUS AND CS LINES TO RDM
                ;       4.  COMPARE RESULTS ON VBUS AND CS LINES (EXPECTED AND RECEIVED)
                ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING PTE'S
                ;
                ; TEST PATTERNS:
                ;
                ;       ITERATION       PTE             VBUS    CS LINES
                ;
                ;           1           00000140        0011      1803
                ;           2           000000C8        0001      1801
                ;           3           00000080        0001      1801
                ;           4           000000A8        0000      1800
                ;           5           00000148        0111      1807
                ;           6           00000138        0000      1800
                ;           7           00000150        0000      1800
                ;           8           00000060        0000      1800
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       THIS TEST CHECKS THAT THE "PTE ACCESS CHECK, WRITE" OPERATES
                ;       CORRECTLY.  MOST OF THE LOGIC USED HAS ALL READY BEEN VERIFIED
                ;       BY PREVIOUS TESTS OF THE PROBE MICRO ORDERS.  ONE THING TO
                ;       LOOKOUT FOR IS THAT THE PTE CHECK ALGORITHM IS IMPLEMENTED
                ;       PROPERLY BY THE ACV GATE ARRAY.  IF THE VBUS BITS ARE
                ;       INCORRECT THEN ACV IS YOU BEST BET TO SWAP FIRST.  ANOTHER
                ;       AREA TO WATCH IS THE TB CONTROL LOGIC WHICH ALLOWS THE WBUS TO
                ;       BE GATED RATHER THAN THE TB DATA.  THE ADK GATE ARRAY IS RESPONSIBLE
                ;       FOR GATING THE WBUS ONTO THE DBUS AND THROUGH THE AMUX TO THE TB
                ;       OUTPUT MULTIPLEXER.  ADK IS A GOOD SECOND CHOICE TO SWAP IF ACV
                ;       DOES NOT FIX YOUR PROBLEM.  THE ONLY OTHER GATE ARRAY USED IS UTR
```

```
                        002D  2341 ;       WHICH CONTROLS MICRO VECTOR LINES 2 AND 3.
                        002D  2342 ;
                        002D  2343 ;
                        002D  2344 ; ERROR DESCRIPTION:
                        002D  2345 ;
                        002D  2346 ;       THIS TEST CAN FAIL IN ONE OF TWO PLACES.  SO YOU'LL HAVE TO
                        002D  2347 ;       CHECK THE PC TO DETERMINE WHICH IFERROR STATEMENT IS DETECTING
                        002D  2348 ;       THE FAILURE.
                        002D  2349 ;
                        002D  2350 ;       VBUS FAILURE:  BIT <8> CONTAINS THE VALUE
                        002D  2351 ;                      BITS <7:0> CONTAIN THE BIT POSITION
                        002D  2352 ;
                        002D  2353 ;              EXPECTED VBUS DATA
                        002D  2354 ;              RECEIVED VBUS DATA
                        002D  2355 ;              LOOP COUNT
                        002D  2356 ;              BALANCE OF VBUS BITS FOLLOWING FIRST FAILURE
                        002D  2357 ;
                        002D  2358 ;       CS ADDRESS FAILURE:
                        002D  2359 ;
                        002D  2360 ;              EXPECTED CS ADDRESS
                        002D  2361 ;              RECEIVED CS ADDRESS
                        002D  2362 ;              LOOP COUNT
                        002D  2363 ;
                        002D  2364 ;--
                        002D  2365 ;******************************************************************************
                        002D  2366 ;//////////////////////////////////////////////////////////////////////////////
                        002D  2367
                        002D  2368         INITIALIZE                                  ;INIT CPU
                        002F  2369         LOOP            I,1,8                       ;CREATE A TEST LOOP
                        0036  2370         LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO D REGISTER
                        003E  2371         ERRLOOP                                     ;ERROR LOOP
                        0040  2372         FETCH           0                           ;START DCS PROGRAM
                        0045  2373         BRSTCLK         7                           ;END DCS PROGRAM
                        0049  2374         CMPVBUS         2$,I                        ;COMPARE VBUS RESULTS
                        004E  2375         IFERROR         ,<<ACV><ADK><UTR>>,         ;POSSIBLE FAILING ARRAY
                        0056  2376         CMPREGMSK       CSTRP,3$,I,4$,,W,           ;COMPARE CS ADDRESS RESULTS
                        0062  2377         IFERROR         ,,<<DPM>>                   ;POSSIBLE FAILING MODULE
                        0068  2378         ENDLOOP         I                           ;END TEST LOOP
                        006B  2379         SKIP
                        0072  2380
                        0072  2381 BEGIN_TEST_DATA
                        0072
                        0072       ;-------------------------------------------------------------------------
                        0072  2382
00000140                0072  2383 1$:      .LONG           ^X00000140                  ;PTE TABLE
000000C8                0076  2384         .LONG           ^X000000C8
00000080                007A  2385         .LONG           ^X00000080
000000A8                007E  2386         .LONG           ^X000000A8
00000148                0082  2387         .LONG           ^X00000148
00000138                0086  2388         .LONG           ^X00000138
00000150                008A  2389         .LONG           ^X00000150
00000060                008E  2390         .LONG           ^X00000060
                        0092  2391
                     04 0092  2392 2$:      .BYTE           ^X4                         ;VBUS DATA TABLE
                        0093  2393         VBUSDATA        <^X08>,0                    ;ITERATION #1
```

```
                0094  2394          VBUSDATA        <^X09>,0
                0095  2395          VBUSDATA        <^X0A>,1
                0096  2396          VBUSDATA        <^X0B>,1
                0097  2397
          04    0097  2398          .BYTE           ^X4                     ;ITERATION #2
                0098  2399          VBUSDATA        <^X08>,0
                0099  2400          VBUSDATA        <^X09>,0
                009A  2401          VBUSDATA        <^X0A>,0
                009B  2402          VBUSDATA        <^X0B>,1
                009C  2403
          04    009C  2404          .BYTE           ^X4                     ;ITERATION #3
                009D  2405          VBUSDATA        <^X08>,0
                009E  2406          VBUSDATA        <^X09>,0
                009F  2407          VBUSDATA        <^X0A>,0
                00A0  2408          VBUSDATA        <^X0B>,1
                00A1  2409
          04    00A1  2410          .BYTE           ^X4                     ;ITERATION #4
                00A2  2411          VBUSDATA        <^X08>,0
                00A3  2412          VBUSDATA        <^X09>,0
                00A4  2413          VBUSDATA        <^X0A>,0
                00A5  2414          VBUSDATA        <^X0B>,0
                00A6  2415
          04    00A6  2416          .BYTE           ^X4                     ;ITERATION #5
                00A7  2417          VBUSDATA        <^X08>,0
                00A8  2418          VBUSDATA        <^X09>,1
                00A9  2419          VBUSDATA        <^X0A>,1
                00AA  2420          VBUSDATA        <^X0B>,1
                00AB  2421
          04    00AB  2422          .BYTE           ^X4                     ;ITERATION #6
                00AC  2423          VBUSDATA        <^X08>,0
                00AD  2424          VBUSDATA        <^X09>,0
                00AE  2425          VBUSDATA        <^X0A>,0
                00AF  2426          VBUSDATA        <^X0B>,0
                00B0  2427
          04    00B0  2428          .BYTE           ^X4                     ;ITERATION #7
                00B1  2429          VBUSDATA        <^X08>,0
                00B2  2430          VBUSDATA        <^X09>,0
                00B3  2431          VBUSDATA        <^X0A>,0
                00B4  2432          VBUSDATA        <^X0B>,0
                00B5  2433
          04    00B5  2434          .BYTE           ^X4                     ;ITERATION #8
                00B6  2435          VBUSDATA        <^X08>,0
                00B7  2436          VBUSDATA        <^X09>,0
                00B8  2437          VBUSDATA        <^X0A>,0
                00B9  2438          VBUSDATA        <^X0B>,0
                00BA  2439
        1803    00BA  2440 3$:      .WORD           ^X1803                  ;CS ADDRESS TABLE
        1801    00BC  2441          .WORD           ^X1801
        1801    00BE  2442          .WORD           ^X1801
        1800    00C0  2443          .WORD           ^X1800
        1807    00C2  2444          .WORD           ^X1807
        1800    00C4  2445          .WORD           ^X1800
        1800    00C6  2446          .WORD           ^X1800
        1800    00C8  2447          .WORD           ^X1800
                00CA  2448
```

```
                        3FFF  00CA  2449 4$:     .WORD           ^X3FFF                  ;MASK FOR CMPREGMSK PSEUDO OP
                              00CC  2450
                              00CC  2451 BEGIN_CPU_DATA
                              0002  2452
                              0002  2453 DCS_DATA
                              0001  2454
U 00, 7700,C800,0364,0300,0470,1801   0001  2455 ;* 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C  2459 ;* 01   MEMSCAR_R[TEMP0]                ;PHY/VIR ADDRESS TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803   0017  2463 ;* 02   MEMSCR_R[TEMP1]                 ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804   0022  2467 ;* 03   MEMSCAR_R[TEMP2]                ;REF CACHE ONLY TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805   002D  2471 ;* 04   MEMSCR_R[TEMP3]                 ;TURN CMI OFF
U 05, 7700,8800,5BE4,0310,6A70,1806   0038  2475 ;* 05   PSL(PREV_CURM ISCURM_R[TEMP4])  ;SET MODE TO SUPERVISOR
U 06, 5600,4800,0364,7B00,0520,9800   0043  2479 ;* 06   STROBE.V.BUS,WB_RDM,PTE CHECK WRITE?,NEXT/1800
                              004E  2483                                              ;PERFORM PTE CHECK
U 07, 7B00,C800,0364,0300,0470,1808   004E  2484 ;* 07   NOP,STOP.CLOCK,LATCH.CS.ADDR  ;STOP CPU CLOCK
                              0059  2488
                              0059  2489 RTEMP_DATA
                              0001  2490
                    00000000  0001  2491         .LONG   ^X00000000                    ;PHY/VIR REG ADDRESS
                    00000000  0005  2492         .LONG   ^X00000000                    ;SET MME OFF
                    0E000000  0009  2493         .LONG   ^X0E000000                    ;REF CACHE ONLY REG ADDRESS
                    01000000  000D  2494         .LONG   ^X01000000                    ;TURN CMI OFF
                    02000000  0011  2495         .LONG   ^X02000000                    ;SET SUPERVISOR MODE
                              0015  2496
                              0015  2497 END_CPU_DATA
                              00CC  2498
                              00CC  2499 END_TEST_DATA
                              00CC
                              00CC       ;-----------------------------------------------------------------------------
                              00CC  2500
                              00CC  2501 ENDTEST
```

```
0038        .SBTTL  "TEST   06A     TEST PROBE ACCESS, READ, MODE SPECIFIED MICRO ORDER
0038  2503 ;*******************************************************************************
0038  2504 ;++
0038  2505 ;
0038  2506 ; FUNCTIONAL DESCRIPTION:
0038  2507 ;
0038  2508 ;       THIS TEST CONSISTS OF 8 SUBTESTS TO VERIFY CORRECT OPERATION OF
0038  2509 ;       THE "BUS/PRB.RD.MODE" MICRO ORDER.  PTE'S WITH ALL ACCESS CODES
0038  2510 ;       ARE COMPARED AGAINST ALL MODES.
0038  2511 ;
0038  2512 ;
0038  2513 ; TEST ALGORITHM:
0038  2514 ;
0038  2515 ;       1.  CREATE A TEST LOOP
0038  2516 ;       2.  SELECT A PTE FROM A TABLE AND LOAD INTO WD REGISTER (RDM)
0038  2517 ;       3.  EXECUTE DCS PROGRAM
0038  2518 ;               1.  LOAD PHYSICAL/VIRTUAL ADDRESSING REGISTER ADDRESS INTO
0038  2519 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
0038  2520 ;                   2.  SET MEMORY MANAGEMENT BIT ON.
0038  2521 ;                   3.  LOAD REFERENCE CACHE ONLY REGISTER ADDRESS INTO MEMSCAR
0038  2522 ;                   4.  SET CMI OFF
0038  2523 ;                   5.  SET CURRENT MODE REGISTER TO USER MODE
0038  2524 ;                   6.  LOAD 0 INTO VA REGISTER
0038  2525 ;                   7.  WRITE PTE FROM WD REGISTER (RDM) TO TB LOCATION POINTED
0038  2526 ;                       TO BY VA
0038  2527 ;                   8.  PERFORM BUS/PRB.RD.MODE WHILE ENABLING MODE FROM
0038  2528 ;                       RTEMP ONTO WBUS
0038  2529 ;                   9.  STROBE VBUS AND CS LINES TO RDM
0038  2530 ;       4.  COMPARE RESULTS ON VBUS AND CS LINES (EXPECTED AND RECEIVED)
0038  2531 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING PTE'S
0038  2532 ;
0038  2533 ;
0038  2534 ; TEST PATTERNS:
0038  2535 ;
0038  2536 ;                                   PTE TABLE
0038  2537 ;
0038  2538 ;                               1       00000100
0038  2539 ;                               2       00000120
0038  2540 ;                               3       00000130
0038  2541 ;                               4       00000150
0038  2542 ;                               5       00000160
0038  2543 ;                               6       00000170
0038  2544 ;                               7       00000180
0038  2545 ;                               8       00000190
0038  2546 ;                               9       000001A0
0038  2547 ;                               10      000001B0
0038  2548 ;                               11      00000140
0038  2549 ;                               12      000001C0
0038  2550 ;                               13      000001D0
0038  2551 ;                               14      000001E0
0038  2552 ;                               15      000001F0
0038  2553 ;
0038  2554 ;       SUBTEST:
0038  2555 ;
0038  2556 ;           1    USES PTE #1 AND CHECKS FOR ACCESS VIOLATION AGAINST
```

```
0038  2557 ;                   KERNEL MODE.
0038  2558 ;
0038  2559 ;           2       USES PTE'S #2 THROUGH #15 AND CHECKS FOR ACCESS ALLOWED
0038  2560 ;                   AGAINST KERNEL MODE.
0038  2561 ;
0038  2562 ;           3       USES PTE'S #1 THROUGH #3 AND CHECKS FOR ACCESS VIOLATION
0038  2563 ;                   AGAINST EXECUTIVE MODE.
0038  2564 ;
0038  2565 ;           4       USES PTE'S #4 THROUGH #15 AND CHECKS FOR ACCESS ALLOWED
0038  2566 ;                   AGAINST EXECUTIVE MODE.
0038  2567 ;
0038  2568 ;           5       USES PTE'S #1 THROUGH #6 AND CHECKS FOR ACCESS VIOLATION
0038  2569 ;                   AGAINST SUPERVISOR MODE.
0038  2570 ;
0038  2571 ;           6       USES PTE'S #7 THROUGH #15 AND CHECKS FOR ACCESS ALLOWED
0038  2572 ;                   AGAINST SUPERVISOR MODE.
0038  2573 ;
0038  2574 ;           7       USES PTE'S #1 THROUGH #10 AND CHECKS FOR ACCESS VIOLATION
0038  2575 ;                   AGAINST USER MODE.
0038  2576 ;
0038  2577 ;           8       USES PTE'S #11 THROUGH #15 AND CHECKS FOR ACCESS ALLOWED
0038  2578 ;                   AGAINST USER MODE.
0038  2579 ;
0038  2580 ;
0038  2581 ; LOGIC DESCRIPTION:
0038  2582 ;
0038  2583 ;           THIS TEST IS ESSENTIALLY THE SAME AS THE "PROBE ACCESS, READ"
0038  2584 ;           TEST EXCEPT ACCESS IS CHECKED AGAINST WBUS<25:24> INSTEAD OF CURRENT
0038  2585 ;           MODE.  TO ENSURE THAT THE WBUS IS BEING SELECTED THE CURRENT MODE
0038  2586 ;           REGISTER IS LOADED WITH USER MODE.  IF WBUS IS NOT BEING SELECTED
0038  2587 ;           THEN SUBTEST #2 SHOULD FAIL ON THE FIRST LOOP WITH INCORRECT VBUS
0038  2588 ;           BITS.  REPLACING THE ACV GATE ARRAY SHOULD FIX YOU UP.
0038  2589 ;
0038  2590 ;
0038  2591 ; ERROR DESCRIPTION:
0038  2592 ;
0038  2593 ;           THIS TEST CAN FAIL IN ONE OF TWO PLACES.  SO YOU'LL HAVE TO
0038  2594 ;           CHECK THE PC TO DETERMINE WHICH IFERROR STATEMENT IS DETECTING
0038  2595 ;           THE FAILURE.
0038  2596 ;
0038  2597 ;           VBUS FAILURE:  BIT <8> CONTAINS THE VALUE
0038  2598 ;                          BITS <7:0> CONTAIN THE BIT POSITION
0038  2599 ;
0038  2600 ;                   EXPECTED VBUS DATA
0038  2601 ;                   RECEIVED VBUS DATA
0038  2602 ;
0038  2603 ;           CS ADDRESS FAILURE:
0038  2604 ;
0038  2605 ;                   EXPECTED CS ADDRESS
0038  2606 ;                   RECEIVED CS ADDRESS
0038  2607 ;
0038  2608 ;--
0038  2609 ;*******************************************************************************
0038  2610
0038  2611           SUBTEST                                         ;SUBTEST #1
```

```
                003B
                003B        ;//////////////////////////////////////////////////////////////////////////////
                003B  2612  ;+
                003B  2613  ;SUBTEST #1
                003B  2614  ;
                003B  2615  ;       TEST KERNEL MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
                003B  2616  ;-
                003B  2617          INITIALIZE                                  ;INIT CPU
                003D  2618          LOADDCS         9$,,<^X0A>,1                ;MODIFY DCS ADDRESS 09
                0044  2619          LOADREG         WDREG,1$,,L                 ;LOAD PTE INTO WD REGISTER
                004C  2620          ERRLOOP                                     ;ERROR LOOP
                004E  2621          FETCH           0                           ;START DCS PROGRAM
                0053  2622          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                0057  2623          CMPVBUS         2$                          ;COMPARE VBUS RESULTS
                005C  2624          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
                0063  2625          CMPREGMSK       CSTRP,4$,,10$,,W,           ;COMPARE CS ADDRESS RESULTS
                006F  2626          IFERROR         ,,<<DPM>>                   ;FAILING MODULE
                0075  2627
                0075  2628          SUBTEST                                     ;SUBTEST #2
                0078
                0078        ;//////////////////////////////////////////////////////////////////////////////
                0078  2629  ;+
                0078  2630  ;SUBTEST #2
                0078  2631  ;
                0078  2632  ;       TEST KERNEL MODE AGAINST PTE'S WITH ACCESS ALLOWED
                0078  2633  ;-
                0078  2634          INITIALIZE                                  ;INIT CPU
                007A  2635          LOOP            I,2,15                      ;CREATE TEST LOOP
                0081  2636          LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO WD REGISTER
                0089  2637          ERRLOOP                                     ;ERROR LOOP
                008B  2638          FETCH           0                           ;START DCS PROGRAM
                0090  2639          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                0094  2640          CMPVBUS         3$                          ;COMPARE VBUS RESULTS
                0099  2641          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
                00A0  2642          CMPREGMSK       CSTRP,5$,,10$,,W,           ;COMPARE CS ADDRESS RESULTS
                00AC  2643          IFERROR         ,,<<DPM>>                   ;FAILING MODULE
                00B2  2644          ENDLOOP         I                           ;END TEST LOOP
                00B5  2645
                00B5  2646          SUBTEST                                     ;SUBTEST #3
                00B8
                00B8        ;//////////////////////////////////////////////////////////////////////////////
                00B8  2647  ;+
                00B8  2648  ;SUBTEST #3
                00B8  2649  ;
                00B8  2650  ;       TEST EXECUTIVE MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
                00B8  2651  ;-
                00B8  2652          INITIALIZE                                  ;INIT CPU
                00BA  2653          LOADDCS         6$,,<^X0A>,1                ;MODIFY DCS ADDRESS 09
                00C1  2654          LOOP            I,1,3                       ;CREATE A TEST LOOP
                00C8  2655          LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO WD REGISTER
                00D0  2656          ERRLOOP                                     ;ERROR LOOP
                00D2  2657          FETCH           0                           ;START DCS PROGRAM
                00D7  2658          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
                00DB  2659          CMPVBUS         2$                          ;COMPARE VBUS RESULTS
                00E0  2660          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
```

```
00E7          CMPREGMSK       CSTRP,4$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
00F3          IFERROR         ,,<<DPM>>               ;FAILING MODULE
00F9          ENDLOOP         I                       ;END TEST LOOP
00FC
00FC          SUBTEST                                 ;SUBTEST #4
00FF
00FF      ;////////////////////////////////////////////////////////////////////////////
00FF      ;+
00FF      ;SUBTEST #4
00FF      ;
00FF      ;       TEST EXECUTIVE MODE AGAINST PTE'S WITH ACCESS ALLOWED
00FF      ;-
00FF          INITIALIZE                              ;INIT CPU
0101          LOOP            I,4,15                  ;CREATE A TEST LOOP
0108          LOADREG         WDREG,1$,I,L            ;LOAD PTE INTO WD REGISTER
0110          ERRLOOP                                 ;ERROR LOOP
0112          FETCH           0                       ;START DCS PROGRAM
0117          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
011B          CMPVBUS         3$                      ;COMPARE VBUS RESULTS
0120          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
0127          CMPREGMSK       CSTRP,5$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
0133          IFERROR         ,,<<DPM>>               ;FAILING MODULE
0139          ENDLOOP         I                       ;END TEST LOOP
013C
013C          SUBTEST                                 ;SUBTEST #5
013F
013F      ;////////////////////////////////////////////////////////////////////////////
013F      ;+
013F      ;SUBTEST #5
013F      ;
013F      ;       TEST SUPERVISOR MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
013F      ;-
013F          INITIALIZE                              ;INIT CPU
0141          LOADDCS         7$,,<^X0A>,1            ;MODIFY DCS ADDRESS 09
0148          LOOP            I,1,6                   ;CREATE A TEST LOOP
014F          LOADREG         WDREG,1$,I,L            ;LOAD PTE INTO WD REGISTER
0157          ERRLOOP                                 ;ERROR LOOP
0159          FETCH           0                       ;START DCS PROGRAM
015E          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
0162          CMPVBUS         2$                      ;COMPARE VBUS RESULTS
0167          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
016E          CMPREGMSK       CSTRP,4$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
017A          IFERROR         ,,<<DPM>>               ;FAILING MODULE
0180          ENDLOOP         I                       ;END TEST LOOP
0183
0183          SUBTEST                                 ;SUBTEST #6
0186
0186      ;////////////////////////////////////////////////////////////////////////////
0186      ;+
0186      ;SUBTEST #6
0186      ;
0186      ;       TEST SUPERVISOR MODE AGAINST PTE'S WITH ACCESS ALLOWED
0186      ;-
0186          INITIALIZE                              ;INIT CPU
0188          LOOP            I,7,15                  ;CREATE A TEST LOOP
```

```
018F  2710          LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO WD REGISTER
0197  2711          ERRLOOP                                     ;ERROR LOOP
0199  2712          FETCH           0                           ;START DCS PROGRAM
019E  2713          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
01A2  2714          CMPVBUS         3$                          ;COMPARE VBUS RESULTS
01A7  2715          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
01AE  2716          CMPREGMSK       CSTRP,5$,,10$,,W,           ;COMPARE CS ADDRESS RESULTS
01BA  2717          IFERROR         ,,<<DPM>>                   ;FAILING MODULE
01C0  2718          ENDLOOP         I                           ;END TEST LOOP
01C3  2719
01C3  2720          SUBTEST                                     ;SUBTEST #7
01C6
01C6        ;////////////////////////////////////////////////////////////////////////////////
01C6  2721  ;+
01C6  2722  ;SUBTEST #7
01C6  2723  ;
01C6  2724  ;       TEST USER MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
01C6  2725  ;-
01C6  2726          INITIALIZE                                  ;INIT CPU
01C8  2727          LOADDCS         8$,,<^X0A>,1                ;LOAD PTE INTO WD REGISTER
01CF  2728          LOOP            I,1,10                      ;CREATE A TEST LOOP
01D6  2729          LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO WD REGISTER
01DE  2730          ERRLOOP                                     ;ERROR LOOP
01E0  2731          FETCH           0                           ;START DCS PROGRAM
01E5  2732          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
01E9  2733          CMPVBUS         2$                          ;COMPARE VBUS RESULTS
01EE  2734          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
01F5  2735          CMPREGMSK       CSTRP,4$,,10$,,W,           ;COMPARE CS ADDRESS RESULTS
0201  2736          IFERROR         ,,<<DPM>>                   ;FAILING MODULE
0207  2737          ENDLOOP         I                           ;END TEST LOOP
020A  2738
020A  2739          SUBTEST                                     ;SUBTEST #8
020D
020D        ;////////////////////////////////////////////////////////////////////////////////
020D  2740  ;+
020D  2741  ;SUBTEST #8
020D  2742  ;
020D  2743  ;       TEST USER MODE AGAINST PTE'S WITH ACCESS ALLOWED
020D  2744  ;-
020D  2745          INITIALIZE                                  ;INIT CPU
020F  2746          LOOP            I,11,15                     ;CREATE A TEST LOOP
0216  2747          LOADREG         WDREG,1$,I,L                ;LOAD PTE INTO WD REGISTER
021E  2748          ERRLOOP                                     ;ERROR LOOP
0220  2749          FETCH           0                           ;START DCS PROGRAM
0225  2750          BRSTCLK         <^X0B>                      ;END DCS PROGRAM
0229  2751          CMPVBUS         3$                          ;COMPARE VBUS RESULTS
022E  2752          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
0235  2753          CMPREGMSK       CSTRP,5$,,10$,,W,           ;COMPARE CS ADDRESS RESULTS
0241  2754          IFERROR         ,,<<DPM>>                   ;FAILING MODULE
0247  2755          ENDLOOP         I                           ;END TEST LOOP
024A  2756          SKIP
0251  2757
0251  2758 BEGIN_TEST_DATA
0251
0251        ;-----------------------------------------------------------------------
```

```
                        0251  2759
00000100                0251  2760 1$:      .LONG           ^X00000100              ;PTE TABLE
00000120                0255  2761           .LONG           ^X00000120
00000130                0259  2762           .LONG           ^X00000130
00000150                025D  2763           .LONG           ^X00000150
00000160                0261  2764           .LONG           ^X00000160
00000170                0265  2765           .LONG           ^X00000170
00000180                0269  2766           .LONG           ^X00000180
00000190                026D  2767           .LONG           ^X00000190
000001A0                0271  2768           .LONG           ^X000001A0
000001B0                0275  2769           .LONG           ^X000001B0
00000140                0279  2770           .LONG           ^X00000140
000001C0                027D  2771           .LONG           ^X000001C0
000001D0                0281  2772           .LONG           ^X000001D0
000001E0                0285  2773           .LONG           ^X000001E0
000001F0                0289  2774           .LONG           ^X000001F0
                        028D  2775
                     04 028D  2776 2$:      .BYTE           ^X4                     ;VBUS DATA SUBTEST 1,3,5,7
                        028E  2777           VBUSDATA        <^X08>,0
                        028F  2778           VBUSDATA        <^X09>,0
                        0290  2779           VBUSDATA        <^X0A>,1
                        0291  2780           VBUSDATA        <^X0B>,0
                        0292  2781
                     04 0292  2782 3$:      .BYTE           ^X4                     ;VBUS DATA SUBTEST 2,4,6,8
                        0293  2783           VBUSDATA        <^X08>,1
                        0294  2784           VBUSDATA        <^X09>,0
                        0295  2785           VBUSDATA        <^X0A>,1
                        0296  2786           VBUSDATA        <^X0B>,1
                        0297  2787
                   1802 0297  2788 4$:      .WORD           ^X1802                  ;CS ADDRESS SUBTEST 1,3,5,7
                   180B 0299  2789 5$:      .WORD           ^X180B                  ;CS ADDRESS SUBTEST 2,4,6,8
                        029B  2790
                        029B  2791 6$:
U 0A, 7601,0800,5BE4,7A18,05E0,9800 029B  2792 ;% 0A    STROBE.V.BUS,WB_R[TEMP6],SIZE[LONG],PROBE READ MODE ON WB?,NEXT/1800
                        02A6  2796                                                  ;MODE SELECT FOR SUBTEST 3 & 4
                        02A6  2797 7$:
U 0A, 7601,4800,5BE4,7A1C,05E0,9800 02A6  2798 ;% 0A    STROBE.V.BUS,WB_R[TEMP7],SIZE[LONG],PROBE READ MODE ON WB?,NEXT/1800
                        02B1  2802                                                  ;MODE SELECT FOR SUBTEST 5 & 6
                        02B1  2803 8$:
U 0A, 7601,4800,5BE4,7A20,05E0,9800 02B1  2804 ;% 0A    STROBE.V.BUS,WB_R[TEMP8],SIZE[LONG],PROBE READ MODE ON WB?,NEXT/1800
                        02BC  2808                                                  ;MODE SELECT FOR SUBTEST 7 & 8
                        02BC  2809 9$:
U 0A, 7601,0800,5BE4,7A14,05E0,9800 02BC  2810 ;% 0A    STROBE.V.BUS,WB_R[TEMP5],SIZE[LONG],PROBE READ MODE ON WB?,NEXT/1800
                        02C7  2814                                                  ;MODE SELECT FOR SUBTEST 1 & 2
                   3FFF 02C7  2815 10$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                        02C9  2816
                        02C9  2817 BEGIN_CPU_DATA
                        0002  2818
                        0002  2819 DCS_DATA
                        0001  2820
U 00, 7700,C800,0364,0300,0470,1801 0001  2821 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  2825 ;% 01    MEMSCAR_R[TEMP0]            ;PHY/VIR ADDRESS TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803 0017  2829 ;% 02    MEMSCR_R[TEMP1]             ;SET MME ON
U 03, 7700,8800,5BE4,0308,6870,1804 0022  2833 ;% 03    MEMSCAR_R[TEMP2]            ;REF CACHE ONLY TO MEMSCAR
U 04, 7700,8800,5BE4,0304,6070,1805 002D  2837 ;% 04    MEMSCR_R[TEMP1]             ;TURN CMI OFF
```

```
U 05. 7700.C800.5BE4.030C.6A70.1806  0038  2841 ;% 05   PSL(PREV_CURM ISCURM_R[TEMP3])                ;SET MODE TO USER
U 06. 7700.C800.5BE6.0310.0470.1807  0043  2845 ;% 06   D_R[TEMP4]                                    ;ADDRESS TO ACCESS TB
U 07. 7700.1868.DB10.0300.0470.1808  004E  2849 ;% 07   M[TEMP8]_D+ZLIT8[0]                          ;store in address in mtemp8
U 08. 7700.9800.DB12.0300.5360.1809  0059  2853 ;% 08   VA_D_D+ZLIT8[0] INVALIDATE TB               ;clear tb at the address in VA
U 09. 5700.C81B.0364.0300.5070.180A  0064  2857 ;% 09   WB_RDM,WCTRL/TB_WB,MSRC/VA                  ;PTE TO TB
U 0A. 7601.0800.5BE4.7A14.05E0.9800  006F  2861 ;% 0A   STROBE.V.BUS,WB_R[TEMP5],SIZE[LONG],PROBE READ MODE ON WB?,NEXT/1800
                                     007A  2865                                                    ;PERFORM PROBE
U 0B. 7B00.4800.0364.0300.0470.180C  007A  2866 ;% 0B   NOP,STOP.CLOCK,LATCH.CS.ADDR               ;STOP CPU CLOCK
                                     0085  2870
                                     0085  2871 RTEMP_DATA
                                     0001  2872
                           00000000  0001  2873          .LONG   ^X00000000                       ;PHY/VIR REG ADDRESS
                           01000000  0005  2874          .LONG   ^X01000000                       ;DATA
                           0E000000  0009  2875          .LONG   ^X0E000000                       ;REF CACHE ONLY REG ADDRESS
                           03000000  000D  2876          .LONG   ^X03000000                       ;SET USER MODE
                           00000000  0011  2877          .LONG   ^X00000000                       ;ADDRESS FOR VA
                           00000000  0015  2878          .LONG   ^X00000000                       ;KERNEL MODE
                           01000000  0019  2879          .LONG   ^X01000000                       ;EXECUTIVE MODE
                           02000000  001D  2880          .LONG   ^X02000000                       ;SUPERVISOR MODE
                           03000000  0021  2881          .LONG   ^X03000000                       ;USER MODE
                                     0025  2882
                                     0025  2883 END_CPU_DATA
                                     02C9  2884
                                     02C9  2885 END_TEST_DATA
                                     02C9
                                     02C9          ;-----------------------------------------------------------------------
                                     02C9  2886
                                     02C9  2887 ENDTEST
```

```
0033        .SBTTL  "TEST   06B     TEST PROBE ACCESS, READ AND WRITE MICRO ORDERS
0033  2889 ;*******************************************************************************
0033  2890 ;++
0033  2891 ;
0033  2892 ; FUNCTIONAL DESCRIPTION:
0033  2893 ;
0033  2894 ;       THIS TEST WILL IMPLICITLY CHECK THE BUS/PRB.RD AND BUS/PRB.WR MICRO
0033  2895 ;       ORDERS BY PERFORMING A "PROBE ACCESS, READ AND WRITE ON VARIOUS PTE'S
0033  2896 ;       AND CHECKING THE MICRO VECTOR AND CS LINES FOR CORRECT RESULTS.
0033  2897 ;
0033  2898 ;
0033  2899 ; TEST ALGORITHM:
0033  2900 ;
0033  2901 ;       1.  SET UP DCS FOR EACH SUBTEST
0033  2902 ;       2.  EXECUTE DCS PROGRAM
0033  2903 ;               1.  SET UP PHYSICAL/VIRTUAL ADDRESSING REGISTER FOR
0033  2904 ;                   MEMORY MANAGEMENT ON OR OFF
0033  2905 ;               2.  SET TB GROUP DISABLE REGISTER TO FORCE GROUP 0
0033  2906 ;               3.  SET CURRENT MODE REGISTER TO USER MODE
0033  2907 ;               4.  LOAD VA REGISTER WITH ADDRESS FOR TEST
0033  2908 ;               5.  LOAD TB WITH A PTE
0033  2909 ;               6.  PERFORM PROBE ACCESS, READ OR WRITE WHILE READING VBUS
0033  2910 ;                   AND CS LINES TO RDM
0033  2911 ;       3.  COMPARE RESULTS ON VBUS AND CS LINES
0033  2912 ;       4.  IF NO ERROR GO TO NEXT SUBTEST
0033  2913 ;
0033  2914 ;
0033  2915 ; TEST PATTERNS:
0033  2916 ;
0033  2917 ;       SUBTEST         PTE             VBUS            CS LINES
0033  2918 ;
0033  2919 ;        1 & 2          00000140        1011              180B
0033  2920 ;        3 & 4          00000108        0010              1802
0033  2921 ;        5 & 6          00000048        0000              1800
0033  2922 ;        7 & 8          00000148        0111              1807
0033  2923 ;        9 & 10         00000008        1111              180F
0033  2924 ;
0033  2925 ;
0033  2926 ; LOGIC DESCRIPTION:
0033  2927 ;
0033  2928 ;       THIS TEST IS DESIGNED TO CHECK THE CORRECT OPERATION OF THE
0033  2929 ;       BUS/PROBE ACCESS, READ AND WRITE MICRO ORDERS.  THE ONLY WAY TO CHECK
0033  2930 ;       THE RESULTS IS TO VERIFY THAT THE CORRECT MICRO VECTOR AND CS ADDRESS
0033  2931 ;       ARE GENERATED.  NATURALLY THIS EXERCISES A LARGE CHUNK OF LOGIC.
0033  2932 ;       THE MICRO VECTOR LINES ARE CHECKED BY LOOKING AT THE VISABILITY
0033  2933 ;       BUS<08:0B>.  IF THESE ARE INCORRECT THEN THE FAILURE IS MOST
0033  2934 ;       LIKELY ON THE MIC MODULE.  THE CS LINES ARE ALSO CHECKED TO BE
0033  2935 ;       SURE THE CORRECT ADDRESS IS GENERATED BY THE MICRO SEQUENCER.
0033  2936 ;       IF THESE ARE INCORRECT THEN THE DPM MODULE IS PROBABLY AT FAULT.
0033  2937 ;       CONSULT THE "BUT/UVCTR CHART" IN THE CPU DOCUMENTATION TO
0033  2938 ;       DETERMINE THE ALGORITHM USED TO GENERATE EACH MICRO VECTOR.
0033  2939 ;       THE ONLY GATE ARRAYS THAT ARE INVOLVED WITH THIS TEST ARE ACV AND
0033  2940 ;       UTR, BOTH OF WHICH ARE ON THE MIC MODULE.  THEY ARE RESPONSIBLE
0033  2941 ;       FOR GENERATING THE CORRECT MICRO VECTOR.
0033  2942 ;
```

```
0033  2943 ;
0033  2944 ; ASSUMPTIONS:
0033  2945 ;
0033  2946 ;       THIS TEST ASSUMES WBUS, DATA PATH, VA, AND TB ARE OPERATIONAL.
0033  2947 ;
0033  2948 ;
0033  2949 ; ERROR DESCRIPTION:
0033  2950 ;
0033  2951 ;       THIS TEST CAN FAIL IN ONE OF TWO PLACES.  SO YOU'LL HAVE TO
0033  2952 ;       CHECK THE PC TO DETERMINE WHICH IFERROR STATEMENT IS DETECTING
0033  2953 ;       THE FAILURE.
0033  2954 ;
0033  2955 ;       VBUS FAILURE: BIT <8> CONTAINS THE VALUE
0033  2956 ;                     BITS <7:0> CONTAIN THE BIT POSITION
0033  2957 ;
0033  2958 ;               EXPECTED VBUS DATA
0033  2959 ;               RECEIVED VBUS DATA
0033  2960 ;
0033  2961 ;       CS ADDRESS FAILURE:
0033  2962 ;
0033  2963 ;               EXPECTED CS ADDRESS
0033  2964 ;               RECEIVED CS ADDRESS
0033  2965 ;
0033  2966 ;--
0033  2967 ;*****************************************************************************
0033  2968
0033  2969         SUBTEST                                         ;SUBTEST #1
0036
0036        ;/////////////////////////////////////////////////////////////////////////////
0036  2970 ;+
0036  2971 ;SUBTEST #1
0036  2972 ;
0036  2973 ;       THIS SUBTEST PERFORMS A PROBE ACCESS READ ON THE FOLLOWING
0036  2974 ;       CONDITIONS:
0036  2975 ;               1.  NOT CROSSING A PAGE BOUNDARY
0036  2976 ;               2.  VALID BIT = 1
0036  2977 ;               3.  NO ACCESS VIOLATION
0036  2978 ;               4.  MEMORY MANAGEMENT ON
0036  2979 ;               5.  MODIFY BIT = 0
0036  2980 ;               6.  PTE = 00000140
0036  2981 ;-
0036  2982         INITIALIZE                                      ;INIT CPU
0038  2983         LOADDCS         18$,,02,1                       ;MODIFY DCS ADDRESS 02
003F  2984         LOADDCS         19$,,06,3                       ;MODIFY DCS ADDRESSES 06,07,08,
0046  2985         ERRLOOP                                         ;ERROR LOOP
0048  2986         FETCH           0                               ;START DCS PROGRAM
004D  2987         BRSTCLK         9                               ;END DCS PROGRAM
0051  2988         CMPVBUS         1$                              ;COMPARE VBUS RESULTS
0056  2989         IFERROR         ,<<ACV><UTR>>,                  ;POSSIBLE FAILING ARRAYS
005D  2990         CMPREGMSK       CSTRP,2$,,20$,,W,               ;COMPARE CS ADDRESS RESULTS
0069  2991         IFERROR         ,<<>>,<<DPM>>                   ;FAILING MODULE
0070  2992
0070  2993         SUBTEST                                         ;SUBTEST #2
0073
0073        ;/////////////////////////////////////////////////////////////////////////////
```

```
                     2994 ;+
                     2995 ;SUBTEST #2
                     2996 ;
                     2997 ;       THIS SUBTEST PERFORMS A PROBE ACCESS WRITE ON THE CONDITIONS
                     2998 ;       SET UP IN SUBTEST #1
                     2999 ;-
0073                 3000         INITIALIZE                                     ;INIT CPU
0075                 3001         LOADDCS         16$,,08,1                      ;MODIFY DCS ADDRESS 08
007C                 3002         ERRLOOP                                        ;ERROR LOOP
007E                 3003         FETCH           0                              ;START DCS PROGRAM
0083                 3004         BRSTCLK         9                              ;END DCS PROGRAM
0087                 3005         CMPVBUS         1$                             ;COMPARE VBUS RESULTS
008C                 3006         IFERROR         ,<<ACV><UTR>>,                 ;POSSIBLE FAILING ARRAYS
0093                 3007         CMPREGMSK       CSTRP,2$,,20$,,W,              ;COMPARE CS ADDRESS RESULTS
009F                 3008         IFERROR         ,<<>>,<<DPM>>                  ;FAILING MODULE
00A6                 3009
00A6                 3010         SUBTEST                                        ;SUBTEST #3
00A9
00A9                     ;////////////////////////////////////////////////////////////////////////////////
00A9                 3011 ;+
00A9                 3012 ;SUBTEST #3
00A9                 3013 ;
00A9                 3014 ;       THIS SUBTEST PERFORMS A PROBE ACCESS WRITE ON THE FOLLOWING
00A9                 3015 ;       CONDITIONS:
00A9                 3016 ;               1.  NOT CROSSING A PAGE BOUNDARY
00A9                 3017 ;               2.  VALID BIT = 1
00A9                 3018 ;               3.  ACCESS VIOLATION
00A9                 3019 ;               4.  MEMORY MANAGEMENT ON
00A9                 3020 ;               5.  MODIFY BIT = 1
00A9                 3021 ;               6.  PTE = 00000108
00A9                 3022 ;-
00A9                 3023         INITIALIZE                                     ;INIT CPU
00AB                 3024         LOADDCS         3$,,07,1                       ;MODIFY DCS ADDRESS 07
00B2                 3025         ERRLOOP                                        ;ERROR LOOP
00B4                 3026         FETCH           0                              ;START DCS PROGRAM
00B9                 3027         BRSTCLK         9                              ;END DCS PROGRAM
00BD                 3028         CMPVBUS         4$                             ;COMPARE VBUS RESULTS
00C2                 3029         IFERROR         ,<<ACV><UTR>>,                 ;POSSIBLE FAILING ARRAYS
00C9                 3030         CMPREGMSK       CSTRP,5$,,20$,,W,              ;COMPARE CS ADDRESS RESULTS
00D5                 3031         IFERROR         ,<<>>,<<DPM>>                  ;FAILING MODULE
00DC                 3032
00DC                 3033         SUBTEST                                        ;SUBTEST #4
00DF
00DF                     ;////////////////////////////////////////////////////////////////////////////////
00DF                 3034 ;+
00DF                 3035 ;SUBTEST #4
00DF                 3036 ;
00DF                 3037 ;       THIS SUBTEST PERFORMS A PROBE ACCESS READ ON THE CONDITIONS
00DF                 3038 ;       SET UP IN SUBTEST #3
00DF                 3039 ;-
00DF                 3040         INITIALIZE                                     ;INIT CPU
00E1                 3041         LOADDCS         17$,,08,1                      ;MODIFY DCS ADDRESS 08
00E8                 3042         ERRLOOP                                        ;ERROR LOOP
00EA                 3043         FETCH           0                              ;START DCS PROGRAM
00EF                 3044         BRSTCLK         9                              ;END DCS PROGRAM
```

```
00F3  3045          CMPVBUS         4$                      ;COMPARE VBUS RESULTS
00F8  3046          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
00FF  3047          CMPREGMSK       CSTRP,5$,,20$,,W,       ;COMPARE CS ADDRESS RESULTS
010B  3048          IFERROR         ,<<>>,<<DPM>>           ;FAILING MODULE
0112  3049
0112  3050          SUBTEST                                 ;SUBTEST #5
0115
0115              ;////////////////////////////////////////////////////////////////////////////////
0115  3051        ;+
0115  3052        ;SUBTEST #5
0115  3053        ;
0115  3054        ;       THIS SUBTEST PERFORMS A PROBE ACCESS READ ON THE FOLLOWING
0115  3055        ;       CONDITIONS:
0115  3056        ;               1.  NOT CROSSING A PAGE BOUNDARY
0115  3057        ;               2.  VALID BIT = 0
0115  3058        ;               3.  NO ACCESS VIOLATION
0115  3059        ;               4.  MEMORY MANAGEMENT ON
0115  3060        ;               5.  MODIFY BIT = 1
0115  3061        ;               6.  PTE = 00000048
0115  3062        ;-
0115  3063          INITIALIZE                              ;INIT CPU
0117  3064          LOADDCS         6$,,07,1                ;MODIFY DCS ADDRESS 07
011E  3065          ERRLOOP                                 ;ERROR LOOP
0120  3066          FETCH           0                       ;START DCS PROGRAM
0125  3067          BRSTCLK         9                       ;END DCS PROGRAM
0129  3068          CMPVBUS         7$                      ;COMPARE VBUS RESULTS
012E  3069          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
0135  3070          CMPREGMSK       CSTRP,8$,,20$,,W,       ;COMPARE CS ADDRESS RESULTS
0141  3071          IFERROR         ,<<>>,<<DPM>>           ;FAILING MODULE
0148  3072
0148  3073          SUBTEST                                 ;SUBTEST #6
014B
014B              ;////////////////////////////////////////////////////////////////////////////////
014B  3074        ;+
014B  3075        ;SUBTEST #6
014B  3076        ;
014B  3077        ;       THIS SUBTEST PERFORMS A PROBE ACCESS WRITE ON THE CONDITIONS
014B  3078        ;       SET UP IN SUBTEST #5
014B  3079        ;-
014B  3080          INITIALIZE                              ;INIT CPU
014D  3081          LOADDCS         16$,,08,1               ;MODIFY DCS ADDRESS 08
0154  3082          ERRLOOP                                 ;ERROR LOOP
0156  3083          FETCH           0                       ;START DCS PROGRAM
015B  3084          BRSTCLK         9                       ;END DCS PROGRAM
015F  3085          CMPVBUS         7$                      ;COMPARE VBUS RESULTS
0164  3086          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
016B  3087          CMPREGMSK       CSTRP,8$,,20$,,W,       ;COMPARE CS ADDRESS RESULTS
0177  3088          IFERROR         ,<<>>,<<DPM>>           ;FAILING MODULE
017E  3089
017E  3090          SUBTEST                                 ;SUBTEST #7
0181
0181              ;////////////////////////////////////////////////////////////////////////////////
0181  3091        ;+
0181  3092        ;SUBTEST #7
0181  3093        ;
```

```
                     3094 ;       THIS SUBTEST PERFORMS A PROBE ACCESS WRITE ON THE FOLLOWING
                     3095 ;       CONDITIONS:
                     3096 ;               1.  CROSSING A PAGE BOUNDARY
                     3097 ;               2.  VALID BIT = 1
                     3098 ;               3.  NO ACCESS VIOLATION
                     3099 ;               4.  MEMORY MANAGEMENT ON
                     3100 ;               5.  MODIFY BIT = 1
                     3101 ;               6.  PTE = 00000148
                     3102 ;-
0181                 3103         INITIALIZE                                  ;INIT CPU
0183                 3104         LOADDCS         9$,,06,2                    ;MODIFY DCS ADDRESS 06,07
018A                 3105         ERRLOOP                                     ;ERROR LOOP
018C                 3106         FETCH           0                           ;START DCS PROGRAM
0191                 3107         BRSTCLK         9                           ;END DCS PROGRAM
0195                 3108         CMPVBUS         10$                         ;COMPARE VBUS RESULTS
019A                 3109         IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
01A1                 3110         CMPREGMSK       CSTRP,11$,,20$,,W,          ;COMPARE CS ADDRESS RESULTS
01AD                 3111         IFERROR         ,<<>>,<<DPM>>               ;FAILING MODULE
01B4                 3112
01B4                 3113         SUBTEST                                     ;SUBTEST #8
01B7
01B7                 ;////////////////////////////////////////////////////////////////////////////////
01B7                 3114 ;+
01B7                 3115 ;SUBTEST #8
01B7                 3116 ;
01B7                 3117 ;       THIS SUBTEST PERFORMS A PROBE ACCESS READ ON THE CONDITIONS
01B7                 3118 ;       SET UP IN SUBTEST #7
01B7                 3119 ;-
01B7                 3120         INITIALIZE                                  ;INIT CPU
01B9                 3121         LOADDCS         17$,,08,1                   ;MODIFY DCS ADDESS 08
01C0                 3122         ERRLOOP                                     ;ERROR LOOP
01C2                 3123         FETCH           0                           ;START DCS PROGRAM
01C7                 3124         BRSTCLK         9                           ;END DCS PROGRAM
01CB                 3125         CMPVBUS         10$                         ;COMPARE VBUS RESULTS
01D0                 3126         IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
01D7                 3127         CMPREGMSK       CSTRP,11$,,20$,,W,          ;COMPARE CS ADDRESS RESULTS
01E3                 3128         IFERROR         ,<<>>,<<DPM>>               ;FAILING MODULE
01EA                 3129
01EA                 3130         SUBTEST                                     ;SUBTEST #9
01ED
01ED                 ;////////////////////////////////////////////////////////////////////////////////
01ED                 3131 ;+
01ED                 3132 ;SUBTEST #9
01ED                 3133 ;
01ED                 3134 ;       THIS SUBTEST PERFORMS A PROBE ACCESS READ ON THE FOLLOWING
01ED                 3135 ;       CONDITIONS:
01ED                 3136 ;               1.  CROSSING A PAGE BOUNDARY
01ED                 3137 ;               2.  VALID BIT = 0
01ED                 3138 ;               3.  ACCESS VIOLATION
01ED                 3139 ;               4.  MEMORY MANAGEMENT OFF
01ED                 3140 ;               5.  MODIFY BIT = 1
01ED                 3141 ;               6.  PTE = 00000008
01ED                 3142 ;-
01ED                 3143         INITIALIZE                                  ;INIT CPU
01EF                 3144         LOADDCS         12$,,02,1                   ;MODIFY DCS ADDRESS 02
```

```
                    01F6  3145          LOADDCS         13$,,07,1                   ;MODIFY DCS ADDRESS 07
                    01FD  3146          ERRLOOP                                     ;ERROR LOOP
                    01FF  3147          FETCH           0                           ;START DCS PROGRAM
                    0204  3148          BRSTCLK         9                           ;END DCS PROGRAM
                    0208  3149          CMPVBUS         14$                         ;COMPARE VBUS RESULTS
                    020D  3150          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
                    0214  3151          CMPREGMSK       CSTRP,15$,,20$,,W,          ;COMPARE CS ADDRESS RESULTS
                    0220  3152          IFERROR         ,<<>>,<<DPM>>               ;FAILING MODULE
                    0227  3153
                    0227  3154          SUBTEST                                     ;SUBTEST #10
                    022A
                    022A          ;////////////////////////////////////////////////////////////////////////////////
                    022A  3155 ;+
                    022A  3156 ;SUBTEST #10
                    022A  3157 ;
                    022A  3158 ;        THIS SUBTEST PERFORMS A PROBE ACCESS WRITE ON THE CONDITIONS
                    022A  3159 ;        SET UP IN SUBTEST #9
                    022A  3160 ;-
                    022A  3161          INITIALIZE                                  ;INIT CPU
                    022C  3162          LOADDCS         16$,,08,1                   ;MODIFY DCS ADDRESS 08
                    0233  3163          ERRLOOP                                     ;ERROR LOOP
                    0235  3164          FETCH           0                           ;START DCS PROGRAM
                    023A  3165          BRSTCLK         9                           ;END DCS PROGRAM
                    023E  3166          CMPVBUS         14$                         ;COMPARE VBUS RESULTS
                    0243  3167          IFERROR         ,<<ACV><UTR>>,              ;POSSIBLE FAILING ARRAYS
                    024A  3168          CMPREGMSK       CSTRP,15$,,20$,,W,          ;COMPARE CS ADDRESS RESULTS
                    0256  3169          IFERROR         ,<<>>,<<DPM>>               ;FAILING MODULE
                    025D  3170          SKIP                                        ;GO TO NEXT TEST
                    0264  3171
                    0264  3172 BEGIN_TEST_DATA
                    0264
                    0264          ;-----------------------------------------------------------------------------
                    0264  3173
                 04 0264  3174 1$:      .BYTE           ^X4                         ;VBUS DATA TABLE SUB #1 & 2
                    0265  3175          VBUSDATA        <^X08>,1
                    0266  3176          VBUSDATA        <^X09>,0
                    0267  3177          VBUSDATA        <^X0A>,1
                    0268  3178          VBUSDATA        <^X0B>,1
                    0269  3179
               180B 0269  3180 2$:      .WORD           ^X180B                      ;CS ADDRESS SUB #1 & 2
                    026B  3181
                    026B  3182 3$:
U 07, 7700,481B,5BE4,0318,5070,1808 026B  3183 ;% 07   TB_R[TEMP6]                  ;DCS ADDRESS 07
                    0276  3187
                 04 0276  3188 4$:      .BYTE           ^X4                         ;VBUS DATA TABLE SUB #3 & 4
                    0277  3189          VBUSDATA        <^X08>,0
                    0278  3190          VBUSDATA        <^X09>,0
                    0279  3191          VBUSDATA        <^X0A>,1
                    027A  3192          VBUSDATA        <^X0B>,0
                    027B  3193
               1802 027B  3194 5$:      .WORD           ^X1802                      ;CS ADDRESS SUB #3 & 4
                    027D  3195
                    027D  3196 6$:
U 07, 7700,081B,5BE4,031C,5070,1808 027D  3197 ;% 07   TB_R[TEMP7]                  ;DCS ADDRESS 07
                    0288  3201
```

```
                                  04  0288  3202 7$:     .BYTE           ^X4                     ;VBUS DATA TABLE SUB #5 & 6
                                      0289  3203         VBUSDATA        <^X08>,0
                                      028A  3204         VBUSDATA        <^X09>,0
                                      028B  3205         VBUSDATA        <^X0A>,0
                                      028C  3206         VBUSDATA        <^X0B>,0
                                      028D  3207
                                1800  028D  3208 8$:     .WORD           ^X1800                  ;CS ADDRESS SUB #5 & 6
                                      028F  3209
                                      028F  3210 9$:
U 06, 7700,8800,5BE4,0320,4A70,1807   028F  3211 ;% 06   VA_R[TEMP8]                            ;DCS ADDRESS 06
U 07, 7700,481B,5BE4,0324,5070,1808   029A  3215 ;% 07   TB_R[TEMP9]                            ;DCS ADDRESS 07
                                      02A5  3219
                                  04  02A5  3220 10$:    .BYTE           ^X4                     ;VBUS DATA TABLE SUB #7 & 8
                                      02A6  3221         VBUSDATA        <^X08>,0
                                      02A7  3222         VBUSDATA        <^X09>,1
                                      02A8  3223         VBUSDATA        <^X0A>,1
                                      02A9  3224         VBUSDATA        <^X0B>,1
                                      02AA  3225
                                1807  02AA  3226 11$:    .WORD           ^X1807                  ;CS ADDRESS SUB #7 & 8
                                      02AC  3227
                                      02AC  3228 12$:
U 02, 7700,8800,5BE4,0304,6070,1803   02AC  3229 ;% 02   MEMSCR_R[TEMP1]                        ;DCS ADDRESS 02
                                      02B7  3233
                                      02B7  3234 13$:
U 07, 7700,481B,5BE4,0328,5070,1808   02B7  3235 ;% 07   TB_R[TEMP10]                           ;DCS ADDRESS 07
                                      02C2  3239
                                  04  02C2  3240 14$:    .BYTE           ^X4                     ;VBUS DATA TABLE SUB #9 & 10
                                      02C3  3241         VBUSDATA        <^X08>,1
                                      02C4  3242         VBUSDATA        <^X09>,1
                                      02C5  3243         VBUSDATA        <^X0A>,1
                                      02C6  3244         VBUSDATA        <^X0B>,1
                                      02C7  3245
                                180F  02C7  3246 15$:    .WORD           ^X180F                  ;CS ADDRESS SUB #9 & 10
                                      02C9  3247
                                      02C9  3248 16$:
U 08, 7601,0800,0364,7A00,05D0,9800   02C9  3249 ;% 08   STROBE.V.BUS,SIZE[LONG],PROBE WRITE?,NEXT/1800
                                      02D4  3253                                                ;DCS ADDRESS 08
                                      02D4  3254 17$:
U 08, 7601,8800,0364,7A00,05F0,9800   02D4  3255 ;% 08   STROBE.V.BUS,SIZE[LONG],PROBE READ?,NEXT/1800
                                      02DF  3259                                                ;DCS ADDRESS 08
                                      02DF  3260 18$:
U 02, 7700,8800,5BE4,032C,6070,1803   02DF  3261 ;% 02   MEMSCR_R[TEMP11]                       ;DCS ADD 02 SUBTEST #1
                                      02EA  3265
                                      02EA  3266 19$:
U 06, 7700,8800,5BE4,0304,4A70,1807   02EA  3267 ;% 06   VA_R[TEMP1]                            ;DCS ADD 06 SUBTEST #1
U 07, 7700,481B,5BE4,0314,5070,1808   02F5  3271 ;% 07   TB_R[TEMP5]                            ;DCS ADD 07 SUBTEST #1
U 08, 7601,8800,0364,7A00,05F0,9800   0300  3275 ;% 08   STROBE.V.BUS,SIZE[LONG],PROBE READ?,NEXT/1800
                                      030B  3279                                                ;DCS ADD 08 SUBTEST #1
                                3FFF  030B  3280 20$:    .WORD   ^X3FFF                         ;MASK FOR CMPREGMSK PSEUDO OP
                                      030D  3281
                                      030D  3282 BEGIN_CPU_DATA
                                      0002  3283
                                      0002  3284 DCS_DATA
                                      0001  3285
U 00, 7700,C800,0364,0300,0470,1801   0001  3286 ;% 00   NOP
```

```
U 01, 7700,C800,5BE4,0300,6870,1802  000C  3290 ;* 01   MEMSCAR_R[TEMP0]                          ;PHY/VIR REG ADDRESS
U 02, 7700,8800,5BE4,032C,6070,1803  0017  3294 ;* 02   MEMSCR_R[TEMP11]                         ;SET MME BIT ON OR OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  3298 ;* 03   MEMSCAR_R[TEMP2]                         ;TB GROUP DISABLE REG ADD
U 04, 7700,C800,5BE4,030C,6070,1805  002D  3302 ;* 04   MEMSCR_R[TEMP3]                          ;FORCE TB GROUP 0
U 05, 7700,8800,5BE4,0310,6A70,1806  0038  3306 ;* 05   PSL(PREV_CURM ISCURM_R[TEMP4])          ;SET MODE TO USER
U 06, 7700,8800,5BE4,0304,4A70,1807  0043  3310 ;* 06   VA_R[TEMP1]                             ;ADDRESS TO ACCESS TB
U 07, 7700,481B,5BE4,0314,5070,1808  004E  3314 ;* 07   TB_R[TEMP5]                             ;PTE TO TB
U 08, 7601,8800,0364,7A00,05F0,9800  0059  3318 ;* 08   STROBE.V.BUS,SIZE[LONG],PROBE READ?,NEXT/1800
                                     0064  3322                                               ;PERFORM PROBE
U 09, 7B00,4800,0364,0300,0470,180A  0064  3323 ;* 09   NOP,STOP.CLOCK,LATCH.CS.ADDR           ;STOP CPU CLOCK
                                     006F  3327
                                     006F  3328 RTEMP_DATA
                                     0001  3329
                         00000000    0001  3330          .LONG   ^X00000000                   ;PHYSICAL/VIRTUAL REG ADDRESS
                         00000000    0005  3331          .LONG   ^X00000000                   ;DATA
                         03000000    0009  3332          .LONG   ^X03000000                   ;TB GROUP DISABLE REG ADDRESS
                         0A000000    000D  3333          .LONG   ^X0A000000                   ;FORCE TB GROUP 0
                         03000000    0011  3334          .LONG   ^X03000000                   ;SETS MODE TO USER
                         00000140    0015  3335          .LONG   ^X00000140                   ;PTE SUBTEST #1 & 2
                         00000108    0019  3336          .LONG   ^X00000108                   ;PTE SUBTEST #3 & 4
                         00000048    001D  3337          .LONG   ^X00000048                   ;PTE SUBTEST #5 & 6
                         000001FF    0021  3338          .LONG   ^X000001FF                   ;FORCE CROSS PAGE BOUNDARY
                         00000148    0025  3339          .LONG   ^X00000148                   ;PTE SUBTEST #7 & 8
                         00000008    0029  3340          .LONG   ^X00000008                   ;PTE SUBTEST #9 & 10
                         01000000    002D  3341          .LONG   ^X01000000                   ;SET MME ON
                                     0031  3342
                                     0031  3343 END_CPU_DATA
                                     030D  3344
                                     030D  3345 END_TEST_DATA
                                     030D
                                     030D       ;-------------------------------------------------------------------------
                                     030D  3346
                                     030D  3347 ENDTEST
```

```
                        0039        .SBTTL  "TEST   06C     TEST PROBE ACCESS, WRITE, MODE SPECIFIED MICRO ORDER
                        0039  3349 ;*******************************************************************************
                        0039  3350 ;++
                        0039  3351 ;
                        0039  3352 ; FUNCTIONAL DESCRIPTION:
                        0039  3353 ;
                        0039  3354 ;       THIS TEST CONSISTS OF 8 SUBTESTS TO VERIFY CORRECT OPERATION OF
                        0039  3355 ;       THE "BUS/PRB.WR.MODE" MICRO ORDER.  PTE'S WITH ALL ACCESS CODES
                        0039  3356 ;       ARE COMPARED AGAINST ALL MODES.
                        0039  3357 ;
                        0039  3358 ;
                        0039  3359 ; TEST ALGORITHM:
                        0039  3360 ;
                        0039  3361 ;       1.  CREATE A TEST LOOP
                        0039  3362 ;       2.  SELECT A PTE FROM A TABLE AND LOAD INTO WD REGISTER (RDM)
                        0039  3363 ;       3.  EXECUTE DCS PROGRAM
                        0039  3364 ;               1.  LOAD PHYSICAL/VIRTUAL ADDRESSING REGISTER ADDRESS INTO
                        0039  3365 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
                        0039  3366 ;               2.  SET MEMORY MANAGEMENT BIT ON
                        0039  3367 ;               3.  LOAD REFERENCE CACHE ONLY REGISTER ADDRESS INTO MEMSCAR
                        0039  3368 ;               4.  SET CMI OFF
                        0039  3369 ;               5.  SET CURRENT MODE REGISTER TO USER MODE
                        0039  3370 ;               6.  LOAD 0 INTO VA REGISTER
                        0039  3371 ;               7.  WRITE PTE FROM WD REGISTER (RDM) TO TB LOCATION POINTED
                        0039  3372 ;                   TO BY VA
                        0039  3373 ;               8.  PERFORM BUS/PRB.WR.MODE WHILE ENABLING MODE FROM
                        0039  3374 ;                   RTEMP ONTO WBUS
                        0039  3375 ;               9.  STROBE VBUS AND CS LINES TO RDM
                        0039  3376 ;       4.  COMPARE RESULTS ON VBUS AND CS LINES (EXPECTED AND RECEIVED)
                        0039  3377 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING PTE'S
                        0039  3378 ;
                        0039  3379 ;
                        0039  3380 ; TEST PATTERNS:
                        0039  3381 ;
                        0039  3382 ;           PTE TABLE
                        0039  3383 ;
                        0039  3384 ;       1       00000100
                        0039  3385 ;       2       00000130
                        0039  3386 ;       3       00000170
                        0039  3387 ;       4       000001B0
                        0039  3388 ;       5       000001F0
                        0039  3389 ;       6       00000160
                        0039  3390 ;       7       000001A0
                        0039  3391 ;       8       000001E0
                        0039  3392 ;       9       00000120
                        0039  3393 ;       10      00000150
                        0039  3394 ;       11      00000190
                        0039  3395 ;       12      000001D0
                        0039  3396 ;       13      00000180
                        0039  3397 ;       14      000001C0
                        0039  3398 ;       15      00000140
                        0039  3399 ;
                        0039  3400 ;       SUBTEST:
                        0039  3401 ;
                        0039  3402 ;           1    USES PTE'S #1 THROUGH #5 AND CHECKS FOR ACCESS VIOLATION
```

```
0039  3403 ;                    AGAINST KERNEL MODE.
0039  3404 ;
0039  3405 ;          2         USES PTE'S #6 THROUGH #15 AND CHECKS FOR ACCESS ALLOWED
0039  3406 ;                    AGAINST KERENEL MODE.
0039  3407 ;
0039  3408 ;          3         USES PTE'S #1 THROUGH #9 AND CHECKS FOR ACCESS VIOLATION
0039  3409 ;                    AGAINST EXECUTIVE MODE.
0039  3410 ;
0039  3411 ;          4         USES PTE'S #10 THROUGH #15 AND CHECKS FOR ACCESS ALLOWED
0039  3412 ;                    AGAINST EXECUTIVE MODE.
0039  3413 ;
0039  3414 ;          5         USES PTE'S #1 THROUGH #12 AND CHECKS FOR ACCESS VIOLATION
0039  3415 ;                    AGAINST SUPERVISOR MODE.
0039  3416 ;
0039  3417 ;          6         USES PTE'S #13 THROUGH #15 AND CHECKS FOR ACCESS ALLOWED
0039  3418 ;                    AGAINST SUPERVISOR MODE.
0039  3419 ;
0039  3420 ;          7         USES PTE'S #1 THROUGH #14 AND CHECKS FOR ACCESS VIOLATION
0039  3421 ;                    AGAINST USER MODE.
0039  3422 ;
0039  3423 ;          8         USES PTE #15 AND CHECKS FOR ACCESS ALLOWED AGAINST USER MODE
0039  3424 ;
0039  3425 ;
0039  3426 ; LOGIC DESCRIPTION:
0039  3427 ;
0039  3428 ;          THIS TEST IS ESSENTIALLY THE SANE AS THE "PROBE ACCESS, WRITE"
0039  3429 ;          TEST EXCEPT ACCESS IS CHECKED AGAINST WBUS<25:24> INSTEAD OF CURRENT
0039  3430 ;          MODE.  TO ENSURE THAT THE WBUS IS BEING SELECTED THE CURRENT MODE
0039  3431 ;          REGISTER IS LOADED WITH USER MODE.  IF WBUS IS NOT BEING SELECTED
0039  3432 ;          THEN SUBTEST #2 SHOULD FAIL ON THE FIRST LOOP WITH INCORRECT VBUS
0039  3433 ;          BITS (MICRO VECTOR).  REPLACING THE ACV GATE ARRAY SHOULD FIX YOU UP.
0039  3434 ;
0039  3435 ;
0039  3436 ; ERROR DESCRIPTION:
0039  3437 ;
0039  3438 ;          THIS TEST CAN FAIL IN ONE OF TWO PLACES.  SO YOU'LL HAVE TO
0039  3439 ;          CHECK THE PC TO DETERMINE WHICH IFERROR STATEMENT IS DETECTING
0039  3440 ;          THE FAILURE.
0039  3441 ;
0039  3442 ;          VBUS FAILURE:  BIT <8> CONTAINS THE VALUE
0039  3443 ;                         BITS <7:0> CONTAIN THE BIT POSITION
0039  3444 ;
0039  3445 ;                   EXPECTED VBUS DATA
0039  3446 ;                   RECEIVED VBUS DATA
0039  3447 ;
0039  3448 ;          CS ADDRESS FAILURE:
0039  3449 ;
0039  3450 ;                   EXPECTED CS ADDRESS
0039  3451 ;                   RECEIVED CS ADDRESS
0039  3452 ;
0039  3453 ;--
0039  3454 ;*****************************************************************************
0039  3455
0039  3456          SUBTEST                                          ;SUBTEST #1
003C
```

```
003C            ;//////////////////////////////////////////////////////////////////////////////
003C    3457    ;+
003C    3458    ;SUBTEST #1
003C    3459    ;
003C    3460    ;       TEST KERNEL MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
003C    3461    ;-
003C    3462            INITIALIZE                                      ;INIT CPU
003E    3463            LOADDCS         9$,,<^X0A>,1                    ;MODIFY DCS ADDRESS 09
0045    3464            LOOP            I,1,5                           ;CREATE A TEST LOOP
004C    3465            LOADREG         WDREG,1$,I,L                    ;LOAD PTE INTO WD REGISTER
0054    3466            ERRLOOP                                         ;ERROR LOOP
0056    3467            FETCH           0                               ;START DCS PROGRAM
005B    3468            BRSTCLK         <^X0B>                          ;END DCS PROGRAM
005F    3469            CMPVBUS         2$                              ;COMPARE VBUS RESULTS
0064    3470            IFERROR         ,<<ACV><UTR>>,                  ;POSSIBLE FAILING ARRAYS
006B    3471            CMPREGMSK       CSTRP,3$,,10$,,W,               ;COMPARE CS ADDRESS RESULTS
0077    3472            IFERROR         ,,<<DPM>>                       ;FAILING MODULE
007D    3473            ENDLOOP         I                               ;END TEST LOOP
0080    3474
0080    3475            SUBTEST                                         ;SUBTEST #2
0083
0083            ;//////////////////////////////////////////////////////////////////////////////
0083    3476    ;+
0083    3477    ;SUBTEST #2
0083    3478    ;
0083    3479    ;       TEST KERNEL MODE AGAINST PTE'S WITH ACCESS ALLOWED
0083    3480    ;-
0083    3481            INITIALIZE                                      ;INIT CPU
0085    3482            LOOP            I,6,15                          ;CREATE A TEST LOOP
008C    3483            LOADREG         WDREG,1$,I,L                    ;LOAD PTE INTO WD REGISTER
0094    3484            ERRLOOP                                         ;ERROR LOOP
0096    3485            FETCH           0                               ;START DCS PROGRAM
009B    3486            BRSTCLK         <^X0B>                          ;END DCS PROGRAM
009F    3487            CMPVBUS         4$                              ;COMPARE VBUS RESULTS
00A4    3488            IFERROR         ,<<ACV><UTR>>,                  ;POSSIBLE FAILING ARRAYS
00AB    3489            CMPREGMSK       CSTRP,5$,,10$,,W,               ;COMPARE CS ADDRESS RESULTS
00B7    3490            IFERROR         ,,<<DPM>>                       ;FAILING MODULE
00BD    3491            ENDLOOP         I                               ;END TEST LOOP
00C0    3492
00C0    3493            SUBTEST                                         ;SUBTEST #3
00C3
00C3            ;//////////////////////////////////////////////////////////////////////////////
00C3    3494    ;+
00C3    3495    ;SUBTEST #3
00C3    3496    ;
00C3    3497    ;       TEST EXECUTIVE MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
00C3    3498    ;-
00C3    3499            INITIALIZE                                      ;INIT CPU
00C5    3500            LOADDCS         6$,,<^X0A>,1                    ;MODIFY DCS ADDRESS 09
00CC    3501            LOOP            I,1,9                           ;CREATE A TEST LOOP
00D3    3502            LOADREG         WDREG,1$,I,L                    ;LOAD PTE INTO WD REGISTER
00DB    3503            ERRLOOP                                         ;ERROR LOOP
00DD    3504            FETCH           0                               ;START DCS PROGRAM
00E2    3505            BRSTCLK         <^X0B>                          ;END DCS PROGRAM
00E6    3506            CMPVBUS         2$                              ;COMPARE VBUS RESULTS
```

```
00EB  3507          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
00F2  3508          CMPREGMSK       CSTRP,3$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
00FE  3509          IFERROR         ,,<<DPM>>               ;FAILING MODULE
0104  3510          ENDLOOP         I                       ;END TEST LOOP
0107  3511
0107  3512          SUBTEST                                 ;SUBTEST #4
010A
010A          ;////////////////////////////////////////////////////////////////////////////////
010A  3513    ;+
010A  3514    ;SUBTEST #4
010A  3515    ;
010A  3516    ;       TEST EXECUTIVE MODE AGAINST PTE'S WITH ACCESS ALLOWED
010A  3517    ;-
010A  3518          INITIALIZE                              ;INIT CPU
010C  3519          LOOP            I,10,15                 ;CREATE A TEST LOOP
0113  3520          LOADREG         WDREG,1$,I,L            ;LOAD PTE INTO WD REGISTER
011B  3521          ERRLOOP                                 ;ERROR LOOP
011D  3522          FETCH           0                       ;START DCS PROGRAM
0122  3523          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
0126  3524          CMPVBUS         4$                      ;COMPARE VBUS RESULTS
012B  3525          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
0132  3526          CMPREGMSK       CSTRP,5$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
013E  3527          IFERROR         ,,<<DPM>>               ;FAILING MODULE
0144  3528          ENDLOOP         I                       ;END TEST LOOP
0147  3529
0147  3530          SUBTEST                                 ;SUBTEST #5
014A
014A          ;////////////////////////////////////////////////////////////////////////////////
014A  3531    ;+
014A  3532    ;SUBTEST #5
014A  3533    ;
014A  3534    ;       TEST SUPERVISOR MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
014A  3535    ;-
014A  3536          INITIALIZE                              ;INIT CPU
014C  3537          LOADDCS         7$,,<^X0A>,1            ;MODIFY DCS ADDRESS 09
0153  3538          LOOP            I,1,12                  ;CREATE A TEST LOOP
015A  3539          LOADREG         WDREG,1$,I,L            ;LOAD PTE INTO WD REGISTER
0162  3540          ERRLOOP                                 ;ERROR LOOP
0164  3541          FETCH           0                       ;START DCS PROGRAM
0169  3542          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
016D  3543          CMPVBUS         2$                      ;COMPARE VBUS RESULTS
0172  3544          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
0179  3545          CMPREGMSK       CSTRP,3$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
0185  3546          IFERROR         ,,<<DPM>>               ;FAILING MODULE
018B  3547          ENDLOOP         I                       ;END TEST LOOP
018E  3548
018E  3549          SUBTEST                                 ;SUBTEST #6
0191
0191          ;////////////////////////////////////////////////////////////////////////////////
0191  3550    ;+
0191  3551    ;SUBTEST #6
0191  3552    ;
0191  3553    ;       TEST SUPERVISOR MODE AGAINST PTE'S WITH ACCESS ALLOWED
0191  3554    ;-
0191  3555          INITIALIZE                              ;INIT CPU
```

```
                0193  3556          LOOP            I,13,14                 ;CREATE A TEST LOOP
                019A  3557          LOADREG         WDREG,1$,I,L            ;LOAD PTE INTO WD REGISTER
                01A2  3558          ERRLOOP                                 ;ERROR LOOP
                01A4  3559          FETCH           0                       ;START DCS PROGRAM
                01A9  3560          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
                01AD  3561          CMPVBUS         4$                      ;COMPARE VBUS RESULTS
                01B2  3562          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
                01B9  3563          CMPREGMSK       CSTRP,5$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
                01C5  3564          IFERROR         ,,<<DPM>>               ;FAILING MODULE
                01CB  3565          ENDLOOP         I                       ;END TEST LOOP
                01CE  3566
                01CE  3567          SUBTEST                                 ;SUBTEST #7
                01D1
                01D1              ;////////////////////////////////////////////////////////////////////////////////
                01D1  3568        ;+
                01D1  3569        ;SUBTEST #7
                01D1  3570        ;
                01D1  3571        ;       TEST USER MODE AGAINST PTE'S CAUSING AN ACCESS VIOLATION
                01D1  3572        ;-
                01D1  3573          INITIALIZE                              ;INIT CPU
                01D3  3574          LOADDCS         8$,,<^X0A>,1            ;MODIFY DCS ADDRESS 09
                01DA  3575          LOOP            I,1,14                  ;CREATE A TEST LOOP
                01E1  3576          LOADREG         WDREG,1$,I,L            ;LOAD PTE INTO WD REGISTER
                01E9  3577          ERRLOOP                                 ;ERROR LOOP
                01EB  3578          FETCH           0                       ;START DCS PROGRAM
                01F0  3579          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
                01F4  3580          CMPVBUS         2$                      ;COMPARE VBUS RESULTS
                01F9  3581          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
                0200  3582          CMPREGMSK       CSTRP,3$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
                020C  3583          IFERROR         ,,<<DPM>>               ;FAILING MODULE
                0212  3584          ENDLOOP         I                       ;END TEST LOOP
                0215  3585
                0215  3586          SUBTEST                                 ;SUBTEST #8
                0218
                0218              ;////////////////////////////////////////////////////////////////////////////////
                0218  3587        ;+
                0218  3588        ;SUBTEST #8
                0218  3589        ;
                0218  3590        ;       TEST USER MODE AGAINST A PTE WITH ACCESS ALLOWED
                0218  3591        ;-
                0218  3592          INITIALIZE                              ;INIT CPU
                021A  3593          LOOP            I,15,15                 ;CREATE A TEST LOOP
                0221  3594          LOADREG         WDREG,1$,I,L            ;LOAD PTE INTO WD REGISTER
                0229  3595          ERRLOOP                                 ;ERROR LOOP
                022B  3596          FETCH           0                       ;START DCS PROGRAM
                0230  3597          BRSTCLK         <^X0B>                  ;END DCS PROGRAM
                0234  3598          CMPVBUS         4$                      ;COMPARE VBUS RESULTS
                0239  3599          IFERROR         ,<<ACV><UTR>>,          ;POSSIBLE FAILING ARRAYS
                0240  3600          CMPREGMSK       CSTRP,5$,,10$,,W,       ;COMPARE CS ADDRESS RESULTS
                024C  3601          IFERROR         ,,<<DPM>>               ;FAILING MODULE
                0252  3602          ENDLOOP         I                       ;END TEST LOOP
                0255  3603          SKIP                                    ;GO TO NEXT TEST
                025C  3604
                025C  3605 BEGIN_TEST_DATA
                025C
```

```
                                  025C        ;-----------------------------------------------------------------------
                                  025C  3606
                        00000100  025C  3607 1$:     .LONG           ^X00000100                  ;PTE TABLE
                        00000130  0260  3608         .LONG           ^X00000130
                        00000170  0264  3609         .LONG           ^X00000170
                        000001B0  0268  3610         .LONG           ^X000001B0
                        000001F0  026C  3611         .LONG           ^X000001F0
                        00000160  0270  3612         .LONG           ^X00000160
                        000001A0  0274  3613         .LONG           ^X000001A0
                        000001E0  0278  3614         .LONG           ^X000001E0
                        00000120  027C  3615         .LONG           ^X00000120
                        00000150  0280  3616         .LONG           ^X00000150
                        00000190  0284  3617         .LONG           ^X00000190
                        000001D0  0288  3618         .LONG           ^X000001D0
                        00000180  028C  3619         .LONG           ^X00000180
                        000001C0  0290  3620         .LONG           ^X000001C0
                        00000140  0294  3621         .LONG           ^X00000140
                                  0298  3622
                              04  0298  3623 2$:     .BYTE           ^X4                         ;VBUS DATA SUBTEST 1,3,5,7
                                  0299  3624         VBUSDATA        <^X08>,0
                                  029A  3625         VBUSDATA        <^X09>,0
                                  029B  3626         VBUSDATA        <^X0A>,1
                                  029C  3627         VBUSDATA        <^X0B>,0
                                  029D  3628
                            1802  029D  3629 3$:     .WORD           ^X1802                      ;CS ADDRESS SUBTEST 1,3,5,7
                                  029F  3630
                              04  029F  3631 4$:     .BYTE           ^X4                         ;VBUS DATA SUBTEST 2,4,6,8
                                  02A0  3632         VBUSDATA        <^X08>,1
                                  02A1  3633         VBUSDATA        <^X09>,0
                                  02A2  3634         VBUSDATA        <^X0A>,1
                                  02A3  3635         VBUSDATA        <^X0B>,1
                                  02A4  3636
                            180B  02A4  3637 5$:     .WORD           ^X180B                      ;CS ADDRESS SUBTEST 2,4,6,8
                                  02A6  3638
                                  02A6  3639 6$:
U 0A, 7601,8800,5BE4,7A18,05C0,9800  02A6  3640 ;% 0A   STROBE.V.BUS,WB_R[TEMP6],SIZE[LONG],PROBE WRITE MODE ON WB?,NEXT/1800
                                  02B1  3644                                                      ;MODE SELECT FOR SUBTEST 3 & 4
                                  02B1  3645 7$:
U 0A, 7601,C800,5BE4,7A1C,05C0,9800  02B1  3646 ;% 0A   STROBE.V.BUS,WB_R[TEMP7],SIZE[LONG],PROBE WRITE MODE ON WB?,NEXT/1800
                                  02BC  3650                                                      ;MODE SELECT FOR SUBTEST 5 & 6
                                  02BC  3651 8$:
U 0A, 7601,C800,5BE4,7A20,05C0,9800  02BC  3652 ;% 0A   STROBE.V.BUS,WB_R[TEMP8],SIZE[LONG],PROBE WRITE MODE ON WB?,NEXT/1800
                                  02C7  3656                                                      ;MODE SELECT FOR SUBTEST 7 & 8
                                  02C7  3657 9$:
U 0A, 7601,8800,5BE4,7A14,05C0,9800  02C7  3658 ;% 0A   STROBE.V.BUS,WB_R[TEMP5],SIZE[LONG],PROBE WRITE MODE ON WB?,NEXT/1800
                                  02D2  3662                                                      ;MODE SELECT FOR SUBTEST 1 & 2
                            3FFF  02D2  3663 10$:    .WORD   ^X3FFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                                  02D4  3664
                                  02D4  3665 BEGIN_CPU_DATA
                                  0002  3666
                                  0002  3667 DCS_DATA
                                  0001  3668
U 00, 7700,C800,0364,0300,0470,1801  0001  3669 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  3673 ;% 01   MEMSCAR_R[TEMP0]                            ;PHY/VIR ADDRESS TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  3677 ;% 02   MEMSCR_R[TEMP1]                             ;SET MME ON
```

```
U 03. 7700.8800.5BE4.0308.6870.1804  0022  3681 ;% 03   MEMSCAR_R[TEMP2]                                  ;REF CACHE ONLY TO MEMSCAR
U 04. 7700.8800.5BE4.0304.6070.1805  002D  3685 ;% 04   MEMSCR_R[TEMP1]                                   ;TURN CMI OFF
U 05. 7700.C800.5BE4.030C.6A70.1806  0038  3689 ;% 05   PSL(PREV_CURM ISCURM_R[TEMP3])                    ;SET CURRENT MODE TO USER
U 06. 7700.C800.5BE6.0310.0470.1807  0043  3693 ;% 06   D_R[TEMP4]                                       ;ADDRESS TO ACCESS TB
U 07. 7700.1868.DB10.0300.0470.1808  004E  3697 ;% 07   M[TEMP8]_D+ZLIT8[0]                              ;store address in mtemp8
U 08. 7700.9800.DB12.0300.5360.1809  0059  3701 ;% 08   VA_D_D+ZLIT8[0] INVALIDATE TB                    ;invalidate TB at VA
U 09. 5700.C81B.0364.0300.5070.180A  0064  3705 ;% 09   WB_RDM,WCTRL/TB_WB,MSRC/VA                      ;PTE TO TB
U 0A. 7601.8800.5BE4.7A14.05C0.9800  006F  3709 ;% 0A   STROBE.V.BUS,WB_R[TEMP5],SIZE[LONG],PROBE WRITE MODE ON WB?,NEXT/1800
                                     007A  3713                                                        ;PERFORM PROBE
U 0B. 7B00.4800.0364.0300.0470.180C  007A  3714 ;% 0B   NOP,STOP.CLOCK,LATCH.CS.ADDR                   ;STOP CPU CLOCK
                                     0085  3718
                                     0085  3719 RTEMP_DATA
                                     0001  3720
                            00000000 0001  3721          .LONG   ^X00000000                           ;PHY/VIR REG ADDRESS
                            01000000 0005  3722          .LONG   ^X01000000                           ;DATA
                            0E000000 0009  3723          .LONG   ^X0E000000                           ;REF CACHE ONLY REG ADDRESS
                            03000000 000D  3724          .LONG   ^X03000000                           ;CURRENT MODE TO USER
                            00000000 0011  3725          .LONG   ^X00000000                           ;ADDRESS FOR VA
                            00000000 0015  3726          .LONG   ^X00000000                           ;KERNEL MODE
                            01000000 0019  3727          .LONG   ^X01000000                           ;EXECUTIVE MODE
                            02000000 001D  3728          .LONG   ^X02000000                           ;SUPERVISOR MODE
                            03000000 0021  3729          .LONG   ^X03000000                           ;USER MODE
                                     0025  3730
                                     0025  3731 END_CPU_DATA
                                     02D4  3732
                                     02D4  3733 END_TEST_DATA
                                     02D4
                                     02D4          ;-------------------------------------------------------------------------
                                     02D4  3734
                                     02D4  3735 ENDTEST
```

```
0038        .SBTTL  "TEST   06D     TEST BUS FIELD NOP, READ, AND READ.PHY MICRO ORDERS
0038  3737 ;*******************************************************************************
0038  3738 ;++
0038  3739 ;
0038  3740 ; FUNCTIONAL DESCRIPTION:
0038  3741 ;
0038  3742 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE FOLLOWING
0038  3743 ;       "BUS" FIELD MICRO ORDERS:
0038  3744 ;               1.  NOP
0038  3745 ;               2.  READ
0038  3746 ;               3.  READ.PHY
0038  3747 ;
0038  3748 ;
0038  3749 ; TEST ALGORITHM:
0038  3750 ;
0038  3751 ;       1.  CREATE A TEST LOOP OF 3 ITERATIONS
0038  3752 ;       2.  MODIFY DCS FOR THIS ITERATION
0038  3753 ;       3.  EXECUTE DCS PROGRAM
0038  3754 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
0038  3755 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
0038  3756 ;               2.  SET MME OFF
0038  3757 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
0038  3758 ;               4.  TURN CMI OFF
0038  3759 ;               5.  LOAD 0'S INTO VIRTUAL ADDRESS REGISTER
0038  3760 ;               6.  WRITE ALL 1'S INTO CACHE ADDRESS 0
0038  3761 ;               7.  LOAD MEMORY DATA REGISTER (MDR) WITH 0'S
0038  3762 ;               8.  PERFORM SELECTED BUS FUNCTION
0038  3763 ;               9.  READ MDR BACK TO RDM
0038  3764 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0038  3765 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING ITERATIONS
0038  3766 ;
0038  3767 ;
0038  3768 ; TEST PATTERNS:
0038  3769 ;
0038  3770 ;       ITERATION       BUS FUNCTION            EXPECTED DATA
0038  3771 ;           1               NOP                     00000000
0038  3772 ;           2               READ                    FFFFFFFF
0038  3773 ;           3               READ.PHY                FFFFFFFF
0038  3774 ;
0038  3775 ;
0038  3776 ; LOGIC DESCRIPTION:
0038  3777 ;
0038  3778 ;       THIS TEST VERIFIES THE CORRECT OPERATION OF 3 OF THE BUS FIELD
0038  3779 ;       READ MICRO ORDERS BY READING DATA FROM CACHE ADDRESS 0.  THE "MDR"
0038  3780 ;       GATE ARRAY IS THE BASIC DATA PATH USED AND IS A GOOD FIRST CHOICE
0038  3781 ;       TO SWAP.  THE "CAK" GATE ARRAY IS RESPONSIBLE FOR CACHE CONTROL,
0038  3782 ;       WHILE "CML" CONTROLS THE CMI.  EITHER OF THESE GATE ARRAYS IS A
0038  3783 ;       GOOD SECOND CHOICE TO REPLACE.
0038  3784 ;
0038  3785 ;
0038  3786 ; ASSUMPTIONS:
0038  3787 ;
0038  3789 ;
0038  3790 ;
```

```
                                0038  3791 ; ERROR DESCRIPTION:
                                0038  3792 ;
                                0038  3793 ;       EXPECTED CACHE DATA
                                0038  3794 ;       RECEIVED CACHE DATA
                                0038  3795 ;       LOOP COUNT
                                0038  3796 ;
                                0038  3797 ;--
                                0038  3798 ;*******************************************************************************
                                0038  3799 ;///////////////////////////////////////////////////////////////////////////////
                                0038  3800
                                0038  3801         INITIALIZE                              ;INIT CPU
                                003A  3802         LOOP            I,1,3                   ;CREATE A TEST LOOP
                                0041  3803         LOADDCS         1$,I,09,1               ;MODIFY DCS ADDRESS 09
                                0048  3804         ERRLOOP                                 ;ERROR LOOP
                                004A  3805         FETCH           0                       ;START DCS PROGRAM
                                004F  3806         BRSTCLK         <^X0B>                  ;END DCS PROGRAM
                                0053  3807         CMPREG          WHREG,2$,I,L            ;COMPARE RESULTS
                                005C  3808         IFERROR         ,<<MDR><CAK><CML>>,     ;POSSIBLE FAILING ARRAYS
                                0064  3809         FETCH           <^X0C>                  ;START DCS PROGRAM
                                0069  3810         BRSTCLK         <^X0F>                  ;END DCS PROGRAM
                                006D  3811         CMPREGMSK       WHREG,3$,I,4$,,L,       ;TEST SAVED MODE REGISTER
                                0079  3812         IFERROR         ,<<ADK>>,               ;FAILING ARRAY
                                007F  3813         ENDLOOP         I                       ;END TEST LOOP
                                0082  3814         SKIP
                                0089  3815
                                0089  3816 BEGIN_TEST_DATA
                                0089
                                0089          ;-------------------------------------------------------------------
                                0089  3817
                                0089  3818 1$:
U 09, 7701,0800,0364,0200,0470,180A  0089  3819 ;% 09    BUS/NOP,SIZE[LONG]                      ;DCS ADD 09 LOOP 1
U 09, 7701,0800,0364,0200,0500,180A  0094  3823 ;% 09    BUS/READ,SIZE[LONG]                     ;DCS ADD 09 LOOP 2
U 09, 7700,C800,0364,0300,0400,180A  009F  3827 ;% 09    BUS/READ.PHY                            ;DCS ADD 09 LOOP 3
                                     00AA  3831
                          00000000   00AA  3832 2$:      .LONG   ^X00000000                     ;RESULTS LOOP 1
                          FFFFFFFF   00AE  3833          .LONG   ^XFFFFFFFF                     ;RESULTS LOOP 2
                          FFFFFFFF   00B2  3834          .LONG   ^XFFFFFFFF                     ;RESULTS LOOP 3
                                     00B6  3835
                          04000000   00B6  3836 3$:      .LONG   ^X04000000                     ;RESULTS IN SAVED MODE REGISTER
                          0C000000   00BA  3837          .LONG   ^X0C000000                     ;...
                          0C000000   00BE  3838          .LONG   ^X0C000000                     ;...
                                     00C2  3839
                          0F000000   00C2  3840 4$:      .LONG   ^X0F000000                     ;MASK FOR CMPREGMSK PSEUDO-OP
                                     00C6  3841
                                     00C6  3842 BEGIN_CPU_DATA
                                     0002  3843
                                     0002  3844 DCS_DATA
                                     0001  3845
U 00, 7700,C800,0364,0300,0470,1801  0001  3846 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  3850 ;% 01    MEMSCAR_R[TEMP0]                        ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  3854 ;% 02    MEMSCR_R[TEMP1]                         ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  3858 ;% 03    MEMSCAR_R[TEMP2]                        ;REF CACHE ONLY ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805  002D  3862 ;% 04    MEMSCR_R[TEMP3]                         ;TURN CMI OFF
U 05, 7700,0800,5BE4,0304,4A70,1806  0038  3866 ;% 05    VA_R[TEMP1]                             ;0'S TO VA REGISTER
U 06, 7700,0800,5BE4,0310,5C80,1807  0043  3870 ;% 06    WRITE.PHY R[TEMP4]                      ;1'S TO CACHE ADD 0
```

```
U 07. 7700.8800.5BE4.0310.5470.1808  004E  3874 ;% 07   WB_R[TEMP4],WCTRL/WDR_WB.UR                ;HOLD DATA FOR EXTRA CYCLE
U 08. 7700.0800.5BE4.0304.4670.1809  0059  3878 ;% 08   WB_R[TEMP1],WCTRL/MDR_WB                   ;0'S TO MDR REGISTER
U 09. 7701.0800.0364.0200.0470.180A  0064  3882 ;% 09   BUS/NOP,SIZE[LONG]                        ;PERFORM BUS FUNCTION
U 0A. 6700.8812.5924.0300.0470.180B  006F  3886 ;% 0A   RDM_WB,WB_M[MDR]                         ;READ MDR TO RDM
U 0B. 7300.4800.0364.0300.0470.180C  007A  3890 ;% 0B   NOP,STOP.CLOCK                           ;STOP CPU CLOCK
U 0C. 7700.C800.0364.0300.0470.180D  0085  3894 ;% 0C   NOP
U 0D. 7700.C800.5BE4.0314.6870.180E  0090  3898 ;% 0D   MEMSCAR_R[TEMP5]                         ;SAVED MODE REG ADD TO MEMSCAR
U 0E. 6700.4800.0364.0300.6470.180F  009B  3902 ;% 0E   RDM_WB,WCTRL/MEMSCR                      ;READ SAVED MODE REG TO RDM
U 0F. 7300.C800.0364.0300.0470.1810  00A6  3906 ;% 0F   NOP,STOP.CLOCK                           ;STOP CPU CLOCK
                                     00B1  3910
                                     00B1  3911 RTEMP_DATA
                                     0001  3912
                          00000000   0001  3913          .LONG   ^X00000000                    ;PHY/VIR REG ADDRESS
                          00000000   0005  3914          .LONG   ^X00000000                    ;DATA
                          0E000000   0009  3915          .LONG   ^X0E000000                    ;REF CACHE ONLY ADDRESS
                          01000000   000D  3916          .LONG   ^X01000000                    ;TURN CMI OFF
                          FFFFFFFF   0011  3917          .LONG   ^XFFFFFFFF                    ;TEST DATA FOR CACHE ADD 0
                          01000000   0015  3918          .LONG   ^X01000000                    ;ADD OF SAVED MODE REGISTER
                                     0019  3919
                                     0019  3920 END_CPU_DATA
                                     00C6  3921
                                     00C6  3922 END_TEST_DATA
                                     00C6
                                     00C6          ;-----------------------------------------------------------------------------
                                     00C6  3923
                                     00C6  3924 ENDTEST
```

```
                        .SBTTL  "TEST   06E     TEST BUS FIELD READ.LNG MICRO ORDER
0028  3926 ;*******************************************************************************
0028  3927 ;++
0028  3928 ;
0028  3929 ; FUNCTIONAL DESCRIPTION:
0028  3930 ;
0028  3931 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE "BUS/READ.LNG"
0028  3932 ;       MICRO ORDER.
0028  3933 ;
0028  3934 ;
0028  3935 ; TEST ALGORITHM:
0028  3936 ;
0028  3937 ;       1.  EXECUTE DCS PROGRAM
0028  3938 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
0028  3939 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
0028  3940 ;               2.  SET MME OFF
0028  3941 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
0028  3942 ;               4.  TURN CMI OFF
0028  3943 ;               5.  LOAD A 3 INTO VIRTUAL ADDRESS REGISTER
0028  3944 ;               6.  LOAD 0'S INTO MDR
0028  3945 ;               7.  PERFORM READ LONG BUS FUNCTION
0028  3946 ;               8.  READ MDR BACK TO RDM
0028  3947 ;       2.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0028  3948 ;
0028  3949 ;
0028  3950 ; TEST PATTERNS:
0028  3951 ;
0028  3952 ;       BUS FUNCTION            TEST PATTERN
0028  3953 ;
0028  3954 ;       BUS/READ.LNG              FFFFFFFF
0028  3955 ;
0028  3956 ;
0028  3957 ; LOGIC DESCRIPTION:
0028  3958 ;
0028  3959 ;       THIS TEST CHECKS TO SEE IF THE 2 LEAST SIGNIFICANT BITS OF THE VA ARE
0028  3960 ;       IGNORED WHEN PERFORMING A BUS/READ.LNG MICRO ORDER.  THE "MDR" GATE
0028  3961 ;       ARRAY IS THE BASIC DATA PATH INVOLVED AND IS THEREFORE A PRIME
0028  3962 ;       CANDIDATE TO SWAP.  SECOND CHOICE SHOULD BE "CAK" OR "CML".
0028  3963 ;
0028  3964 ;
0028  3965 ; ASSUMPTIONS:
0028  3966 ;
0028  3967 ;       THIS TEST ASSUMES THE WBUS, MBUS AND DPM ARE OPERATIONAL
0028  3968 ;
0028  3969 ;
0028  3970 ; ERROR DESCRIPTION:
0028  3971 ;
0028  3972 ;       EXPECTED CACHE DATA
0028  3973 ;       RECEIVED CACHE DATA
0028  3974 ;
0028  3975 ;--
0028  3976 ;*******************************************************************************
0028  3977 ;///////////////////////////////////////////////////////////////////////////////
0028  3978
0028  3979         INITIALIZE                                      ;INIT CPU
```

```
                                002A          EXPUTRAP                                  ;POSSIBLE FAILURE MODE
                                002C          ERRLOOP                                   ;ERROR LOOP
                                002E          FETCH          0                          ;START DCS PROGRAM
                                0033          BRSTCLK        <^X09>                     ;END DCS PROGRAM
                                0037          CMPREGMSK      CSTRP,2$,,3$,,W,NE         ;CHECK FOR UNDR U-TRAP
                                0043          IFERROR        ,<<MDR><CAK><CML>>,        ;POSSIBLE FAILING ARRAYS
                                004B          CMPREGMSK      CSTRP,4$,,3$,,W,           ;CHECK FOR UNEXPECTED U-TRAP
                                0057          IFERROR        ,<<MDR><CAK><CML>>,        ;POSSIBLE FAILING ARRAYS
                                005F          CMPREG         WHREG,1$,,L,               ;COMPARE RESULTS
                                0068          IFERROR        ,<<MDR><CAK><CML>>,        ;POSSIBLE FAILING ARRAYS
                                0070          FETCH          <^X0A>                     ;START DCS PROGRAM
                                0075          BRSTCLK        <^X10>,INH                 ;END DCS PROGRAM
                                0079          SKIP                                      ;GO TO NEXT TEST
                                0080
                                0080  BEGIN_TEST_DATA
                                0080
                                0080  ;-----------------------------------------------------------------------
                                0080
                     FFFFFFFF   0080  1$:     .LONG   ^XFFFFFFFF                        ;TEST RESULTS
                         0021   0084  2$:     .WORD   ^X0021                            ;UNALIGNED DATA READ U-TRAP
                         3FFF   0086  3$:     .WORD   ^X3FFF                            ;MASK FOR CMPREGMSK PSEUDO-OP
                         1809   0088  4$:     .WORD   ^X1809                            ;ADD DCS SHOULD HALT AT
                                008A
                                008A  BEGIN_CPU_DATA
                                0002
                                0002  DCS_DATA
                                0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  ;% 01   MEMSCAR_R[TEMP0]                         ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  ;% 02   MEMSCR_R[TEMP1]                          ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  ;% 03   MEMSCAR_R[TEMP2]                         ;REF CACHE ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805  002D  ;% 04   MEMSCR_R[TEMP3]                          ;TURN CMI OFF
U 05, 7700,0800,5BE4,0310,4A70,1806  0038  ;% 05   VA_R[TEMP4]                              ;LOAD VA WITH 3
U 06, 7700,8800,5BE4,0304,4670,1807  0043  ;% 06   WB_R[TEMP1],WCTRL/MDR_WB                  ;LOAD MDR WITH 0'S
U 07, 7700,4800,0364,0300,0510,1808  004E  ;% 07   BUS/READ.LNG                             ;PERFORM READ LONG
U 08, 6700,0812,5924,0300,0470,1809  0059  ;% 08   RDM_WB,WB_M[MDR]                         ;READ MDR TO RDM
U 09, 7300,4800,0364,0300,0470,180A  0064  ;% 09   NOP,STOP.CLOCK                           ;STOP CPU CLOCK
                                006F  ;
                                006F  ; This section of microcode is use to se the VA past any valid cache data.
                                006F  ; This is necessary in case this test is run in continous mode.  In this mode
                                006F  ; if the VA points to valid cache data then the INITIALIZE pseudo will cause
                                006F  ; that cache location to be invalidated
                                006F  ;
U 0A, 7700,C800,0364,0300,0470,180B  006F  ;% 0A   NOP
U 0B, 7700,1800,CB70,0307,6870,180C  007A  ;% 0B   MEMSCAR_ZLIT24[00E]                      ;Ref. cache addr to memscar
U 0C, 7700,D800,CB70,0300,E070,180D  0085  ;% 0C   MEMSCR_ZLIT24[1]                         ;Turn the CMI off
U 0D, 7700,1800,C370,0380,4A70,180E  0090  ;% 0D   VA_ZLIT0[100]                            ;Set VA to phy addr 100(hex)
U 0E, 7700,4800,5BE4,03D8,4670,180F  009B  ;% 0E   WB_R[ZERO],WCTRL/MDR_WB                   ;Zero the MDR out
U 0F, 7700,4800,0364,0300,0400,1810  00A6  ;% 0F   READ.PHY                                 ;Latch VA to MA address
U 10, 7300,4800,0364,0300,0470,1811  00B1  ;% 10   NOP,STOP.CLOCK
                                00BC
                                00BC
                                00BC  RTEMP_DATA
                                0001
                     00000000   0001          .LONG   ^X00000000                        ;PHY/VIR REGISTER ADDRESS
```

```
00000000  0005  4084          .LONG   ^X00000000                      ;DATA
0E000000  0009  4085          .LONG   ^X0E000000                      ;REF CACHE ONLY REG ADDRESS
01000000  000D  4086          .LONG   ^X01000000                      ;TURN CMI OFF
00000003  0011  4087          .LONG   ^X00000003                      ;DATA FOR VIRTUAL ADD REG
          0015  4088
          0015  4089 CACHE_DATA
          0001  4090
FFFFFFFF  0001  4091          .LONG   ^XFFFFFFFF                      ;TEST PATTERN
00000000  0005  4092          .LONG   ^X00000000
          0009  4093
          0009  4094 END_CPU_DATA
          008A  4095
          008A  4096 END_TEST_DATA
          008A
          008A      ;-------------------------------------------------------------------------
          008A  4097
          008A  4098 ENDTEST
```

```
                        002C        .SBTTL  "TEST   06F     TEST BUS FIELD READ.LNG.MOD MICRO ORDER
                        002C  4100 ;*******************************************************************************
                        002C  4101 ;++
                        002C  4102 ;
                        002C  4103 ; FUNCTIONAL DESCRIPTION:
                        002C  4104 ;
                        002C  4105 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE "BUS/READ.LNG.MOD"
                        002C  4106 ;       MICRO ORDER.
                        002C  4107 ;
                        002C  4108 ;
                        002C  4109 ; TEST ALGORITHM:
                        002C  4110 ;
                        002C  4111 ;       1.  MODIFY DCS ADDRESS 06 (SUBTEST #2)
                        002C  4112 ;       2.  EXECUTE DCS PROGRAM
                        002C  4113 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
                        002C  4114 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER. (MEMSCAR)
                        002C  4115 ;               2.  SET MME ON
                        002C  4116 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
                        002C  4117 ;               4.  TURN CMI OFF
                        002C  4118 ;               5.  LOAD A 3 INTO THE VIRTUAL ADDRESS REGISTER
                        002C  4119 ;               6.  LOAD PTE INTO THE LONGLIT REGISTER
                        002C  4120 ;               7.  WRITE PTE IN LONGLIT INTO TRANSLATION BUFFER
                        002C  4121 ;               8.  SET PSL CURRENT MODE TO EXECUTIVE
                        002C  4122 ;               9.  LOAD 0'S INTO MDR
                        002C  4123 ;              10.  PERFORM BUS/READ.LNG.MOD FUNCTION
                        002C  4124 ;              11.  READ MDR TO RDM
                        002C  4125 ;       3.  COMPARE RESULTS (EXPECTED AN RECEIVED)
                        002C  4126 ;
                        002C  4127 ;       NOTE:  SUBTEST #1 SHOULD COMPLETE NORMALLY WITH DATA FROM CACHE
                        002C  4128 ;              ADDRESS 0 ENDING UP IN THE RDM.
                        002C  4129 ;              SUBTEST #2 SHOULD EXPERIENCE AN ACCESS VIOLATION TRAP WHEN
                        002C  4130 ;              THE "BUS/READ.LNG.MOD" IS PERFORMED.  THE CS LINES ARE TESTED
                        002C  4131 ;              TO INSURE THIS IS THE CASE.
                        002C  4132 ;
                        002C  4133 ;
                        002C  4134 ; TEST PATTERNS:
                        002C  4135 ;
                        002C  4136 ;       SUBTEST          PTE                 COMPARE DATA
                        002C  4137 ;          1           00000150        (CACHE ADD 0) FFFFFFFF
                        002C  4138 ;          2           00000160        (CS LINES) 002F
                        002C  4139 ;
                        002C  4140 ;
                        002C  4141 ; LOGIC DESCRIPTION:
                        002C  4142 ;
                        002C  4143 ;       THIS TEST IS SIMILAR TO THE "BUS/READ.MOD TEST".  THE ONLY DIFFERENCE
                        002C  4144 ;       BEING THE BUS/READ.LNG.MOD FUNCTION IS TESTED USING A VIRTUAL ADDRESS
                        002C  4145 ;       OF 3 IN EXECUTIVE MODE.  THE "ACV" GATE ARRAY SHOULD BE YOUR FIRST
                        002C  4146 ;       AND ONLY CHOICE FOR REPLACEMENT.
                        002C  4147 ;
                        002C  4148 ;
                        002C  4149 ; ASSUMPTIONS:
                        002C  4150 ;
                        002C  4151 ;       THIS TEST ASSUMES THE DPM, WBUS, MBUS, VA, CACHE, AND PSL ARE
                        002C  4152 ;       OPERATIONAL.
                        002C  4153 ;
```

```
                        4154 ;
                        4155 ; ERROR DESCRIPTION:
                        4156 ;
                        4157 ;       SUBTEST #1
                        4158 ;               EXPECTED CACHE DATA
                        4159 ;               RECEIVED CACHE DATA
                        4160 ;
                        4161 ;       SUBTEST #2
                        4162 ;               EXPECTED CS ADDRESS
                        4163 ;               RECEIVED CS ADDRESS
                        4164 ;
                        4165 ;--
                        4166 ;*****************************************************************************
                        4167
                        4168         SUBTEST                                        ;SUBTEST #1
                        4169 ;+
                        4170 ;SUBTEST #1
                        4171 ;
                        4172 ;       TEST "BUS/READ.LNG.MOD" WITH NO MICRO TRAP
                        4173 ;-
                        4174         INITIALIZE                                     ;INIT CPU
                        4175         LOADDCS         4$,,08,1                       ;MODIFY DCS ADDRESS 07
                        4176         EXPUTRAP                                       ;POSSIBLE FAILURE MODE
                        4177         ERRLOOP                                        ;ERROR LOOP
                        4178         FETCH           0                              ;START DCS PROGRAM
                        4179         BRSTCLK         <^X0E>                         ;END DCS PROGRAM
                        4180         CMPREGMSK       CSTRP,3$,,5$,,W,NE             ;CHECK FOR ACVR U-TRAP
                        4181         IFERROR         ,<<ACV>>,
                        4182         CMPREGMSK       CSTRP,7$,,5$,,W,NE             ;CHECK FOR UNDR U-TRAP
                        4183         IFERROR         ,<<ACV>>,
                        4184         CMPREGMSK       CSTRP,6$,,5$,,W,               ;CHECK FOR UNEXPECTED U-TRAP
                        4185         IFERROR         ,<<ACV>>,
                        4186         CMPREG          WHREG,1$,,L,                   ;COMPARE RESULTS
                        4187         IFERROR         ,<<ACV>>,
                        4188
                        4189         SUBTEST                                        ;SUBTEST #2
                        4190 ;+
                        4191 ;SUBTEST #2
                        4192 ;
                        4193 ;       TEST "BUS/READ.LNG.MOD" WITH MICRO TRAP
                        4194 ;-
                        4195         INITIALIZE                                     ;INIT CPU
                        4196         EXPUTRAP                                       ;EXPECTED U-TRAP
                        4197         LOADDCS         2$,,08,1                       ;MODIFY DCS ADDRESS 07
                        4198         ERRLOOP                                        ;ERROR LOOP
                        4199         FETCH           0                              ;START DCS PROGRAM
                        4200         BRSTCLK         <^X0E>                         ;END DCS PROGRAM
                        4201         CMPREGMSK       CSTRP,3$,,5$,,W,               ;COMPARE CS ADDRESS
                        4202         IFERROR         ,<<ACV>>,                      ;FAILING ARRAY AND MODULE
                        4203         FETCH           <^X0F>                         ;START DCS PROGRAM
                        4204         BRSTCLK         <^X12>                         ;END DCS PROGRAM
```

```
                                00BE  4205          CMPREGMSK       WHREG,8$,,9$,,L,        ;TEST SAVED MODE REGISTER
                                00CA  4206          IFERROR         ,<<ADK>>,               ;FAILING ARRAY
                                00D0  4207          SKIP
                                00D7  4208
                                00D7  4209 BEGIN_TEST_DATA
                                00D7
                                00D7          ;-------------------------------------------------------------------------
                                00D7  4210
                    FFFFFFFF    00D7  4211 1$:      .LONG   ^XFFFFFFFF                      ;RESULTS SUBTEST #1
                                00DB  4212
                                00DB  4213 2$:
U 08, 7700,B800,7FFF,FF4F,8470,1809 00DB  4214 ;% 08   LONLIT_[00000160]                   ;PTE SUBTEST #2
                                00E6  4218
                         002E   00E6  4219 3$:      .WORD   ^X002E                          ;ACCESS VIOLATION READ U-TRAP
                                00E8  4220
                                00E8  4221 4$:
U 08, 7700,B800,7FFF,FF57,8470,1809 00E8  4222 ;% 08   LONLIT_[00000150]                   ;PTE SUBTEST #1
                                00F3  4226
                         3FFF   00F3  4227 5$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO-OP
                         180E   00F5  4228 6$:      .WORD   ^X180E                          ;ADD DCS SHOULD HALT AT
                         0021   00F7  4229 7$:      .WORD   ^X0021                          ;UNALIGNED DATA READ U-TRAP
                    00000000    00F9  4230 8$:      .LONG   ^X00000000                      ;SAVED MODE REGISTER RESULTS
                    0F000000    00FD  4231 9$:      .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO-OP
                                0101  4232
                                0101  4233 BEGIN_CPU_DATA
                                0002  4234
                                0002  4235 DCS_DATA
                                0001  4236
U 00, 7700,C800,0364,0300,0470,1801 0001  4237 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  4241 ;% 01   MEMSCAR_R[TEMP0]                    ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803 0017  4245 ;% 02   MEMSCR_R[TEMP1]                     ;SET MME ON
U 03, 7700,8800,5BE4,0308,6870,1804 0022  4249 ;% 03   MEMSCAR_R[TEMP2]                    ;REF CACHE ONLY ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805 002D  4253 ;% 04   MEMSCR_R[TEMP3]                     ;TURN CMI OFF
U 05, 7700,4800,5BE6,0310,0470,1806 0038  4257 ;% 05   D_R[TEMP4]                          ;3 TO D REGISTER
U 06, 7700,1868,DB10,0300,0470,1807 0043  4261 ;% 06   M[TEMP8]_D+ZLIT8[0]                ;store address in mtemp8
U 07, 7700,1800,DB12,0300,5360,1808 004E  4265 ;% 07   VA_D_D+ZLIT8[0] INVALIDATE TB      ;invalidate the TB at VA
U 08, 7700,B800,7FFF,FF57,8470,1809 0059  4269 ;% 08   LONLIT_[00000150]                   ;PTE
U 09, 7700,C81B,5BE4,03D4,5070,180A 0064  4273 ;% 09   TB_R[LONLIT]                       ;TB GETS PTE
U 0A, 7700,0800,5BF4,0314,0070,180B 006F  4277 ;% 0A   PSL_R[TEMP5]                      ;SET EXECUTIVE MODE IN PSL
U 0B, 7700,4800,5BE4,0318,4670,180C 007A  4281 ;% 0B   WB_R[TEMP6],WCTRL/MDR_WB          ;0'S TO MDR
U 0C, 7700,C800,0364,0300,0550,180D 0085  4285 ;% 0C   BUS/READ.LNG.MOD                  ;PERFORM READ LONG MODIFY
U 0D, 6700,8812,5924,0300,0470,180E 0090  4289 ;% 0D   RDM_WB,WB_M[MDR]                  ;READ MDR TO RDM
U 0E, 7300,4800,0364,0300,0470,180F 009B  4293 ;% 0E   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
U 0F, 7700,C800,0364,0300,0470,1810 00A6  4297 ;% 0F   NOP
U 10, 7700,0800,5BE4,031C,6870,1811 00B1  4301 ;% 10   MEMSCAR_R[TEMP7]                   ;ADD OF SAVED MODE REG
U 1A, 6700,4800,0364,0300,6470,181B 00BC  4305 ;% 1A   RDM_WB,WCTRL/MEMSCR               ;READ SAVED MODE REG TO RDM
U 12, 7300,C800,0364,0300,0470,1813 00C7  4309 ;% 12   NOP,STOP.CLOCK                    ;STOP CPU CLOCK
                                00D2  4313
                                00D2  4314 RTEMP_DATA
                                0001  4315
                    00000000    0001  4316         .LONG   ^X00000000                  ;PHY/VIR REG ADDRESS
                    01000000    0005  4317         .LONG   ^X01000000                  ;SET MME ON
                    0E000000    0009  4318         .LONG   ^X0E000000                  ;REFERENCE CACHE ONLY ADD
                    01000000    000D  4319         .LONG   ^X01000000                  ;TURN CMI OFF
                    00000003    0011  4320         .LONG   ^X00000003                  ;ADDRESS FOR VA REGISTER
```

```
01000000  0015  4321          .LONG   ^X01000000                      ;SET EXECUTIVE MODE IN PSL
00000000  0019  4322          .LONG   ^X00000000                      ;0'S FOR MDR
01000000  001D  4323          .LONG   ^X01000000                      ;ADD OF SAVED MODE REGISTER
          0021  4324
          0021  4325 CACHE_DATA
          0001  4326
FFFFFFFF  0001  4327          .LONG   ^XFFFFFFFF                      ;TEST PATTERN
          0005  4328
          0005  4329 END_CPU_DATA
          0101  4330
          0101  4331 END_TEST_DATA
          0101
          0101        ;-------------------------------------------------------------------------
          0101  4332
          0101  4333 ENDTEST
```

```
                        .SBTTL  "TEST   070     TEST BUS FIELD READ.MOD MICRO ORDER
                4335 ;*******************************************************************************
                4336 ;++
                4337 ;
                4338 ; FUNCTIONAL DESCRIPTION:
                4339 ;
                4340 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE "BUS/READ.MOD"
                4341 ;       MICRO ORDER.
                4342 ;
                4343 ;
                4344 ; TEST ALGORITHM:
                4345 ;
                4346 ;       1.  MODIFY DCS ADDRESS 06 (SUBTEST #2)
                4347 ;       2.  EXECUTE DCS PROGRAM
                4348 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
                4349 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
                4350 ;               2.  SET MME ON
                4351 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
                4352 ;               4.  TURN CMI OFF
                4353 ;               5.  LOAD 0'S INTO VIRTUAL ADDRESS REGISTER
                4354 ;               6.  LOAD PTE INTO LONGLIT REGISTER
                4355 ;               7.  WRITE PTE INTO TRANSLATION BUFFER
                4356 ;               8.  SET CURRENT MODE TO KERNEL
                4357 ;               9.  LOAD 0'S INTO MDR
                4358 ;              10.  PERFORM "BUS/READ.MOD" FUNCTION
                4359 ;              11.  READ MDR TO RDM
                4360 ;       3.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                4361 ;
                4362 ;       NOTE:  SUBTEST #1 SHOULD COMPLETE NORMALLY WITH DATA FROM CACHE
                4363 ;              ADDRESS 0 ENDING UP IN THE RDM.
                4364 ;              SUBTEST #2 SHOULD EXPERIENCE AN ACCESS VIOLATION TRAP WHEN
                4365 ;              THE "BUS/READ.MOD" IS PERFORMED.  THE CS LINES ARE TESTED
                4366 ;              TO INSURE THIS IS THE CASE.
                4367 ;
                4368 ;
                4369 ; TEST PATTERNS:
                4370 ;
                4371 ;       SUBTEST         PTE                     COMPARE DATA
                4372 ;          1            00000120            (CACHE ADD 0) FFFFFFFF
                4373 ;          2            00000130            (CS LINES) 002F
                4374 ;
                4375 ;
                4376 ; LOGIC DESCRIPTION:
                4377 ;
                4378 ;       THIS TEST CHECKS THE ABILITY OF THE "ACV" GATE ARRAY TO DECODE AND
                4379 ;       EXECUTE THE "BUS/READ.MOD" MICRO ORDER CORRECTLY.  SUBTEST #1 USES
                4380 ;       A PTE THAT ALLOWS WRITE ACCESS WHILE SUBTEST #2 USES A PTE WHICH
                4381 ;       CAUSES AN ACCESS VIOLATION.
                4382 ;
                4383 ;
                4384 ; ASSUMPTIONS:
                4385 ;
                4386 ;       THIS TEST ASSUMES THE DPM, WBUS, MBUS, VA, CACHE, AND PSL ARE
                4387 ;       OPERATIONAL.
                4388 ;
```

```
                     0028  4389 ;
                     0028  4390 ; ERROR DESCRIPTION:
                     0028  4391 ;
                     0028  4392 ;       SUBTEST #1
                     0028  4393 ;               EXPECTED CACHE DATA
                     0028  4394 ;               RECEIVED CACHE DATA
                     0028  4395 ;
                     0028  4396 ;       SUBTEST #2
                     0028  4397 ;               EXPECTED CS ADDRESS
                     0028  4398 ;               RECEIVED CS ADDRESS
                     0028  4399 ;
                     0028  4400 ;--
                     0028  4401 ;*****************************************************************************
                     0028  4402
                     0028  4403         SUBTEST                                         ;SUBTEST #1
                     002B
                     002B       ;/////////////////////////////////////////////////////////////////////////////
                     002B  4404 ;+
                     002B  4405 ;SUBTEST #1
                     002B  4406 ;
                     002B  4407 ;       TEST "BUS/READ.MOD" WITH NO MICRO TRAP
                     002B  4408 ;-
                     002B  4409         INITIALIZE                                      ;INIT CPU
                     002D  4410         EXPUTRAP                                        ;POSSIBLE FAILURE MODE
                     002F  4411         LOADDCS         4$,,08,1                        ;MODIFY DCS ADDRESS 07
                     0036  4412         ERRLOOP                                         ;ERROR LOOP
                     0038  4413         FETCH           0                               ;START DCS PROGRAM
                     003D  4414         BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                     0041  4415         CMPREGMSK       CSTRP,3$,,5$,,W,NE              ;CHECK FOR ACVR U-TRAP
                     004D  4416         IFERROR         ,<<ACV>>,                       ;POSSIBLE FAILING ARRAY
                     0053  4417         CMPREGMSK       CSTRP,6$,,5$,,W,                ;CHECK FOR UNEXPECTED U-TRAP
                     005F  4418         IFERROR         ,<<ACV>>,                       ;POSSIBLE FAILING ARRAY
                     0065  4419         CMPREG          WHREG,1$,,L,                    ;COMPARE RESULTS
                     006E  4420         IFERROR         ,<<ACV>>,                       ;POSSIBLE FAILING ARRAY
                     0074  4421
                     0074  4422         SUBTEST                                         ;SUBTEST #2
                     0077
                     0077       ;/////////////////////////////////////////////////////////////////////////////
                     0077  4423 ;+
                     0077  4424 ;SUBTEST #2
                     0077  4425 ;
                     0077  4426 ;       TEST "BUS/READ.MOD" WITH MICRO TRAP
                     0077  4427 ;-
                     0077  4428         INITIALIZE                                      ;INIT CPU
                     0079  4429         EXPUTRAP                                        ;EXPECTED U-TRAP
                     007B  4430         LOADDCS         2$,,08,1                        ;MODIFY DCS ADDRESS 07
                     0082  4431         ERRLOOP                                         ;ERROR LOOP
                     0084  4432         FETCH           0                               ;START DCS PROGRAM
                     0089  4433         BRSTCLK         <^X0E>                          ;END DCS PROGRAM
                     008D  4434         CMPREGMSK       CSTRP,3$,,5$,,W,                ;COMPARE CS ADDRESS
                     0099  4435         IFERROR         ,<<ACV>>,                       ;POSSIBLE FAILING ARRAY
                     009F  4436         FETCH           <^X0F>                          ;START DCS PROGRAM
                     00A4  4437         BRSTCLK         <^X12>                          ;END DCS PROGRAM
                     00A8  4438         CMPREGMSK       WHREG,7$,,8$,,L,                ;TEST SAVED MODE REGISTER
                     00B4  4439         IFERROR         ,<<ADK>>,                       ;FAILING ARRAY
```

```
                                00BA  4440          SKIP
                                00C1  4441
                                00C1  4442 BEGIN_TEST_DATA
                                00C1
                                00C1          ;-------------------------------------------------------------------------
                                00C1  4443
                     FFFFFFFF   00C1  4444 1$:      .LONG    ^XFFFFFFFF                      ;RESULTS SUBTEST #1
                                00C5  4445
                                00C5  4446 2$:
U 08, 7700,B800,7FFF,FF67,8470,1809 00C5  4447 ;% 08   LONLIT_[00000130]                    ;PTE SUBTEST #2
                                00D0  4451
                         002E   00D0  4452 3$:      .WORD    ^X002E                          ;RESULTS SUBTEST #2
                                00D2  4453
                                00D2  4454 4$:
U 08, 7700,F800,7FFF,FF6F,8470,1809 00D2  4455 ;% 08   LONLIT_[00000120]                    ;PTE SUBTEST #1
                                00DD  4459
                         3FFF   00DD  4460 5$:      .WORD    ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO-OP
                         180E   00DF  4461 6$:      .WORD    ^X180E                          ;ADD DCS SHOULD HALT AT
                     01000000   00E1  4462 7$:      .LONG    ^X01000000                      ;SAVED MODE REGISTER RESULTS
                     0F000000   00E5  4463 8$:      .LONG    ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO-OP
                                00E9  4464
                                00E9  4465 BEGIN_CPU_DATA
                                0002  4466
                                0002  4467 DCS_DATA
                                0001  4468
U 00, 7700,C800,0364,0300,0470,1801 0001  4469 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  4473 ;% 01   MEMSCAR_R[TEMP0]                     ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803 0017  4477 ;% 02   MEMSCR_R[TEMP1]                      ;SET MME ON
U 03, 7700,8800,5BE4,0308,6870,1804 0022  4481 ;% 03   MEMSCAR_R[TEMP2]                     ;REF CACHE ONLY ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805 002D  4485 ;% 04   MEMSCR_R[TEMP3]                      ;TURN CMI OFF
U 05, 7700,4800,5BE6,0310,0470,1806 0038  4489 ;% 05   D_R[TEMP4]                           ;0'S TO D REGISTER
U 06, 7700,5800,DB13,0300,4A70,1807 0043  4493 ;% 06   VA_Q_D_D+ZLIT8[0]                    ;0'S TO THE VA REGISTER
U 07, 7700,1800,DB12,0300,5360,1808 004E  4497 ;% 07   VA_D_D+ZLIT8[0] INVALIDATE TB       ;INVALIDATE TB AT VA
U 08, 7700,F800,7FFF,FF6F,8470,1809 0059  4501 ;% 08   LONLIT_[00000120]                    ;PTE
U 09, 7700,C81B,5BE4,03D4,5070,180A 0064  4505 ;% 09   TB_R[LONLIT]                        ;TB GETS PTE
U 0A, 7700,4800,5BF4,0310,0070,180B 006F  4509 ;% 0A   PSL_R[TEMP4]                        ;SET KERNEL MODE IN PSL
U 0B, 7700,0800,5BE4,0310,4670,180C 007A  4513 ;% 0B   WB_R[TEMP4],WCTRL/MDR_WB           ;0'S TO MDR
U 0C, 7701,0800,0364,0200,0540,180D 0085  4517 ;% 0C   BUS/READ.MOD,SIZE[LONG]            ;PERFORM READ MODIFY
U 0D, 6700,8812,5924,0300,0470,180E 0090  4521 ;% 0D   RDM_WB,WB_M[MDR]                   ;READ MDR TO RDM
U 0E, 7300,4800,0364,0300,0470,180F 009B  4525 ;% 0E   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
U 0F, 7700,C800,0364,0300,0470,1810 00A6  4529 ;% 0F   NOP
U 10, 7700,0800,5BE4,0310,6870,1811 00B1  4533 ;% 10   MEMSCAR_R[TEMP4]                    ;SAVED MODE REG ADDRESS
U 11, 6700,4800,0364,0300,6470,1812 00BC  4537 ;% 11   RDM_WB,WCTRL/MEMSCR                ;READ SAVED MODE REG TO RDM
U 12, 7300,C800,0364,0300,0470,1813 00C7  4541 ;% 12   NOP,STOP.CLOCK                     ;STOP CPU CLOCK
                                00D2  4545
                                00D2  4546 RTEMP_DATA
                                0001  4547
                     00000000   0001  4548          .LONG    ^X00000000                  ;PHY/VIR REG ADDRESS
                     01000000   0005  4549          .LONG    ^X01000000                  ;SET MME ON
                     0E000000   0009  4550          .LONG    ^X0E000000                  ;REFERENCE CACHE ONLY ADD
                     01000000   000D  4551          .LONG    ^X01000000                  ;TURN CMI OFF
                     00000000   0011  4552          .LONG    ^X00000000                  ;DATA
                     01000000   0015  4553          .LONG    ^X01000000                  ;SAVED MODE REG ADDRESS
                                0019  4554
                                0019  4555 CACHE_DATA
```

```
                    0001  4556
FFFFFFFF            0001  4557          .LONG   ^XFFFFFFFF                      ;TEST PATTERN
                    0005  4558
                    0005  4559 END_CPU_DATA
                    00E9  4560
                    00E9  4561 END_TEST_DATA
                    00E9
                    00E9          ;-----------------------------------------------------------------------
                    00E9  4562
                    00E9  4563 ENDTEST
```

```
0027          .SBTTL  "TEST   071     TEST BUS FIELD READ.NT MICRO ORDER
0027  4565 ;*******************************************************************************
0027  4566 ;++
0027  4567 ;
0027  4568 ; FUNCTIONAL DESCRIPTION:
0027  4569 ;
0027  4570 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE "BUS/READ.NT"
0027  4571 ;       MICRO ORDER.
0027  4572 ;
0027  4573 ;
0027  4574 ; TEST ALGORITHM:
0027  4575 ;
0027  4576 ;       1.  EXECUTE DCS PROGRAM
0027  4577 ;               1.  LOAD ADDRESS OF PHYSICAL/VIRTUAL ADDRESSING REGISTER INTO
0027  4578 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
0027  4579 ;               2.  SET MME ON
0027  4580 ;               3.  LOAD ADDRESS OF REFERENCE CACHE ONLY REGISTER INTO MEMSCAR
0027  4581 ;               4.  TURN CMI OFF
0027  4582 ;               5.  LOAD A 3 INTO VIRTUAL ADDRESS REGISTER (CAUSE UNALIGNED
0027  4583 ;                   DATA)
0027  4584 ;               6.  WRITE PTE INTO TRANSLATION BUFFER (SET NO ACCESS ALLOWED)
0027  4585 ;               7.  LOAD 0'S INTO THE PROCESSOR STATUS LONGWORD REGISTER
0027  4586 ;                   (SETS CURRENT MODE TO KERNEL)
0027  4587 ;               8.  LOAD 0'S INTO THE MDR
0027  4588 ;               9.  PERFORM BUS/READ.NT
0027  4589 ;              10.  READ MDR TO THE RDM
0027  4590 ;       2.  COMPARE RESULTS (EXPECTED AND RECEIVED)
0027  4591 ;
0027  4592 ;
0027  4593 ; TEST PATTERNS:
0027  4594 ;
0027  4595 ;       BUS FUNCTION            TEST PATTERN
0027  4596 ;
0027  4597 ;       BUS/READ.NT               FFFFFFFF
0027  4598 ;
0027  4599 ;
0027  4600 ; LOGIC DESCRIPTION:
0027  4601 ;
0027  4602 ;       THIS TEST VERIFIES THE CORRECT OPERATION OF THE "BUS/READ.NT"
0027  4603 ;       MICRO ORDER BY EXECUTING THE FUNCTION WITH UNALIGNED DATA AND
0027  4604 ;       AN ACCESS VIOLATION.  THE "ACV" GATE ARRAY IS RESPONSIBLE FOR
0027  4605 ;       DECODING THE MICRO FIELD AND BLOCKING THE MICRO TRAP.
0027  4606 ;
0027  4607 ;
0027  4608 ; ASSUMPTIONS:
0027  4609 ;
0027  4610 ;       THIS TEST ASSUMES THE WBUS, MBUS, DPM, CACHE, AND TB ARE ALL
0027  4611 ;       OPERATIONAL.
0027  4612 ;
0027  4613 ;
0027  4614 ; ERROR DESCRIPTION:
0027  4615 ;
0027  4616 ;       EXPECTED CACHE DATA
0027  4617 ;       RECEIVED CACHE DATA
0027  4618 ;
```

```
                     0027  4619 ;--
                     0027  4620 ;*******************************************************************************
                     0027  4621 ;///////////////////////////////////////////////////////////////////////////////
                     0027  4622
                     0027  4623         INITIALIZE                                      ;INIT CPU
                     0029  4624         EXPUTRAP                                        ;POSSIBLE U-TRAP
                     002B  4625         ERRLOOP                                         ;ERROR LOOP
                     002D  4626         FETCH           0                               ;START DCS PROGRAM
                     0032  4627         BRSTCLK         <^X0D>                          ;END DCS PROGRAM
                     0036  4628         CMPREGMSK       CSTRP,2$,,3$,,W,                ;TEST FOR U-TRAP
                     0042  4629         IFERROR         ,<<ACV>>,                       ;FAILING ARRAY
                     0048  4630         CMPREG          WHREG,1$,,L,                    ;COMPARE RESULTS
                     0051  4631         IFERROR         ,<<MDR>>,                       ;POSSIBLE FAILING ARRAY
                     0057  4632         FETCH           <^X0E>                          ;START DCS PROGRAM
                     005C  4633         BRSTCLK         <^X14>,INH                      ;END DCS PROGRAM
                     0060  4634         SKIP
                     0067  4635
                     0067  4636 BEGIN_TEST_DATA
                     0067
                     0067               ;-------------------------------------------------------------------------
                     0067  4637
FFFFFFFF             0067  4638 1$:      .LONG   ^XFFFFFFFF                              ;EXPECTED RESULTS
    180D             006B  4639 2$:      .WORD   ^X180D                                  ;ENDING ADDRESS OF TEST
    3FFF             006D  4640 3$:      .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO-OP
                     006F  4641
                     006F  4642 BEGIN_CPU_DATA
                     0002  4643
                     0002  4644 DCS_DATA
                     0001  4645
U 00, 7700,C800,0364,0300,0470,1801  0001  4646 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  4650 ;% 01   MEMSCAR_R[TEMP0]                  ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  4654 ;% 02   MEMSCR_R[TEMP1]                   ;SET MME ON
U 03, 7700,8800,5BE4,0308,6870,1804  0022  4658 ;% 03   MEMSCAR_R[TEMP2]                  ;REF CACHE ONLY ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805  002D  4662 ;% 04   MEMSCR_R[TEMP3]                   ;TURN CMI OFF
U 05, 7700,4800,5BE6,0310,0470,1806  0038  4666 ;% 05   D_R[TEMP4]                       ;3 TO D REGISTER
U 06, 7700,5800,DB13,0300,4A70,1807  0043  4670 ;% 06   VA_Q_D_D+ZLIT8[0]               ;VA REGISTER GETS ADDRESS
U 07, 7700,1800,DB12,0300,5360,1808  004E  4674 ;% 07   VA_D_D+ZLIT8[0] INVALIDATE TB   ;INVALIDATE TB AT VA
U 08, 7700,C81B,5BE4,0314,5070,1809  0059  4678 ;% 08   TB_R[TEMP5]                      ;PTE TO TB
U 09, 7700,8800,5BF4,0318,0070,180A  0064  4682 ;% 09   PSL_R[TEMP6]                     ;SET KERNEL MODE
U 0A, 7700,C800,5BE4,0318,4670,180B  006F  4686 ;% 0A   WB_R[TEMP6],WCTRL/MDR_WB        ;0'S TO MDR
U 0B, 7700,4800,0364,0300,0420,180C  007A  4690 ;% 0B   BUS/READ.NT                     ;READ NO TRAP
U 0C, 6700,8812,5924,0300,0470,180D  0085  4694 ;% 0C   RDM_WB,WB_M[MDR]                ;READ MDR TO RDM
U 0D, 7300,C800,0364,0300,0470,180E  0090  4698 ;% 0D   NOP,STOP.CLOCK                  ;STOP CPU CLOCK
                     009B  4702 ;
                     009B  4703 ; This section of microcode is use to se the VA past any valid cache data.
                     009B  4704 ; This is necessary in case this test is run in continous mode.  In this mode
                     009B  4705 ; if the VA points to valid cache data then the INITIALIZE pseudo will cause
                     009B  4706 ; that cache location to be invalidated
                     009B  4707 ;
U 0E, 7700,4800,0364,0300,0470,180F  009B  4708 ;% 0E   NOP
U 0F, 7700,9800,CB70,0307,6870,1810  00A6  4712 ;% 0F   MEMSCAR_ZLIT24[00E]             ;Ref. cache addr to memscar
U 10, 7700,5800,CB70,0300,E070,1811  00B1  4716 ;% 10   MEMSCR_ZLIT24[1]               ;Turn the CMI off
U 11, 7700,9800,C370,0380,4A70,1812  00BC  4720 ;% 11   VA_ZLIT0[100]                  ;Set VA to phy addr 100(hex)
U 12, 7700,C800,5BE4,03D8,4670,1813  00C7  4724 ;% 12   WB_R[ZERO],WCTRL/MDR_WB       ;Zero the MDR out
U 13, 7700,C800,0364,0300,0400,1814  00D2  4728 ;% 13   READ.PHY                       ;Latch VA to MA address
```

```
U 14, 7300,C800,0364,0300,0470,1815  00DD  4732 ;% 14   NOP,STOP.CLOCK
                                     00E8  4736
                                     00E8  4737
                                     00E8  4738 RTEMP_DATA
                                     0001  4739
                            00000000 0001  4740         .LONG   ^X00000000                  ;PHY/VIR REGISTER ADD
                            01000000 0005  4741         .LONG   ^X01000000                  ;SET MME ON
                            0E000000 0009  4742         .LONG   ^X0E000000                  ;REF CACHE ONLY ADD
                            01000000 000D  4743         .LONG   ^X01000000                  ;TURN CMI OFF
                            00000003 0011  4744         .LONG   ^X00000003                  ;ADDRESS FOR VA
                            00000100 0015  4745         .LONG   ^X00000100                  ;PTE FOR TB
                            00000000 0019  4746         .LONG   ^X00000000                  ;DATA
                                     001D  4747
                                     001D  4748 CACHE_DATA
                                     0001  4749
                            FFFFFFFF 0001  4750         .LONG   ^XFFFFFFFF                  ;TEST PATTERN
                            00000000 0005  4751         .LONG   ^X00000000
                                     0009  4752
                                     0009  4753 END_CPU_DATA
                                     006F  4754
                                     006F  4755 END_TEST_DATA
                                     006F
                                     006F       ;-------------------------------------------------------------------------
                                     006F  4756
                                     006F  4757 ENDTEST
```

```
                        .SBTTL  "TEST   072     TEST BUS FIELD READ.SEC MICRO ORDER
                4759 ;*****************************************************************************
                4760 ;++
                4761 ;
                4762 ; FUNCTIONAL DESCRIPTION:
                4763 ;
                4764 ;       THIS TEST WILL VERIFY CORRECT OPERATION OF THE "BUS/READ.SEC" MICRO
                4765 ;       ORDER USING VARIOUS VIRTUAL ADDRESSES.
                4766 ;
                4767 ;
                4768 ; TEST ALGORITHM:
                4769 ;
                4770 ;       1.  CREATE A TEST LOOP OF 4 ITERATIONS
                4771 ;       2.  LOAD WD REGISTER (RDM) WITH ADDRESS FOR VIRTUAL ADDRESS REGISTER
                4772 ;       3.  EXECUTE DCS PROGRAM
                4773 ;               1.  LOAD PHYSICAL/VIRTUAL ADDRESSING REGISTER ADDRESS INTO
                4774 ;                   MEMORY STATUS AND CONTROL ADDRESS REGISTER (MEMSCAR)
                4775 ;               2.  SET MME OFF
                4776 ;               3.  LOAD REFERENCE CACHE ONLY REGISTER ADDRESS INTO MEMSCAR
                4777 ;               4.  TURN CMI OFF
                4778 ;               5.  LOAD VIRTUAL ADDRESS REGISTER WITH ADDRESS IN WD REG (RDM)
                4779 ;               6.  LOAD MDR WITH 0'S
                4780 ;               7.  PERFORM "BUS/READ.SEC"
                4781 ;               8.  READ MDR BACK TO RDM
                4782 ;       4.  COMPARE RESULTS (EXPECTED AND RECEIVED)
                4783 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ITERATIONS
                4784 ;
                4785 ;
                4786 ; TEST PATTERNS:
                4787 ;
                4788 ;       ITERATION       VA REGISTER             EXPECTED DATA
                4789 ;
                4790 ;           1           00000000                00000000
                4791 ;           2           00000001                FF000000
                4792 ;           3           00000002                FFFF0000
                4793 ;           4           00000003                FFFFFF00
                4794 ;
                4795 ;
                4796 ; LOGIC DESCRIPTION:
                4797 ;
                4798 ;       THIS TEST INSURES THAT THE LOGIC WITHIN THE "MDR" GATE ARRAY WHICH
                4799 ;       IS RESPONSIBLE FOR LOADING THE MEMORY DATA REGISTER IS OPERATING
                4800 ;       PROPERLY.  IF PROBLEMS ARISE AND REPLACING "MDR" DOES NOT HELP
                4801 ;       THEN YOUR NEXT CHOICE SHOULD BE "ADK" OR "CAK".  BOTH OF THESE
                4802 ;       ARRAYS HAVE LOGIC WHICH IS USED TO CONTROL THE MDR DURING
                4803 ;       A BUS/READ.SEC.
                4804 ;
                4805 ;
                4806 ; ASSUMPTIONS:
                4807 ;
                4808 ;       THIS TEST ASSUMES THE WBUS, MBUS, AND DPM ARE OPERATIONAL
                4809 ;
                4810 ;
                4811 ; ERROR DESCRIPTION:
                4812 ;
```

```
                                  0028  ;       EXPECTED MDR DATA
                                  0028  ;       RECEIVED MDR DATA
                                  0028  ;       LOOP COUNT
                                  0028  ;
                                  0028  ;--
                                  0028  ;*******************************************************************************
                                  0028  ;///////////////////////////////////////////////////////////////////////////////
                                  0028
                                  0028          INITIALIZE                                  ;INIT CPU
                                  002A          LOOP            I,1,4                       ;CREATE A TEST LOOP
                                  0031          LOADREG         WDREG,1$,I,L                ;SELECT ADD FOR VA
                                  0039          ERRLOOP                                     ;ERROR LOOP
                                  003B          FETCH           0                           ;START DCS PROGRAM
                                  0040          BRSTCLK         <^X9>                       ;END DCS PROGRAM
                                  0044          CMPREG          WHREG,2$,I,L,               ;COMPARE RESULTS
                                  004D          IFERROR         ,<<MDR><ADK><CAK>>,         ;POSSIBLE FAILING ARRAYS
                                  0055          ENDLOOP         I                           ;ENDLOOP
                                  0058          FETCH           <^XA>                       ;START DCS PROGRAM
                                  005D          BRSTCLK         <^X10>,INH
                                  0061          SKIP                                        ;GO TO NEXT TEST
                                  0068
                                  0068  BEGIN_TEST_DATA
                                  0068
                                  0068          ;-----------------------------------------------------------------------
                                  0068
                        00000000  0068  1$:     .LONG   ^X00000000                          ;ADDRESSES FOR VA REGISTER
                        00000001  006C          .LONG   ^X00000001
                        00000002  0070          .LONG   ^X00000002
                        00000003  0074          .LONG   ^X00000003
                                  0078
                        00000000  0078  2$:     .LONG   ^X00000000                          ;RESULTS FOR EACH ITERATION
                        FF000000  007C          .LONG   ^XFF000000
                        FFFF0000  0080          .LONG   ^XFFFF0000
                        FFFFFF00  0084          .LONG   ^XFFFFFF00
                                  0088
                                  0088  BEGIN_CPU_DATA
                                  0002
                                  0002  DCS_DATA
                                  0001
U 00, 7700,C800,0364,0300,0470,1801  0001  ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  ;% 01   MEMSCAR_R[TEMP0]                           ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  ;% 02   MEMSCR_R[TEMP1]                            ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  ;% 03   MEMSCAR_R[TEMP2]                           ;REF CACHE ONLY ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805  002D  ;% 04   MEMSCR_R[TEMP3]                            ;TURN CMI OFF
U 05, 5700,4800,0364,0300,4A70,1806  0038  ;% 05   VA_RDM                                      ;ADD IN WD REG (RDM) TO VA
U 06, 7700,8800,5BE4,0304,4670,1807  0043  ;% 06   WB_R[TEMP1],WCTRL/MDR_WB                    ;0'S TO MDR
U 07, 7700,4800,0364,0300,0460,1808  004E  ;% 07   BUS/READ.SEC                               ;PERFORM READ SECOND REF
U 08, 6700,0812,5924,0300,0470,1809  0059  ;% 08   RDM_WB,WB_M[MDR]                           ;READ MDR TO RDM
U 09, 7300,4800,0364,0300,0470,180A  0064  ;% 09   NOP,STOP.CLOCK                             ;STOP CPU CLOCK
                                  006F  ;
                                  006F  ; This section of microcode is use to se the VA past any valid cache data.
                                  006F  ; This is necessary in case this test is run in continous mode.  In this mode
                                  006F  ; if the VA points to valid cache data then the INITIALIZE pseudo will cause
                                  006F  ; that cache location to be invalidated
                                  006F  ;
```

```
U 0A, 7700,C800,0364,0300,0470,180B  006F  4896 ;% 0A   NOP
U 0B, 7700,1800,CB70,0307,6870,180C  007A  4900 ;% 0B   MEMSCAR_ZLIT24[00E]                    ;Ref. cache addr to memscar
U 0C, 7700,D800,CB70,0300,E070,180D  0085  4904 ;% 0C   MEMSCR_ZLIT24[1]                      ;Turn the CMI off
U 0D, 7700,1800,C370,0380,4A70,180E  0090  4908 ;% 0D   VA_ZLIT0[100]                        ;Set VA to phy addr 100(hex)
U 0E, 7700,4800,5BE4,03D8,4670,180F  009B  4912 ;% 0E   WB_R[ZERO],WCTRL/MDR_WB             ;Zero the MDR out
U 0F, 7700,4800,0364,0300,0400,1810  00A6  4916 ;% 0F   READ.PHY                            ;Latch VA to MA address
U 10, 7300,4800,0364,0300,0470,1811  00B1  4920 ;% 10   NOP,STOP.CLOCK
                                     00BC  4924
                                     00BC  4925
                                     00BC  4926
                                     00BC  4927 RTEMP_DATA
                                     0001  4928
                          00000000   0001  4929          .LONG   ^X00000000                 ;PHY/VIR REG ADD
                          00000000   0005  4930          .LONG   ^X00000000                 ;DATA
                          0E000000   0009  4931          .LONG   ^X0E000000                 ;REF CACHE ONLY REG ADD
                          01000000   000D  4932          .LONG   ^X01000000                 ;TURN CMI OFF
                          06000000   0011  4933          .LONG   ^X06000000
                          FFFFFFFF   0015  4934          .LONG   ^XFFFFFFFF
                                     0019  4935
                                     0019  4936 CACHE_DATA
                                     0001  4937
                          FFFFFFFF   0001  4938          .LONG   ^XFFFFFFFF                 ;TEST PATTERN
                          00000000   0005  4939          .LONG   ^X00000000
                                     0009  4940
                                     0009  4941 END_CPU_DATA
                                     0088  4942
                                     0088  4943 END_TEST_DATA
                                     0088
                                     0088          ;-------------------------------------------------------------------------------
                                     0088  4944
                                     0088  4945 ENDTEST
```

```
001D        .SBTTL  "TEST   073     TEST SECOND MDR REGISTER
001D  4947 ;*******************************************************************************
001D  4948 ;++
001D  4949 ;
001D  4950 ; FUNCTIONAL DESCRIPTION:
001D  4951 ;
001D  4952 ;       THIS TEST WILL CHECK THE OPERATION OF THE SECOND MEMORY DATA REGISTER
001D  4953 ;       (MDR).
001D  4954 ;
001D  4955 ;
001D  4956 ; TEST ALGORITHM:
001D  4957 ;
001D  4958 ;       1.  CREATE A TEST LOOP OF 6
001D  4959 ;       2.  GENERATE A TEST PATTERN
001D  4960 ;       3.  LOAD TEST PATTERN INTO RDM WDREG
001D  4961 ;       4.  EXECUTE DCS PROGRAM
001D  4962 ;               a.  MOVE TEST PATTERN INTO MDR REGISTER
001D  4963 ;               b.  ZERO THE VA REGISTER
001D  4964 ;               c.  PERFORM A READ WHILE SOURCING THE MDR REGISTER
001D  4965 ;               d.  PERFORM A RETURN MICRO ORDER TO SELECT THE SECOND MDR
001D  4966 ;               e.  PERFORM A READ WHILE SOURCING THE SECOND MDR TO THE RDM
001D  4967 ;       5.  TEST THE VBUS FOR RTUT DINH
001D  4968 ;       6.  IF NO ERROR TEST THE WHREG FOR THE CORRECT TEST PATTERN
001D  4969 ;       7.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
001D  4970 ;
001D  4971 ;
001D  4972 ; TEST PATTERNS:
001D  4973 ;
001D  4974 ;       ITERATION       TEST PATTERN
001D  4975 ;           1             AAAAAAAA
001D  4976 ;           2             55555555
001D  4977 ;           3             33333333
001D  4978 ;           4             0F0F0F0F
001D  4979 ;           5             00FF00FF
001D  4980 ;           6             0000FFFF
001D  4981 ;
001D  4982 ;
001D  4983 ; LOGIC DESCRIPTION:
001D  4984 ;
001D  4985 ;       THIS TEST WILL VERIFY THE OPERATION OF THE SECOND MDR REGISTER.  ANY
001D  4986 ;       FAILURES SHOULD BE CONFINED TO THE MDR GATE ARRAY.  THE ONLY OTHER
001D  4987 ;       POSSIBLE FAILURE IS THAT UBI DIDNOT DECODE THE MISC/RSBC MICRO ORDER.
001D  4988 ;       THIS WOULD RESULT IN THE VBUS FAILING.
001D  4989 ;
001D  4990 ;
001D  4991 ; ASSUMPTIONS:
001D  4992 ;
001D  4993 ;       THIS TEST ASSUMES THE CACHE IS OPERATIONAL.
001D  4994 ;
001D  4995 ;
001D  4996 ; ERROR DESCRIPTION:
001D  4997 ;
001D  4998 ;       FIRST IFERROR:
001D  4999 ;               EXPECTED VBUS DATA
001D  5000 ;               RECEIVED VBUS DATA
```

```
                        001D  5001 ;               LOOP COUNT I
                        001D  5002 ;                       (BITS <7:0> = VBUS BIT POSITION)
                        001D  5003 ;                       (BITS <8> = BIT STATE)
                        001D  5004 ;
                        001D  5005 ;       SECOND IFERROR:
                        001D  5006 ;               EXPECTED SECOND MDR REGISTER CONTENTS
                        001D  5007 ;               RECEIVED SECOND MDR REGISTER CONTENTS
                        001D  5008 ;
                        001D  5009 ;--
                        001D  5010 ;*******************************************************************************
                        001D  5011
                        001D  5012
                        001D  5013
                        001D  5014          INITIALIZE                                  ;INIT CPU
                        001F  5015          LOOP            I,1,6                       ;CREATE A TEST LOOP
                        0026  5016          SA01PAT         I,1$,,                      ;GENERATE A TEST PATTERN
                        002E  5017          LOADREG         WDREG,1$,,L                 ;LOAD TEST PATTERN INTO WD REG
                        0036  5018          ERRLOOP                                     ;ERROR LOOP
                        0038  5019          FETCH           0                           ;START DCS PROGRAM
                        003D  5020          BRSTCLK         6                           ;END DCS PROGRAM
                        0041  5021          CMPVBUS         2$                          ;TEST FOR RTUT DINH L
                        0046  5022          IFERROR         ,,<<UBI>>                   ;SIGNAL NOT PRESENT
                        004C  5023          CMPREG          WHREG,1$,,L,                ;COMPARE RESULTS
                        0055  5024          IFERROR         ,<<MDR>>,                   ;CAUSE ERROR PRINTOUT
                        005B  5025          ENDLOOP         I                           ;END TEST LOOP
                        005E  5026          SKIP                                        ;GO TO NEXT TEST
                        0065  5027
                        0065  5028
                        0065  5029
                        0065  5030          BEGIN_TEST_DATA
                        0065
                        0065        ;-------------------------------------------------------------------------
                        0065  5031
                00000000  0065  5032 1$:       .LONG   ^X00000000                      ;TEST PATTERN STORAGE LOCATION
                      01  0069  5033 2$:       .BYTE   ^X1
                        006A  5034          VBUSDATA        <^X03>,0                    ;RTUT DINH L ON VBUS
                        006B  5035
                        006B  5036
                        006B  5037
                        006B  5038 BEGIN_CPU_DATA
                        0002  5039
                        0002  5040 DCS_DATA
                        0001  5041
U 00, 7700,C800,0364,0300,0470,1801   0001  5042 ;% 00    NOP
U 01, 5700,C800,0364,0300,4670,1802   000C  5046 ;% 01    WCTRL/MDR_WB,WB_RDM                 ;TEST PATTERN TO MDR
U 02, 7700,4800,5BE4,03D8,4A70,1803   0017  5050 ;% 02    VA_R[ZERO]                          ;ZERO THE VA REGISTER
U 03, 7700,0812,5924,0300,0400,1804   0022  5054 ;% 03    READ,PHY,WB_M[MDR]                  ;LOAD THE SECOND MDR
U 04, 7700,CD80,0364,0300,0470,1805   002D  5058 ;% 04    MISC/RSBC                           ;SET UP RTUT DINH L
U 05, 6600,8812,5924,0300,0400,1806   0038  5062 ;% 05    RDM_WB,WB_M[MDR],READ.PHY,STROBE.V.BUS  ;READ SECOND MDR TO RDM
U 06, 7300,C800,0364,0300,0470,1807   0043  5066 ;% 06    NOP,STOP.CLOCK                      ;STOP THE CPU CLOCK
                        004E  5070
                        004E  5071 CACHE_DATA
                        0001  5072
                00000000  0001  5073          .LONG   ^X00000000                      ;ZERO CACHE ADDRESS 0
                        0005  5074
```

```
                0005  5075 END_CPU_DATA
                006B  5076
                006B  5077
                006B  5078
                006B  5079 END_TEST_DATA
                006B
                006B       ;-----------------------------------------------------------------------
                006B  5080 ENDTEST
```

```
                        001C        .SBTTL  "TEST   074     TEST CMI DATA INTEGRITY
                        001C  5082 ;*******************************************************************************
                        001C  5083 ;++
                        001C  5084 ;
                        001C  5085 ; FUNCTIONAL DESCRIPTION:
                        001C  5086 ;
                        001C  5087 ;       THIS IS THE FIRST TEST OF THE CMI BUS.  IT WILL CHECK TO SEE IF
                        001C  5088 ;       ANY DATA BITS ARE STUCK.
                        001C  5089 ;
                        001C  5090 ;
                        001C  5091 ; TEST ALGORITHM:
                        001C  5092 ;
                        001C  5093 ;       SUBTEST #1
                        001C  5094 ;       1.  CREATE A TEST LOOP OF 6
                        001C  5095 ;       2.  GENERATE A TEST PATTERN
                        001C  5096 ;       3.  LOAD THE TEST PATTERN INTO THE RDM MA REGISTER
                        001C  5097 ;       4.  ENABLE MA REGISTER ONTO CMI BUS
                        001C  5098 ;       5.  READ CMI BUS INTO RDM MH REGISTER
                        001C  5099 ;       6.  CLEAR THE RDM MAIN32 REGISTER
                        001C  5100 ;       7.  TEST THE MH REGISTER FOR THE CORRECT PATTERN
                        001C  5101 ;       8.  IF AN ERROR GO TO SUBTEST 3 ELSE GO TO STEP 2 FOR REMAINING PATT.
                        001C  5102 ;
                        001C  5103 ;       SUBTEST #2
                        001C  5104 ;       1.  CREATE A TEST LOOP OF 6
                        001C  5105 ;       2.  GENERATE A TEST PATTERN
                        001C  5106 ;       3.  LOAD THE TEST PATTERN INTO THE RDM MD REGISTER
                        001C  5107 ;       4.  ENABLE MD REGISTER ONTO CMI BUS
                        001C  5108 ;       5.  READ CMI BUS INTO RDM MH REGISTER
                        001C  5109 ;       6.  CLEAR THE RDM MAIN32 REGISTER
                        001C  5110 ;       7.  TEST THE MH REGISTER FOR THE CORRECT PATTERN
                        001C  5111 ;       8.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                        001C  5112 ;
                        001C  5113 ;       SUBTEST #3
                        001C  5114 ;       1.  CREATE A TEST LOOP OF 6
                        001C  5115 ;       2.  GENERATE A TEST PATTERN
                        001C  5116 ;       3.  LOAD THE TEST PATTERN INTO THE RDM MD REGISTER
                        001C  5117 ;       4.  ENABLE MD REGISTER ONTO CMI BUS
                        001C  5118 ;       5.  READ CMI BUS INTO RDM MH REGISTER
                        001C  5119 ;       6.  CLEAR THE RDM MAIN32 REGISTER
                        001C  5120 ;       7.  TEST THE MH REGISTER FOR THE CORRECT PATTERN
                        001C  5121 ;       8.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                        001C  5122 ;       9.  IF LOOP EXHUSTED FORCE AN ERROR TO CALL OUT RDM MA REGISTER
                        001C  5123 ;
                        001C  5124 ;
                        001C  5125 ; TEST PATTERNS:
                        001C  5126 ;
                        001C  5127 ;       LOOP COUNT      TEST PATTERN
                        001C  5128 ;           1             AAAAAAAA
                        001C  5129 ;           2             55555555
                        001C  5130 ;           3             33333333
                        001C  5131 ;           4             0F0F0F0F
                        001C  5132 ;           5             00FF00FF
                        001C  5133 ;           6             0000FFFF
                        001C  5134 ;
                        001C  5135 ;
```

```
                        ; LOGIC DESCRIPTION:
                        ;
                        ;       THIS TEST SIMPLY TRIES TO DRIVE BITS ONTO THE CMI DATA BUS FROM THE
                        ;       RDM MA AND MD REGISTERS.  THEY ARE THEN READ BACK ONTO THE RDM MH
                        ;       REGISTER.  IF THE TEST FAILS, IT IS EITHER THE RDM, OR BITS ARE BEING
                        ;       HELD BY SOMEOTHER MODULE ON THE CMI.  WHAT MODULES ARE ON THE CMI WILL
                        ;       DEPEND ON THE SYSTEM CONFIGURATION.  DON'T FORGET THINGS LIKE WCS, MBA,
                        ;       etc.
                        ;
                        ; ASSUMPTIONS:
                        ;
                        ;       THIS TEST ASSUMES THE RDM POWER-UP SELF TEST HAS RUN SUCCESSFULLY
                        ;
                        ;
                        ; ERROR DESCRIPTION:
                        ;
                        ;       EXPECTED TEST PATTERN ON CMI
                        ;       RECEIVED TEST PATTERN FROM CMI
                        ;       LOOP COUNT
                        ;
                        ;--
                        ;*****************************************************************************
                        ;/////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       TEST MAREG TO CMI
                        ;-
001C  5165              SUBTEST         1                               ;SUBTEST #1
001F
001F                    ;/////////////////////////////////////////////////////////////////////////////
001F  5166              INITIALIZE                                      ;INIT THE CPU
0021  5167              LOOP            I,1,6                           ;CREATE A TEST LOOP OF 6
0028  5168              SA01PAT         I,1$,,                          ;GENERATE A TEST PATTERN
0030  5169              LOADREG         MAREG,1$,,L                     ;PUT TEST PATTERN IN MAREG
0038  5170              LOADREG         MAIN32,2$,,B                    ;ENABLE MAREG TO CMI
0040  5171              LOADREG         MAIN32,3$,,B                    ;STROBE MHREG
0048  5172              LOADREG         MAIN32,6$,,B                    ;CLEAR THE MAIN32 REGISTER
0050  5173              CMPREG          MHREG,1$,,L,                    ;TEST FOR CORRECT TEST PATTERN
0059  5174              SKIP            100$,ONERROR                    ;TRY CPU TO MAIN MEMORY
0060  5175              ENDLOOP         I                               ;END TEST LOOP
0063  5176
0063  5177
0063  5178
0063  5179  ;/////////////////////////////////////////////////////////////////////////////
0063  5180  ;+
0063  5181  ;SUBTEST #2
0063  5182  ;
0063  5183  ;       TEST MDREG TO CMI
0063  5184  ;-
0063  5185          SUBTEST         2                               ;SUBTEST #2
0066
0066            ;/////////////////////////////////////////////////////////////////////////////
0066  5186          INITIALIZE                                      ;INIT THE CPU
```

```
                     0068  5187          LOOP            I,1,6                   ;CREATE A TEST LOOP OF 6
                     006F  5188          SA01PAT         I,1$,,                  ;GENERATE A TEST PATTERN
                     0077  5189          ERRLOOP                                 ;ERROR LOOP POINT
                     0079  5190          LOADREG         MDREG,1$,,L             ;PUT TEST PATTERN INTO MDREG
                     0081  5191          LOADREG         MAIN32,4$,,B            ;ENABLE MDREG TO CMI
                     0089  5192          LOADREG         MAIN32,5$,,B            ;STROBE MHREG
                     0091  5193          LOADREG         MAIN32,6$,,B            ;CLEAR THE MAIN32 REGISTER
                     0099  5194          CMPREG          MHREG,1$,,L,            ;TEST FOR CORRECT PATTERN
                     00A2  5195          IFERROR         ,,<<RDM>>               ;MDREG ON RDM IS BAD
                     00A8  5196          ENDLOOP         I                       ;END TEST LOOP
                     00AB  5197          SKIP                                    ;GO TO NEXT TEST
                     00B2  5198
                     00B2  5199
                     00B2  5200 ;////////////////////////////////////////////////////////////////////////////
                     00B2  5201 ;+
                     00B2  5202 ;SUBTEST #3
                     00B2  5203 ;
                     00B2  5204 ;       TEST MDREG TO CMI
                     00B2  5205 ;-
                     00B2  5206 100$:    SUBTEST         3                       ;SUBTEST #3
                     00B5
                     00B5       ;////////////////////////////////////////////////////////////////////////////
                     00B5  5207          INITIALIZE                              ;INIT THE CPU
                     00B7  5208          LOOP            I,1,6                   ;CREATE A TEST LOOP OF 6
                     00BE  5209          SA01PAT         I,1$,,                  ;GENERATE A TEST PATTERN
                     00C6  5210          ERRLOOP                                 ;ERROR LOOP POINT
                     00C8  5211          LOADREG         MDREG,1$,,L             ;PUT TEST PATTERN INTO MDREG
                     00D0  5212          LOADREG         MAIN32,4$,,B            ;ENABLE MDREG TO CMI
                     00D8  5213          LOADREG         MAIN32,5$,,B            ;STROBE MHREG
                     00E0  5214          LOADREG         MAIN32,6$,,B            ;CLEAR MAIN32 REGISTER
                     00E8  5215          CMPREG          MHREG,1$,,L,            ;TEST FOR CORRECT PATTERN
                     00F1  5216          IFERROR         ,,<<RDM><CMI>>          ;SOMETHING IS TIED TO CMI
                     00F8  5217          ENDLOOP         I                       ;END TEST LOOP
                     00FB  5218          CMPREG          MHREG,1$,,L,NE          ;FORCE AN ERROR
                     0104  5219          IFERROR         ,,<<RDM>>               ;MAREG ON RDM IS BAD
                     010A  5220          SKIP                                    ;GO TO NEXT TEST
                     0111  5221
                     0111  5222
                     0111  5223
                     0111  5224 BEGIN_TEST_DATA
                     0111
                     0111       ;-----------------------------------------------------------------------------
                     0111  5225
00000000             0111  5226 1$:      .LONG    ^X00000000                     ;TEST PATTERN STORAGE
       81            0115  5227 2$:      .BYTE    ^X81                           ;ENABLE MAREG TO CMI
       A1            0116  5228 3$:      .BYTE    ^XA1                           ;STROBE MHREG
       41            0117  5229 4$:      .BYTE    ^X41                           ;ENABLE MDREG TO CMI
       61            0118  5230 5$:      .BYTE    ^X61                           ;STROBE MHREG
       01            0119  5231 6$:      .BYTE    ^X01                           ;CLEAR MAIN32 REGISTER
                     011A  5232
                     011A  5233 END_TEST_DATA
                     011A
                     011A       ;-----------------------------------------------------------------------------
                     011A  5234 ENDTEST
```

```
0020                    .SBTTL  "TEST   075     TEST CMI CONTROL AND STATUS
0020    5236    ;*****************************************************************************
0020    5237    ;++
0020    5238    ;
0020    5239    ; FUNCTIONAL DESCRIPTION:
0020    5240    ;
0020    5241    ;       THIS TEST WILL PERFORM A CMI BUS CYCLE FROM THE RDM TO DETERMINE IF
0020    5242    ;       THE CONTROL AND STATUS LINES ARE OPERATIONAL.
0020    5243    ;
0020    5244    ; TEST ALGORITHM:
0020    5245    ;
0020    5246    ;       SUBTEST #1
0020    5247    ;       1.  CREATE A TEST LOOP OF 6
0020    5248    ;       2.  GENERATE A TEST PATTERN
0020    5249    ;       3.  LOAD THE FIRST DCS MICROINSTRUCTION
0020    5250    ;       4.  ALLOW THE CPU CLOCK TO FREE RUN
0020    5251    ;       5.  CLEAR THE CMICTL REGISTER
0020    5252    ;       6.  LOAD THE MAREG WITH A WRITE COMMAND
0020    5253    ;       7.  LOAD THE TEST PATTERN INTO THE MDREG
0020    5254    ;       8.  ALLOW THE RDM TO PERFORM THE BUS CYCLE
0020    5255    ;       9.  STOP THE CPU CLOCK
0020    5256    ;      10.  TEST THE MHREG FOR THE TEST PATTERN
0020    5257    ;      11.  IF NO ERROR TEST THE STATUS REGISTER FOR GOOD STATUS
0020    5258    ;      12.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
0020    5259    ;
0020    5260    ;       SUBTEST #2
0020    5261    ;       1.  SAME AS SUBTEST 1 ONLY CMI CYCLE IS TO A NONEXISTANT ADDRESS
0020    5262    ;       2.  TEST STATUS REGISTER FOR NO RESPONSE
0020    5263    ;
0020    5264    ;
0020    5265    ; TEST PATTERNS:
0020    5266    ;
0020    5267    ;       ITERATION       TEST PATTERN
0020    5268    ;           1             AAAAAAAA
0020    5269    ;           2             55555555
0020    5270    ;           3             33333333
0020    5271    ;           4             0F0F0F0F
0020    5272    ;           5             00FF00FF
0020    5273    ;           6             0000FFFF
0020    5274    ;
0020    5275    ;
0020    5276    ; LOGIC DESCRIPTION:
0020    5277    ;
0020    5278    ;       THIS TEST ATTEMPTS TO CHECK THE CONTROL AND STATUS LINES ON THE CMI BY
0020    5279    ;       PERFORMING A WRITE OPERATION TO MAIN MEMORY.  BY SIMULTANEOUSLY SETTING
0020    5280    ;       THE WRITE AND READ BITS IN THE RDM CMICTL REGISTER YOU CAN CAUSE THE
0020    5281    ;       RDM TO PERFORM A CMI CYCLE.  THE DATA LOADED INTO THE MHREG WILL BE
0020    5282    ;       WHAT IS SENT FROM THE MDREG.
0020    5283    ;
0020    5284    ;
0020    5285    ; ASSUMPTIONS:
0020    5286    ;
0020    5287    ;       THIS TEST ASSUMES THE CPU CLOCK IS OPERATIONAL.
0020    5288    ;
0020    5289    ;
```

```
                        ; ERROR DESCRIPTION:
                        ;
                        ;       SUBTEST #1:
                        ;       FIRST IFERROR
                        ;               EXPECTED DATA FROM CMI
                        ;               RECEIVED DATA FROM CMI
                        ;               LOOP COUNT I
                        ;
                        ;       SECOND IFERROR
                        ;               EXPECTED STATUS FROM CMI
                        ;               RECEIVED STATUS FROM CMI
                        ;               LOOP COUNT I
                        ;
                        ;       SUBTEST #2:
                        ;               EXPECTED STATUS FROM CMI
                        ;               RECEIVED STATUS FROM CMI
                        ;
                        ;--
                        ;*****************************************************************************
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       TEST CMI CONTROL AND STATUS LINES
                        ;-
0020  5314              SUBTEST                                         ;SUBTEST #1
0023
0023                    ;/////////////////////////////////////////////////////////////////////////////
0023  5315              INITIALIZE                                      ;INIT THE CPU
0025  5316              LOOP            I,1,6                           ;CREATE A TEST LOOP OF 6
002C  5317              SA01PAT         I,5$,,                          ;GENERATE A TEST PATTERN
0034  5318              ERRLOOP                                         ;ERROR LOOP POINT
0036  5319              FETCH           0                               ;START DCS PROGRAM
003B  5320              LOADREG         DCSCR,2$,,B                     ;START THE CPU CLOCK
0043  5321              LOADREG         CMICTL,3$,,B                    ;CLEAR THE CMICTL REGISTER
004B  5322              LOADREG         MAREG,4$,,L                     ;LOAD MAREG WITH WRITE COMMAND
0053  5323              LOADREG         MDREG,5$,,L                     ;LOAD MDREG WITH WRITE DATA
005B  5324              LOADREG         CMICTL,6$,,B                    ;PERFORM CMI WRITE/READ
0063  5325              LOADREG         DCSCR,7$,,B                     ;STOP THE CPU CLOCK
006B  5326              CMPREG          MHREG,5$,,L,                    ;TEST FOR CORRECT TEST PATTERN
0074  5327              IFERROR         ,,<<RDM><CMI>>                  ;CMI OR RDM FAILURE
007B  5328              CMPREGMSK       STATUS,8$,,8$,,B,               ;TEST FOR GOOD STATUS
0087  5329              IFERROR         ,,<<RDM><MC0><MC1><CMI>>;CMI, CMC OR RDM FAILURE
0090  5330              ENDLOOP         I                               ;END TEST LOOP
0093  5331
0093  5332
0093  5333
0093  5334  ;/////////////////////////////////////////////////////////////////////////////
0093  5335  ;+
0093  5336  ;SUBTEST #2
0093  5337  ;
0093  5338  ;       TEST STATUS LINES FOR NONEXISTANT MEMORY
0093  5339  ;-
0093  5340          SUBTEST                                             ;SUBTEST #2
0096
0096        ;/////////////////////////////////////////////////////////////////////////////
```

```
                     0096  5341          INITIALIZE                                ;INIT THE CPU
                     0098  5342          ERRLOOP                                   ;ERROR LOOP POINT
                     009A  5343          FETCH           0                         ;START DCS PROGRAM
                     009F  5344          LOADREG         DCSCR,2$,,B               ;START CPU CLOCK
                     00A7  5345          LOADREG         CMICTL,3$,,B              ;CLEAR THE CMICTL REGISTER
                     00AF  5346          LOADREG         MAREG,9$,,L               ;LOAD NONEXISTANT ADDRESS
                     00B7  5347          LOADREG         CMICTL,6$,,L              ;PERFORM CMI WRITE/READ
                     00BF  5348          LOADREG         DCSCR,7$,,B               ;STOP THE CPU CLOCK
                     00C7  5349          CMPREGMSK       STATUS,10$,,11$,,B,       ;TEST FOR NO RESPONSE
                     00D3  5350          IFERROR         ,,<<RDM><CMI>>            ;INCORRECT STATUS
                     00DA  5351          INITIALIZE                                ;INIT THE CPU
                     00DC  5352          SKIP                                      ;GO TO NEXT TEST
                     00E3  5353
                     00E3  5354
                     00E3  5355
                     00E3  5356 BEGIN_TEST_DATA
                     00E3
                     00E3          ;-------------------------------------------------------------------------
                     00E3  5357
           00        00E3  5358 2$:      .BYTE   ^X00                              ;CLEAR REGISTERS
           01        00E4  5359 3$:      .BYTE   ^X01                              ;CLEAR CMICTL REGISTER
   F8000000        00E5  5360 4$:      .LONG   ^XF8000000                        ;WRITE COMMAND TO ADDRESS ZERO
   00000000        00E9  5361 5$:      .LONG   ^X00000000                        ;TEST PATTERN STORAGE
           68        00ED  5362 6$:      .BYTE   ^X68                              ;PERFORM CMI WRITE/READ
           60        00EE  5363 7$:      .BYTE   ^X60                              ;STOP CPU CLOCK
           38        00EF  5364 8$:      .BYTE   ^X38                              ;GOOD STATUS
   F8F40000        00F0  5365 9$:      .LONG   ^XF8F40000                        ;WRITE TO NONEXISTANT MEMORY
           20        00F4  5366 10$:     .BYTE   ^X20                              ;NO RESPONSE STATUS
           38        00F5  5367 11$:     .BYTE   ^X38                              ;MASK FOR CMPREGMSK PSEUDO OP
                     00F6  5368
                     00F6  5369
                     00F6  5370 BEGIN_CPU_DATA
                     0002  5371
                     0002  5372 DCS_DATA
                     0001  5373
U 00, 7700,C800,0364,0300,0470,1801  0001  5374 ;% 00    NOP
U 01, 7500,C800,0364,0300,0470,1802  000C  5378 ;% 01    NOP,DCS.ADDR_0            ;RUN B CLOCK
                     0017  5382
                     0017  5383 END_CPU_DATA
                     00F6  5384
                     00F6  5385
                     00F6  5386
                     00F6  5387 END_TEST_DATA
                     00F6
                     00F6          ;-------------------------------------------------------------------------
                     00F6  5388 ENDTEST
```

```
0019          .SBTTL  "TEST   076     TEST CPU CMI DRIVERS
0019  5390 ;*******************************************************************************
0019  5391 ;++
0019  5392 ;
0019  5393 ; FUNCTIONAL DESCRIPTION:
0019  5394 ;
0019  5395 ;       THIS TEST WILL VERIFY THAT THE MIC MODULE CAN DRIVE DATA ONTO THE
0019  5396 ;       CMI BUS.
0019  5397 ;
0019  5398 ;
0019  5399 ; TEST ALGORITHM:
0019  5400 ;
0019  5401 ;       1.  CREATE A TEST LOOP OF 6
0019  5402 ;       2.  GENERATE A TEST PATTERN
0019  5403 ;       3.  LOAD THE TEST PATTERN INTO THE RDM WDREG
0019  5404 ;       4.  EXECUTE THE DCS PROGRAM
0019  5405 ;               a.  TURN THE CMI ON
0019  5406 ;               b.  TURN THE CACHE OFF
0019  5407 ;               c.  ZERO THE VA REGISTER
0019  5408 ;               d.  WRITE THE TEST PATTERN TO MAIN MEMORY
0019  5409 ;               e.  LATCH THE CMI DATA IN THE RDM MHREG
0019  5410 ;       5.  TEST THE MHREG FOR THE CORRECT TEST PATTERN
0019  5411 ;       6.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST PATTERNS
0019  5412 ;
0019  5413 ;
0019  5414 ; TEST PATTERNS:
0019  5415 ;
0019  5416 ;       ITERATION       TEST PATTERNS
0019  5417 ;           1             AAAAAAAA
0019  5418 ;           2             55555555
0019  5419 ;           3             33333333
0019  5420 ;           4             0F0F0F0F
0019  5421 ;           5             00FF00FF
0019  5422 ;           6             0000FFFF
0019  5423 ;
0019  5424 ;
0019  5425 ; LOGIC DESCRIPTION:
0019  5426 ;
0019  5427 ;       THE CMI DRIVERS LOCATED IN THE MDR GATE ARRAY ARE TESTED BY DRIVING
0019  5428 ;       TEST PATTERNS THROUGH THEM WHILE ALLOWING THE RDM TO READ THE CMI
0019  5429 ;       VIA THE MHREG.
0019  5430 ;
0019  5431 ;
0019  5432 ; ASSUMPTIONS:
0019  5433 ;
0019  5434 ;       THIS TEST ASSUMES THAT PREVIOUS MIC TESTS HAVE RUN SUCCESSFULLY
0019  5435 ;
0019  5436 ;
0019  5437 ; ERROR DESCRIPTION:
0019  5438 ;
0019  5439 ;       EXPECTED CMI DATA
0019  5440 ;       RECEIVED CMI DATA
0019  5441 ;       LOOP COUNT I
0019  5442 ;
0019  5443 ;--
```

```
                                0019  5444 ;*****************************************************************************
                                0019  5445
                                0019  5446
                                0019  5447         INITIALIZE                                  ;INIT THE CPU
                                001B  5448         LOOP            I,1,6                       ;CREATE A TEST LOOP OF 6
                                0022  5449         SA01PAT         I,1$,,                      ;GENERATE A TEST PATTERN
                                002A  5450         LOADREG         WDREG,1$,,L                 ;TEST PATTERN TO WDREG
                                0032  5451         ERRLOOP                                     ;ERROR LOOP POINT
                                0034  5452         FETCH           0                           ;START DCS PROGRAM
                                0039  5453         BRSTCLK         8                           ;END DCS PROGRAM
                                003D  5454         CMPREG          MHREG,1$,,L,                ;TEST FOR CORRECT PATTERN
                                0046  5455         IFERROR         ,<<MDR>>,                   ;WRONG PATTERN
                                004C  5456         ENDLOOP         I                           ;END TEST LOOP
                                004F  5457         SKIP                                        ;GO TO NEXT TEST
                                0056  5458
                                0056  5459
                                0056  5460 BEGIN_TEST_DATA
                                0056
                                0056          ;-------------------------------------------------------------------------
                                0056  5461
                    00000000    0056  5462 1$:      .LONG   ^X00000000                          ;TEST PATTERN STORAGE
                                005A  5463
                                005A  5464
                                005A  5465 BEGIN_CPU_DATA
                                0002  5466
                                0002  5467 DCS_DATA
                                0001  5468
U 00, 7700,C800,0364,0300,0470,1801   0001  5469 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C  5473 ;% 01    MEMSCAR_R[TEMP0]                            ;TURN CMI ON
U 02, 7700,8800,5BE4,0304,6070,1803   0017  5477 ;% 02    MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,6870,1804   0022  5481 ;% 03    MEMSCAR_R[TEMP2]                            ;TURN CACHE OFF
U 04, 7700,C800,5BE4,030C,6070,1805   002D  5485 ;% 04    MEMSCR_R[TEMP3]
U 05, 7700,4800,5BE4,03D8,4A70,1806   0038  5489 ;% 05    VA_R[ZERO]                                  ;LOAD VA WITH ZEROS
U 06, 5701,8800,0364,0200,5590,1807   0043  5493 ;% 06    WRITE.LONG,SIZE[LONG],WB_RDM                ;WRITE PATTERN TO ADDRESS 0
U 07, 3700,C800,0364,0300,0470,1808   004E  5497 ;% 07    NOP,RDM_CMI                                 ;LATCH CMI IN RDM MHREG
U 08, 7300,4800,0364,0300,0470,1809   0059  5501 ;% 08    NOP,STOP.CLOCK                              ;STOP THE CPU CLOCK
                                0064  5505
                                0064  5506 RTEMP_DATA
                                0001  5507
                    0E000000    0001  5508         .LONG   ^X0E000000                          ;REFERENCE CACHE ONLY ADDRESS
                    00000000    0005  5509         .LONG   ^X00000000                          ;CMI ON
                    06000000    0009  5510         .LONG   ^X06000000                          ;CACHE CONTROL REGISTER ADDRESS
                    01000000    000D  5511         .LONG   ^X01000000                          ;TURN CACHE OFF
                                0011  5512
                                0011  5513 END_CPU_DATA
                                005A  5514
                                005A  5515
                                005A  5516 END_TEST_DATA
                                005A
                                005A          ;-------------------------------------------------------------------------
                                005A  5517 ENDTEST
```

```
0026        .SBTTL  "TEST   077     TEST WRITE VECTOR FROM RDM TO CPU
0026  5519 ;*******************************************************************************
0026  5520 ;++
0026  5521 ;
0026  5522 ; FUNCTIONAL DESCRIPTION:
0026  5523 ;
0026  5524 ;       THIS TEST WILL CHECK THE CMI RECEIVERS ON MIC BY FEEDING DATA THROUGH
0026  5525 ;       THEM VIA A WRITE VECTOR OPERATION.
0026  5526 ;
0026  5527 ;
0026  5528 ; TEST ALGORITHM:
0026  5529 ;
0026  5530 ;       1.  LOAD MICRO-CODE TO TURN ON CMI
0026  5531 ;       2.  EXECUTE DCS PROGRAM
0026  5532 ;               a.  ENABLE CMI
0026  5533 ;       3.  CREATE BUS/GRANT MICRO ORDER
0026  5534 ;       4.  LOAD TEST MICRO-CODE INTO DCS
0026  5535 ;       5.  CREATE A TEST LOOP OF 6
0026  5536 ;       6.  GENERATE A TEST PATTERN
0026  5537 ;       7.  LOAD FIRST MICROINSTRUCTION INTO CPU
0026  5538 ;       8.  START THE CPU CLOCK
0026  5539 ;       9.  CLEAR THE CMICTL REGISTER
0026  5540 ;      10.  LOAD A WRITE VECTOR COMMAND INTO THE MAREG
0026  5541 ;      11.  LOAD THE TEST PATTERN INTO THE MDREG
0026  5542 ;      12.  PERFORM THE WRITE VECTOR OPERATION
0026  5543 ;      13.  STOP THE CPU CLOCK
0026  5544 ;      14.  TEST FOR GOOD STATUS FROM CPU
0026  5545 ;      15.  IF NO ERROR EXECUTE MORE DCS MICROINSTRUCTIONS
0026  5546 ;               a.  MOVE CONTENTS OF MDR REGISTER TO THE RDM WHREG
0026  5547 ;      16.  TEST THE WHREG FOR THE CORRECT TEST PATTERN
0026  5548 ;      17.  IF NO ERROR EXECUTE MORE DCS MICROINSTRUCTIONS
0026  5549 ;               a.  READ CONTENTS OF WRITE VECTOR OCCURRED REGISTER TO RDM
0026  5550 ;      18.  TEST WHREG FOR WRITE VECTOR OCCURRED BIT SET
0026  5551 ;      19.  IF NO ERROR GO TO STEP 6 FOR REMAINING TEST PATTERNS
0026  5552 ;
0026  5553 ;
0026  5554 ; TEST PATTERNS:
0026  5555 ;
0026  5556 ;       ITERATIONS      TEST PATTERN
0026  5557 ;           1             AAAAAAAA
0026  5558 ;           2             55555555
0026  5559 ;           3             33333333
0026  5560 ;           4             0F0F0F0F
0026  5561 ;           5             00FF00FF
0026  5562 ;           6             0000FFFF
0026  5563 ;
0026  5564 ;
0026  5565 ; LOGIC DESCRIPTION:
0026  5566 ;
0026  5567 ;       THIS TEST WILL VERIFY THAT THE CMI RECEIVERS ON THE MIC MODULE ARE
0026  5568 ;       OPERATIONAL.  THIS IS ACCOMPLISHED BY PERFORMING A WRITE VECTOR
0026  5569 ;       OPERATION FROM THE RDM MODULE TO THE CPU.  IN ORDER TO LATCH
0026  5570 ;       DATA IN THE MDR REGISTER IS WAS NECESSARY TO HANG THE CPU
0026  5571 ;       WITH A BUS/GRANT MICRO ORDER.  NORMALLY THIS IS AN ILLEGAL MICRO -
0026  5572 ;       INSTRUCTION WITHOUT A WCTRL/GRANT MICRO ORDER ACCOMPANYING IT.  THAT'S
```

```
0026  5573 ;       THE REASON I FORMED THE MICRO WORD WITH A LDFIELD PSEUDO OP.
0026  5574 ;
0026  5575 ;
0026  5576 ; ASSUMPTIONS:
0026  5577 ;
0026  5578 ;       THIS TEST ASSUMES THE CMI BUS HAS PASSED ALL PREVIOUS TESTS
0026  5579 ;
0026  5580 ;
0026  5581 ; ERROR DESCRIPTION:
0026  5582 ;
0026  5583 ;       FIRST IFERROR:
0026  5584 ;               EXPECTED STATUS FROM THE CPU
0026  5585 ;               RECEIVED STATUS
0026  5586 ;               LOOP COUNT I
0026  5587 ;
0026  5588 ;       SECOND IFERROR:
0026  5589 ;               EXPECTED CONTENTS OF MDR REGISTER
0026  5590 ;               RECEIVED MDR REGISTER
0026  5591 ;               LOOP COUNT I
0026  5592 ;
0026  5593 ;       THIRD IFERROR:
0026  5594 ;               EXPECTED CONTENTS OF WRITE VECTOR OCCURRED REGISTER
0026  5595 ;               RECEIVED CONTENTS OF WRITE VECTOR OCCURRED REGISTER
0026  5596 ;               LOOP COUNT I
0026  5597 ;
0026  5598 ;-
0026  5599 ;*******************************************************************************
0026  5600
0026  5601
0026  5602
0026  5603          INITIALIZE                                      ;INIT THE CPU
0028  5604          LOADDCS         20$,,0,4                        ;LOAD CODE TO TURN CMI ON
002F  5605          FETCH           0                               ;START DCS PROGRAM
0034  5606          BRSTCLK         3                               ;END DCS PROGRAM
0038  5607          LDFIELD         10$,,B,30$,20,24,               ;SETUP BUS/GRANT
0049  5608          LOADDCS         30$,,0,9                        ;LOAD CODE FOR TEST
0050  5609          LOOP            I,1,6                           ;CREATE A TEST LOOP OF 6
0057  5610          SA01PAT         I,1$,,                          ;GENERATE A TEST PATTERN
005F  5611          ERRLOOP                                         ;ERROR LOOP POINT
0061  5612          FETCH           0                               ;START DCS PROGRAM
0066  5613          LOADREG         DCSCR,2$,,B                     ;START THE CPU CLOCK
006E  5614          LOADREG         CMICTL,3$,,B                    ;CLEAR THE CMICTL REGISTER
0076  5615          LOADREG         MAREG,4$,,L                     ;LOAD WRITE VECTOR COMMAND
007E  5616          LOADREG         MDREG,1$,,L                     ;LOAD TEST PATTERN
0086  5617          LOADREG         CMICTL,5$,,B                    ;START CMI OPERATION
008E  5618          LOADREG         DCSCR,6$,,B                     ;STOP THE CPU CLOCK
0096  5619          CMPREGMSK       STATUS,7$,,7$,,L,               ;TEST FOR GOOD STATUS
00A2  5620          IFERROR         ,<<CML>>,                       ;INCORRECT STATUS
00A8  5621          FETCH           2                               ;START DCS PROGRAM
00AD  5622          BRSTCLK         4,INH                           ;END DCS PROGRAM
00B1  5623          CMPREG          WHREG,1$,,L,                    ;TEST FOR CORRECT PATTERN
00BA  5624          IFERROR         ,<<MDR><CML>>,                  ;INCORRECT DATA
00C1  5625          FETCH           5                               ;START DCS PROGRAM
00C6  5626          BRSTCLK         8,INH                           ;END DCS PROGRAM
00CA  5627          CMPREGMSK       WHREG,8$,,9$,,L,                ;TEST FOR WRITE VECTOR OCCURED
```

```
                                00D6  5628          IFERROR         ,<<ADK><CML>>,          ;REGISTER NOT SET
                                00DD  5629          ENDLOOP         I                       ;END TEST LOOP
                                00E0  5630          SKIP                                    ;GO TO NEXT TEST
                                00E7  5631
                                00E7  5632
                                00E7  5633
                                00E7  5634 BEGIN_TEST_DATA
                                00E7
                                00E7          ;-------------------------------------------------------------------------
                                00E7  5635
                    00000000    00E7  5636 1$:      .LONG   ^X00000000                      ;TEST PATTERN STORAGE
                          00    00EB  5637 2$:      .BYTE   ^X00                            ;CLEAR REGISTERS
                          01    00EC  5638 3$:      .BYTE   ^X01                            ;CLEAR CMICTL REGISTER
                    FC000000    00ED  5639 4$:      .LONG   ^XFC000000                      ;WRITE VECTOR COMMAND
                          28    00F1  5640 5$:      .BYTE   ^X28                            ;START CMI OPERATION
                          60    00F2  5641 6$:      .BYTE   ^X60                            ;STOP THE CPU CLOCK
                          38    00F3  5642 7$:      .BYTE   ^X38                            ;GOOD STATUS
                    01000000    00F4  5643 8$:      .LONG   ^X01000000                      ;WRITE VECTOR OCCURED
                    0F000000    00F8  5644 9$:      .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                          0F    00FC  5645 10$:     .BYTE   ^X0F                            ;CODE FOR BUS/GRANT
                                00FD  5646 20$:
U 00, 7700,C800,0364,0300,0470,1801 00FD  5647 ;* 00    NOP
U 01, 7700,8800,5BE4,0304,6870,1802 0108  5651 ;* 01    MEMSCAR_R[TEMP1]                        ;TURN CMI ON
U 02, 7700,C800,5BE4,03D8,6070,1803 0113  5655 ;* 02    MEMSCR_R[ZERO]
U 03, 7300,C800,0364,0300,0470,1804 011E  5659 ;* 03    NOP,STOP.CLOCK
                                0129  5663 30$:
U 00, 7700,C800,0364,0300,0470,1801 0129  5664 ;* 00    NOP                                     ;WILL BECOME BUS/GRANT
U 01, 7300,C800,0364,0300,0470,1802 0134  5668 ;* 01    NOP,STOP.CLOCK
U 02, 7700,4800,0364,0300,0470,1803 013F  5672 ;* 02    NOP
U 03, 6700,8812,5924,0300,0470,1804 014A  5676 ;* 03    RDM_WB,WB_M[MDR]                        ;READ TEST PATTERN TO RDM
U 04, 7300,4800,0364,0300,0470,1805 0155  5680 ;* 04    NOP,STOP.CLOCK                          ;STOP CPU CLOCK
U 05, 7700,4800,0364,0300,0470,1806 0160  5684 ;* 05    NOP
U 06, 7700,C800,5BE4,0300,6870,1807 016B  5688 ;* 06    MEMSCAR_R[TEMP0]                        ;WRITE VECTOR OCCURED ADDRESS
U 07, 6700,C800,0364,0300,6470,1808 0176  5692 ;* 07    RDM_WB,WCTRL/MEMSCR                     ;READ REGISTER TO RDM
U 08, 7300,4800,0364,0300,0470,1809 0181  5696 ;* 08    NOP,STOP.CLOCK                          ;STOP CPU CLOCK
                                018C  5700
                                018C  5701 BEGIN_CPU_DATA
                                0002  5702
                                0002  5703 RTEMP_DATA
                                0001  5704
                    02000000    0001  5705          .LONG   ^X02000000                      ;WRITE VECTOR OCCURED ADDRESS
                    0E000000    0005  5706          .LONG   ^X0E000000                      ;REFERENCE CACHE ONLY ADDRESS
                                0009  5707
                                0009  5708 END_CPU_DATA
                                018C  5709
                                018C  5710
                                018C  5711 END_TEST_DATA
                                018C
                                018C          ;-------------------------------------------------------------------------
                                018C  5712 ENDTEST
```

```
0027          .SBTTL  "TEST   078     TEST RDM TO MAIN MEMORY WRITE/READ
0027  5714 ;*******************************************************************************
0027  5715 ;++
0027  5716 ;
0027  5717 ; FUNCTIONAL DESCRIPTION:
0027  5718 ;
0027  5719 ;       THIS TEST WILL CHECK THE ABILITY OF MAIN MEMORY TO BE READ AND WRITTEN
0027  5720 ;       FROM THE RDM.
0027  5721 ;
0027  5722 ;
0027  5723 ; TEST ALGORITHM:
0027  5724 ;
0027  5725 ;       1.  LOAD THE FIRST DCS MICROINSTRUCTION
0027  5726 ;       2.  LET THE CPU CLOCK FREE RUN
0027  5727 ;       3.  LOAD THE MAREG WITH A WRITE COMMAND TO ADDRESS ZERO
0027  5728 ;       4.  LOAD ZEROS INTO THE MDREG
0027  5729 ;       5.  PERFORM THE WRITE OPERATION
0027  5730 ;       6.  LOAD THE MAREG WITH A READ COMMAND TO ADDRESS ZERO
0027  5731 ;       7.  PERFORM THE READ OPERATION
0027  5732 ;       8.  STOP THE CPU CLOCK
0027  5733 ;       9.  TEST THE STATUS REGISTER FOR GOOD STATUS
0027  5734 ;      10.  IF NO ERROR TEST THE MHREG FOR THE CORRECT DATA
0027  5735 ;
0027  5736 ;
0027  5737 ; TEST PATTERNS:
0027  5738 ;
0027  5739 ;       WRITE DATA = 00000000
0027  5740 ;
0027  5741 ;
0027  5742 ; LOGIC DESCRIPTION:
0027  5743 ;
0027  5744 ;       THIS IS THE FIRST TEST THAT GOES TO MAIN MEMORY FOR A READ.  ANY
0027  5745 ;       FAILURES SHOULD BE CONFINED TO THE MEMORY SUBSYSTEM.
0027  5746 ;
0027  5747 ;
0027  5748 ; ASSUMPTIONS:
0027  5749 ;
0027  5750 ;       THIS TEST ASSUMES THAT THE PREVIOUS CMI TESTS HAVE RUN SUCCESSFULLY
0027  5751 ;
0027  5752 ;
0027  5753 ; ERROR DESCRIPTION:
0027  5754 ;
0027  5755 ;       FIRST IFERROR
0027  5756 ;               EXPECTED STATUS FROM MAIN MEMORY
0027  5757 ;               RECEIVED STATUS FROM MAIN MEMORY
0027  5758 ;
0027  5759 ;       SECOND IFERROR
0027  5760 ;               EXPECTED DATA FROM MAIN MEMORY
0027  5761 ;               RECEIVED DATA FROM MAIN MEMORY
0027  5762 ;
0027  5763 ;--
0027  5764 ;*******************************************************************************
0027  5765
0027  5766
0027  5767         INITIALIZE                                      ;INIT THE CPU
```

```
                        0029  5768          FETCH           0                               ;START DCS PROGRAM
                        002E  5769          LOADREG         DCSCR,2$,,B                     ;START THE CPU CLOCK
                        0036  5770          LOADREG         CMICTL,3$,,B                    ;CLEAR THE CMICTL REGISTER
                        003E  5771          LOADREG         MAREG,4$,,L                     ;LOAD WRITE ADDRESS AND COMMAND
                        0046  5772          LOADREG         MDREG,5$,,L                     ;LOAD TEST PATTERN INTO MDREG
                        004E  5773          LOADREG         CMICTL,6$,,B                    ;PERFORM CMI WRITE OPERATION
                        0056  5774          LOADREG         MAREG,7$,,L                     ;LOAD READ ADDRESS AND COMMAND
                        005E  5775          LOADREG         CMICTL,8$,,B                    ;PERFORM READ OPERATION
                        0066  5776          LOADREG         DCSCR,9$,,B                     ;STOP THE CPU CLOCK
                        006E  5777          CMPREGMSK       STATUS,10$,,10$,,B,             ;TEST FOR GOOD STATUS
                        007A  5778          IFERROR         ,,<<MC0><MC1><MC2>>             ;INCORRECT STATUS
                        0082  5779          CMPREG          MHREG,5$,,L                     ;TEST FOR TEST PATTERN
                        008B  5780          IFERROR         ,<<MDL>>,<<MC0><MC1><MC2><MA0><MA1><MA2>>;WRONG DATA
                        0097  5781          SKIP                                            ;GO TO NEXT TEST
                        009E  5782
                        009E  5783
                        009E  5784 BEGIN_TEST_DATA
                        009E
                        009E          ;-----------------------------------------------------------------------
                        009E  5785
                     00 009E  5786 2$:      .BYTE   ^X00                                    ;START THE CPU CLOCK
                     01 009F  5787 3$:      .BYTE   ^X01                                    ;CLEAR THE CMICTL REGISTER
              F8000000  00A0  5788 4$:      .LONG   ^XF8000000                              ;WRITE COMMAND TO ADDRESS ZERO
              00000000  00A4  5789 5$:      .LONG   ^X00000000                              ;TEST PATTERN
                     28 00A8  5790 6$:      .BYTE   ^X28                                    ;CMI WRITE OPERATION
              00000000  00A9  5791 7$:      .LONG   ^X00000000                              ;READ COMMAND TO ADDRESS ZERO
                     48 00AD  5792 8$:      .BYTE   ^X48                                    ;CMI READ OPERATION
                     60 00AE  5793 9$:      .BYTE   ^X60                                    ;STOP CPU CLOCK
                     30 00AF  5794 10$:     .BYTE   ^X30                                    ;GOOD STATUS
                        00B0  5795
                        00B0  5796
                        00B0  5797
                        00B0  5798 BEGIN_CPU_DATA
                        0002  5799
                        0002  5800 DCS_DATA
                        0001  5801
U 00, 7700,C800,0364,0300,0470,1801 0001  5802 ;% 00    NOP
U 01, 7500,C800,0364,0300,0470,1802 000C  5806 ;% 01    NOP,DCS.ADDR_0                      ;RUN CPU CLOCK
                        0017  5810
                        0017  5811 END_CPU_DATA
                        00B0  5812
                        00B0  5813
                        00B0  5814 END_TEST_DATA
                        00B0
                        00B0          ;-----------------------------------------------------------------------
                        00B0  5815 ENDTEST
```

```
                     0027        .SBTTL  "TEST   079     TEST CPU TO MAIN MEMORY WRITE/READ
                     0027  5817 ;*******************************************************************************
                     0027  5818 ;++
                     0027  5819 ;
                     0027  5820 ; FUNCTIONAL DESCRIPTION:
                     0027  5821 ;
                     0027  5822 ;       TEST THE ABILITY OF CPU TO WRITE/READ LOCATION ZERO OF MAIN MEMORY.
                     0027  5823 ;
                     0027  5824 ;
                     0027  5825 ; TEST ALGORITHM:
                     0027  5826 ;
                     0027  5827 ;       1.  SET UP A TEST LOOP OF 6 TO SELECT TEST PATTERNS
                     0027  5828 ;       2.  GENERATE A TEST PATTERN
                     0027  5829 ;       3.  LOAD TEST PATTERN INTO RDM WD REGISTER
                     0027  5830 ;       4.  EXECUTE DCS PROGRAM
                     0027  5831 ;               a.  TURN CMI ON
                     0027  5832 ;               b.  DISABLE CACHE
                     0027  5833 ;               c.  LOAD VA REGISTER WITH ZERO
                     0027  5834 ;               d.  WRITE TEST PATTERN TO ADDRESS ZERO
                     0027  5835 ;               e.  READ PATTERN BACK TO CPU
                     0027  5836 ;               f.  MOVE PATTERN TO RDM WH REGISTER
                     0027  5837 ;       5.  TEST WH REGISTER FOR CORRECT PATTERN
                     0027  5838 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                     0027  5839 ;
                     0027  5840 ;
                     0027  5841 ; TEST PATTERNS:
                     0027  5842 ;
                     0027  5843 ;       ITERATION       TEST PATTERN
                     0027  5844 ;           1             AAAAAAAA
                     0027  5845 ;           2             55555555
                     0027  5846 ;           3             33333333
                     0027  5847 ;           4             0F0F0F0F
                     0027  5848 ;           5             00FF00FF
                     0027  5849 ;           6             0000FFFF
                     0027  5850 ;
                     0027  5851 ;
                     0027  5852 ; LOGIC DESCRIPTION:
                     0027  5853 ;
                     0027  5854 ;       THIS TEST VERIFIES THAT CPU CAN WRITE AND READ MAIN MEMORY.
                     0027  5855 ;       PREVIOUS TESTS HAVE CONFIRMED THE INTEGRITY OF THE CMI.  ANY FAILURES
                     0027  5856 ;       IN THIS TEST SHOULD BE CONFINED TO THE CML GATE ARRAY OR THE MEMORY
                     0027  5857 ;       CONTROLLER.
                     0027  5858 ;
                     0027  5859 ;
                     0027  5860 ; ASSUMPTIONS:
                     0027  5861 ;
                     0027  5862 ;       THIS TEST ASSUMES THE CMI IS NOT HUNG BY ANYOTHER DEVICE.
                     0027  5863 ;
                     0027  5864 ;
                     0027  5865 ; ERROR DESCRIPTION:
                     0027  5866 ;
                     0027  5867 ;       EXPECTED DATA FROM MAIN MEMORY
                     0027  5868 ;       RECEIVED DATA FROM MAIN MEMORY
                     0027  5869 ;       LOOP COUNT I
                     0027  5870 ;
```

```
                     0027  5871 ;--
                     0027  5872 ;*****************************************************************************
                     0027  5873
                     0027  5874
                     0027  5875          INITIALIZE                                  ;INIT THE CPU
                     0029  5876          FETCH           0                           ;start dcs
                     002E  5877          BRSTCLK         4                           ;end dcs
                     0032  5878          CMPREGMSK       MHREG,2$,,3$,,L             ;check controller type
                     003E  5879          SKIP            5$,ONERROR                  ;skip if L0022
                     0045  5880          EXPUTRAP                                    ;POSSIBLE FAILURE MODE
                     0047  5881          LOADDCS         10$,,0,<^X0A>               ;test ucode
                     004E  5882          LOOP            I,1,6                       ;TEST LOOP OF 6
                     0055  5883          SA01PAT         I,1$,,                      ;GENERATE A TEST PATTERN
                     005D  5884          LOADREG         WDREG,1$,,L                 ;TEST PATTERN TO WD REGISTER
                     0065  5885          ERRLOOP                                     ;ERROR LOOP POINT
                     0067  5886          FETCH           0                           ;START DCS PROGRAM
                     006C  5887          BRSTCLK         9                           ;END DCS PROGRAM
                     0070  5888          CMPREG          WHREG,1$,,L,                ;TEST FOR CORRECT PATTERN
                     0079  5889          IFERROR         ,<<CML><MDL><MEC>>,<<MC0><MC1><MA0><MA1>>
                     0085  5890                                                      ;INCORRECT DATA
                     0085  5891          ENDLOOP         I                           ;END TEST LOOP
                     0088  5892          SKIP                                        ;GO TO NEXT TEST
                     008F  5893 5$:      EXPUTRAP                                    ;POSSIBLE FAILURE MODE
                     0091  5894          LOADDCS         10$,,0,<^X0A>               ;test ucode
                     0098  5895          LOOP            I,1,6                       ;TEST LOOP OF 6
                     009F  5896          SA01PAT         I,1$,,                      ;GENERATE A TEST PATTERN
                     00A7  5897          LOADREG         WDREG,1$,,L                 ;TEST PATTERN TO WD REGISTER
                     00AF  5898          ERRLOOP                                     ;ERROR LOOP POINT
                     00B1  5899          FETCH           0                           ;START DCS PROGRAM
                     00B6  5900          BRSTCLK         9                           ;END DCS PROGRAM
                     00BA  5901          CMPREG          WHREG,1$,,L,                ;TEST FOR CORRECT PATTERN
                     00C3  5902          IFERROR         ,<<CML><MDL><MEC>>,<<MC2><MA1><MA2>>
                     00CE  5903                                                      ;INCORRECT DATA
                     00CE  5904          ENDLOOP         I                           ;END TEST LOOP
                     00D1  5905          SKIP                                        ;GO TO NEXT TEST
                     00D8  5906
                     00D8  5907
                     00D8  5908 BEGIN_TEST_DATA
                     00D8                ;-------------------------------------------------------------------------
                     00D8  5909
                 00000000  00D8  5910 1$:     .LONG   ^X00000000                     ;TEST PATTERN STORAGE
                 00000000  00DC  5911 2$:     .LONG   ^X00000000                     ;dest for cmpregmsk
                 02000000  00E0  5912 3$:     .LONG   ^X02000000                     ;mask for controller type
                           00E4  5913 10$:
U 00, 7700,C800,0364,0300,0470,1801  00E4  5914 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  00EF  5918 ;% 01    MEMSCAR_R[TEMP0]            ;TURN CMI ON
U 02, 7700,8800,5BE4,0304,6070,1803  00FA  5922 ;% 02    MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,6870,1804  0105  5926 ;% 03    MEMSCAR_R[TEMP2]            ;TURN CACHE OFF
U 04, 7700,C800,5BE4,030C,6070,1805  0110  5930 ;% 04    MEMSCR_R[TEMP3]
U 05, 7700,4800,5BE4,03D8,4A70,1806  011B  5934 ;% 05    VA_R[ZERO]                  ;LOAD VA WITH ZEROS
U 06, 5701,8800,0364,0200,5590,1807  0126  5938 ;% 06    WRITE.LONG,SIZE[LONG],WB_RDM ;WRITE MAIN MEMORY
U 07, 7700,4800,0364,0300,0510,1808  0131  5942 ;% 07    READ.LONG                   ;READ MAIN MEMORY
U 08, 6700,0812,5924,0300,0470,1809  013C  5946 ;% 08    RDM_WB,WB_M[MDR]            ;MOVE DATA TO RDM
U 09, 7300,4800,0364,0300,0470,180A  0147  5950 ;% 09    NOP,STOP.CLOCK              ;STOP CPU CLOCK
```

```
                                0152  5954
                                0152  5955 BEGIN_CPU_DATA
                                0002  5956
                                0002  5957 DCS_DATA
                                0001  5958
U 00, 7700,C800,0364,0300,0470,1801  0001  5959 ;% 00   NOP
U 01, 7700,8800,5BE4,0310,4A70,1802  000C  5963 ;% 01   VA_R[TEMP4]                    ;load VA with CSR2 addr.
U 02, 7700,C800,0364,0300,0400,1803  0017  5967 ;% 02   READ.PHY                      ;read CSR2
U 03, 3700,8812,5924,0300,0470,1804  0022  5971 ;% 03   WB_M[MDR],RDM_CMI            ;WHREG gets contents of CSR2
U 04, 7300,4800,0364,0300,0470,1805  002D  5975 ;% 04   NOP,STOP.CLOCK
                                0038  5979
                                0038  5980 RTEMP_DATA
                                0001  5981
                     0E000000   0001  5982         .LONG   ^X0E000000               ;REFERENCE CACHE ONLY ADDRESS
                     00000000   0005  5983         .LONG   ^X00000000               ;CMI ON
                     06000000   0009  5984         .LONG   ^X06000000               ;CACHE CONTROL REGISTER ADDRESS
                     01000000   000D  5985         .LONG   ^X01000000               ;CACHE OFF
                     00F20008   0011  5986         .LONG   ^X00F20008               ;CSR2 address
                                0015  5987
                                0015  5988 END_CPU_DATA
                                0152  5989
                                0152  5990
                                0152  5991 END_TEST_DATA
                                0152
                                0152        ;-----------------------------------------------------------------------
                                0152  5992 ENDTEST
```

```
                        .SBTTL  "TEST   07A     TEST WRITE BUS ERROR INTERRUPT
0023  5994 ;*******************************************************************************
0023  5995 ;++
0023  5996 ;
0023  5997 ; FUNCTIONAL DESCRIPTION:
0023  5998 ;
0023  5999 ;       THIS TEST CHECKS THAT THE CPU CAN DETECT AND REPORT A WRITE
0023  6000 ;       BUS ERROR ON THE CMI.
0023  6001 ;
0023  6002 ;
0023  6003 ; TEST ALGORITHM:
0023  6004 ;
0023  6005 ;       1.  EXECUTE DCS PROGRAM
0023  6006 ;               a.  ENABLE THE CACHE MEMORY
0023  6007 ;               b.  DISABLE THE CMI
0023  6008 ;               c.  LOAD PC WITH ZERO TO CAUSE A PREFETCH
0023  6009 ;               d.  DISABLE ^P INTERRUPTS
0023  6010 ;               e.  DISABLE CACHE MEMORY
0023  6011 ;               f.  ENABLE THE CMI
0023  6012 ;               g.  LOAD THE VA WITH A NON-EXISTANT ADDRESS
0023  6013 ;               h.  WRITE TO THIS ADDRESS
0023  6014 ;               i.  PERFORM AN INSTRUCTION DECODF TO ALLOW INTERRUPT TO HAPPEN
0023  6015 ;       2.  TEST THE VBUS FOR WRITE ERROR INTERRUPT
0023  6016 ;       3.  IF NO ERROR TEST THE CS ADDRESS FOR THE CORRECT VECTOR
0023  6017 ;       4.  IF NO ERROR EXECUTE MORE DCS CODE
0023  6018 ;               a.  READ THE BUS ERROR REGISTER TO THE RDM
0023  6019 ;       5.  TEST THE BUS ERROR REGISTER FOR NON-EXISTANT MEMORY
0023  6020 ;       6.  IF NO ERROR EXECUTE MORE DCS CODE
0023  6021 ;               a.  READ THE MACHINE CHECK REGISTER TO THE RDM
0023  6022 ;               b.  CLEAR THE MACHINE CHECK REGISTER
0023  6023 ;       7.  TEST THE REGISTER FOR BUS ERROR
0023  6024 ;       8.  IF NO ERROR GO TO NEXT TEST
0023  6025 ;
0023  6026 ;
0023  6027 ; TEST PATTERNS:
0023  6028 ;
0023  6029 ;       NONE
0023  6030 ;
0023  6031 ;
0023  6032 ; LOGIC DESCRIPTION:
0023  6033 ;
0023  6034 ;       THIS TEST USES THE MIC, UBI, AND DPM MODULES TO DETERMINE IF A WRITE
0023  6035 ;       BUS ERROR IS HANDLED CORRECTLY.  THE MIC MODULE (CML, UTR) DETECTS THAT
0023  6036 ;       A WRITE ERROR HAS OCCURRED AND REPORTS SUCH TO THE INT GATE ARRAY ON
0023  6037 ;       UBI.  THE INT INTURN GENERATES INTERRUPT PENDING WHICH IT SENDS TO THE
0023  6038 ;       SAC ARRAY ON DPM.  SAC FORCES THE INTERRUPT AT IRD TIME.
0023  6039 ;
0023  6040 ;
0023  6041 ; ERROR DESCRIPTION:
0023  6042 ;
0023  6043 ;       FIRST IFERROR:
0023  6044 ;               EXPECTED VBUS DATA
0023  6045 ;               RECEIVED VBUS DATA
0023  6046 ;                       (BITS <7:0> = VBUS BIT POSITION)
0023  6047 ;                       (BITS <8> = BIT STATE)
```

```
                        0023  6048 ;
                        0023  6049 ;       SECOND IFERROR:
                        0023  6050 ;               EXPECTED CS ADDRESS
                        0023  6051 ;               RECEIVED CS ADDRESS
                        0023  6052 ;
                        0023  6053 ;       THIRD IFERROR:
                        0023  6054 ;               EXPECTED BUS ERROR REGISTER CONTENTS
                        0023  6055 ;               RECEIVED BUS ERROR REGISTER CONTENTS
                        0023  6056 ;
                        0023  6057 ;       FORTH IFERROR:
                        0023  6058 ;               EXPECTED MACHINE CHECK REGISTER CONTENTS
                        0023  6059 ;               RECEIVED MACHINE CHECK REGISTER CONTENTS
                        0023  6060 ;
                        0023  6061 ;--
                        0023  6062 ;*******************************************************************************
                        0023  6063
                        0023  6064
                        0023  6065
                        0023  6066         INITIALIZE                              ;INIT THE CPU
                        0025  6067         EXPUTRAP                                ;EXPECT A WRITE MICRO TRAP
                        0027  6068         ERRLOOP                                 ;ERROR LOOP POINT
                        0029  6069         FETCH           0                       ;START DCS PROGRAM
                        002E  6070         BRSTCLK         <^X12>                  ;END DCS PROGRAM
                        0032  6071         CMPVBUS         1$                      ;TEST FOR WRITE ERROR
                        0037  6072         IFERROR         ,<<CML><UTR>>,          ;NO WRITE BUS ERROR
                        003E  6073         CMPREGMSK       CSTRP,2$,,3$,,W,        ;TEST CS ADDRESS
                        004A  6074         IFERROR         ,<INT>,<<UBI><DPM>>     ;WRONG VECTOR
                        0052  6075         FETCH           <^X13>                  ;START DCS PROGRAM
                        0057  6076         BRSTCLK         <^X16>                  ;END DCS PROGRAM
                        005B  6077         CMPREGMSK       WHREG,4$,,5$,,L,        ;TEST BUS ERROR REGISTER
                        0067  6078         IFERROR         ,<UTR>,                 ;NOT SET CORRECTLY
                        006D  6079         FETCH           <^X17>                  ;START DCS PROGRAM
                        0072  6080         BRSTCLK         <^X1B>                  ;END DCS PROGRAM
                        0076  6081         CMPREGMSK       WHREG,4$,,5$,,L,        ;TEST MACHINE CHECK REGISTER
                        0082  6082         IFERROR         ,<UTR>,                 ;NOT SET CORRECTLY
                        0088  6083         SKIP                                    ;GO TO NEXT TEST
                        008F  6084
                        008F  6085
                        008F  6086 BEGIN_TEST_DATA
                        008F               ;-------------------------------------------------------------------------
                        008F  6087
                     01 008F  6088 1$:      .BYTE   ^X1                             ;TEST VBUS DATA
                        0090  6089          VBUSDATA        <^X15>,0                ;WRITE BUS ERR INT L
                   003E 0091  6090 2$:      .WORD   ^X003E                          ;INTERRUPT VECTOR
                   3FFF 0093  6091 3$:      .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK
               08000000 0095  6092 4$:      .LONG   ^X08000000                      ;BUS/MACHINE CHECK REGISTERS
               0F000000 0099  6093 5$:      .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                        009D  6094
                        009D  6095
                        009D  6096 BEGIN_CPU_DATA
                        0002  6097
                        0002  6098 DCS_DATA
                        0001  6099
U 00, 7700,C800,0364,0300,0470,1801 0001  6100 ;* 00    NOP
```

```
U 01. 7700.8800.5BE4.0308.6870.1802  000C  6104 ;% 01   MEMSCAR_R[TEMP2]                    ;TURN CACHE ON
U 02. 7700.C800.5BE4.03D8.6070.1803  0017  6108 ;% 02   MEMSCR_R[ZERO]                     ;...
U 03. 7700.8800.5BE4.0310.6870.1804  0022  6112 ;% 03   MEMSCAR_R[TEMP4]                   ;TURN CMI OFF
U 04. 7700.C800.5BE4.030C.6070.1805  002D  6116 ;% 04   MEMSCR_R[TEMP3]                    ;...
U 05. 7700.C800.5BE4.03D8.4870.1806  0038  6120 ;% 05   PC_R[ZERO]                        ;CAUSE PREFETCH
U 06. 7700.4800.5BE4.0300.2470.1807  0043  6124 ;% 06   WCTRL/LOADCRAR,WB_R[TEMP0]       ;DISABLE ^P INTERRUPTS
U 07. 7700.8800.5BE4.0304.2C70.1808  004E  6128 ;% 07   WCTRL/CONWRITE,WB_R[TEMP1]       ;...
U 08. 7700.0800.5BE4.0308.6870.1809  0059  6132 ;% 08   MEMSCAR_R[TEMP2]                   ;TURN CACHE OFF
U 09. 7700.C800.5BE4.030C.6070.180A  0064  6136 ;% 09   MEMSCR_R[TEMP3]                    ;...
U 0A. 7700.8800.5BE4.0310.6870.180B  006F  6140 ;% 0A   MEMSCAR_R[TEMP4]                   ;TURN CMI ON
U 0B. 7700.C800.5BE4.03D8.6070.180C  007A  6144 ;% 0B   MEMSCR_R[ZERO]                     ;...
U 0C. 7700.C800.5BE4.0314.4A70.180D  0085  6148 ;% 0C   VA_R[TEMP5]                       ;LOAD NON-EXISTANT ADDRESS
U 0D. 7700.C800.0364.0300.0480.180E  0090  6152 ;% 0D   WRITE.PHY                        ;PERFORM WRITE OPERATION
U 0E. 7700.4800.0364.0300.0470.180F  009B  6156 ;% 0E   NOP                             ;ALLOW INTERRUPT TO SET UP
U 0F. 7600.C800.0364.0300.0470.1810  00A6  6160 ;% 0F   NOP,STROBE.V.BUS                 ;...
U 10. 7700.4800.0364.1700.0470.1811  00B1  6164 ;% 10   IRD1TEST                         ;ALLOW INTERRUPT TO HAPPEN
U 11. 7700.4800.0364.0300.0470.1812  00BC  6168 ;% 11   NOP                             ;STROBE THE VBUS
U 12. 7300.C800.0364.0300.0470.1813  00C7  6172 ;% 12   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 13. 7700.4800.0364.0300.0470.1814  00D2  6176 ;% 13   NOP
U 14. 7700.C800.5BE4.0318.6870.1815  00DD  6180 ;% 14   MEMSCAR_R[TEMP6]                  ;BUS ERROR REGISTER
U 15. 6700.C800.0364.0300.6470.1816  00E8  6184 ;% 15   RDM_WB,WCTRL/MEMSCR              ;MOVE TO RDM WHREGR
U 16. 7300.4800.0364.0300.0470.1817  00F3  6188 ;% 16   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 17. 7700.4800.0364.0300.0470.1818  00FE  6192 ;% 17   NOP
U 18. 7700.8800.5BE4.031C.6870.1819  0109  6196 ;% 18   MEMSCAR_R[TEMP7]                  ;MACHINE CHECK REGISTER
U 19. 6700.C800.0364.0300.6470.181A  0114  6200 ;% 19   RDM_WB,WCTRL/MEMSCR              ;MOVE TO RDM WHREG
U 1A. 7700.C800.5BE4.03D8.6070.181B  011F  6204 ;% 1A   MEMSCR_R[ZERO]                    ;CLEAR THE REGISTER
U 1B. 7300.C800.0364.0300.0470.181C  012A  6208 ;% 1B   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
                                     0135  6212
                                     0135  6213 RTEMP_DATA
                                     0001  6214
                     00C00000        0001  6215         .LONG   ^X00C00000             ;DISABLE ^P INTERRUPTS
                     00400000        0005  6216         .LONG   ^X00400000             ;...
                     06000000        0009  6217         .LONG   ^X06000000             ;CACHE DISABLE REGISTER
                     01000000        000D  6218         .LONG   ^X01000000             ;DATA
                     0E000000        0011  6219         .LONG   ^X0E000000             ;REFERENCE CACHE ONLY REGISTER
                     00F10000        0015  6220         .LONG   ^X00F10000             ;NON-EXISTANT MEMORY ADDRESS
                     09000000        0019  6221         .LONG   ^X09000000             ;BUS ERROR REGISTER ADDRESS
                     08000000        001D  6222         .LONG   ^X08000000             ;MACHINE CHECK REG ADDRESS
                                     0021  6223
                                     0021  6224 CACHE_DATA
                                     0001  6225
                     00000000        0001  6226         .LONG   ^X00000000             ;VALIDATE THE CACHE
                     00000000        0005  6227         .LONG   ^X00000000             ;...
                                     0009  6228
                                     0009  6229 END_CPU_DATA
                                     009D  6230
                                     009D  6231
                                     009D  6232 END_TEST_DATA
                                     009D
                                     009D       ;-----------------------------------------------------------------------------
                                     009D  6233 ENDTEST
```

```
0023          .SBTTL  "TEST   07B     TEST MAIN MEMORY CSR0 AND CSR1
0023  6235  ;*****************************************************************************
0023  6236  ;++
0023  6237  ;
0023  6238  ; FUNCTIONAL DESCRIPTION:
0023  6239  ;
0023  6240  ;       THIS TEST WILL VERIFY THAT CPU CAN WRITE AND READ THE CONTROL AND
0023  6241  ;       STATUS REGISTERS IN MAIN MEMORY.
0023  6242  ;
0023  6243  ;
0023  6244  ; TEST ALGORITHM:
0023  6245  ;
0023  6246  ;       1.  MODIFY DCS ACCORDING TO SUBTEST
0023  6247  ;       2.  CREATE A TEST LOOP OF 6
0023  6248  ;       3.  GENERATE A TEST PATTERN
0023  6249  ;       4.  LOAD TEST PATTERN INTO THE RDM WDREG
0023  6250  ;       5.  EXECUTE DCS PROGRAM
0023  6251  ;               a.  ENABLE THE CMI
0023  6252  ;               b.  DISABLE CACHE
0023  6253  ;               c.  LOAD VA REGISTER WITH ADDRESS OF CSR UNDER TEST
0023  6254  ;               d.  WRITE TEST PATTERN TO CSR
0023  6255  ;               e.  READ TEST PATTERN BACK TO RDM
0023  6256  ;       6.  TEST CSTRP REGISTER TO BE SURE WE DIDN'T MICRO TRAP
0023  6257  ;       7.  IF NO ERROR TEST FOR CORRECT TEST PATTERN
0023  6258  ;       8.  IF NO ERROR GO STEP 3 FOR THE REMAINING TEST PATTERNS
0023  6259  ;
0023  6260  ;
0023  6261  ; TEST PATTERNS:
0023  6262  ;
0023  6263  ;       ITERATION       TEST PATTERN
0023  6264  ;           1             AAAAAAAA
0023  6265  ;           2             55555555
0023  6266  ;           3             33333333
0023  6267  ;           4             0F0F0F0F
0023  6268  ;           5             00FF00FF
0023  6269  ;           6             0000FFFF
0023  6270  ;
0023  6271  ;
0023  6272  ; LOGIC DESCRIPTION:
0023  6273  ;
0023  6274  ;       THIS TEST VERIFIES THAT CPU CAN WRITE AND READ THE CONTROL AND STATUS
0023  6275  ;       REGISTERS IN MAIN MEMORY.  IF THE TEST MICRO TRAPS IT IS PROBABLE
0023  6276  ;       BECAUSE THERE WAS NO RESPONSE FROM MEMORY.  SWAPPING THE MEMORY
0023  6277  ;       CONTROLLER AND/OR ASSOCIATED GATE ARRAYS SHOULD FIX ANY FAILURES.
0023  6278  ;
0023  6279  ;
0023  6280  ; ASSUMPTIONS:
0023  6281  ;
0023  6282  ;       THIS TEST ASSUMES THE CMI IS OPERATIONAL.
0023  6283  ;
0023  6284  ;
0023  6285  ; ERROR DESCRIPTION:
0023  6286  ;
0023  6287  ;       FIRST IFERROR:
0023  6288  ;               EXPECTED CS ADDRESS
```

```
                        ;               RECEIVED CS ADDRESS
                        ;               LOOP COUNT I
                        ;
                        ;       SECOND IFERROR:
                        ;               EXPECTED CONTENTS OF THE CSR UNDER TEST
                        ;               RECEIVED CONTENTS OF THE CSR UNDER TEST
                        ;               LOOP COUNT I
                        ;
                        ;--
                        ;*******************************************************************************
                        ;///////////////////////////////////////////////////////////////////////////////
                        ;+
                        ; SUBTEST #1
                        ;
                        ;       TEST CSR0
                        ;-
                        SUBTEST                                         ;SUBTEST #1

                        ;///////////////////////////////////////////////////////////////////////////////
                        INITIALIZE                                      ;INIT THE CPU
                        LOADDCS         4$,,5,1                         ;MODIFY U-CODE FOR SUBTEST #1
                        EXPUTRAP                                        ;POSSIBLE FAILURE MODE
                        LOOP            I,1,6                           ;TEST LOOP OF 6
                        SA01PAT         I,1$,,                          ;GENERATE A TEST PATTERN
                        LOADREG         WDREG,1$,,L                     ;LOAD TEST PATTERN INTO WDREG
                        ERRLOOP                                         ;ERROR LOOP POINT
                        FETCH           0                               ;START DCS PROGRAM
                        BRSTCLK         9                               ;END DCS PROGRAM
                        CMPREGMSK       CSTRP,2$,,3$,,W,                ;TEST FOR A U-TRAP
                        IFERROR         ,,<<MC0><MC1><MC2>>             ;TEST U-TRAPPED
                        CMPREGMSK       WHREG,1$,,6$,,L,                ;TEST FOR CORRECT DATA
                        IFERROR         ,<<MDL>>,<<MC0><MC1><MC2>>;INCORRECT TEST PATTERN
                        ENDLOOP         I                               ;END TEST LOOP

                        ;///////////////////////////////////////////////////////////////////////////////
                        ;+
                        ; SUBTEST #2
                        ;
                        ;       TEST CSR1
                        ;-
                        SUBTEST                                         ;SUBTEST #2

                        ;///////////////////////////////////////////////////////////////////////////////
                        INITIALIZE                                      ;INIT THE CPU
                        LOADDCS         5$,,5,1                         ;MODIFY U-CODE FOR SUBTEST #2
                        EXPUTRAP                                        ;POSSIBLE FAILURE MODE
                        LOOP            I,1,6                           ;TEST LOOP OF 6
                        SA01PAT         I,1$,,                          ;GENERATE A TEST PATTERN
                        LOADREG         WDREG,1$,,L                     ;LOAD TEST PATTERN INTO WDREG
                        ERRLOOP                                         ;ERROR LOOP POINT
                        FETCH           0                               ;START DCS PROGRAM
                        BRSTCLK         9                               ;END DCS PROGRAM
                        CMPREGMSK       CSTRP,2$,,3$,,W,                ;TEST FOR A U-TRAP
                        IFERROR         ,,<<MC0><MC1><MC2>>             ;TEST U-TRAPPED
```

```
                                00C3  6340          CMPREG          WHREG,1$,,L,                    ;TEST FOR CORRECT DATA
                                00CC  6341          IFERROR         ,<<MDL>>,<<MC0><MC1><MC2>>;INCORRECT TEST PATTERN
                                00D5  6342          ENDLOOP         I                               ;END TEST LOOP
                                00D8  6343          SKIP                                            ;GO TO NEXT TEST
                                00DF  6344
                                00DF  6345
                                00DF  6346
                                00DF  6347 BEGIN_TEST_DATA
                                00DF
                                00DF          ;-------------------------------------------------------------------------
                                00DF  6348
                    00000000    00DF  6349 1$:      .LONG   ^X00000000                              ;TEST PATTERN STORAGE
                        1809    00E3  6350 2$:      .WORD   ^X1809                                  ;LAST DCS LOCATION
                        3FFF    00E5  6351 3$:      .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO OP
                                00E7  6352 4$:
U 05, 7700,4800,5BE4,030C,4A70,1806 00E7  6353 ;% 05    VA_R[TEMP3]                                     ;U-CODE FOR SUBTEST #1
                                00F2  6357 5$:
U 05, 7700,0800,5BE4,0310,4A70,1806 00F2  6358 ;% 05    VA_R[TEMP4]                                     ;U-CODE FOR SUBTEST #2
                    1FFFFFFF    00FD  6362 6$:      .LONG   ^X1FFFFFFF                              ;MASK FOR CMPREGMSK PSEUDO OP
                                0101  6363
                                0101  6364
                                0101  6365 BEGIN_CPU_DATA
                                0002  6366
                                0002  6367 DCS_DATA
                                0001  6368
U 00, 7700,C800,0364,0300,0470,1801 0001  6369 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  6373 ;% 01    MEMSCAR_R[TEMP0]                                ;ENABLE CMI
U 02, 7700,C800,5BE4,03D8,6070,1803 0017  6377 ;% 02    MEMSCR_R[ZERO]                                  ;...
U 03, 7700,8800,5BE4,0304,6870,1804 0022  6381 ;% 03    MEMSCAR_R[TEMP1]                                ;DISABLE CACHE
U 04, 7700,8800,5BE4,0308,6070,1805 002D  6385 ;% 04    MEMSCR_R[TEMP2]                                 ;...
U 05, 7700,4800,5BE4,030C,4A70,1806 0038  6389 ;% 05    VA_R[TEMP3]                                     ;ADDRESS OF CSR TO TEST
U 06, 5701,8800,0364,0200,5590,1807 0043  6393 ;% 06    WRITE.LONG,SIZE[LONG],WCTRL/WDR_WB.UR,WB_RDM
                                004E  6397                                                            ;WRITE TEST PATTERN
U 07, 7700,4800,0364,0300,0400,1808 004E  6398 ;% 07    READ.PHY                                        ;READ DATA BACK
U 08, 6700,0812,5924,0300,0470,1809 0059  6402 ;% 08    RDM_WB,WB_M[MDR]                                ;MOVE DATA TO RDM WHREG
U 09, 7300,4800,0364,0300,0470,180A 0064  6406 ;% 09    NOP,STOP.CLOCK                                  ;STOP THE CPU CLOCK
                                006F  6410
                                006F  6411 RTEMP_DATA
                                0001  6412
                    0E000000    0001  6413          .LONG   ^X0E000000                              ;REFERENCE CACHE ONLY ADDRESS
                    06000000    0005  6414          .LONG   ^X06000000                              ;CACHE CONTROL REGISTER ADDRESS
                    01000000    0009  6415          .LONG   ^X01000000                              ;DISABLE CACHE
                    00F20000    000D  6416          .LONG   ^X00F20000                              ;ADDRESS OF CSR0
                    00F20004    0011  6417          .LONG   ^X00F20004                              ;ADDRESS OF CSR1
                                0015  6418
                                0015  6419 END_CPU_DATA
                                0101  6420
                                0101  6421
                                0101  6422 END_TEST_DATA
                                0101
                                0101          ;-------------------------------------------------------------------------
                                0101  6423 ENDTEST
```

```
                        .SBTTL  "TEST   07C     TEST FOR CONTROLLER TYPE AND ARRAY CONFIGURATION-PART 1
                ;;;
```

```
                        003C  6479 ;       SLOT [4] = yyyy
                        003C  6480 ;       SLOT [5] = yyyy
                        003C  6481 ;       SLOT [6] = yyyy
                        003C  6482 ;       SLOT [7] = yyyy
                        003C  6483 ;
                        003C  6484 ; ERROR DESCRIPTION:
                        003C  6485 ;
                        003C  6486 ;       EXPECTED CS ADDRESS
                        003C  6487 ;       RECEIVED CS ADDRESS
                        003C  6488 ;--
                        003C  6489 ;;;
```

```
00F5  6534 205$:   PRINT_S        52$                     ;this is slot 1 test
00F9  6535         CMPREGMSK      WHREG,8$,,30$,,W,       ;check for 256kb present
0105  6536         SKIP           206$,ONERROR            ;skip if not present
010C  6537         PRINT_S        61$                     ;present, print "= 256KB"
0110  6538         SKIP           210$                    ;goto slot 2 test
0117  6539 206$:   CMPREGMSK      WHREG,9$,,30$,,W,       ;check for 128kb present
0123  6540         SKIP           207$,ONERROR            ;skip if not present
012A  6541         PRINT_S        62$                     ;present, print "= 128KB"
012E  6542         SKIP           210$                    ;goto slot 2 test
0135  6543 207$:   CMPREGMSK      WHREG,10$,,30$,,W,      ;check for 1mb present
0141  6544         SKIP           208$,ONERROR            ;skip if not present
0148  6545         PRINT_S        60$                     ;present, print "= 1MB"
014C  6546         SKIP           210$                    ;goto slot 2 test
0153  6547 208$:   PRINT_S        63$                     ;empty, print "= EMPTY"
0157  6548
0157  6549 ;-------------------------------------------------------------------------------
0157  6550 ;
0157  6551 ;       Slot 2 configuration test
0157  6552 ;
0157  6553 ;-------------------------------------------------------------------------------
0157  6554
0157  6555 210$:   PRINT_S        53$                     ;this is slot 2 test
015B  6556         CMPREGMSK      WHREG,11$,,31$,,W,      ;check for 256kb present
0167  6557         SKIP           211$,ONERROR            ;skip if not present
016E  6558         PRINT_S        61$                     ;present, print "= 256KB"
0172  6559         SKIP           215$                    ;goto slot 3 test
0179  6560 211$:   CMPREGMSK      WHREG,12$,,31$,,W,      ;check for 128kb present
0185  6561         SKIP           212$,ONERROR            ;skip if not present
018C  6562         PRINT_S        62$                     ;present, print "= 128KB"
0190  6563         SKIP           215$                    ;goto slot 3 test
0197  6564 212$:   CMPREGMSK      WHREG,13$,,31$,,W,      ;check for 1mb present
01A3  6565         SKIP           213$,ONERROR            ;skip if not present
01AA  6566         PRINT_S        60$                     ;present, print "= 1MB"
01AE  6567         SKIP           215$                    ;goto slot 3 test
01B5  6568 213$:   PRINT_S        63$                     ;empty, print "= EMPTY"
01B9  6569
01B9  6570 ;-------------------------------------------------------------------------------
01B9  6571 ;
01B9  6572 ;       Slot 3 configuration test
01B9  6573 ;
01B9  6574 ;-------------------------------------------------------------------------------
01B9  6575
01B9  6576 215$:   PRINT_S        54$                     ;this is slot 3 test
01BD  6577         CMPREGMSK      WHREG,14$,,32$,,W,      ;check for 256kb present
01C9  6578         SKIP           216$,ONERROR            ;skip if not present
01D0  6579         PRINT_S        61$                     ;present, print "= 256KB"
01D4  6580         SKIP           220$                    ;goto slot 4 test
01DB  6581 216$:   CMPREGMSK      WHREG,15$,,32$,,W,      ;check for 128kb present
01E7  6582         SKIP           217$,ONERROR            ;skip if not present
01EE  6583         PRINT_S        62$                     ;present, print "= 128KB"
01F2  6584         SKIP           220$                    ;goto slot 4 test
01F9  6585 217$:   CMPREGMSK      WHREG,16$,,32$,,W,      ;check for 1mb present
0205  6586         SKIP           218$,ONERROR            ;skip if not present
020C  6587         PRINT_S        60$                     ;present, print "= 1MB"
0210  6588         SKIP           220$                    ;goto slot 4 test
```

```
0217  6589 218$:   PRINT_S         63$                        ;empty, print "= EMPTY"
021B  6590
021B  6591 ;-------------------------------------------------------------------------------
021B  6592 ;
021B  6593 ;       Slot 4 configuration test
021B  6594 ;
021B  6595 ;-------------------------------------------------------------------------------
021B  6596
021B  6597 220$:   PRINT_S         55$                        ;this is slot 4 test
021F  6598         CMPREGMSK       WHREG,17$,,33$,,W,         ;check for 256kb present
022B  6599         SKIP            221$,ONERROR               ;skip if not present
0232  6600         PRINT_S         61$                        ;present, print "= 256KB"
0236  6601         SKIP            225$                       ;goto slot 5 test
023D  6602 221$:   CMPREGMSK       WHREG,18$,,33$,,W,         ;check for 128kb present
0249  6603         SKIP            222$,ONERROR               ;skip if not present
0250  6604         PRINT_S         62$                        ;present, print "= 128KB"
0254  6605         SKIP            225$                       ;goto slot 5 test
025B  6606 222$:   CMPREGMSK       WHREG,19$,,33$,,W,         ;check for 1mb present
0267  6607         SKIP            223$,ONERROR               ;skip if not present
026E  6608         PRINT_S         60$                        ;present, print "= 1MB"
0272  6609         SKIP            225$                       ;goto slot 5 test
0279  6610 223$:   PRINT_S         63$                        ;empty, print "= EMPTY"
027D  6611
027D  6612 ;-------------------------------------------------------------------------------
027D  6613 ;
027D  6614 ;       Slot 5 configuration test
027D  6615 ;
027D  6616 ;-------------------------------------------------------------------------------
027D  6617
027D  6618 225$:   PRINT_S         56$                        ;this is slot 5 test
0281  6619         CMPREGMSK       WHREG,20$,,34$,,W,         ;check for 256kb present
028D  6620         SKIP            226$,ONERROR               ;skip if not present
0294  6621         PRINT_S         61$                        ;present, print "= 256KB"
0298  6622         SKIP            230$                       ;goto slot 6 test
029F  6623 226$:   CMPREGMSK       WHREG,21$,,34$,,W,         ;check for 128kb present
02AB  6624         SKIP            227$,ONERROR               ;skip if not present
02B2  6625         PRINT_S         62$                        ;present, print "= 128KB"
02B6  6626         SKIP            230$                       ;goto slot 6 test
02BD  6627 227$:   CMPREGMSK       WHREG,22$,,34$,,W,         ;check for 1mb present
02C9  6628         SKIP            228$,ONERROR               ;skip if not present
02D0  6629         PRINT_S         60$                        ;present, print "= 1MB"
02D4  6630         SKIP            230$                       ;goto slot 6 test
02DB  6631 228$:   PRINT_S         63$                        ;empty, print "= EMPTY"
02DF  6632
02DF  6633 ;-------------------------------------------------------------------------------
02DF  6634 ;
02DF  6635 ;       Slot 6 configuration test
02DF  6636 ;
02DF  6637 ;-------------------------------------------------------------------------------
02DF  6638
02DF  6639 230$:   PRINT_S         57$                        ;this is slot 6 test
02E3  6640         CMPREGMSK       WHREG,23$,,35$,,W,         ;check for 256kb present
02EF  6641         SKIP            231$,ONERROR               ;skip if not present
02F6  6642         PRINT_S         61$                        ;present, print "= 256KB"
02FA  6643         SKIP            235$                       ;goto slot 7 test
```

```
                0301  6644 231$:   CMPREGMSK       WHREG,24$,,35$,,W,      ;check for 128kb present
                030D  6645          SKIP           232$,ONERROR            ;skip if not present
                0314  6646          PRINT_S        62$                     ;present, print "= 128KB"
                0318  6647          SKIP           235$                    ;goto slot 7 test
                031F  6648 232$:    CMPREGMSK      WHREG,25$,,35$,,W,      ;check for 1mb present
                032B  6649          SKIP           233$,ONERROR            ;skip if not present
                0332  6650          PRINT_S        60$                     ;present, print "= 1MB"
                0336  6651          SKIP           235$                    ;goto slot 7 test
                033D  6652 233$:    PRINT_S        63$                     ;empty, print "= EMPTY"
                0341  6653
                0341  6654 ;-------------------------------------------------------------------------------
                0341  6655 ;
                0341  6656 ;       Slot 7 configuration test
                0341  6657 ;
                0341  6658 ;-------------------------------------------------------------------------------
                0341  6659
                0341  6660 235$:    PRINT_S        58$                     ;this is slot 7 test
                0345  6661          CMPREGMSK      WHREG,26$,,36$,,W,      ;check for 256kb present
                0351  6662          SKIP           236$,ONERROR            ;skip if not present
                0358  6663          PRINT_S        61$                     ;present, print "= 256KB"
                035C  6664          SKIP                                   ;done, goto next test
                0363  6665 236$:    CMPREGMSK      WHREG,27$,,36$,,W,      ;check for 128kb present
                036F  6666          SKIP           237$,ONERROR            ;skip if not present
                0376  6667          PRINT_S        62$                     ;present, print "= 128KB"
                037A  6668          SKIP                                   ;done, goto next test
                0381  6669 237$:    CMPREGMSK      WHREG,28$,,36$,,W,      ;check for 1mb present
                038D  6670          SKIP           238$,ONERROR            ;skip if not present
                0394  6671          PRINT_S        60$                     ;present, print "= 1MB"
                0398  6672          SKIP                                   ;done, goto next test
                039F  6673 238$:    PRINT_S        63$                     ;empty, print "= EMPTY"
                03A3  6674          SKIP                                   ;done, goto next test
                03AA  6675
                03AA  6676
                03AA  6677 BEGIN_TEST_DATA
                03AA
                03AA       ;-------------------------------------------------------------------------
                03AA  6678
                03AA  6679
            1808  03AA  6680 1$:      .WORD    ^X1808                     ;last dcs location
            3FFF  03AC  6681 2$:      .WORD    ^X3FFF                     ;mask for first cmpregmsk
        00000000  03AE  6682 3$:      .LONG    ^X00000000                 ;dest. for 2nd and 3rd cmpregmsk
        02000000  03B2  6683 37$:     .LONG    ^X02000000                 ;mask for second cmpregmsk
        01000000  03B6  6684 4$:      .LONG    ^X01000000                 ;mask for third cmpregmsk
            0003  03BA  6685 5$:      .WORD    ^X0003                     ;slot 0 256kb
            0001  03BC  6686 6$:      .WORD    ^X0001                     ;slot 0 128kb
            0002  03BE  6687 7$:      .WORD    ^X0002                     ;slot 0 1mb
            000C  03C0  6688 8$:      .WORD    ^X000C                     ;slot 1 256kb
            0004  03C2  6689 9$:      .WORD    ^X0004                     ;slot 1 128kb
            0008  03C4  6690 10$:     .WORD    ^X0008                     ;slot 1 1mb
            0030  03C6  6691 11$:     .WORD    ^X0030                     ;slot 2 256kb
            0010  03C8  6692 12$:     .WORD    ^X0010                     ;slot 2 128kb
            0020  03CA  6693 13$:     .WORD    ^X0020                     ;slot 2 1mb
            00C0  03CC  6694 14$:     .WORD    ^X00C0                     ;slot 3 256kb
            0040  03CE  6695 15$:     .WORD    ^X0040                     ;slot 3 128kb
            0080  03D0  6696 16$:     .WORD    ^X0080                     ;slot 3 1mb
```

```
                        0300  03D2  6697 17$:    .WORD   ^X0300                          ;slot 4 256kb
                        0100  03D4  6698 18$:    .WORD   ^X0100                          ;slot 4 128kb
                        0200  03D6  6699 19$:    .WORD   ^X0200                          ;slot 4 1mb
                        0C00  03D8  6700 20$:    .WORD   ^X0C00                          ;slot 5 256kb
                        0400  03DA  6701 21$:    .WORD   ^X0400                          ;slot 5 128kb
                        0800  03DC  6702 22$:    .WORD   ^X0800                          ;slot 5 1mb
                        3000  03DE  6703 23$:    .WORD   ^X3000                          ;slot 6 256kb
                        1000  03E0  6704 24$:    .WORD   ^X1000                          ;slot 6 128kb
                        2000  03E2  6705 25$:    .WORD   ^X2000                          ;slot 6 1mb
                        C000  03E4  6706 26$:    .WORD   ^XC000                          ;slot 7 256kb
                        4000  03E6  6707 27$:    .WORD   ^X4000                          ;slot 7 128kb
                        8000  03E8  6708 28$:    .WORD   ^X8000                          ;slot 7 1mb
                        0003  03EA  6709 29$:    .WORD   ^X0003                          ;slot 0 mask
                        000C  03EC  6710 30$:    .WORD   ^X000C                          ;slot 1 mask
                        0030  03EE  6711 31$:    .WORD   ^X0030                          ;slot 2 mask
                        00C0  03F0  6712 32$:    .WORD   ^X00C0                          ;slot 3 mask
                        0300  03F2  6713 33$:    .WORD   ^X0300                          ;slot 4 mask
                        0C00  03F4  6714 34$:    .WORD   ^X0C00                          ;slot 5 mask
                        3000  03F6  6715 35$:    .WORD   ^X3000                          ;slot 6 mask
                        C000  03F8  6716 36$:    .WORD   ^XC000                          ;slot 7 mask
4F 43 20 59 52 4F 4D 45 4D 0A 0D 00' 03FA  6717 50$:    .ASCIC  <13><10>/MEMORY CONTROLLER = L0011/<13><10>
4C 20 3D 20 52 45 4C 4C 4F 52 54 4E  0406
                  0A 0D 31 31 30 30  0412
                              1D     03FA
                                     0418  6718                                        ;message for L0011
            5D 30 5B 54 4F 4C 53 00' 0418  6719 51$:    .ASCIC  /SLOT[0]/               ;message for slot 0
                                 07  0418
            5D 31 5B 54 4F 4C 53 00' 0420  6720 52$:    .ASCIC  /SLOT[1]/               ;message for slot 1
                                 07  0420
            5D 32 5B 54 4F 4C 53 00' 0428  6721 53$:    .ASCIC  /SLOT[2]/               ;message for slot 2
                                 07  0428
            5D 33 5B 54 4F 4C 53 00' 0430  6722 54$:    .ASCIC  /SLOT[3]/               ;message for slot 3
                                 07  0430
            5D 34 5B 54 4F 4C 53 00' 0438  6723 55$:    .ASCIC  /SLOT[4]/               ;message for slot 4
                                 07  0438
            5D 35 5B 54 4F 4C 53 00' 0440  6724 56$:    .ASCIC  /SLOT[5]/               ;message for slot 5
                                 07  0440
            5D 36 5B 54 4F 4C 53 00' 0448  6725 57$:    .ASCIC  /SLOT[6]/               ;message for slot 6
                                 07  0448
            5D 37 5B 54 4F 4C 53 00' 0450  6726 58$:    .ASCIC  /SLOT[7]/               ;message for slot 7
                                 07  0450
         0A 0D 42 4D 31 20 3D 20 00' 0458  6727 60$:    .ASCIC  / = 1MB/<13><10>        ;array equals 1mb
                                 08  0458
   0A 0D 42 4B 36 35 32 20 3D 20 00' 0461  6728 61$:    .ASCIC  / = 256KB/<13><10>      ;array equals 256kb
                                 0A  0461
   0A 0D 42 4B 38 32 31 20 3D 20 00' 046C  6729 62$:    .ASCIC  / = 128KB/<13><10>      ;array equals 128kb
                                 0A  046C
   0A 0D 59 54 50 4D 45 20 3D 20 00' 0477  6730 63$:    .ASCIC  / = EMPTY/<13><10>
                                 0A  0477
                                     0482  6731                                        ;slot is empty
4F 43 20 59 52 4F 4D 45 4D 0A 0D 00' 0482  6732 70$:    .ASCIC  <13><10>/MEMORY CONTROLLER = L0016/<13><10>
4C 20 3D 20 52 45 4C 4C 4F 52 54 4E  048E
                  0A 0D 36 31 30 30  049A
                              1D     0482
                                     04A0  6733                                        ;message for L0016
```

```
                                04A0  6734
                                04A0  6735 BEGIN_CPU_DATA
                                0002  6736
                                0002  6737
                                0002  6738 DCS_DATA
                                0001  6739
U 00. 7700.C800.0364.0300.0470.1801   0001  6740 ;% 00    NOP
U 01. 7700.C800.5BE4.0300.6870.1802   000C  6744 ;% 01    MEMSCAR_R[TEMP0]                    ;point to MEMSCR #E
U 02. 7700.C800.5BE4.03D8.6070.1803   0017  6748 ;% 02    MEMSCR_R[ZERO]                      ;enable the cmi
U 03. 7700.8800.5BE4.0304.6870.1804   0022  6752 ;% 03    MEMSCAR_R[TEMP1]                    ;point to MEMSCR #6 (CADR)
U 04. 7700.8800.5BE4.0308.6070.1805   002D  6756 ;% 04    MEMSCR_R[TEMP2]                     ;disable cache
U 05. 7700.4800.5BE4.030C.4A70.1806   0038  6760 ;% 05    VA_R[TEMP3]                         ;address of CSR2
U 06. 7700.4800.0364.0300.0400.1807   0043  6764 ;% 06    READ.PHY                            ;read the contents of CSR2
U 07. 6700.8812.5924.0300.0470.1808   004E  6768 ;% 07    RDM_WB,WB_M[MDR]                    ;rdm WHREG <-- (CSR2)
U 08. 7300.4800.0364.0300.0470.1809   0059  6772 ;% 08    NOP,STOP.CLOCK                      ;stop the cpu clock
                                0064  6776
                                0064  6777 RTEMP_DATA
                                0001  6778
                     0E000000   0001  6779         .LONG   ^X0E000000                      ;cmi disable register address
                     06000000   0005  6780         .LONG   ^X06000000                      ;cache disable register address
                     01000000   0009  6781         .LONG   ^X01000000                      ;data to disable cache
                     00F20008   000D  6782         .LONG   ^X00F20008                      ;CSR2 physical address
                                0011  6783
                                0011  6784
                                0011  6785 END_CPU_DATA
                                04A0  6786
                                04A0  6787 END_TEST_DATA
                                04A0
                                04A0        ;-------------------------------------------------------------------------------
                                04A0  6788
                                04A0  6789 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
003C          .SBTTL  "TEST   07D     TEST FOR CONTROLLER TYPE AND ARRAY CONFIGURATION-PART 2
003C  6791 ;;;
```

```
                     6845 ;       SLOT [5] = yyyy
                     6846 ;       SLOT [6] = yyyy
                     6847 ;       SLOT [7] = yyyy
                     6848 ;
                     6849 ; ERROR DESCRIPTION:
                     6850 ;
                     6851 ;       EXPECTED CS ADDRESS
                     6852 ;       RECEIVED CS ADDRESS
                     6853 ;--
                     6854 ;;;
```

```
00F9  6900 206$:   CMPREGMSK       WHREG,9$,,30$,,W,       ;check for 4mb present
0105  6901         SKIP            207$,ONERROR            ;skip if not present
010C  6902         PRINT_S         62$                     ;present, print "= 4MB"
0110  6903         SKIP            210$                    ;goto slot 2 test
0117  6904 207$:   CMPREGMSK       WHREG,10$,,30$,,W,      ;check for 1mb present
0123  6905         SKIP            208$,ONERROR            ;skip if not present
012A  6906         PRINT_S         60$                     ;present, print "= 1MB"
012E  6907         SKIP            210$                    ;goto slot 2 test
0135  6908 208$:   PRINT_S         63$                     ;empty, print "= EMPTY"
0139  6909
0139  6910 ;-------------------------------------------------------------------------------
0139  6911 ;
0139  6912 ;       Slot 2 configuration test
0139  6913 ;
0139  6914 ;-------------------------------------------------------------------------------
0139  6915
0139  6916 210$:   PRINT_S         53$                     ;this is slot 2 test
013D  6917         CMPREGMSK       WHREG,11$,,31$,,W,      ;check for 256kb present
0149  6918         SKIP            211$,ONERROR            ;skip if not present
0150  6919         PRINT_S         61$                     ;present, print "= 256KB"
0154  6920         SKIP            215$                    ;goto slot 3 test
015B  6921 211$:   CMPREGMSK       WHREG,12$,,31$,,W,      ;check for 4mb present
0167  6922         SKIP            212$,ONERROR            ;skip if not present
016E  6923         PRINT_S         62$                     ;present, print "= 4MB"
0172  6924         SKIP            215$                    ;goto slot 3 test
0179  6925 212$:   CMPREGMSK       WHREG,13$,,31$,,W,      ;check for 1mb present
0185  6926         SKIP            213$,ONERROR            ;skip if not present
018C  6927         PRINT_S         60$                     ;present, print "= 1MB"
0190  6928         SKIP            215$                    ;goto slot 3 test
0197  6929 213$:   PRINT_S         63$                     ;empty, print "= EMPTY"
019B  6930
019B  6931 ;-------------------------------------------------------------------------------
019B  6932 ;
019B  6933 ;       Slot 3 configuration test
019B  6934 ;
019B  6935 ;-------------------------------------------------------------------------------
019B  6936
019B  6937 215$:   PRINT_S         54$                     ;this is slot 3 test
019F  6938         CMPREGMSK       WHREG,14$,,32$,,W,      ;check for 256kb present
01AB  6939         SKIP            216$,ONERROR            ;skip if not present
01B2  6940         PRINT_S         61$                     ;present, print "= 256KB"
01B6  6941         SKIP            220$                    ;goto slot 4 test
01BD  6942 216$:   CMPREGMSK       WHREG,15$,,32$,,W,      ;check for 4mb present
01C9  6943         SKIP            217$,ONERROR            ;skip if not present
01D0  6944         PRINT_S         62$                     ;present, print "= 4MB"
01D4  6945         SKIP            220$                    ;goto slot 4 test
01DB  6946 217$:   CMPREGMSK       WHREG,16$,,32$,,W,      ;check for 1mb present
01E7  6947         SKIP            218$,ONERROR            ;skip if not present
01EE  6948         PRINT_S         60$                     ;present, print "= 1MB"
01F2  6949         SKIP            220$                    ;goto slot 4 test
01F9  6950 218$:   PRINT_S         63$                     ;empty, print "= EMPTY"
01FD  6951
01FD  6952 ;-------------------------------------------------------------------------------
01FD  6953 ;
01FD  6954 ;       Slot 4 configuration test
```

```
01FD  ;
01FD  ;-------------------------------------------------------------------------------
01FD
01FD  220$:   PRINT_S        55$                        ;this is slot 4 test
0201          CMPREGMSK      WHREG,17$,,33$,,W,         ;check for 256kb present
020D          SKIP           221$,ONERROR               ;skip if not present
0214          PRINT_S        61$                        ;present, print "= 256KB"
0218          SKIP           225$                       ;goto slot 5 test
021F  221$:   CMPREGMSK      WHREG,18$,,33$,,W,         ;check for 4mb present
022B          SKIP           222$,ONERROR               ;skip if not present
0232          PRINT_S        62$                        ;present, print "= 4MB"
0236          SKIP           225$                       ;goto slot 5 test
023D  222$:   CMPREGMSK      WHREG,19$,,33$,,W,         ;check for 1mb present
0249          SKIP           223$,ONERROR               ;skip if not present
0250          PRINT_S        60$                        ;present, print "= 1MB"
0254          SKIP           225$                       ;goto slot 5 test
025B  223$:   PRINT_S        63$                        ;empty, print "= EMPTY"
025F
025F  ;-------------------------------------------------------------------------------
025F  ;
025F  ;       Slot 5 configuration test
025F  ;
025F  ;-------------------------------------------------------------------------------
025F
025F  225$:   PRINT_S        56$                        ;this is slot 5 test
0263          CMPREGMSK      WHREG,20$,,34$,,W,         ;check for 256kb present
026F          SKIP           226$,ONERROR               ;skip if not present
0276          PRINT_S        61$                        ;present, print "= 256KB"
027A          SKIP           230$                       ;goto slot 6 test
0281  226$:   CMPREGMSK      WHREG,21$,,34$,,W,         ;check for 4mb present
028D          SKIP           227$,ONERROR               ;skip if not present
0294          PRINT_S        62$                        ;present, print "= 4MB"
0298          SKIP           230$                       ;goto slot 6 test
029F  227$:   CMPREGMSK      WHREG,22$,,34$,,W,         ;check for 1mb present
02AB          SKIP           228$,ONERROR               ;skip if not present
02B2          PRINT_S        60$                        ;present, print "= 1MB"
02B6          SKIP           230$                       ;goto slot 6 test
02BD  228$:   PRINT_S        63$                        ;empty, print "= EMPTY"
02C1
02C1  ;-------------------------------------------------------------------------------
02C1  ;
02C1  ;       Slot 6 configuration test
02C1  ;
02C1  ;-------------------------------------------------------------------------------
02C1
02C1  230$:   PRINT_S        57$                        ;this is slot 6 test
02C5          CMPREGMSK      WHREG,23$,,35$,,W,         ;check for 256kb present
02D1          SKIP           231$,ONERROR               ;skip if not present
02D8          PRINT_S        61$                        ;present, print "= 256KB"
02DC          SKIP           235$                       ;goto slot 7 test
02E3  231$:   CMPREGMSK      WHREG,24$,,35$,,W,         ;check for 4mb present
02EF          SKIP           232$,ONERROR               ;skip if not present
02F6          PRINT_S        62$                        ;present, print "= 4MB"
02FA          SKIP           235$                       ;goto slot 7 test
0301  232$:   CMPREGMSK      WHREG,25$,,35$,,W,         ;check for 1mb present
```

```
                030D  7010          SKIP            233$,ONERROR            ;skip if not present
                0314  7011          PRINT_S         60$                     ;present, print "= 1MB"
                0318  7012          SKIP            235$                    ;goto slot 7 test
                031F  7013 233$:    PRINT_S         63$                     ;empty, print "= EMPTY"
                0323  7014
                0323  7015 ;-------------------------------------------------------------------------------
                0323  7016 ;
                0323  7017 ;       Slot 7 configuration test
                0323  7018 ;
                0323  7019 ;-------------------------------------------------------------------------------
                0323  7020
                0323  7021 235$:    PRINT_S         58$                     ;this is slot 7 test
                0327  7022          CMPREGMSK       WHREG,26$,,36$,,W,      ;check for 256kb present
                0333  7023          SKIP            236$,ONERROR            ;skip if not present
                033A  7024          PRINT_S         61$                     ;present, print "= 256KB"
                033E  7025          SKIP                                    ;done, goto next test
                0345  7026 236$:    CMPREGMSK       WHREG,27$,,36$,,W,      ;check for 4mb present
                0351  7027          SKIP            237$,ONERROR            ;skip if not present
                0358  7028          PRINT_S         62$                     ;present, print "= 4MB"
                035C  7029          SKIP                                    ;done, goto next test
                0363  7030 237$:    CMPREGMSK       WHREG,28$,,36$,,W,      ;check for 1mb present
                036F  7031          SKIP            238$,ONERROR            ;skip if not present
                0376  7032          PRINT_S         60$                     ;present, print "= 1MB"
                037A  7033          SKIP                                    ;done, goto next test
                0381  7034 238$:    PRINT_S         63$                     ;empty, print "= EMPTY"
                0385  7035          SKIP                                    ;done, goto next test
                038C  7036
                038C  7037
                038C  7038 BEGIN_TEST_DATA
                038C
                038C        ;-------------------------------------------------------------------------
                038C  7039
                038C  7040
            1808  038C  7041 1$:      .WORD   ^X1808                  ;last dcs location
            3FFF  038E  7042 2$:      .WORD   ^X3FFF                  ;mask for first cmpregmsk
        00000000  0390  7043 3$:      .LONG   ^X00000000              ;dest. for second cmpregmsk
        02000000  0394  7044 4$:      .LONG   ^X02000000              ;mask for second cmpregmsk
            0003  0398  7045 5$:      .WORD   ^X0003                  ;slot 0 256kb
            0001  039A  7046 6$:      .WORD   ^X0001                  ;slot 0 4mb
            0002  039C  7047 7$:      .WORD   ^X0002                  ;slot 0 1mb
            000C  039E  7048 8$:      .WORD   ^X000C                  ;slot 1 256kb
            0004  03A0  7049 9$:      .WORD   ^X0004                  ;slot 1 4mb
            0008  03A2  7050 10$:     .WORD   ^X0008                  ;slot 1 1mb
            0030  03A4  7051 11$:     .WORD   ^X0030                  ;slot 2 256kb
            0010  03A6  7052 12$:     .WORD   ^X0010                  ;slot 2 4mb
            0020  03A8  7053 13$:     .WORD   ^X0020                  ;slot 2 1mb
            00C0  03AA  7054 14$:     .WORD   ^X00C0                  ;slot 3 256kb
            0040  03AC  7055 15$:     .WORD   ^X0040                  ;slot 3 4mb
            0080  03AE  7056 16$:     .WORD   ^X0080                  ;slot 3 1mb
            0300  03B0  7057 17$:     .WORD   ^X0300                  ;slot 4 256kb
            0100  03B2  7058 18$:     .WORD   ^X0100                  ;slot 4 4mb
            0200  03B4  7059 19$:     .WORD   ^X0200                  ;slot 4 1mb
            0C00  03B6  7060 20$:     .WORD   ^X0C00                  ;slot 5 256kb
            0400  03B8  7061 21$:     .WORD   ^X0400                  ;slot 5 4mb
            0800  03BA  7062 22$:     .WORD   ^X0800                  ;slot 5 1mb
```

```
                        3000  03BC  7063 23$:    .WORD   ^X3000                          ;slot 6 256kb
                        1000  03BE  7064 24$:    .WORD   ^X1000                          ;slot 6 4mb
                        2000  03C0  7065 25$:    .WORD   ^X2000                          ;slot 6 1mb
                        C000  03C2  7066 26$:    .WORD   ^XC000                          ;slot 7 256kb
                        4000  03C4  7067 27$:    .WORD   ^X4000                          ;slot 7 4mb
                        8000  03C6  7068 28$:    .WORD   ^X8000                          ;slot 7 1mb
                        0003  03C8  7069 29$:    .WORD   ^X0003                          ;slot 0 mask
                        000C  03CA  7070 30$:    .WORD   ^X000C                          ;slot 1 mask
                        0030  03CC  7071 31$:    .WORD   ^X0030                          ;slot 2 mask
                        00C0  03CE  7072 32$:    .WORD   ^X00C0                          ;slot 3 mask
                        0300  03D0  7073 33$:    .WORD   ^X0300                          ;slot 4 mask
                        0C00  03D2  7074 34$:    .WORD   ^X0C00                          ;slot 5 mask
                        3000  03D4  7075 35$:    .WORD   ^X3000                          ;slot 6 mask
                        C000  03D6  7076 36$:    .WORD   ^XC000                          ;slot 7 mask
4F 43 20 59 52 4F 4D 45 4D 0A 0D 00' 03D8  7077 50$:    .ASCIC  <13><10>/MEMORY CONTROLLER = L0022/<13><10>
4C 20 3D 20 52 45 4C 4C 4F 52 54 4E  03E4
                  0A 0D 32 32 30 30  03F0
                              1D     03D8
                                     03F6  7078                                      ;message for L0022
            5D 30 5B 54 4F 4C 53 00' 03F6  7079 51$:    .ASCIC  /SLOT[0]/                       ;message for slot 0
                                 07  03F6
            5D 31 5B 54 4F 4C 53 00' 03FE  7080 52$:    .ASCIC  /SLOT[1]/                       ;message for slot 1
                                 07  03FE
            5D 32 5B 54 4F 4C 53 00' 0406  7081 53$:    .ASCIC  /SLOT[2]/                       ;message for slot 2
                                 07  0406
            5D 33 5B 54 4F 4C 53 00' 040E  7082 54$:    .ASCIC  /SLOT[3]/                       ;message for slot 3
                                 07  040E
            5D 34 5B 54 4F 4C 53 00' 0416  7083 55$:    .ASCIC  /SLOT[4]/                       ;message for slot 4
                                 07  0416
            5D 35 5B 54 4F 4C 53 00' 041E  7084 56$:    .ASCIC  /SLOT[5]/                       ;message for slot 5
                                 07  041E
            5D 36 5B 54 4F 4C 53 00' 0426  7085 57$:    .ASCIC  /SLOT[6]/                       ;message for slot 6
                                 07  0426
            5D 37 5B 54 4F 4C 53 00' 042E  7086 58$:    .ASCIC  /SLOT[7]/                       ;message for slot 7
                                 07  042E
         0A 0D 42 4D 31 20 3D 20 00' 0436  7087 60$:    .ASCIC  / = 1MB/<13><10>                ;array equals 1mb
                                 08  0436
4C 4C 49 2D 42 4B 36 35 32 3D 20 00' 043F  7088 61$:    .ASCIC  / =256KB-ILLEGAL CONFIG/<13><10>;array equals 256kb
0D 47 49 46 4E 4F 43 20 4C 41 47 45  044B
                                 0A  0457
                                 18  043F
         0A 0D 42 4D 34 20 3D 20 00' 0458  7089 62$:    .ASCIC  / = 4MB/<13><10>                ;array equals 4mb
                                 08  0458
   0A 0D 59 54 50 4D 45 20 3D 20 00' 0461  7090 63$:    .ASCIC  / = EMPTY/<13><10>
                                 0A  0461
                                     046C  7091                                      ;slot is empty
                                     046C  7092
                                     046C  7093 BEGIN_CPU_DATA
                                     0002  7094
                                     0002  7095
                                     0002  7096 DCS_DATA
                                     0001  7097
U 00, 7700,C800,0364,0300,0470,1801  0001  7098 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  7102 ;% 01    MEMSCAR_R[TEMP0]                ;point to MEMSCR #E
U 02, 7700,C800,5BE4,03D8,6070,1803  0017  7106 ;% 02    MEMSCR_R[ZERO]                  ;enable the cmi
```

```
U 03. 7700,8800,5BE4,0304,6870,1804  0022  7110 ;% 03   MEMSCAR_R[TEMP1]                    ;point to MEMSCR #6 (CADR)
U 04. 7700,8800,5BE4,0308,6070,1805  002D  7114 ;% 04   MEMSCR_R[TEMP2]                     ;disable cache
U 05. 7700,4800,5BE4,030C,4A70,1806  0038  7118 ;% 05   VA_R[TEMP3]                        ;address of CSR2
U 06. 7700,4800,0364,0300,0400,1807  0043  7122 ;% 06   READ.PHY                          ;read the contents of CSR2
U 07. 6700,8812,5924,0300,0470,1808  004E  7126 ;% 07   RDM_WB,WB_M[MDR]                  ;rdm WHREG <-- (CSR2)
U 08. 7300,4800,0364,0300,0470,1809  0059  7130 ;% 08   NOP,STOP.CLOCK                     ;stop the cpu clock
                                     0064  7134
                                     0064  7135 RTEMP_DATA
                                     0001  7136
                         0E000000    0001  7137         .LONG   ^X0E000000               ;cmi disable register address
                         06000000    0005  7138         .LONG   ^X06000000               ;cache disable register address
                         01000000    0009  7139         .LONG   ^X01000000               ;data to disable cache
                         00F20008    000D  7140         .LONG   ^X00F20008               ;CSR2 physical address
                                     0011  7141
                                     0011  7142
                                     0011  7143 END_CPU_DATA
                                     046C  7144
                                     046C  7145 END_TEST_DATA
                                     046C
                                     046C       ;-----------------------------------------------------------------------
                                     046C  7146
                                     046C  7147 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
0029        .SBTTL  "TEST   07E     TEST DATA INTEGRITY FOR ADDRESS TEST
0029  7149 ;*****************************************************************************
0029  7150 ;++
0029  7151 ;
0029  7152 ; FUNCTIONAL DESCRIPTION:
0029  7153 ;
0029  7154 ;       THIS TEST WILL VERIFY THE DATA INTEGRITY OF LOACTIONS IN MAIN MEMORY
0029  7155 ;       THAT WILL BE USED IN THE DUAL ADDRESSING TEST.
0029  7156 ;
0029  7157 ;
0029  7158 ; TEST ALGORITHM:
0029  7159 ;
0029  7160 ;       1.  LOAD DCS WITH CODE TO DISABLE CACHE AND ENABLE THE CMI
0029  7161 ;       2.  EXECUTE DCS PROGRAM
0029  7162 ;               a.  TURN CMI ON
0029  7163 ;               b.  DISABLE CACHE
0029  7164 ;       3.  LOAD DCS WITH TEST MICRO-CODE
0029  7165 ;       4.  CREATE A TEST LOOP OF 6 (J)
0029  7166 ;       5.  GENERATE A TEST PATTERN
0029  7167 ;       6.  LOAD TEST PATTERN INTO DCS
0029  7168 ;       7.  CREATE A SECOND LOOP OF 15 (I)
0029  7169 ;       8.  LOAD TEST ADDRESS INTO RDM WDREG
0029  7170 ;       9.  EXECUTE DCS PROGRAM
0029  7171 ;               a.  PUT TEST ADDRESS INTO VA AND RTEMP0
0029  7172 ;               b.  PUT TEST PATTERN INTO LONGLIT REGISTER
0029  7173 ;               c.  WRITE TEST PATTERN TO MAIN MEMORY
0029  7174 ;               d.  READ TEST PATTERN BACK TO CPU
0029  7175 ;               e.  MOVE TEST PATTERN TO RDM WHREG
0029  7176 ;      10.  INCASE OF MICRO-TRAP REEXECUTE LAST MICRO-INSTRUCTION
0029  7177 ;      11.  TEST WHREG FOR CORRECT TEST PATTERN
0029  7178 ;      12.  IF NO ERROR GO TO STEP 7 FOR REMAINING TEST ADDRESSES
0029  7179 ;      13.  GO TO STEP 4 FOR REMAINING TEST PATTERNS
0029  7180 ;
0029  7181 ;
0029  7182 ; TEST PATTERNS:
0029  7183 ;
0029  7184 ;       ITERATION (J)   TEST PATTERN
0029  7185 ;            1            AAAAAAAA
0029  7186 ;            2            55555555
0029  7187 ;            3            33333333
0029  7188 ;            4            0F0F0F0F
0029  7189 ;            5            00FF00FF
0029  7190 ;            6            0000FFFF
0029  7191 ;
0029  7192 ;
0029  7193 ; LOGIC DESCRIPTION:
0029  7194 ;
0029  7195 ;
0029  7196 ;       THIS TEST WILL VERIFY THAT THE LOCATIONS USED IN THE DUAL ADDRESSING
0029  7197 ;       TEST CAN HOLD DATA.  THIS WILL ELIMINATE GETTING DATA AND ADDRESSING
0029  7198 ;       ERRORS AT THE SAME TIME.
0029  7199 ;
0029  7200 ;
0029  7201 ; ASSUMPTIONS:
0029  7202 ;
```

```
                        0029  7203 ;       THIS TEST ASSUMES THAT THE PREVIOUS CMI TESTS HAVE RUN SUCCESSFULLY
                        0029  7204 ;
                        0029  7205 ;
                        0029  7206 ; ERROR DESCRIPTION:
                        0029  7207 ;
                        0029  7208 ;       EXPECTED MEMORY DATA
                        0029  7209 ;       RECEIVED MEMORY DATA
                        0029  7210 ;       LOOP COUNT I (POINTER TO TEST ADDRESS)
                        0029  7211 ;       LOOP COUNT J (POINTER TO TEST PATTERN)
                        0029  7212 ;       RTEMP0 (ADDRESS BEING TESTED)
                        0029  7213 ;
                        0029  7214 ;--
                        0029  7215 ;*******************************************************************************
                        0029  7216
                        0029  7217
                        0029  7218
                        0029  7219         INITIALIZE                                      ;INIT THE CPU
                        002B  7220         EXPUTRAP                                        ;POSSIBLE U-TRAP
                        002D  7221         LOADDCS         1$,,0,6                         ;CODE TO DISABLE CACHE
                        0034  7222         FETCH           0                               ;START DCS PROGRAM
                        0039  7223         BRSTCLK         5                               ;END DCS PROGRAM
                        003D  7224         LOADDCS         2$,,0,8                         ;CODE TO TEST MEMORY
                        0044  7225         LOOP            J,1,6                           ;LOOP TO CREATE TEST PATTERNS
                        004B  7226         SA01PAT         J,4$,LON,5$                     ;GENERATE A TEST PATTERN
                        0053  7227         LOADDCS         4$,,3,1                         ;LOAD TEST PATTERN TO DCS
                        005A  7228         LOOP            I,1,15                          ;LOOP TO TEST MEMORY
                        0061  7229         LOADREG         WDREG,3$,I,L                    ;LOAD TEST ADDRESS
                        0069  7230         ERRLOOP                                         ;ERROR LOOP POINT
                        006B  7231         FETCH           0                               ;START DCS PROGRAM
                        0070  7232         BRSTCLK         7                               ;END DCS PROGRAM
                        0074  7233         FETCH           6                               ;START DCS PROGRAM
                        0079  7234         BRSTCLK         7                               ;END DCS PROGRAM
                        007D  7235         CMPREG          WHREG,5$,,L,                    ;TEST FOR CORRECT PATTERN
                        0086  7236         IFERROR         1,<<MDL><MEC>>,-
                        0086  7237         <<MC0><MC1><MC2><MA0><MA1><MA2>>                ;INCORRECT DATA
                        0093  7238         ENDLOOP         I                               ;END TEST LOOP
                        0096  7239         ENDLOOP         J                               ;END TEST PATTERN LOOP
                        0099  7240         SKIP                                            ;GO TO NEXT TEST
                        00A0  7241
                        00A0  7242
                        00A0  7243 BEGIN_TEST_DATA
                        00A0               ;-------------------------------------------------------------------------
                        00A0  7244
                        00A0  7245 1$:
U 00, 7700,C800,0364,0300,0470,1801     00A0  7246 ;% 00    NOP
U 01, 7700,8800,5BE4,0304,6870,1802     00AB  7250 ;% 01    MEMSCAR_R[TEMP1]                ;TURN CMI ON
U 02, 7700,C800,5BE4,03D8,6070,1803     00B6  7254 ;% 02    MEMSCR_R[ZERO]
U 03, 7700,8800,5BE4,0308,6870,1804     00C1  7258 ;% 03    MEMSCAR_R[TEMP2]                ;TURN CACHE OFF
U 04, 7700,C800,5BE4,030C,6070,1805     00CC  7262 ;% 04    MEMSCR_R[TEMP3]
U 05, 7300,4800,0364,0300,0470,1806     00D7  7266 ;% 05    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                        00E2  7270 2$:
U 00, 7700,C800,0364,0300,0470,1801     00E2  7271 ;% 00    NOP
U 01, 7700,C800,0364,0300,0470,1802     00ED  7275 ;% 01    NOP
U 02, 5700,0840,0364,0300,4A70,1803     00F8  7279 ;% 02    VA_RDM,R[TEMP0]_WB              ;ADDRESS TO VA AND RTEMP 0
```

```
U 03. 7700.7800.7FFF.FFFF.8470.1804  0103  7283 ;% 03   LONLIT_[0]                          ;TEST PATTERN TO LONGLIT REG
U 04. 7700.C800.5BE4.03D4.5C80.1805  010E  7287 ;% 04   WRITE.PHY R[LONLIT]                 ;WRITE TEST PATTERN TO MEMORY
U 05. 7700.C800.0364.0300.0400.1806  0119  7291 ;% 05   READ.PHY                           ;READ TEST PATTERN BACK TO RDM
U 06. 6700.8812.5924.0300.0470.1807  0124  7295 ;% 06   RDM_WB,WB_M[MDR]                  ;READ PATTERN TO RDM
U 07. 7300.C800.0364.0300.0470.1808  012F  7299 ;% 07   NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
                          00000000   013A  7303 3$:     .LONG   ^X00000000               ;TEST ADDRESS TABLE
                          00000004   013E  7304         .LONG   ^X00000004
                          00000008   0142  7305         .LONG   ^X00000008
                          00000010   0146  7306         .LONG   ^X00000010
                          00000020   014A  7307         .LONG   ^X00000020
                          00000040   014E  7308         .LONG   ^X00000040
                          00000080   0152  7309         .LONG   ^X00000080
                          00000100   0156  7310         .LONG   ^X00000100
                          00000200   015A  7311         .LONG   ^X00000200
                          00000400   015E  7312         .LONG   ^X00000400
                          00000800   0162  7313         .LONG   ^X00000800
                          00001000   0166  7314         .LONG   ^X00001000
                          00002000   016A  7315         .LONG   ^X00002000
                          00004000   016E  7316         .LONG   ^X00004000
                          00008000   0172  7317         .LONG   ^X00008000
                                     0176  7318 4$:
U 02. 7700.F800.7FFF.FFFF.8470.1803  0176  7319 ;% 02   LONLIT_[0]                          ;TEST PATTERN TO LONGLIT
                          00000000   0181  7323 5$:     .LONG   ^X00000000               ;TEST PATTERN STORAGE
                                     0185  7324
                                     0185  7325
                                     0185  7326 BEGIN_CPU_DATA
                                     0002  7327
                                     0002  7328 RTEMP_DATA
                                     0001  7329
                   00000000          0001  7330         .LONG   ^X00000000               ;VA STORAGE
                   0E000000          0005  7331         .LONG   ^X0E000000               ;REFERENCE CACHE ONLY ADDRESS
                   06000000          0009  7332         .LONG   ^X06000000               ;CACHE CONTROL REG ADDRESS
                   01000000          000D  7333         .LONG   ^X01000000               ;TURN CACHE OFF
                                     0011  7334
                                     0011  7335 END_CPU_DATA
                                     0185  7336
                                     0185  7337
                                     0185  7338 END_TEST_DATA
                                     0185
                                     0185       ;-----------------------------------------------------------------------
                                     0185  7339 ENDTEST
```

```
0025          .SBTTL  "TEST   07F     MAIN MEMORY DUAL ADDRESSING TEST
0025  7341 ;*****************************************************************************
0025  7342 ;++
0025  7343 ;
0025  7344 ; FUNCTIONAL DESCRIPTION:
0025  7345 ;
0025  7346 ;       THIS TEST WILL ATTEMPT TO DETECT ANY ADDRESSING PROBLEMS IN THE FIRST
0025  7347 ;       256K BYTES OF MAIN MEMORY (FIRST ARRAY CARD)
0025  7348 ;
0025  7349 ;
0025  7350 ; TEST ALGORITHM:
0025  7351 ;
0025  7352 ;       1.  LOAD MICRO-CODE TO DISABLE CACHE AND ENABLE THE CMI AND GET CSR2
0025  7353 ;           TO DETERMINE WHICH CONTROLLER IS INSTALLED
0025  7354 ;       2.  EXECUTE DCS PROGRAM
0025  7355 ;               a.  TURN CMI ON
0025  7356 ;               b.  TURN CACHE OFF
0025  7357 ;       3.  CREATE A TEST LOOP OF 15 (I)
0025  7358 ;       4.  LOAD MICRO CODE TO WRITE A TEST ADDRESS POINTED TO BY (I)
0025  7359 ;       5.  LOAD TEST ADDRESS INTO WDREG
0025  7360 ;       6.  EXECUTE DCS PROGRAM
0025  7361 ;               a.  PUT TEST ADDRESS INTO VA AND RTEMP0
0025  7362 ;               b.  WRITE TEST ADDRESS WITH ITSELF
0025  7363 ;       7.  LOAD MICRO-CODE TO PERFORM TEST
0025  7364 ;       8.  CREATE A TEST LOOP OF 15 (J)
0025  7365 ;       9.  IF I = J INCREMENT J AND CONTINUE
0025  7366 ;      10.  LOAD WDREG WITH AN ADDRESS POINTED TO BY (J)
0025  7367 ;      11.  EXECUTE DCS PROGRAM
0025  7368 ;               a.  PUT TEST ADDRESS IN VA AND RTEMP1
0025  7369 ;               b.  WRITE TO ADDRESS IN VA WITH ITSELF
0025  7370 ;               c.  PUT TEST ADDRESS IN RTEMP0 INTO VA
0025  7371 ;               d.  READ THIS ADDRESS BACK TO RDM
0025  7372 ;      12.  REPEAT MEMORY READ INCASE OF A MICRO TRAP
0025  7373 ;      13.  TEST WHREG FOR CORRECT TEST PATTERN
0025  7374 ;      14.  IF NO ERROR GO TO STEP 9 UNTIL ALL ADDRESSES ARE TESTED
0025  7375 ;      15.  GO TO STEP 4 UNTIL ALL ADDRESSES ARE TESTED
0025  7376 ;
0025  7377 ;
0025  7378 ; TEST PATTERNS:
0025  7379 ;
0025  7380 ;        ADDRESSES TESTED
0025  7381 ;            00000000
0025  7382 ;            00000004
0025  7383 ;            00000008
0025  7384 ;            00000010
0025  7385 ;            00000020
0025  7386 ;            00000040
0025  7387 ;            00000080
0025  7388 ;            00000100
0025  7389 ;            00000200
0025  7390 ;            00000400
0025  7391 ;            00000800
0025  7392 ;            00001000
0025  7393 ;            00002000
0025  7394 ;            00004000
```

```
0025  7395 ;            00008000
0025  7396 ;
0025  7397 ;
0025  7398 ; LOGIC DESCRIPTION:
0025  7399 ;
0025  7400 ;       THIS TEST WILL VERIFY THAT THE ADDRESSING TO THE MEMORY CONTROLLER AND
0025  7401 ;       FIRST ARRAY IS FUNCTIONING PROPERLY.  ANY ERRORS IN THIS TEST SHOULD BE
0025  7402 ;       CONFINED TO MAIN MEMORY.
0025  7403 ;
0025  7404 ;
0025  7405 ; ASSUMPTIONS:
0025  7406 ;
0025  7407 ;       THIS TEST ASSUMES THAT THE PREVIOUS CMI TESTS HAVE RUN SUCCESSFULLY
0025  7408 ;
0025  7409 ;
0025  7410 ; ERROR DESCRIPTION:
0025  7411 ;
0025  7412 ;       EXPECTED CONTENTS OF MAIN MEMORY
0025  7413 ;       RECEIVED CONTENTS OF MAIN MEMORY
0025  7414 ;       LOOP COUNT I
0025  7415 ;       LOOP COUNT J
0025  7416 ;       RTEMP0 (MEMORY LOCATION OF FIRST WRITE; INDEX I)
0025  7417 ;       RTEMP1 (MEMORY LOCATION OF SECOND WRITE; INDEX J)
0025  7418 ;
0025  7419 ;--
0025  7420 ;*******************************************************************************
0025  7421
0025  7422
0025  7423
0025  7424          INITIALIZE                              ;INIT THE CPU
0027  7425          LOADDCS         10$,,0,9                ;CODE TO DISABLE CACHE
002E  7426          FETCH           0                       ;START DCS PROGRAM
0033  7427          BRSTCLK         8                       ;END DCS PROGRAM
0037  7428          EXPUTRAP                                ;POSSIBLE U-TRAP
0039  7429          LOOP            I,1,15                  ;LOOP TO SELECT TEST ADDRESS
0040  7430          LOADDCS         20$,,0,9                ;CODE TO WRITE TEST ADDRESS
0047  7431          LOADREG         WDREG,2$,I,L            ;LOAD TEST ADDRESS TO WDREG
004F  7432          FETCH           0                       ;START DCS PROGRAM
0054  7433          BRSTCLK         8                       ;END DCS PROGRAM
0058  7434          LOADDCS         30$,,0,9                ;CODE FOR ALTERNATE ADDRESS
005F  7435          LOOP            J,1,15                  ;LOOP TO SELECT ALTERNATE ADD
0066  7436          SKIP            1$,ONEQUAL,I,J          ;SKIP ON I = J
006D  7437          LOADREG         WDREG,2$,J,L            ;LOAD ALTERNATE ADDRESS
0075  7438          ERRLOOP                                 ;ERROR LOOP POINT
0077  7439          FETCH           0                       ;START DCS PROGRAM
007C  7440          BRSTCLK         8                       ;END DCS PROGRAM
0080  7441          FETCH           7                       ;RETRY INCASE OF U-TRAP
0085  7442          BRSTCLK         8                       ;END DCS PROGRAM
0089  7443          CMPREG          WHREG,2$,I,L            ;TEST FOR CORRECT DATA
0092  7444          SKIP            1$,NOERROR              ;skip if no address problem
0099  7445          CMPREGMSK       MHREG,4$,,5$,,L         ;check cntrlr type
00A5  7446          IFERROR         2,<<MDL><MAP><MAD><MEC>>,<<MC2><MA1><MA2>>
00B1  7447                                                  ;error printout if L0022
00B1  7448          CMPREGMSK       MHREG,4$,,5$,,L,NE      ;check cntrlr type
00BD  7449          IFERROR         2,<<MDL><MAP><MAD><MEC>>,<<MC0><MC1><MA0><MA1>>
```

```
                                  00CA                                                  ;error printout if L001X
                                  00CA  1$:     ENDLOOP         J                       ;END TEST LOOP J
                                  00CD          ENDLOOP         I                       ;END TEST LOOP I
                                  00D0          SKIP                                    ;GO TO NEXT TEST
                                  00D7
                                  00D7
                                  00D7  BEGIN_TEST_DATA
                                  00D7
                                  00D7      ;-----------------------------------------------------------------------
                                  00D7
                        00000000  00D7  2$:     .LONG   ^X00000000                      ;TEST ADDRESSES
                        00000004  00DB          .LONG   ^X00000004
                        00000008  00DF          .LONG   ^X00000008
                        00000010  00E3          .LONG   ^X00000010
                        00000020  00E7          .LONG   ^X00000020
                        00000040  00EB          .LONG   ^X00000040
                        00000080  00EF          .LONG   ^X00000080
                        00000100  00F3          .LONG   ^X00000100
                        00000200  00F7          .LONG   ^X00000200
                        00000400  00FB          .LONG   ^X00000400
                        00000800  00FF          .LONG   ^X00000800
                        00001000  0103          .LONG   ^X00001000
                        00002000  0107          .LONG   ^X00002000
                        00004000  010B          .LONG   ^X00004000
                        00008000  010F          .LONG   ^X00008000
                        00000000  0113  4$:     .LONG   ^X00000000                      ;dest. for cmpregmsk
                        02000000  0117  5$:     .LONG   ^X02000000                      ;mask for cntrlr type
                                  011B  10$:
U 00, 7700,C800,0364,0300,0470,1801  011B  ;% 00   NOP
U 01, 7700,8800,5BE4,0308,6870,1802  0126  ;% 01   MEMSCAR_R[TEMP2]                        ;TURN CMI ON
U 02, 7700,C800,5BE4,03D8,6070,1803  0131  ;% 02   MEMSCR_R[ZERO]
U 03, 7700,C800,5BE4,030C,6870,1804  013C  ;% 03   MEMSCAR_R[TEMP3]                        ;DISABLE CACHE
U 04, 7700,8800,5BE4,0310,6070,1805  0147  ;% 04   MEMSCR_R[TEMP4]
U 05, 7700,4800,5BE4,0314,4A70,1806  0152  ;% 05   VA_R[TEMP5]                             ;address of CSR2
U 06, 7700,4800,0364,0300,0400,1807  015D  ;% 06   READ.PHY                                ;read CSR2
U 07, 3700,8812,5924,0300,0470,1808  0168  ;% 07   WB_M[MDR],RDM_CMI                       ;MHREG gets CSR2
U 08, 7300,4800,0364,0300,0470,1809  0173  ;% 08   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
                                  017E  20$:
U 00, 7700,C800,0364,0300,0470,1801  017E  ;% 00   NOP
U 01, 7700,C800,0364,0300,0470,1802  0189  ;% 01   NOP
U 02, 7700,4800,0364,0300,0470,1803  0194  ;% 02   NOP
U 03, 5700,8840,0364,0300,4A70,1804  019F  ;% 03   VA_RDM,R[TEMP0]_WB                      ;ADDRESS TO VA & RTEMP0
U 04, 7700,C800,5BE4,0300,5C80,1805  01AA  ;% 04   WRITE.PHY R[TEMP0]                      ;WRITE ADDRESS TO MEMORY
U 05, 7700,4800,0364,0300,0470,1806  01B5  ;% 05   NOP                                     ;ALLOW TIME FOR WRITE
U 06, 7700,C800,0364,0300,0470,1807  01C0  ;% 06   NOP                                     ;...
U 07, 7700,C800,0364,0300,0470,1808  01CB  ;% 07   NOP                                     ;...
U 08, 7300,4800,0364,0300,0470,1809  01D6  ;% 08   NOP,STOP.CLOCK                          ;STOP CPU CLOCK
                                  01E1  30$:
U 00, 7700,C800,0364,0300,0470,1801  01E1  ;% 00   NOP
U 01, 7700,C800,0364,0300,0470,1802  01EC  ;% 01   NOP
U 02, 7700,4800,0364,0300,0470,1803  01F7  ;% 02   NOP
U 03, 5700,C840,0364,0304,4A70,1804  0202  ;% 03   VA_RDM,R[TEMP1]_WB                      ;ADDRESS TO VA & RTEMP1
U 04, 7700,8800,5BE4,0304,5C80,1805  020D  ;% 04   WRITE.PHY R[TEMP1]                      ;WRITE ADDRESS TO MEMORY
U 05, 7700,4800,5BE4,0300,4A70,1806  0218  ;% 05   VA_R[TEMP0]                             ;RELOAD VA TO PREVIOUS ADDRESS
U 06, 7700,4800,0364,0300,0400,1807  0223  ;% 06   READ.PHY                                ;READ THAT ADDRESS
```

```
U 07, 6700,8812,5924,0300,0470,1808  022E  7578 ;% 07   RDM_WB,WB_M[MDR]                    ;MOVE DATA TO RDM
U 08, 7300,4800,0364,0300,0470,1809  0239  7582 ;% 08   NOP,STOP.CLOCK                      ;STOP CPU CLOCK
                                     0244  7586
                                     0244  7587
                                     0244  7588 BEGIN_CPU_DATA
                                     0002  7589
                                     0002  7590 RTEMP_DATA
                                     0001  7591
                           00000000  0001  7592         .LONG   ^X00000000                ;TEST ADDRESS STORAGE
                           00000000  0005  7593         .LONG   ^X00000000                ;TEST ADDRESS STORAGE
                           0E000000  0009  7594         .LONG   ^X0E000000                ;REFERENCE CACHE ONLY ADDRESS
                           06000000  000D  7595         .LONG   ^X06000000                ;CACHE CONTROL REGISTER ADDRESS
                           01000000  0011  7596         .LONG   ^X01000000                ;DISABLE CACHE
                           00F20008  0015  7597         .LONG   ^X00F20008                ;CSR2 address
                                     0019  7598
                                     0019  7599 END_CPU_DATA
                                     0244  7600
                                     0244  7601
                                     0244  7602 END_TEST_DATA
                                     0244
                                     0244       ;-------------------------------------------------------------------------
                                     0244  7603 ENDTEST
```

```
                        .SBTTL  "TEST   080     ECC SINGLE BIT ERROR TEST
                ;*****************************************************************************
                ;++
                ;
                ; FUNCTIONAL DESCRIPTION:
                ;
                ;       THIS TEST WILL VERIFY THAT THE MEMORY CONTROLLER CAN GENERATE AND
                ;       DETECT A SINGLE BIT ERROR.  ALSO TESTED IS THE CORRECTED MEMORY
                ;       DATA INTERRUPT IN THE CPU.
                ;
                ;
                ; TEST ALGORITHM:
                ;
                ;       1.  EXECUTE DCS PROGRAM
                ;               a.  ENABLE CMI
                ;               b.  DISABLE CACHE
                ;               c.  ZERO THE VA REGISTER
                ;               d.  WRITE ZERO'S TO ADDRESS 0 AND 4
                ;               e.  LOAD VA WITH ADDRESS OF CSR1
                ;               f.  SET UP CSR1 FOR SINGLE BIT ERROR
                ;               g.  RELOAD VA WITH ZERO'S
                ;               h.  PERFORM A READ OF LOCATION 0
                ;               i.  MOVE MEMORY DATA TO RDM
                ;       2.  TEST CSTRP REGISTER TO SEE IF WE MICRO TRAPPED
                ;       3.  IF NO ERROR TEST VBUS FOR CORR DATA INT
                ;       4.  IF NO ERROR TEST MEMORY DATA TO SEE IF CORRECTED
                ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
                ;               a.  READ BUS ERROR REGISTER TO THE RDM
                ;       6.  TEST VBUS FOR INT PEND
                ;       7.  IF NO ERROR TEST BUS ERROR REGISTER FOR CORRECTED DATA BIT SET
                ;       8.  IF NO ERROR EXECUTE DCS PROGRAM
                ;               a.  PERFORM A SECOND READ TO ADDRESS 0
                ;               b.  READ BUS ERROR REGISTER TO THE RDM
                ;       9.  TEST BUS ERROR REGISTER FOR LOST ERROR AND CORRECTED DATA
                ;      10.  IF NO ERROR EXECUTE DCS PROGRAM
                ;               a.  CLEAR CSR0 AND CSR1
                ;               b.  CLEAR ^P INTERRUPT
                ;               c.  LOAD PC WITH ZERO TO CAUSE PREFETCH
                ;               d.  READ THE MACHINE CHECK REGISTER TO THE RDM
                ;               e.  CLEAR THE MACHINE CHECK REGISTER
                ;               f.  PERFORM AN IRD1TST TO ALLOW THE INTERRUPT TO OCCURR
                ;      11.  TEST THE VBUS FOR DO SRVC
                ;      12.  IF NO ERROR TEST THE CSTRP FOR CORRECTED DATA INTERRUPT ADDRESS
                ;      13.  IF NO ERROR TEST THE MACHINE CHECK REGISTER FOR BUS ERROR
                ;
                ;
                ; TEST PATTERNS:
                ;
                ;       NONE
                ;
                ;
                ; LOGIC DESCRIPTION:
                ;
                ;       IN A NUTSHELL, WHAT HAPPENS HERE IS; CSR1 IS SET UP TO MAKE THE
                ;       MEMORY CONTROLLER THING THERE IS A SINGLE BIT ERROR IN ADDRESS ZERO.
```

```
001E  7659 ;       AFTER A READ IS PERFORMED ON ADDRESS ZERO THE APPROPRIATE AREAS OF
001E  7660 ;       THE CPU ARE EXAMINED TO SEE IF THE ERROR WAS REPORTED AND DETECTED.
001E  7661 ;       A SECOND READ IS THEN PERFORMED TO SEE IF THE LOST ERROR BIT IS SET
001E  7662 ;       IN THE BUS ERROR REGISTER.  FINALLY THE CORRECTED DATA INTERRUPT IS
001E  7663 ;       ALLOWED TO OCCURR AND THE RESULTS ARE CHECKED.  ONCE THE ERROR IS
001E  7664 ;       REPORTED TO THE CPU ANY FAILURES SHOULD BE CONFINED TO THE PROCESSOR.
001E  7665 ;       ERROR LOOPS WILL HAVE TO LOOP ON THE ENTIRE TEST, SINCE IT IS A SERIES
001E  7666 ;       OF EVENTS THAT LEADS UP TO THE FINAL INTERRUPT.
001E  7667 ;
001E  7668 ;
001E  7669 ; ASSUMPTIONS:
001E  7670 ;
001E  7671 ;       THIS TEST ASSUMES THE CMI IS OPERATIONAL.
001E  7672 ;
001E  7673 ;
001E  7674 ; ERROR DESCRIPTION:
001E  7675 ;
001E  7676 ;       FIRST IFERROR:
001E  7677 ;               EXPECTED CS ADDRESS
001E  7678 ;               RECEIVED CS ADDRESS
001E  7679 ;
001E  7680 ;       SECOND IFERROR:
001E  7681 ;               EXPECTED VBUS BITS
001E  7682 ;               RECEIVED VBUS BITS
001E  7683 ;
001E  7684 ;       THIRD IFERROR:
001E  7685 ;               EXPECTED MAIN MEMORY DATA
001E  7686 ;               RECEIVED MAIN MEMORY DATA
001E  7687 ;
001E  7688 ;       FORTH IFERROR:
001E  7689 ;               EXPECTED VBUS BITS
001E  7690 ;               RECEIVED VBUS BITS
001E  7691 ;                       (NOTE: A FAILURE HERE WOULD INDICATE ECO L0004-TW003
001E  7692 ;                               IS MISSING)
001E  7693 ;
001E  7694 ;       FIFTH IFERROR:
001E  7695 ;               EXPECTED BUS ERROR REGISTER CONTENTS
001E  7696 ;               RECEIVED BUS ERROR REGISTER CONTENTS
001E  7697 ;
001E  7698 ;       SIXTH IFERROR:
001E  7699 ;               EXPECTED BUS ERROR REGISTER CONTENTS
001E  7700 ;               RECEIVED BUS ERROR REGISTER CONTENTS
001E  7701 ;
001E  7702 ;       SEVENTH IFERROR:
001E  7703 ;               EXPECTED VBUS BITS
001E  7704 ;               RECEIVED VBUS BITS
001E  7705 ;
001E  7706 ;       EIGHTH IFERROR:
001E  7707 ;               EXPECTED INTERRUPT ADDRESS
001E  7708 ;               RECEIVED INTERRUPT ADDRESS
001E  7709 ;
001E  7710 ;       NINTH IFERROR:
001E  7711 ;               EXPECTED MACHINE CHECK REGISTER CONTENTS
001E  7712 ;               RECEIVED MACHINE CHECK REGISTER CONTENTS
001E  7713 ;
```

```
                     001E  7714 ;                ON VBUS ERROR: BITS <7:0> = VBUS BIT POSITION
                     001E  7715 ;                               BITS <8> = BIT STATE
                     001E  7716 ;
                     001E  7717 ;--
                     001E  7718 ;*******************************************************************************
                     001E  7719
                     001E  7720
                     001E  7721          INITIALIZE                              ;INIT THE CPU
                     0020  7722          EXPUTRAP                                ;POSSIBLE FAILURE MODE
                     0022  7723          ERRLOOP                                 ;ERROR LOOP POINT
                     0024  7724          FETCH           0                       ;START DCS PROGRAM
                     0029  7725          BRSTCLK         <^X0E>,INH              ;END DCS PROGRAM
                     002D  7726          CMPREGMSK       CSTRP,1$,,2$,,W,        ;TEST FOR U-TRAP
                     0039  7727          IFERROR         ,,<<MC0><MC1><MC2>>     ;TEST U-TRAPPED
                     0041  7728          CMPVBUS         3$                      ;TEST FOR CORR DATA INT
                     0046  7729          IFERROR         ,<<MEC><CML>>,<<MC0><MC1><MC2>>;NO INTERRUPT
                     0050  7730          CMPREG          WHREG,4$,,L,            ;TEST TO SEE IF DATA IS GOOD
                     0059  7731          IFERROR         ,<<MEC>>,<<MC0><MC1><MC2>>      ;INCORRECT DATA
                     0062  7732          FETCH           <^X0F>                  ;START DCS PROGRAM
                     0067  7733          BRSTCLK         <^X12>,INH              ;END DCS PROGRAM
                     006B  7734          CMPVBUS         5$                      ;TEST FOR NOT INT PEND
                     0070  7735          IFERROR         ,<INT>,<UBI>            ;SIGNAL INCORRECT
                     0077  7736          CMPREGMSK       WHREG,8$,,9$,,L,        ;TEST BUS ERROR REGISTER
                     0083  7737          IFERROR         ,<<UTR>>,               ;NOT CORRECT BITS SET
                     0089  7738          FETCH           <^X13>                  ;START DCS PROGRAM
                     008E  7739          BRSTCLK         <^X17>,INH              ;END DCS PROGRAM
                     0092  7740          CMPREGMSK       WHREG,10$,,9$,,L,       ;TEST BUS ERROR REGISTER
                     009E  7741          IFERROR         ,<<UTR>>,               ;NOT LOST ERROR BIT SET
                     00A4  7742          FETCH           <^X18>                  ;START DCS PROGRAM
                     00A9  7743          BRSTCLK         <^X25>,INH              ;END DCS PROGRAM
                     00AD  7744          CMPVBUS         6$                      ;TEST FOR DO SRVC, AND INT PEND
                     00B2  7745          IFERROR         ,<<SAC>>,<<DPM>>        ;SIGNALS INCORRECT
                     00B9  7746          CMPREGMSK       CSTRP,7$,,2$,,W,        ;TEST FOR CORRECT CS ADDRESS
                     00C5  7747          IFERROR         ,<<UTR><INT>>,<<UBI>>   ;NO INTERRUPT ON CS BUS
                     00CD  7748          CMPREGMSK       WHREG,11$,,9$,,L,       ;TEST MACHINE CHECK REGISTER
                     00D9  7749          IFERROR         ,<<UTR>>,               ;NOT BUS ERROR
                     00DF  7750          SKIP                                    ;GO TO NEXT TEST
                     00E6  7751
                     00E6  7752
                     00E6  7753 BEGIN_TEST_DATA
                     00E6               ;-------------------------------------------------------------------------
                     00E6  7754
                180E 00E6  7755 1$:     .WORD   ^X180E                          ;LAST DCS ADDRESS
                3FFF 00E8  7756 2$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                  01 00EA  7757 3$:     .BYTE   ^X1
                     00EB  7758         VBUSDATA        <^X14>,0                ;CORR DATA INT ON VBUS
            00000000 00EC  7759 4$:     .LONG   ^X00000000                      ;EXPECTED DATA FROM MEMORY
                  01 00F0  7760 5$:     .BYTE   ^X1
                     00F1  7761         VBUSDATA        <^X05>,1                ;NOT INT PEND
                  02 00F2  7762 6$:     .BYTE   ^X2
                     00F3  7763         VBUSDATA        <^X05>,0                ;INT PEND ON VBUS
                     00F4  7764         VBUSDATA        <^X19>,0                ;DO SRVC ON VBUS
                003C 00F5  7765 7$:     .WORD   ^X003C                          ;CORR DATA INT VECTOR
            01000000 00F7  7766 8$:     .LONG   ^X01000000                      ;CORR DATA IN BUS ERROR REG
```

```
                    0F000000  00FB  7767 9$:     .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                    03000000  00FF  7768 10$:    .LONG   ^X03000000                      ;LOST ERROR & CORR DATA BITS
                    08000000  0103  7769 11$:    .LONG   ^X08000000                      ;BUS ERROR IN MACHINE CHECK REG
                              0107  7770
                              0107  7771
                              0107  7772 BEGIN_CPU_DATA
                              0002  7773
                              0002  7774 DCS_DATA
                              0001  7775
U 00. 7700,C800,0364,0300,0470,1801   0001  7776 ;* 00   NOP
U 01. 7700,C800,5BE4,0300,6870,1802   000C  7780 ;* 01   MEMSCAR_R[TEMP0]                 ;ENABLE CMI
U 02. 7700,C800,5BE4,03D8,6070,1803   0017  7784 ;* 02   MEMSCR_R[ZERO]                   ;...
U 03. 7700,8800,5BE4,0304,6870,1804   0022  7788 ;* 03   MEMSCAR_R[TEMP1]                 ;DISABLE CACHE
U 04. 7700,8800,5BE4,0308,6070,1805   002D  7792 ;* 04   MEMSCR_R[TEMP2]                  ;...
U 05. 7700,4800,5BE4,03D8,4A70,1806   0038  7796 ;* 05   VA_R[ZERO]                       ;ZERO TO VA REGISTER
U 06. 7700,4800,5BE4,03D8,5C80,1807   0043  7800 ;* 06   WRITE.PHY R[ZERO]               ;WRITE ZERO TO MAIN MEMORY
U 07. 7700,4800,0364,0300,4470,1808   004E  7804 ;* 07   VA_VA+4                         ;INCREMENT VA TO ADDRESS 4
U 08. 7700,C800,5BE4,03D8,5C80,1809   0059  7808 ;* 08   WRITE.PHY R[ZERO]               ;WRITE ZERO TO ADDRESS 4
U 09. 7700,4800,5BE4,030C,4A70,180A   0064  7812 ;* 09   VA_R[TEMP3]                     ;ADDRESS OF CSR1 TO VA REGISTER
U 0A. 7700,0800,5BE4,0310,5C80,180B   006F  7816 ;* 0A   WRITE.PHY R[TEMP4]              ;SET UP FOR SINGLE BIT ERROR
U 0B. 7700,4800,5BE4,03D8,4A70,180C   007A  7820 ;* 0B   VA_R[ZERO]                       ;SET VA BACK TO ZERO
U 0C. 7700,4800,0364,0300,0400,180D   0085  7824 ;* 0C   READ.PHY                        ;READ LOCATION ZERO
U 0D. 6600,8812,5924,0300,0470,180E   0090  7828 ;* 0D   RDM_WB,WB_M[MDR],STROBE.V.BUS   ;READ DATA TO RDM
U 0E. 7300,4800,0364,0300,0470,180F   009B  7832 ;* 0E   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 0F. 7700,C800,0364,0300,0470,1810   00A6  7836 ;* 0F   NOP
U 10. 7700,4800,5BE4,0324,6870,1811   00B1  7840 ;* 10   MEMSCAR_R[TEMP9]                ;READ BUS ERROR REGISTER
U 11. 6600,4800,0364,0300,6470,1812   00BC  7844 ;* 11   RDM_WB,WCTRL/MEMSCR,STROBE.V.BUS ;AND VBUS
U 12. 7300,C800,0364,0300,0470,1813   00C7  7848 ;* 12   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 13. 7700,4800,0364,0300,0470,1814   00D2  7852 ;* 13   NOP
U 14. 7700,4800,0364,0300,0400,1815   00DD  7856 ;* 14   READ.PHY                        ;CAUSE A SECOND ERROR
U 15. 7700,8812,5924,0300,0470,1816   00E8  7860 ;* 15   WB_M[MDR]                       ;...
U 16. 6700,4800,0364,0300,6470,1817   00F3  7864 ;* 16   RDM_WB,WCTRL/MEMSCR             ;READ BUS ERROR REGISTER
U 17. 7300,4800,0364,0300,0470,1818   00FE  7868 ;* 17   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
U 18. 7700,C800,0364,0300,0470,1819   0109  7872 ;* 18   NOP
U 19. 7700,8800,5BE4,031C,4A70,181A   0114  7876 ;* 19   VA_R[TEMP7]                     ;ADDRESS OF CSR0 TO VA
U 1A. 7700,8800,5BE4,0320,5C80,181B   011F  7880 ;* 1A   WRITE.PHY R[TEMP8]              ;CLEAR CSR0
U 1B. 7700,4800,0364,0300,4470,181C   012A  7884 ;* 1B   VA_VA+4                         ;INCREMENT VA TO CSR1
U 1C. 7700,C800,5BE4,03D8,5C80,181D   0135  7888 ;* 1C   WRITE.PHY R[ZERO]               ;CLEAR CSR1
U 1D. 7700,C800,5BE4,0314,2470,181E   0140  7892 ;* 1D   WCTRL/LOADCRAR,WB_R[TEMP5]     ;KILL ^P INTERRUPTS
U 1E. 7700,C800,5BE4,0318,2C70,181F   014B  7896 ;* 1E   WCTRL/CONWRITE,WB_R[TEMP6]     ;...
U 1F. 7700,4800,5BE4,03D8,4870,1820   0156  7900 ;* 1F   PC_R[ZERO]                     ;CAUSE XB'S TO PREFETCH
U 20. 7700,4800,5BE4,0328,6870,1821   0161  7904 ;* 20   MEMSCAR_R[TEMP10]             ;READ MACHINE CHECK REGISTER
U 21. 6700,4800,0364,0300,6470,1822   016C  7908 ;* 21   RDM_WB,WCTRL/MEMSCR           ;...
U 22. 7700,4800,5BE4,03D8,6070,1823   0177  7912 ;* 22   MEMSCR_R[ZERO]                 ;CLEAR MACHINE CHECK REGISTER
U 23. 7700,4800,0364,1700,0470,1824   0182  7916 ;* 23   IRD1TEST                       ;ALLOW THE INIT TO HAPPEN
U 24. 7600,C800,0364,0300,0470,1825   018D  7920 ;* 24   NOP,STROBE.V.BUS              ;LOOK FOR DO SRVC, AND INT PEND
U 25. 7300,C800,0364,0300,0470,1826   0198  7924 ;* 25   NOP,STOP.CLOCK                 ;STOP THE CPU CLOCK
                              01A3  7928
                              01A3  7929 RTEMP_DATA
                              0001  7930
                    0E000000  0001  7931        .LONG   ^X0E000000                      ;REFERENCE CACHE ONLY ADDRESS
                    06000000  0005  7932        .LONG   ^X06000000                      ;CACHE CONTROL REGISTER ADDRESS
                    01000000  0009  7933        .LONG   ^X01000000                      ;DISABLE CACHE
                    00F20004  000D  7934        .LONG   ^X00F20004                      ;ADDRESS OF CSR1
                    1C00003D  0011  7935        .LONG   ^X1C00003D                      ;CAUSE SINGLE BIT ERROR
```

```
00C00000  0015  7936          .LONG   ^X00C00000                      ;ADDRESS FOR CRAR
00400000  0019  7937          .LONG   ^X00400000                      ;DISABLE ^P INTERRUPT
00F20000  001D  7938          .LONG   ^X00F20000                      ;ADDRESS OF CSR0
E0000000  0021  7939          .LONG   ^XE0000000                      ;CLEAR CSR0
09000000  0025  7940          .LONG   ^X09000000                      ;BUS ERROR REGISTER ADDRESS
08000000  0029  7941          .LONG   ^X08000000                      ;MACHINE CHECK REGISTER ADDRESS
          002D  7942
          002D  7943 END_CPU_DATA
          0107  7944
          0107  7945 END_TEST_DATA
          0107
          0107          ;-----------------------------------------------------------------------
          0107  7946 ENDTEST
```

```
001E                    .SBTTL  "TEST   081     ECC DOUBLE BIT ERROR TEST
001E    7948 ;*****************************************************************************
001E    7949 ;++
001E    7950 ;
001E    7951 ; FUNCTIONAL DESCRIPTION:
001E    7952 ;
001E    7953 ;       THIS TEST WILL VERIFY THAT THE MEMORY CONTROLLER CAN GENERATE AND
001E    7954 ;       REPORT A DOUBLE BIT ERROR.  ALSO TESTED IS THE ABILITY OF THE CPU
001E    7955 ;       TO HANDLE SUCH AN ERROR.
001E    7956 ;
001E    7957 ;
001E    7958 ; TEST ALGORITHM:
001E    7959 ;
001E    7960 ;       1.  EXECUTE DCS PROGRAM
001E    7961 ;               a.  ENABLE THE CMI
001E    7962 ;               b.  DISABLE CACHE
001E    7963 ;               c.  ZERO THE VA REGISTER
001E    7964 ;               d.  WRITE ZERO'S TO ADDRESS ZERO
001E    7965 ;               e.  LOAD VA REGISTER WITH ADDRESS OF CSR1
001E    7966 ;               f.  SET UP CSR1 FOR A DOUBLE BIT ERROR
001E    7967 ;               g.  RELOAD VA REGISTER WITH ZERO
001E    7968 ;               h.  PERFORM A READ OF LOCATION ZERO
001E    7969 ;               i.  SOUCRE MDR TO RDM
001E    7970 ;       2.  TEST CS ADDRESS FOR A MACHINE CHECK MICRO TRAP
001E    7971 ;       3.  IF NO ERROR EXECUTE DCS PROGRAM
001E    7972 ;               a.  READ BUS ERROR REGISTER TO THE RDM
001E    7973 ;       4.  TEST BUS ERROR REGISTER FOR UNCORRECTABLE ERROR
001E    7974 ;       5.  IF NO ERROR EXECUTE DCS PROGRAM
001E    7975 ;               a.  READ ADDRESS ZERO A SECOND TIME
001E    7976 ;               b.  READ THE BUS ERROR REGISTER TO THE RDM
001E    7977 ;       6.  TEST BUS ERROR REGISTER FOR LOST ERROR AND UNCORRECTABLE ERROR
001E    7978 ;       7.  IF NO ERROR EXECUTE DCS PROGRAM
001E    7979 ;               a.  LOAD ADDRESS OF CSR0 INTO VA REGISTER
001E    7980 ;               b.  CLEAR CSR0
001E    7981 ;               c.  CLEAR CSR1
001E    7982 ;               d.  READ THE MACHINE CHECK REGISTER TO THE RDM
001E    7983 ;               e.  CLEAR THE MACHINE CHECK REGISTER
001E    7984 ;       8.  TEST MACHINE CHECK REGISTER FOR BUS ERROR
001E    7985 ;
001E    7986 ;
001E    7987 ; TEST PATTERNS:
001E    7988 ;
001E    7989 ;       NONE
001E    7990 ;
001E    7991 ;
001E    7992 ; LOGIC DESCRIPTION:
001E    7993 ;
001E    7994 ;       THIS TEST CAUSES MAIN MEMORY TO CREATE AND REPORT A DOUBLE BIT ERROR.
001E    7995 ;       IF THE MEMORY CONTROLLER IS SUCCESSFULL THEN THE CPU SHOULD
001E    7996 ;       REPORT THE ERROR VIA A MACHINE CHECK MICRO TRAP.  ONCE THE MEMORY
001E    7997 ;       CONTROLLER REPORTS THE ERROR ANY FAILURES SHOULD BE CONFINED TO THE
001E    7998 ;       CPU.
001E    7999 ;
001E    8000 ;
001E    8001 ; ASSUMPTIONS:
```

```
                        001E  8002 ;
                        001E  8003 ;       THIS TEST ASSUMES THE CMI IS OPERATIONAL.
                        001E  8004 ;
                        001E  8005 ;
                        001E  8006 ; ERROR DESCRIPTION:
                        001E  8007 ;
                        001E  8008 ;       FIRST IFERROR:
                        001E  8009 ;              EXPECTED CS ADDRESS
                        001E  8010 ;              RECEIVED CS ADDRESS
                        001E  8011 ;
                        001E  8012 ;       SECOND IFERROR:
                        001E  8013 ;              EXPECTED BUS ERROR REGISTER CONTENTS
                        001E  8014 ;              RECEIVED BUS ERROR REGISTER CONTENTS
                        001E  8015 ;
                        001E  8016 ;       THIRD IFERROR:
                        001E  8017 ;              EXPECTED BUS ERROR REGISTER CONTENTS
                        001E  8018 ;              RECEIVED BUS ERROR REGISTER CONTENTS
                        001E  8019 ;
                        001E  8020 ;       FORTH IFERROR:
                        001E  8021 ;              EXPECTED MACHINE CHECK REGISTER CONTENTS
                        001E  8022 ;              RECEIVED MACHINE CHECK REGISTER CONTENTS
                        001E  8023 ;
                        001E  8024 ;--
                        001E  8025 ;*******************************************************************************
                        001E  8026
                        001E  8027
                        001E  8028         INITIALIZE                                  ;INIT THE CPU
                        0020  8029         EXPUTRAP                                    ;EXPECT A U-TRAP
                        0022  8030         ERRLOOP                                     ;ERROR LOOP POINT
                        0024  8031         FETCH           0                           ;START DCS PROGRAM
                        0029  8032         BRSTCLK         <^X0C>                      ;END DCS PROGRAM
                        002D  8033         CMPREGMSK       CSTRP,1$,,2$,,W,            ;TEST FOR MACHINE CHECK
                        0039  8034         IFERROR         ,<<MEC><CML><UTR>>,<<MC0><MC1><MC2>>
                        0044  8035         FETCH           <^X0D>                      ;START DCS PROGRAM
                        0049  8036         BRSTCLK         <^X10>                      ;END DCS PROGRAM
                        004D  8037         CMPREGMSK       WHREG,3$,,4$,,L,            ;TEST BUS ERROR REGISTER
                        0059  8038         IFERROR         ,<<UTR>>,                   ;BITS NOT RIGHT
                        005F  8039         FETCH           <^X11>                      ;START DCS PROGRAM
                        0064  8040         BRSTCLK         <^X14>                      ;END DCS PROGRAM
                        0068  8041         FETCH           <^X15>                      ;START DCS PROGRAM
                        006D  8042         BRSTCLK         <^X17>                      ;END DCS PROGRAM
                        0071  8043         CMPREGMSK       WHREG,5$,,4$,,L,            ;TEST BUS ERROR REGISTER
                        007D  8044         IFERROR         ,<<UTR>>,                   ;BITS NOT RIGHT
                        0083  8045         FETCH           <^X18>                      ;START DCS PROGRAM
                        0088  8046         BRSTCLK         <^X20>                      ;END DCS PROGRAM
                        008C  8047         CMPREGMSK       WHREG,6$,,4$,,L,            ;TEST MACHINE CHECK REGISTER
                        0098  8048         IFERROR         ,<<UTR>>,                   ;BITS NOT RIGHT
                        009E  8049         SKIP                                        ;GO TO NEXT TEST
                        00A5  8050
                        00A5  8051
                        00A5  8052 BEGIN_TEST_DATA
                        00A5
                        00A5          ;-------------------------------------------------------------------------
                        00A5  8053
                0028    00A5  8054 1$:      .WORD    ^X0028                             ;MACHINE CHECK ADDRESS
```

```
                     3FFF  00A7  8055 2$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
                 04000000  00A9  8056 3$:     .LONG   ^X04000000                      ;UNCORRECTABLE ERROR BIT
                 0F000000  00AD  8057 4$:     .LONG   ^X0F000000                      ;MASK FOR CMPREGMSK PSEUDO OP
                 06000000  00B1  8058 5$:     .LONG   ^X06000000                      ;UNCORRECTABLE & LOST ERROR
                 08000000  00B5  8059 6$:     .LONG   ^X08000000                      ;BUS ERROR BIT
                           00B9  8060
                           00B9  8061
                           00B9  8062 BEGIN_CPU_DATA
                           0002  8063
                           0002  8064 DCS_DATA
                           0001  8065
U 00. 7700.C800.0364,0300.0470,1801  0001  8066 ;* 00   NOP
U 01. 7700.C800.5BE4,0300.6870,1802  000C  8070 ;* 01   MEMSCAR_R[TEMP0]                ;ENABLE CMI
U 02. 7700.C800.5BE4,03D8.6070,1803  0017  8074 ;* 02   MEMSCR_R[ZERO]                  ;...
U 03. 7700.8800.5BE4,0304.6870,1804  0022  8078 ;* 03   MEMSCAR_R[TEMP1]               ;DISABLE CACHE
U 04. 7700.8800.5BE4,0308.6070,1805  002D  8082 ;* 04   MEMSCR_R[TEMP2]                ;...
U 05. 7700.4800.5BE4,03D8.4A70,1806  0038  8086 ;* 05   VA_R[ZERO]                     ;ZERO THE VA REGISTER
U 06. 7700.4800.5BE4,03D8.5C80,1807  0043  8090 ;* 06   WRITE.PHY R[ZERO]             ;WRITE ZERO'S TO MEMORY
U 07. 7700.C800.5BE4,030C.4A70,1808  004E  8094 ;* 07   VA_R[TEMP3]                   ;ADDRESS OF CSR1 TO VA REGISTER
U 08. 7700.8800.5BE4,0310.5C80,1809  0059  8098 ;* 08   WRITE.PHY R[TEMP4]            ;SET UP FOR DOUBLE BIT ERROR
U 09. 7700.4800.5BE4,03D8.4A70,180A  0064  8102 ;* 09   VA_R[ZERO]                     ;SET VA REGISTER BACK TO ZERO
U 0A. 7700.4800.0364,0300.0400,180B  006F  8106 ;* 0A   READ.PHY                      ;READ LOCATION ZERO
U 0B. 6700.0812.5924,0300.0470,180C  007A  8110 ;* 0B   RDM_WB,WB_M[MDR]             ;SHOULD MACHINE CHECK
U 0C. 7300.C800.0364,0300.0470,180D  0085  8114 ;* 0C   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
U 0D. 7700.C800.0364,0300.0470,180E  0090  8118 ;* 0D   NOP
U 0E. 7700.4800.5BE4,0314,6870,180F  009B  8122 ;* 0E   MEMSCAR_R[TEMP5]            ;READ BUS ERROR REGISTER
U 0F. 6700.C800.0364,0300.6470,1810  00A6  8126 ;* 0F   RDM_WB,WCTRL/MEMSCR          ;...
U 10. 7300.4800.0364,0300.0470,1811  00B1  8130 ;* 10   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
U 11. 7700.4800.0364,0300.0470,1812  00BC  8134 ;* 11   NOP
U 12. 7700.4800.0364,0300.0400,1813  00C7  8138 ;* 12   READ.PHY                      ;CAUSE A SECOND MACHINE CHECK
U 13. 6700.0812.5924,0300.0470,1814  00D2  8142 ;* 13   RDM_WB,WB_M[MDR]             ;...
U 14. 7300.C800.0364,0300.0470,1815  00DD  8146 ;* 14   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
U 15. 7700.C800.0364,0300.0470,1816  00E8  8150 ;* 15   NOP
U 16. 6700.4800.0364,0300.6470,1817  00F3  8154 ;* 16   RDM_WB,WCTRL/MEMSCR          ;READ BUS ERROR REGISTER
U 17. 7300.4800.0364,0300.0470,1818  00FE  8158 ;* 17   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
U 18. 7700.C800.0364,0300.0470,1819  0109  8162 ;* 18   NOP
U 19. 7700.C800.5BE4,0318.4A70,181A  0114  8166 ;* 19   VA_R[TEMP6]                   ;ADDRESS OF CSR0 TO VA
U 1A. 7700.8800.5BE4,031C.5C80,181B  011F  8170 ;* 1A   WRITE.PHY R[TEMP7]            ;CLEAR CSR0
U 1B. 7700.4800.0364,0300.4470,181C  012A  8174 ;* 1B   VA_VA+4                      ;INCREMENT VA TO CSR1'S ADDRESS
U 1C. 7700.C800.5BE4,03D8.5C80,181D  0135  8178 ;* 1C   WRITE.PHY R[ZERO]            ;CLEAR CSR1
U 1D. 7700.0800.5BE4,0320.6870,181E  0140  8182 ;* 1D   MEMSCAR_R[TEMP8]            ;READ MACHINE CHECK REGISTER
U 1E. 6700.C800.0364,0300.6470,181F  014B  8186 ;* 1E   RDM_WB,WCTRL/MEMSCR          ;...
U 1F. 7700.4800.5BE4,03D8.6070,1820  0156  8190 ;* 1F   MEMSCR_R[ZERO]              ;CLEAR THE MACHINE CHECK REG
U 20. 7300.4800.0364,0300.0470,1821  0161  8194 ;* 20   NOP,STOP.CLOCK               ;STOP THE CPU CLOCK
                           016C  8198
                           016C  8199 RTEMP_DATA
                           0001  8200
                 0E000000  0001  8201          .LONG   ^X0E000000                    ;REFERENCE CACHE ONLY ADDRESS
                 06000000  0005  8202          .LONG   ^X06000000                    ;CACHE CONTROL REGISTER ADDRESS
                 01000000  0009  8203          .LONG   ^X01000000                    ;DISABLE CACHE
                 00F20004  000D  8204          .LONG   ^X00F20004                    ;ADDRESS OF CSR1
                 1C00003F  0011  8205          .LONG   ^X1C00003F                    ;CAUSE DOUBLE BIT ERROR
                 09000000  0015  8206          .LONG   ^X09000000                    ;BUS ERROR REGISTER ADDRESS
                 00F20000  0019  8207          .LONG   ^X00F20000                    ;ADDRESS OF CSR0
                 E0000000  001D  8208          .LONG   ^XE0000000                    ;CLEAR CSR0
```

```
                08000000  0021  8209          .LONG   ^X08000000                      ;MACHINE CHECK REGISTER ADDRESS
                          0025  8210
                          0025  8211 END_CPU_DATA
                          00B9  8212
                          00B9  8213
                          00B9  8214 END_TEST_DATA
                          00B9
                          00B9        ;-------------------------------------------------------------------------
                          00B9  8215 ENDTEST
```

```
                0000  8217 .END
```

```
$CONTROL_C_CODE= 0000000A
$FILE_NOT_FOUND= 00000009
$STN            = 00000000
$TEST_NUMBER    = 00000082
ACV             = 0000000F
AC_LOW          = 00000080
ADD             = 00000019
ADDR_MATCH_HI   = 00007405
ADDR_MATCH_LO   = 00007404
ADK             = 0000000B
ALK             = 00000000
ALP             = 00000016
ARG_CNT         = 00000099 R     BF
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_HEAD      = 00000000 R     98
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
CML             = 0000000E
COMPOSIT_LENGTH= 00000006
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              = 00000001
D1              = 00000008
D2              = 00000000
DATA_FLAG       = 00000000
DATA_LENGTH     = 00000004
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     C0
DCS_LENGTH      = 00000001
DCS_SCRATCH_ADR= 00000036
DCS_START       = 00001800
DC_LOW          = 00000040
DIAGNOSE_FLAG   = 00000004
DPM             = 00000000
DUM1            = FFFFFFF1
ENDTEST_LOC     = 000000BE R     BF
END_SLICE       = 00000025
END_TEST_100      000002D8 R     1D
END_TEST_101      000001E1 R     23
END_TEST_102      000001E2 R     25
END_TEST_103      000000B5 R     27
END_TEST_104      0000008A R     2D
END_TEST_105      000000CC R     33
END_TEST_106      000002C9 R     39
END_TEST_107      0000030D R     3F
END_TEST_108      000002D4 R     45
END_TEST_109      000000C6 R     4B
END_TEST_110      0000008A R     51
END_TEST_111      00000101 R     57
END_TEST_112      000000E9 R     5D
END_TEST_113      0000006F R     63
END_TEST_114      00000088 R     69
END_TEST_115      0000006B R     6F
END_TEST_116      0000011A R     75
END_TEST_117      000000F6 R     77
END_TEST_118      0000005A R     7D
END_TEST_119      0000018C R     83
END_TEST_120      000000B0 R     89
END_TEST_121      00000152 R     8F
END_TEST_122      0000009D R     95
END_TEST_123      00000101 R     9B
END_TEST_124      000004A0 R     A1
END_TEST_125      0000046C R     A7
END_TEST_126      00000185 R     AD
END_TEST_127      00000244 R     B3
END_TEST_128      00000107 R     B9
END_TEST_129      000000B9 R     BF
END_TEST_95       000000A1 R     02
END_TEST_96       000000BD R     09
END_TEST_97       00000190 R     0F
END_TEST_98       00000103 R     15
END_TEST_99       000001F6 R     17
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          MD_REG_BYTE_2    = 00007416          PB_INIT          = 00000040
IB_KEY           = 00000012          MD_REG_BYTE_3    = 00007417          PB_KEY           = 00000036
INSTR_FLAG       = 00000004          MEC              = 0000001A          PC_KEY           = 00000030
INSTR_KEY        = 00000018          MHREG            = 00007418          PHB              = 00000007
INT              = 00000010          MH_REG_BYTE_0    = 00007418          PRK              = 0000000C
IRD              = 00000005          MH_REG_BYTE_1    = 00007419          PS_KEY           = 0000003E
I_INDEX_FLAG     = 00000000          MH_REG_BYTE_2    = 0000741A          QA_FLAG          = 00000040
I_INIT           = 00000000          MH_REG_BYTE_3    = 0000741B          QA_KEY           = 00000014
J_INDEX_FLAG     = 00000000          MIC              = 00000001          QV_FLAG          = 00000002
J_INIT           = 00000000          MICROWORD        = 0000000A          QV_KEY           = 00000028
K_INDEX_FLAG     = 00000000          MICRO_ADDR_INH   = 00000080          RDCTRL           = 0000741E
K_INIT           = 00000000          MONITOR_SIZE     = 00002C5E          RDM              = 00000004
L                = 00000004          MSQ              = 00000008          RDM_FLAG         = 00000004
LIST_END         = 0000009E R    BF  MT_KEY           = 0000002E          RDM_KEY          = 0000002A
LIST_HEAD        = 0000009A R    BF  N                = 00000001          RDM_LAST         = 00000008
LIST_LENGTH      = 00000004          NER_FLAG         = 00000004          RD_CTRL_REG      = 0000741E
LITERAL          = 00000000          NER_KEY          = 0000000C          REMOTE           = 00000001
LONG             = 00000004          NOERROR          = 00000001          RTEMP_FLAG       = 00000000
LOOP_ERROR_FLAG= 00000020            NOTEQUAL         = 00000003          RTEMP_FLAG1      = 00000000
LOOP_FLAG        = 00000002          ONEQUAL          = 00000002          RTEMP_HEAD       = 00000000 R    C1
LOOP_KEY         = 0000000A          ONERROR          = 00000000          RTEMP_LENGTH     = 00000001
LOST_FLAG        = 00000010          OP_BEGIN_LOOP    = 00000008          RT_KEY           = 0000002C
LOST_KEY         = 0000001A          OP_BURST_CLOCK   = 00000002          SAC              = 00000009
LOW              = 00000000          OP_COMPARE_REG   = 00000004          SA_CLOCK         = 00000080
M                = 0000000A          OP_COMPARE_REGM= 00000006            SA_FLAG          = 00000010
M$TEST_SIZE      = 000007A2          OP_COMPARE_VB    = 00000020          SA_KEY           = 00000010
M$TEST_SIZE_D    = 000003E2          OP_DUMPLOG       = 00000030          SA_START_STOP    = 00000080
MA0              = 00000008          OP_ENABL_STALL   = 00000038          SBE_FLAG         = 00000020
MA1              = 0000000A          OP_ENABL_TRAP    = 0000002E          SBE_KEY          = 00000042
MA2              = 0000000C          OP_END_FILE      = 00000024          SECTION_FLAG     = 00000080
MAD              = 00000013          OP_END_LOOP      = 0000000A          SF_KEY           = 00000040
MAIN32           = 0000741D          OP_END_TEST      = 0000000C          SOMM_FLAG        = 00000001
MAINT_A_TO_CMI   = 00000080          OP_ERRLOG        = 00000032          SOMM_KEY         = 00000006
MAINT_CMI_TO_MH= 00000020            OP_ERROR_LOOP    = 0000000E          SPA              = 00000002
MAINT_DCS_ENABL= 00000004            OP_FETCH         = 00000022          SRK              = 00000001
MAINT_MD_TO_CMI= 00000040            OP_IF_ERROR      = 00000012          SRM              = 00000017
MAINT_REG        = 0000741D          OP_INC_INDEX     = 0000002A          SR_KEY           = 0000003C
MAINT_STB_INH    = 00000002          OP_INIT          = 00000014          STACK_SIZE       = 00000400
MAINT_TRAP_OFF   = 00000001          OP_LOAD_DCS      = 00000000          STATUS           = 00007406
MAINT_WB_TO_WH   = 00000008          OP_LOAD_FIELD    = 00000026          STATUS_REG       = 00007406
MAINT_WD_TO_WB   = 00000010          OP_LOAD_INDEX    = 00000028          STEP_KEY         = 0000001E
MAP              = 00000012          OP_LOAD_REG      = 00000016          STOP_CLOCK       = 00000004
MAREG            = 00007410          OP_MASK          = 00000018          STROBE_CMI       = 00000040
MASTER_HALT_EN   = 00000008          OP_NEW_TEST      = 0000001A          ST_KEY           = 0000001E
MAX_ARGUMENTS    = 00000010          OP_PATTERN_GEN   = 00000010          TEMP             = 00000002 R    BE
MA_KEY           = 00000038          OP_PRINT_N       = 00000036          TEMP1            = 0000006C
MC0              = 00000003          OP_PRINT_S       = 00000034          TEMP2            = 00000192
MC1              = 00000009          OP_SAVE_INDEX    = 0000002C          TEMP3            = 00000002
MC2              = 0000000B          OP_SKIP          = 0000001C          TEMP4            = 00000002 R    BE
MDL              = 0000001B          OP_SUB_TEST      = 0000001E          TEST             = 00000004
MDR              = 00000018          OVERLAY_HEAD     = 00000000 R    C4  TEST_FLAG        = 00000000
MDREG            = 00007414          PARITY_ERROR     = 00000020          TEST_HEAD        = 0000001B R    BF
MD_KEY           = 00000034          PAR_CHK_ENABL    = 00000008          TEST_KEY         = 00000000
MD_REG_BYTE_0    = 00007414          PAR_STOP_ENABL   = 00000010          TEST_LENGTH      = 00000002
MD_REG_BYTE_1    = 00007415          PASS_KEY         = 00000002          TEST_SECT_HEAD   = 00000000 R    BF
```

```
THIS_TEST       = 00000081
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
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 synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| X_0095_D | 00000002 ( 2.) | 01 ( 1.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0095_T | 00000100 ( 256.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0095_GDCS | 000000D2 ( 210.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0095_ERTEMP | 00000009 ( 9.) | 04 ( 4.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0095_FCACHE | 00000001 ( 1.) | 05 ( 5.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0095_HFILL | 00000022 ( 34.) | 06 ( 6.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| DIRECTORY | 00000023 ( 35.) | 07 ( 7.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0096_D | 00000002 ( 2.) | 08 ( 8.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0096_T | 00000100 ( 256.) | 09 ( 9.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0096_GDCS | 00000109 ( 265.) | 0A ( 10.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0096_ERTEMP | 00000015 ( 21.) | 0B ( 11.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0096_FCACHE | 00000001 ( 1.) | 0C ( 12.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0096_HFILL | 0000005F ( 95.) | 0D ( 13.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0097_D | 00000002 ( 2.) | 0E ( 14.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0097_T | 00000200 ( 512.) | 0F ( 15.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0097_GDCS | 00000090 ( 144.) | 10 ( 16.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0097_ERTEMP | 00000011 ( 17.) | 11 ( 17.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0097_FCACHE | 00000001 ( 1.) | 12 ( 18.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0097_HFILL | 0000005C ( 92.) | 13 ( 19.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0098_D | 00000002 ( 2.) | 14 ( 20.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0098_T | 0000017E ( 382.) | 15 ( 21.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0099_D | 00000002 ( 2.) | 16 ( 22.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0099_T | 00000200 ( 512.) | 17 ( 23.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0099_ERTEMP | 00000009 ( 9.) | 18 ( 24.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0099_FCACHE | 00000001 ( 1.) | 19 ( 25.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0099_GDCS | 00000001 ( 1.) | 1A ( 26.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0099_HFILL | 00000073 ( 115.) | 1B ( 27.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0100_D | 00000002 ( 2.) | 1C ( 28.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0100_T | 00000300 ( 768.) | 1D ( 29.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0100_ERTEMP | 00000005 ( 5.) | 1E ( 30.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0100_FCACHE | 00000001 ( 1.) | 1F ( 31.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0100_GDCS | 00000001 ( 1.) | 20 ( 32.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0100_HFILL | 00000077 ( 119.) | 21 ( 33.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0101_D | 00000002 ( 2.) | 22 ( 34.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0101_T | 000001FE ( 510.) | 23 ( 35.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0102_D | 00000002 ( 2.) | 24 ( 36.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0102_T | 000001FE ( 510.) | 25 ( 37.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0103_D | 00000002 ( 2.) | 26 ( 38.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0103_T | 00000100 ( 256.) | 27 ( 39.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0103_GDCS | 00000059 ( 89.) | 28 ( 40.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0103_ERTEMP | 00000015 ( 21.) | 29 ( 41.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0103_FCACHE | 00000001 ( 1.) | 2A ( 42.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0103_HFILL | 0000000F ( 15.) | 2B ( 43.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0104_D | 00000002 ( 2.) | 2C ( 44.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0104_T | 00000100 ( 256.) | 2D ( 45.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0104_GDCS | 00000059 ( 89.) | 2E ( 46.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0104_ERTEMP | 00000011 ( 17.) | 2F ( 47.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0104_FCACHE | 00000001 | ( 1.) | 30 | ( 48.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0104_HFILL | 00000013 | ( 19.) | 31 | ( 49.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0105_D | 00000002 | ( 2.) | 32 | ( 50.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0105_T | 00000100 | ( 256.) | 33 | ( 51.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0105_GDCS | 00000059 | ( 89.) | 34 | ( 52.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0105_ERTEMP | 00000015 | ( 21.) | 35 | ( 53.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0105_FCACHE | 00000001 | ( 1.) | 36 | ( 54.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0105_HFILL | 0000000F | ( 15.) | 37 | ( 55.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0106_D | 00000002 | ( 2.) | 38 | ( 56.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0106_T | 00000300 | ( 768.) | 39 | ( 57.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0106_GDCS | 00000085 | ( 133.) | 3A | ( 58.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0106_ERTEMP | 00000025 | ( 37.) | 3B | ( 59.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0106_FCACHE | 00000001 | ( 1.) | 3C | ( 60.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0106_HFILL | 00000053 | ( 83.) | 3D | ( 61.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0107_D | 00000002 | ( 2.) | 3E | ( 62.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0107_T | 00000380 | ( 896.) | 3F | ( 63.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0107_GDCS | 0000006F | ( 111.) | 40 | ( 64.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0107_ERTEMP | 00000031 | ( 49.) | 41 | ( 65.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0107_FCACHE | 00000001 | ( 1.) | 42 | ( 66.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0107_HFILL | 0000005D | ( 93.) | 43 | ( 67.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0108_D | 00000002 | ( 2.) | 44 | ( 68.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0108_T | 00000300 | ( 768.) | 45 | ( 69.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0108_GDCS | 00000085 | ( 133.) | 46 | ( 70.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0108_ERTEMP | 00000025 | ( 37.) | 47 | ( 71.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0108_FCACHE | 00000001 | ( 1.) | 48 | ( 72.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0108_HFILL | 00000053 | ( 83.) | 49 | ( 73.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0109_D | 00000002 | ( 2.) | 4A | ( 74.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0109_T | 00000100 | ( 256.) | 4B | ( 75.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0109_GDCS | 000000B1 | ( 177.) | 4C | ( 76.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0109_ERTEMP | 00000019 | ( 25.) | 4D | ( 77.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0109_FCACHE | 00000001 | ( 1.) | 4E | ( 78.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0109_HFILL | 00000033 | ( 51.) | 4F | ( 79.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0110_D | 00000002 | ( 2.) | 50 | ( 80.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0110_T | 00000100 | ( 256.) | 51 | ( 81.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0110_GDCS | 000000BC | ( 188.) | 52 | ( 82.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0110_ERTEMP | 00000015 | ( 21.) | 53 | ( 83.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0110_FCACHE | 00000009 | ( 9.) | 54 | ( 84.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0110_HFILL | 00000024 | ( 36.) | 55 | ( 85.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0111_D | 00000002 | ( 2.) | 56 | ( 86.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0111_T | 00000180 | ( 384.) | 57 | ( 87.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0111_GDCS | 000000D2 | ( 210.) | 58 | ( 88.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0111_ERTEMP | 00000021 | ( 33.) | 59 | ( 89.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0111_FCACHE | 00000005 | ( 5.) | 5A | ( 90.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0111_HFILL | 00000006 | ( 6.) | 5B | ( 91.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0112_D | 00000002 | ( 2.) | 5C | ( 92.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0112_T | 00000100 | ( 256.) | 5D | ( 93.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0112_GDCS | 000000D2 | ( 210.) | 5E | ( 94.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0112_ERTEMP | 00000019 | ( 25.) | 5F | ( 95.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0112_FCACHE | 00000005 | ( 5.) | 60 | ( 96.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0112_HFILL | 0000000E | ( 14.) | 61 | ( 97.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0113_D | 00000002 | ( 2.) | 62 | ( 98.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0113_T | 00000080 | ( 128.) | 63 | ( 99.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0113_GDCS | 000000E8 | ( 232.) | 64 | (100.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0113_ERTEMP | 0000001D | ( 29.) | 65 | (101.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0113_FCACHE | 00000009 | ( 9.) | 66 | (102.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |

| | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0113_HFILL | 00000070 | ( 112.) | 67 | (103.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0114_D | 00000002 | ( 2.) | 68 | (104.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0114_T | 00000100 | ( 256.) | 69 | (105.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0114_GDCS | 000000BC | ( 188.) | 6A | (106.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0114_ERTEMP | 00000019 | ( 25.) | 6B | (107.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0114_FCACHE | 00000009 | ( 9.) | 6C | (108.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0114_HFILL | 00000020 | ( 32.) | 6D | (109.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0115_D | 00000002 | ( 2.) | 6E | (110.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0115_T | 00000080 | ( 128.) | 6F | (111.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0115_GDCS | 0000004E | ( 78.) | 70 | (112.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0115_FCACHE | 00000005 | ( 5.) | 71 | (113.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0115_ERTEMP | 00000001 | ( 1.) | 72 | (114.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0115_HFILL | 0000002A | ( 42.) | 73 | (115.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0116_D | 00000002 | ( 2.) | 74 | (116.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0116_T | 0000017E | ( 382.) | 75 | (117.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0117_D | 00000002 | ( 2.) | 76 | (118.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0117_T | 00000100 | ( 256.) | 77 | (119.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0117_GDCS | 00000017 | ( 23.) | 78 | (120.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0117_ERTEMP | 00000001 | ( 1.) | 79 | (121.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0117_FCACHE | 00000001 | ( 1.) | 7A | (122.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0117_HFILL | 00000065 | ( 101.) | 7B | (123.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0118_D | 00000002 | ( 2.) | 7C | (124.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0118_T | 00000080 | ( 128.) | 7D | (125.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0118_GDCS | 00000064 | ( 100.) | 7E | (126.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0118_ERTEMP | 00000011 | ( 17.) | 7F | (127.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0118_FCACHE | 00000001 | ( 1.) | 80 | (128.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0118_HFILL | 00000008 | ( 8.) | 81 | (129.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0119_D | 00000002 | ( 2.) | 82 | (130.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0119_T | 00000200 | ( 512.) | 83 | (131.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0119_ERTEMP | 00000009 | ( 9.) | 84 | (132.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0119_FCACHE | 00000001 | ( 1.) | 85 | (133.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0119_GDCS | 00000001 | ( 1.) | 86 | (134.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0119_HFILL | 00000073 | ( 115.) | 87 | (135.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0120_D | 00000002 | ( 2.) | 88 | (136.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0120_T | 00000100 | ( 256.) | 89 | (137.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0120_GDCS | 00000017 | ( 23.) | 8A | (138.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0120_ERTEMP | 00000001 | ( 1.) | 8B | (139.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0120_FCACHE | 00000001 | ( 1.) | 8C | (140.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0120_HFILL | 00000065 | ( 101.) | 8D | (141.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0121_D | 00000002 | ( 2.) | 8E | (142.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0121_T | 00000180 | ( 384.) | 8F | (143.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0121_GDCS | 00000038 | ( 56.) | 90 | (144.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0121_ERTEMP | 00000015 | ( 21.) | 91 | (145.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0121_FCACHE | 00000001 | ( 1.) | 92 | (146.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0121_HFILL | 00000030 | ( 48.) | 93 | (147.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0122_D | 00000002 | ( 2.) | 94 | (148.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0122_T | 00000100 | ( 256.) | 95 | (149.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0122_GDCS | 00000135 | ( 309.) | 96 | (150.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0122_ERTEMP | 00000021 | ( 33.) | 97 | (151.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0122_FCACHE | 00000009 | ( 9.) | 98 | (152.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0122_HFILL | 0000001F | ( 31.) | 99 | (153.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0123_D | 00000002 | ( 2.) | 9A | (154.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0123_T | 00000180 | ( 384.) | 9B | (155.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0123_GDCS | 0000006F | ( 111.) | 9C | (156.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0123_ERTEMP | 00000015 | ( 21.) | 9D | (157.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0123_FCACHE | 00000001 | ( 1.) | 9E (158.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0123_HFILL | 00000079 | ( 121.) | 9F (159.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0124_D | 00000002 | ( 2.) | A0 (160.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0124_T | 00000500 | ( 1280.) | A1 (161.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0124_GDCS | 00000064 | ( 100.) | A2 (162.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0124_ERTEMP | 00000011 | ( 17.) | A3 (163.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0124_FCACHE | 00000001 | ( 1.) | A4 (164.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0124_HFILL | 00000008 | ( 8.) | A5 (165.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0125_D | 00000002 | ( 2.) | A6 (166.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0125_T | 00000480 | ( 1152.) | A7 (167.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0125_GDCS | 00000064 | ( 100.) | A8 (168.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0125_ERTEMP | 00000011 | ( 17.) | A9 (169.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0125_FCACHE | 00000001 | ( 1.) | AA (170.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0125_HFILL | 00000008 | ( 8.) | AB (171.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0126_D | 00000002 | ( 2.) | AC (172.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0126_T | 00000200 | ( 512.) | AD (173.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0126_ERTEMP | 00000011 | ( 17.) | AE (174.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0126_FCACHE | 00000001 | ( 1.) | AF (175.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0126_GDCS | 00000001 | ( 1.) | B0 (176.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0126_HFILL | 0000006B | ( 107.) | B1 (177.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0127_D | 00000002 | ( 2.) | B2 (178.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0127_T | 00000280 | ( 640.) | B3 (179.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0127_ERTEMP | 00000019 | ( 25.) | B4 (180.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0127_FCACHE | 00000001 | ( 1.) | B5 (181.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0127_GDCS | 00000001 | ( 1.) | B6 (182.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0127_HFILL | 00000063 | ( 99.) | B7 (183.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0128_D | 00000002 | ( 2.) | B8 (184.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0128_T | 00000180 | ( 384.) | B9 (185.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0128_GDCS | 000001A3 | ( 419.) | BA (186.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0128_ERTEMP | 0000002D | ( 45.) | BB (187.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0128_FCACHE | 00000001 | ( 1.) | BC (188.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0128_HFILL | 0000002D | ( 45.) | BD (189.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0129_D | 00000002 | ( 2.) | BE (190.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0129_T | 00000100 | ( 256.) | BF (191.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0129_GDCS | 0000016C | ( 364.) | C0 (192.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0129_ERTEMP | 00000025 | ( 37.) | C1 (193.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0129_FCACHE | 00000001 | ( 1.) | C2 (194.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0129_HFILL | 0000006C | ( 108.) | C3 (195.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |
| X_0130_D | 00000000 | ( 0.) | C4 (196.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT | NOVEC BYTE |

## Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 133 | 00:00:00.37 | 00:00:01.82 |
| Command processing | 864 | 00:00:00.88 | 00:00:01.37 |
| Pass 1 | 3892 | 00:02:18.13 | 00:02:35.52 |
| Symbol table sort | 0 | 00:00:00.94 | 00:00:01.10 |
| Pass 2 | 44 | 00:00:38.32 | 00:00:49.88 |
| Symbol table output | 2 | 00:00:00.22 | 00:00:00.34 |
| Psect synopsis output | 2 | 00:00:00.72 | 00:00:00.94 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 4938 | 00:02:59.58 | 00:03:30.97 |

```
The working set limit was 7500 pages.
1768113 bytes (3454 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 365 non-local and 400 local symbols.
8935 source lines were read in Pass 1, producing 441 object records in Pass 2.
99 pages of virtual memory were used to define 45 macros.
```

+----------------------------+
! Macro library statistics !
+----------------------------+

| Macro library name | Macros defined |
|---|---|
| DISK$UCODPACK00:[VAX750]COMETMAC.MLB;1 | 45 |
| SYS$COMMON:[SYSLIB]STARLET.MLB;1 | 0 |
| TOTALS (all libraries) | 45 |

```
1998 GETS were required to define 45 macros.

There were 0 errors, 2 warnings and 0 information messages, on lines:
 6789 (1)      7147 (1)

MACRO UCOD$0:[VAX750.MIC]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.MIC]ECKAC3/LIS=ECKAC3/OBJ=ECKAC3
```

(1)  24  "MIC Revision History"
(1)  1  ""
(1)  3  "DIAGNOSTIC MICROCODE MACROS"
(1)  98  "  RDM Control File Macros"
(1)  109  "  Diagnostic Macros"
(1)  194  ""
(1)  195  "MODULE GATE ARRAY MAPS"
(1)  197  "  L0001 MULTIPLIER AND GATE ARRAY LOCATION"
(1)  254  "  L0002 GATE ARRAY LOCATION"
(1)  311  "  L0003 GATE ARRAY LOCATION"
(1)  369  "  L0004 GATE ARRAY LOCATION"
(1)  427  "  L0011 & L0016 GATE ARRAY LOCATION"
(1)  485  ""
(1)  490  "EQUATED SYMBOLS
(1)  493  ""
(1)  2  "TEST  082  Array 0 Memory Test Part 1
(1)  720  "TEST  083  Array 0 Memory Test Part 2
(1)  1433  "TEST  084  Array 0 Memory Test Part 3
(1)  2155  "TEST  085  Array 0 Memory Test Part 4
(1)  2882  "TEST  086  Array 1 Memory Test Part 1
(1)  3611  "TEST  087  Array 1 Memory Test Part 2
(1)  4333  "TEST  088  Array 1 Memory Test Part 3
(1)  5058  "TEST  089  Array 1 Memory Test Part 4
(1)  5791  "TEST  08A  Test Main Memory Byte Mask
(1)  5919  "TEST  08B  Test CPU Byte Mask
(1)  6109  "TEST  08C  Test Byte Mask with Write Second Micro Order
(1)  6335  "TEST  08D  Test Read Lock Function
(1)  6716  "TEST  08E  Test Invalidate Cache Logic
(1)  6912  "TEST  08F  Test Bus Field NOP, READ, and READ.PHY Micro Orders to Main Memory
(1)  7081  "TEST  090  Test Bus Field READ.LNG Micro Order using Main Memory
(1)  7248  "TEST  091  Test READ.LNG.MOD Bus Micro Order using Main Memory
(1)  7458  "TEST  092  PC Increment on IRD1
(1)  7595  "TEST  093  Test BUT/WCSENA

```
0000     1          .TITLE  ECKAC VAX-11/750 MIC MICRO DIAGNOSTICS
0000     2          .IDENT  /V08.00/
0000     3 ;
0000     4 ; COPYRIGHT (C) 1980,1982
0000     5 ; DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754
0000     6 ;
0000     7 ; THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE  ONLY ON A SINGLE
0000     8 ; COMPUTER  SYSTEM AND  MAY BE  COPIED ONLY WITH  THE INCLUSION OF THE
0000     9 ; ABOVE COPYRIGHT NOTICE.  THIS SOFTWARE, OR ANY OTHER COPIES THEREOF,
0000    10 ; MAY NOT BE PROVIDED OR  OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
0000    11 ; EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE
0000    12 ; TERMS.  TITLE TO AND  OWNERSHIP OF THE  SOFTWARE  SHALL AT ALL TIMES
0000    13 ; REMAIN IN DEC.
0000    14 ;
0000    15 ; THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000    16 ; AND SHOULD  NOT BE CONSTRUED  AS A COMMITMENT  BY DIGITAL  EQUIPMENT
0000    17 ; CORPORATION.
0000    18 ;
0000    19 ; DEC ASSUMES  NO  RESPONSIBILITY  FOR  THE USE OR  RELIABILITY OF ITS
0000    20 ; SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.
0000    21 ;
0000    22 ;
```

```
0000     24          .SBTTL  "MIC Revision History"
0000     25 ;++
0000     26 ; MIC MICRO-DIAGNOSTIC PACKAGE
0000     27 ;
0000     28 ; REVISION HISTORY
0000     29 ;
0000     30 ; 00-01 Bill Landry and Dave Spain      4-FEB-1980
0000     31 ;       Started file.
0000     32 ;
0000     33 ; 00-02 Bill Landry and Dave Spain      5-FEB-1980
0000     34 ;       Fixed numerous tests to invalidate both groups of TB before using them.
0000     35 ;
0000     36 ; 00-03 Bill Landry                     6-FEB-1980
0000     37 ;       Fixed my typing errors in test 4C.
0000     38 ;
0000     39 ; 00-04 Bill Landry and Dave Spain      7-FEB-1980
0000     40 ;       Skipped subtest 7 - 10 in test 4C. Also added cache data to PC
0000     41 ;       increment and ADD adder tests.
0000     42 ;
0000     43 ; 00-05 Bill Landry     21-FEB-1980
0000     44 ;       Fixed CMPREGMSK psuedo-ops in TB tests.
0000     45 ;
0000     46 ; 00-06 Bill Landry and Dave Spain      27-FEB-1980
0000     47 ;       Changed a CLRTB.VA_VA macro in test 4C to CLRTB.VA_R[]. Also added
0000     48 ;       cache validates to test 12 to prevent random cache parity errors.
0000     49 ;
0000     50 ; 00-07 Bill Landry     3-MAR-1980
0000     51 ;       Added changes for revised SKIP pseudo op.
0000     52 ;
0000     53 ; 00-08 Bill Landry     13-MAR-1980
0000     54 ;       Fixed tests that didnot use a CMPREGMSK pseudo op for testing the
0000     55 ;       CSTRP register.
0000     56 ; 00-09 Bill Landry     25-MAR-1980
0000     57 ;       Added cache tests.
0000     58 ;
0000     59 ; 01-10 Dave Spain       4-APR-1980
0000     60 ;       Added IRD tests to official release version.
0000     61 ;
0000     62 ; 01-11 Bill Landry     4-APR-1980
0000     63 ;       Moved XB, PREFETCH and IRD tests to ECKAB.SRC so manufacturing can
0000     64 ;       run one tape for DPM.
0000     65 ;
0000     66 ; 00-12 Bill Landry     4-APR-1980
0000     67 ;       Let's save version 1.0 for our first SDC release
0000     68 ;
0000     69 ; 00-13 Bill Landry     8-APR-1980
0000     70 ;       Revised cache tests to survive parity error traps.
0000     71 ;
0000     72 ; 00-14 Bill Landry and Dave Spain      15-APR-1980
0000     73 ;       Reassembled using new COMETDEF.
0000     74 ;
0000     75 ; 00-15 Dave Spain      18-APR-1980
0000     76 ;       Changed tests 4F and 50 to call out the ACV and UTR gate arrays instead
0000     77 ;       of the MIC module.
0000     78 ;
```

```
0000     79 ; 00-16 Bill Landry     1-MAY-1980
0000     80 ;       Revised TB tests to loop on error nicely.
0000     81 ;       Deleted redundent TB tests.
0000     82 ;
0000     83 ; 00-17 Bill Landry     14-MAY-1980
0000     84 ;       Added TB parity error tests
0000     85 ;
0000     86 ; 00-18 Bill Landry     10-JUN-1980
0000     87 ;       Added RDM tests to beginning of tape.
0000     88 ;
0000     89 ; 01-00 Bill Landry/Dave Spain  13-JUN-1980
0000     90 ;       Added diagnostic macro file to title section and submitted to
0000     91 ;       RELEASE ENGINEERING.
0000     92 ;
0000     93 ; 01-01 Dave Spain      19-JUN-1980
0000     94 ;       Fixed Prefetch Incrementer test to clear out cache data after
0000     95 ;       every prefetch so that old data from cache will not cause test
0000     96 ;       to mistake bad addresses for good ones.
0000     97 ;
0000     98 ; 01-02 Bill Landry     26-JUN-1980
0000     99 ;       Added tests that check XB trap conditions
0000    100 ;
0000    101 ; 01-03 Bill Landry     30-JUN-1980
0000    102 ;       Added tests that check XB access violations
0000    103 ;
0000    104 ; 01-04 Dave Spain      11-JUL-1980
0000    105 ;       Changed location of BEGINSA pseudo-op to fix signature analysis
0000    106 ;       problem in PC_PC+WBUS test.
0000    107 ;
0000    108 ; 01-05 Bill Landry     8-AUG-1980
0000    109 ;       Added CMI tests in honor of my vacation.
0000    110 ;
0000    111 ; 01-06 Bill Landry     25-AUG-1980
0000    112 ;       Remodeled RDM to CMI tests.
0000    113 ;
0000    114 ; 01-07 Bill Landry     23-SEP-1980
0000    115 ;       Added byte mask testing and improved testing of first memory array.
0000    116 ;
0000    117 ; 02-00 Bill Landry     29-SEP-1980
0000    118 ;       Second release to SDC
0000    119 ;
0000    120 ; 02-01 Bill Landry     5-NOV-1980
0000    121 ;       Added miscellaneous enhancements
0000    122 ;
0000    123 ; 03-00 Bill Landry     17-NOV-1980
0000    124 ;       Third release to SDC
0000    125 ;
0000    126 ; 03-01 Bill Landry     24-NOV-1980
0000    127 ;       Added tests for cache parity logic
0000    128 ;
0000    129 ; 03-02 Bill Landry     30-DEC-1980
0000    130 ;       Separated RDM and MIC tests into two files
0000    131 ;
0000    132 ; 04-00 Bill Landry     05-JAN-1981
0000    133 ;       Submitted to RELEASE ENGINEERING
```

```
0000    134 ;
0000    135 ; 04-01 Bill Landry     14-JAN-1981
0000    136 ;       Revised ECC test to fix bug caused by ECO to INT PEND logic.
0000    137 ;
0000    138 ; 04-02 Bill Landry     27-FEB-1981
0000    139 ;       Udated MDR test to include branching logic.
0000    140 ;       Fixed PC_* tests to fix bug found by field service.
0000    141 ;
0000    142 ; 05-00 Bill Landry     2-MAR-1981
0000    143 ;       Added WRITE BUS ERROR INTERRUPT test
0000    144 ;       Submitted to RELEASE ENGINEERING
0000    145 ;
0000    146 ; 05-01 Bill Landry     18-MAR-1981
0000    147 ;       Made minor changes to various tests as suggested by BT manufacturing.
0000    148 ;
0000    149 ; 05-02 Bill Landry     8-JUN-1981
0000    150 ;       Fixed bug found by Paul Magnet in TB TAG PARITY TESTS.
0000    151 ;
0000    152 ; 05-03 Bill Landry     20-AUG-1981
0000    153 ;       Fixed bugs in memory tests caused by eco to interrupt pending logic.
0000    154 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    155 ;       logic.
0000    156 ;       Replaced all references to COMET with the word CPU.
0000    157 ;       Fixed bug in CACHE TAG MEMORY DUAL ADDRESSING TEST.
0000    158 ;       Submitted to RELEASE ENGINEERING.
0000    159 ;
0000    160 ; 05-04 Bill Landry     4-NOV-1981
0000    161 ;       Moved WCS test from DPM tape to this tape
0000    162 ;       Submitted to RELEASE ENGINEERING.
0000    163 ;
0000    164 ; 05-05 Bill Landry     10-NOV-1981
0000    165 ;       Changed all references of CLRTB.VA_WB to CLRTB.VA_VA.  This fixes a
0000    166 ;       problem caused by plugging in the FPA.
0000    167 ;       Re-released to SDC. Stopped release of 5.4
0000    168 ;
0000    169 ; 05-06 Bill Landry     6-JAN-1982
0000    170 ;       Fixed bugs in memory tests due to memory controller clock switching
0000    171 ;       logic (same as 5.3 above).
0000    172 ;
0000    173 ; 05-07 Bill Landry     23-FEB-1982
0000    174 ;       Fixed more bugs in single bit error tests found by PAUL NATUSH.
0000    175 ;       Same as revision 5.3.
0000    176 ; 05-08 Dave Shull      9-Mar-1982
0000    177 ;       Changed the the IFERROR callouts so when L0011 (MC0) is encountered
0000    178 ;       L0016 (MC1) will be included, when M8728 (MA0) is encountered M8750
0000    179 ;       (MA1) will be included, and when the MAP gate array is encountered
0000    180 ;       MAD gate array will be included.
0000    181 ; 05-09 Dave Shull      10-Mar-1982
0000    182 ;       Added a test to check MS750 controller CSR2 and determine the
0000    183 ;       controller type (ie., L0011 or L0016) and to also print a configuration
0000    184 ;       map of the array's that are installed.
0000    185 ; 06-00 Dave Shull      22-mar-1982
0000    186 ;       Changed the 4 existing memory tests to check for array 0 type then
0000    187 ;       adjust addresses accordingly and test all of the first array versus
0000    188 ;       only testing the first 256kb.
```

```
0000    189 ;       Added 4 new tests that will check to see if array 1 is present and if
0000    190 ;       it is then determine the size and test accordingly.  It looks at both
0000    191 ;       array 0 and array 1, array 0 to determine the starting address of array
0000    192 ;       1 and array 1 to determine the starting and ending address to be tested
0000    193 ; 06-01 Dave Shull      26-Mar-1982
0000    194 ;       Changed READ.SEC MICRO ORDER test so the test could ran in an infinite
0000    195 ;       loop by itself.  Added code to reload Cache everytime the test is
0000    196 ;       restarted in any of the test that are using cache data that could get
0000    197 ;       blown away.
0000    198 ; 06-02 Dave Shull      05-Apr-1982
0000    199 ;       Removed all of the BEGINSA and ENDSA pseudo's
0000    200 ;       Submitted to RELEASE ENGINEERING
0000    201 ; 07-00 Dave Shull      10-May-1982
0000    202 ;       Added new test ( name <Test PCBACK While PC Increment Logic is Active>)
0000    203 ;       to check for side-affects on PCBACK register while incrementin the PC
0000    204 ;       through the execution buffer.
0000    205 ; 07-01 Dave Shull      14-May-1982
0000    206 ;       Changed all CLRTB.VA_VA macro's to prevent translation buffer
0000    207 ;       invalidate failures when manufacturing runs the micro's with an
0000    208 ;       external clock set to fast margins.
0000    209 ; 07-02 Dave Shull      17-May-1982
0000    210 ;       Added coverage in the Read Lock Function Test for the CML gate array.
0000    211 ; 07-03 Dave Shull      14-Jun-1982
0000    212 ;       Changed CMK callouts to CML for new gate array.
0000    213 ; 08-00 Joe Zagarella   11-Feb-1986
0000    214 ;       Enhanced the following tests to be compatible with the new memory
0000    215 ;       controller(L0022) and array card(M7199):
0000    216 ;       Test write bus error interrupt
0000    217 ;       Test for controller type and array configuration
0000    218 ;       Test data integrity for address test
0000    219 ;       Main memory dual addressing test
0000    220 ;       Array 0 memory test (part1 thru part4)
0000    221 ;       Array 1 memory test (part1 thru part4)
0000    222 ;
0000    223 ;--
0000      1         .SBTTL  ""
```

```
0000      3          .SBTTL  "DIAGNOSTIC MICROCODE MACROS"
0000      4 ;++
0000      5 ; Revision History
0000      6 ;
0000      7 ; 28    Dave Spain      22-Jun-82
0000      8 ;       Removed LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000      9 ;       Decided not to use them.
0000     10 ;
0000     11 ; 27    Dave Spain      15-Jun-82
0000     12 ;       Added LOAD FPA SIGN REG and LOAD FPA SIGN REGS macros.
0000     13 ;
0000     14 ; 26    Dave Spain      19-May-82
0000     15 ;       Added M[]_FPA macro.
0000     16 ;
0000     17 ; 25    Rich Lewis      23-Feb-82
0000     18 ;       Added PSL_WB, M[]_R[].OR.NIB0(M), WB_FPA CCR, WB_FPA TCR, Q_R[].ASL.P,
0000     19 ;       RDM_M[].XZ.OR.R[], FPA_WB macros
0000     20 ;
0000     21 ; 24    Dave Spain      02-Feb-82
0000     22 ;       Added FPA_R[]+M[] macro.
0000     23 ;
0000     24 ; 23    Bill Landry     22-OCT-81
0000     25 ;       Added PSL_RDM macro
0000     26 ;
0000     27 ; 22    Dave Spain       5 Oct 81
0000     28 ;       Added RDM_FPA and FPA_M[] FPA_RDM macros.
0000     29 ;
0000     30 ; 21    Bill Landry     29 Sept 81
0000     31 ;       Added FPA macro, FPA_M[] FPA_R[].
0000     32 ;
0000     33 ; 20    Dave Spain      14 Aug 81
0000     34 ;       Added the TESTFPA [] macro.
0000     35 ;
0000     36 ; 19    Dave Spain      18 July 81
0000     37 ;       Added the following FPA macros, FPA_RDM and FPA_R[].
0000     38 ;
0000     39 ; 18    Dave Spain      24 Sept 80
0000     40 ;       Added RDM_Q macro. This macro uses the ALU.
0000     41 ;
0000     42 ; 17    Bill Landry     23 Sept 80
0000     43 ;       Added macro VA_VA-ZLIT0[] for testing memory in decending order.
0000     44 ;
0000     45 ; 16    Dave Spain      27 August 80
0000     46 ;       Changed RDM_Q macro to RDM_Q Q_D. This more accurately reflects macro.
0000     47 ;
0000     48 ; 15    Bill Landry     4 June 80
0000     49 ;       Deleted BUS/PRB.RD.PTE from macro CLRTB.VA_VA and CLRCH.VA_VA to
0000     50 ;       satisfy validity checks.
0000     51 ;
0000     52 ; 14    Dave Spain      23 May 80
0000     53 ;       Added BUS/PRB.RD.PTE to the clear cache and clear TB macros for Mr.
0000     54 ;       Bill.
0000     55 ;
0000     56 ; 13    Dave Spain      14 Apr 80
0000     57 ;       Added SIZE[] or VSIZE/1 to some macros. This allows version 65 of
```

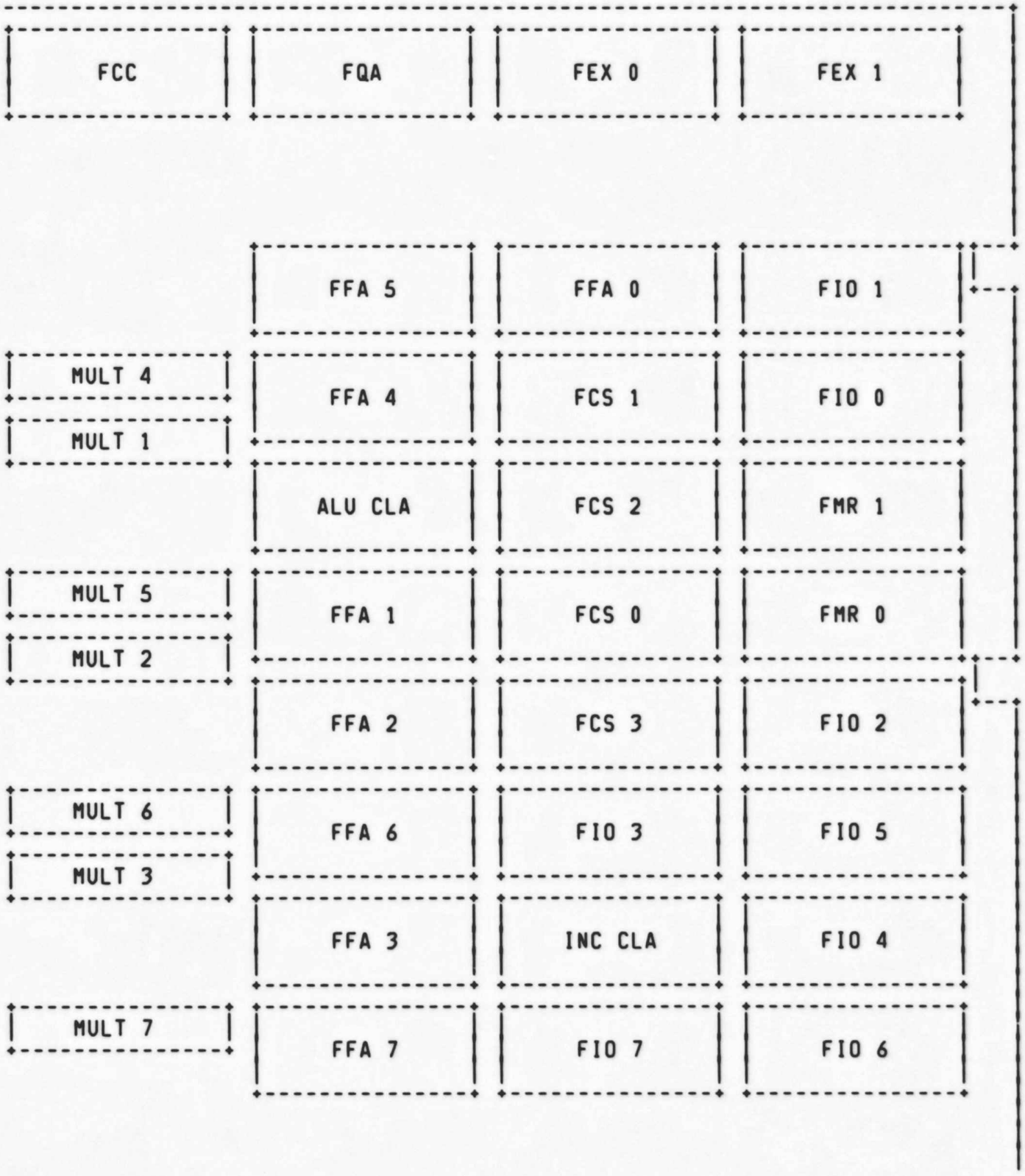
```
0000     58 ;       the COMET micro-code defines to run.
0000     59 ;
0000     60 ; 12    Bill Landry      1 Apr 80
0000     61 ;       Added BUS/PRB.RD micro order to RDM_TB macro to compensate for timing
0000     62 ;       problem in ADD gate array.  This is not an APRIL FOOL!
0000     63 ;
0000     64 ; 11    Dave Spain      27 Mar 80
0000     65 ;       Fixed TB macro's to include MSRC/VA to prevent validity failure.
0000     66 ;
0000     67 ; 10    Bill Landry     19 Mar 80
0000     68 ;       Added Q_R[].RL.P and VA_Q.OR.R[]
0000     69 ;
0000     70 ; 09    Bill Landry     19 Mar 80
0000     71 ;       Deleted TB_R[].RL.P.OR.D
0000     72 ;
0000     73 ; 08    Bill Landry     18 Mar 80
0000     74 ;       Added VA_R[].RL.P
0000     75 ;
0000     76 ; 07    Bill Landry     07 Mar 80
0000     77 ;       Added Q_M[].RL.PTE
0000     78 ;
0000     79 ; 06    Dave Spain      06 Mar 80
0000     80 ;       Changed TB_R[].RL.P macro to TB_R[].RL.P.OR.D.
0000     81 ;
0000     82 ; 05    Dave Spain      03 Mar 80
0000     83 ;       Added TB_R[].RL.P macro for Bill Landry.
0000     84 ;
0000     85 ; 04    Dave Spain      22 Feb 80
0000     86 ;       Added CLRTB.VA_R[] macro for Bill Landry.
0000     87 ;
0000     88 ; 03    Dave Spain      30 Jan 80
0000     89 ;       Added Tom Groetzinger's macro definitions.
0000     90 ;
0000     91 ; 02    Dave Spain      30 Jan 80
0000     92 ;       Added Tony Vezza's macro definitions.
0000     93 ;
0000     94 ; 01    Dave Spain      29 Jan 80
0000     95 ;       Macros initiated.
0000     96 ;--
0000     97
0000     98          .SBTTL  "       RDM Control File Macros"
0000     99
0000    100 ;STROBE.V.BUS            "RDMCF0/V.STROBE"
0000    101 ;DCS.ADDR_0              "RDMCF1/DCS.ADDR.CLR"
0000    102 ;STOP.CLOCK              "RDMCF2/STOP.CPU.CLOCK"
0000    103 ;LATCH.CS.ADDR           "RDMCF3/DIS.STROBE.CS.A"
0000    104 ;RDM_WB                  "RDMCF4/H.REG.FROM.WB"
0000    105 ;WB_RDM                  "RDMCF5/WB.FROM.D.REG"
0000    106 ;RDM_CMI                                "RDMCF6/STROBE.CMI"
0000    107 ;INH.CLOCK.SA            "RDMCF7/INH.SA.CLOCK"
0000    108
0000    109          .SBTTL  "       Diagnostic Macros"
0000    110
0000    111 ;CLOCK.EXT                              "CLKX/XTND"
0000    112 ;CLRCH.VA_RDM                           "WB_RDM,WCTRL/CLRCH.VA_WB,BUS/PRB.RD.PTE"
```

```
                0000   113 ;CLRCH.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRCH.VA_ -
cont> WB"
                0000   114 ;CLRTB.VA_D                     "RSRC/ZERO,MUX/D.R1,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   115 ;                                        BUS/PRB.RD.PTE"
                0000   116 ;CLRTB.VA_RDM                   "WB_RDM,WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   117 ;CLRTB.VA_R[]                   "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/CLRTB.VA_WB,
                0000   118 ;                                        BUS/PRB.RD.PTE"
                0000   119 ;CLRTB.VA_VA                    "RSRC/ZERO,MSRC/VA,MUX/M.R1,ALU/OR,WCTRL/CLRTB.VA_ -
cont> WB"
                0000   120 ;CLRTB.VA_WB                    "WCTRL/CLRTB.VA_WB,BUS/PRB.RD.PTE"
                0000   121 ;D_LONLIT                       "RSRC/LONLIT,ALPCTL/WX_D_R.Q_D"
                0000   122 ;D_M[]_LONLIT                   "D_LONLIT,MSRC/a1"
                0000   123 ;D_VA_0                         "RSRC/ZERO,ROT/ZERO,MUX/R.S,ALU/OR,DQ1/D_WX,
                0000   124 ;                                        WCTRL/VA_WB"
                0000   125 ;FPA_M[] FPA_RDM                        "MSRC/a1,WB_RDM,FPA/FPA_MBUS.FPA_WBUS"
                0000   126 ;FPA_M[] FPA_R[]                       "FPA/FPA_MBUS.FPA_WBUS,MSRC/a1,RSRC/a2,MUX -
cont> /R.S,
                0000   127 ;                                        ROT/ZERO,ALU/OR"
                0000   128 ;FPA_RDM                               "WB_RDM,FPA/FPA_DATA.WBUS"
                0000   129 ;FPA_R[]                               "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,FPA/FPA_D -
cont> ATA.WBUS"
                0000   130 ;FPA_R[]+M[]                    "RSRC/a1,MSRC/a2,MUX/M.R1,ALU/A+B+CI,ALUCI/ZERO,FP -
cont> A/FPA_DATA.WBUS"
                0000   131 ;FPA_WB                         "FPA/FPA_DATA.WBUS"
                0000   132 ;MEMSCAR_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,WCTRL/MEMSCAR_WB"
                0000   133 ;M[]_FPA                               "MSRC/a1,SPW/MLONG,FPA/WBUS_FPA"
                0000   134 ;M[]_LONLIT                    "MSRC/a1,SPW/MLONG,ALPCTL/WX_R.Q_D,RSRC/LONLIT"
                0000   135 ;M[]_R[].OR.NIB0(M)            "MSRC/a1,RSRC/a2,SPW/MLONG,ROT/GETNIB,ALU/OR,
                0000   136 ;                                        MUX/R.S,DQ1/NOP"
                0000   137 ;M[]_RDM                               "WB_RDM,SPW/MLONG,MSRC/a1"
                0000   138 ;PC_PC+RDM                     "WCTRL/PC_PC+WB,WB_RDM"
                0000   139 ;PC_PC+4 MB_XB                 "MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,SIZE[LONG]"
                0000   140 ;PC_PC+4 RDM_XB                "RSRC/ZERO,MSRC/XB.PC_PC+I,ISTRM/ISIZE_DSIZE,
                0000   141 ;                                        MUX/M.R1,ALU/OR,RDM_WB,SIZE[LONG]"
                0000   142 ;PC_RDM                        "WB_RDM,WCTRL/PC_WB"
                0000   143 ;PSL_RDM                               "CCPSL/PSL_WB.CCBR_ALUS,WB_RDM"
                0000   144 ;PSL_WB                        "CCPSL/PSL_WB.CCBR_ALUS"
                0000   145 ;Q_D WX_R                      "ALPCTL/WX_R.Q_D"
                0000   146 ;Q_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_S.Q_R"
                0000   147 ;Q_M[].RL.PTE                  "ALPCTL/WX_Q_S,MSRC/a1,ROT/RL.MM.PTE"
                0000   148 ;Q_R[].AND.Q                   "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/AND"
                0000   149 ;Q_R[].ASL.P                   "ALPCTL/WX_Q_S,RSRC/a1,ROT/ASL.R.P"
                0000   150 ;Q_R[].OR.Q                    "DQ1/Q_WX,RSRC/a1,MUX/R.Q,ALU/OR"
                0000   151 ;Q_R[].RL.P                    "ALPCTL/WX_Q_S,RSRC/a1,ROT/RL.RR.P"
                0000   152 ;RDM_ALUF                      "ALPCTL/WB_ALUF,RDM_WB"
                0000   153 ;RDM_FPA                               "FPA/WBUS_FPA,RDM_WB"
                0000   154 ;RDM_LONLIT                    "WB_LONLIT,RDM_WB"
                0000   155 ;RDM_LONLIT.Q_M                "RSRC/LONLIT,ALPCTL/WX_R.Q_M,RDM_WB"
                0000   156 ;RDM_M[]                               "MSRC/a1,ROT/ZERO,MUX/M.S,ALU/OR,RDM_WB"
                0000   157 ;RDM_M[].OR.R[]                "WB_M[a1].OR.R[a2],RDM_WB"
                0000   158 ;RDM_M[]-R[]                   "WB_M[a1]-R[a2],RDM_WB"
                0000   159 ;RDM_M[].XZ.OR.R[]             "ALU/A+B+CI,MUX/R.S,ROT/XZ.MM,MSRC/a1,RSRC/a2,
                0000   160 ;                                RDM_WB"
                0000   161 ;RDM_PC                        "MSRC/PC,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
                0000   162 ;RDM_PCBACK                    "RSRC/ZERO,MSRC/PCBACK,MUX/M.R1,ALU/OR,RDM_WB"
                0000   163 ;RDM_Q                         "RSRC/ZERO,MUX/R.Q,ALU/OR,RDM_WB"
                0000   164 ;RDM_Q Q_D                     "ALPCTL/WX_Q.Q_D,RDM_WB"
```

```
0000   165 ;RDM_R[]                    "RSRC/a1,ROT/ZERO,MUX/R.S,ALU/OR,RDM_WB"
0000   166 ;RDM_TB                     "WB_M[TB],RDM_WB,BUS/PRB.RD,SIZE[LONG]"
0000   167 ;RDM_VA                     "MSRC/VA,RSRC/ZERO,MUX/M.R1,ALU/OR,RDM_WB"
```

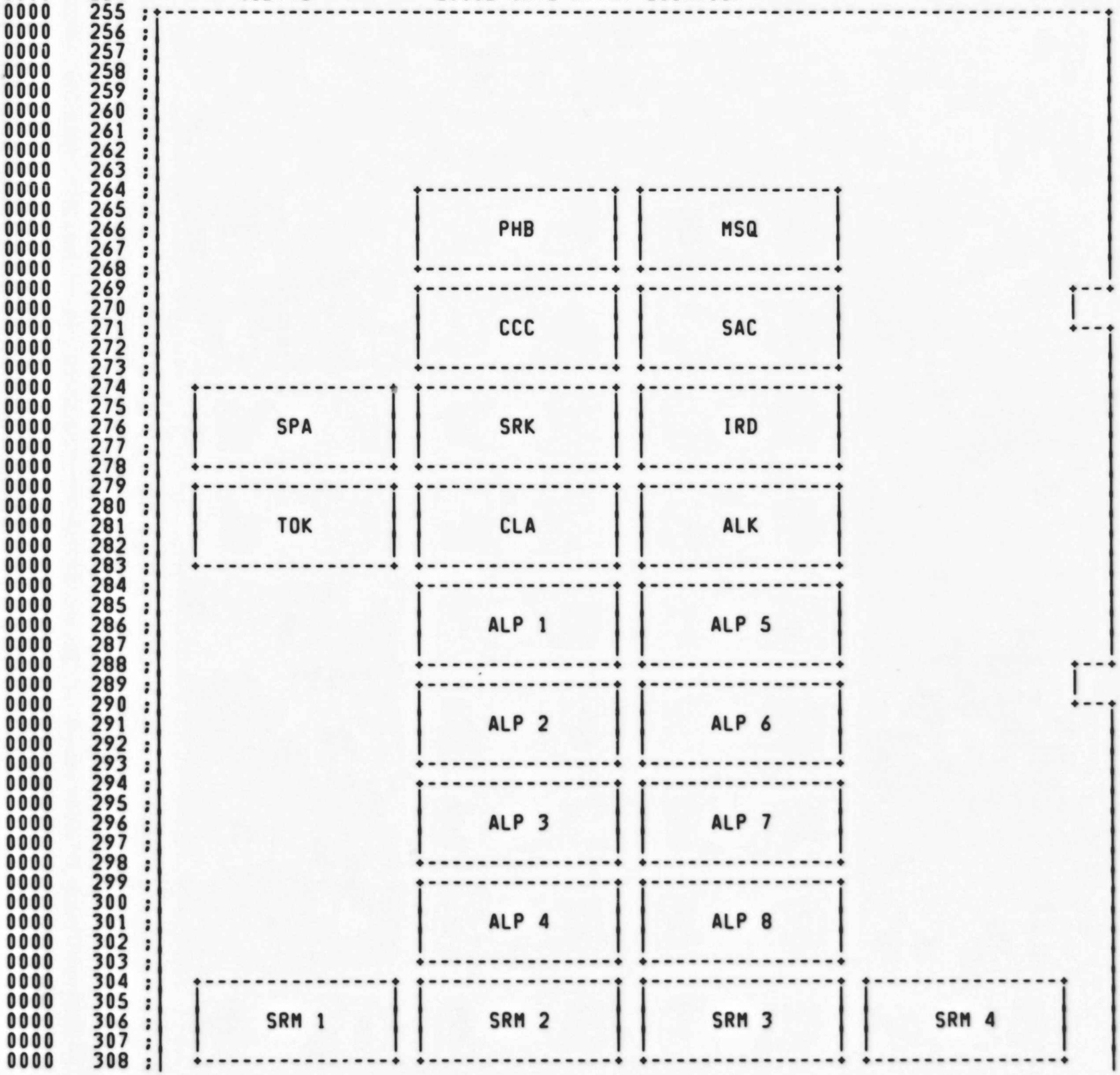
```
                0000   168 ;RNUM_LONLIT                    "ALPCTL/WX_D_R.Q_M,MSRC/RNUM_WBUS,RSRC/LONLIT"
                0000   169 ;R[]_RDM                                "WB_RDM,SPW/RLONG,RSRC/a1"
                0000   170 ;R[].DT_Q                        "RSRC/a1,D_Q Q_D,SPW/RSIZE,VSIZE/1"
                0000   171 ;R[]_VA                         "RSRC/a1,MSRC/VA,ROT/ZERO,MUX/M.S,ALU/OR,SPW/RLONG"
                0000   172 ;R[]_WB                         "RSRC/a1,SPW/RLONG"
                0000   173 ;TB_RDM                         "WCTRL/TB_WB,WB_RDM,MSRC/VA"
                0000   174 ;TB_WB                          "WCTRL/TB_WB,MSRC/VA"
                0000   175 ;TESTFPA []                     "TESTFPA/a1"
                0000   176 ;VA_PC+LONLIT+I.PC_PC+I         "RSRC/LONLIT,MSRC/XB.PC_PC+I,ROT/ZERO,MUX/R.S,ALU/ -
cont> OR,
                0000   177 ;                                       ISTRM/ISIZE_DSIZE,WCTRL/VA_PC+I+W.PC_PC+I,
                0000   178 ;                                       SIZE[LONG]"
                0000   179 ;VA_Q.OR.R[]                    "WCTRL/VA_WB,RSRC/a1,ALU/OR,MUX/R.Q"
                0000   180 ;VA_RDM                         "WB_RDM,WCTRL/VA_WB"
                0000   181 ;VA_R[].RL.P                    "ALPCTL/WX_S,ROT/RL.RR.P,RSRC/a1,WCTRL/VA_WB"
                0000   182 ;VA_VA-ZLIT0[]                  "WCTRL/VA_WB,LIT/LITRL,LITRL/a1,ROT/ZLIT0,MSRC/VA,
                0000   183 ;                                       MUX/M.S,ALU/A-B-CI"
                0000   184 ;VA_WB                          "WCTRL/VA_WB"
                0000   185 ;WB_FPA CCR                     "FPA/WBUS_FPA.CC"
                0000   186 ;WB_FPA TCR                     "WCTRL/FPTCR"
                0000   187 ;WB_LONLIT                      "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000   188 ;WB_LONLIT.Q_M                  "RSRC/LONLIT,ALPCTL/WX_R.Q_M"
                0000   189 ;WB_M[]+R[]                     "ALU/A+B+CI,ALUCI/ZERO,MUX/M.R1,MSRC/a1,RSRC/a2"
                0000   190 ;WB_R[].Q_D                     "RSRC/a1,ALPCTL/WX_R.Q_D"
                0000   191 ;WDR_R[]                                "WCTRL/WDR_WB,ALU/OR,MUX/R.S,RSRC/a1,ROT/Z -
cont> ERO"
                0000   192 ;WRITE.LONG R[]                 "BUS/WRITE.LNG,WCTRL/WDR_WB.UR,RSRC/a1,ROT/ZERO,
                0000   193 ;                                       MUX/R.S,ALU/OR,SIZE[LONG]"
                0000   194         .SBTTL  ""
                0000   195         .SBTTL  "MODULE GATE ARRAY MAPS"
```

```
                        .SBTTL  "       L0001 MULTIPLIER AND GATE ARRAY LOCATION"
```



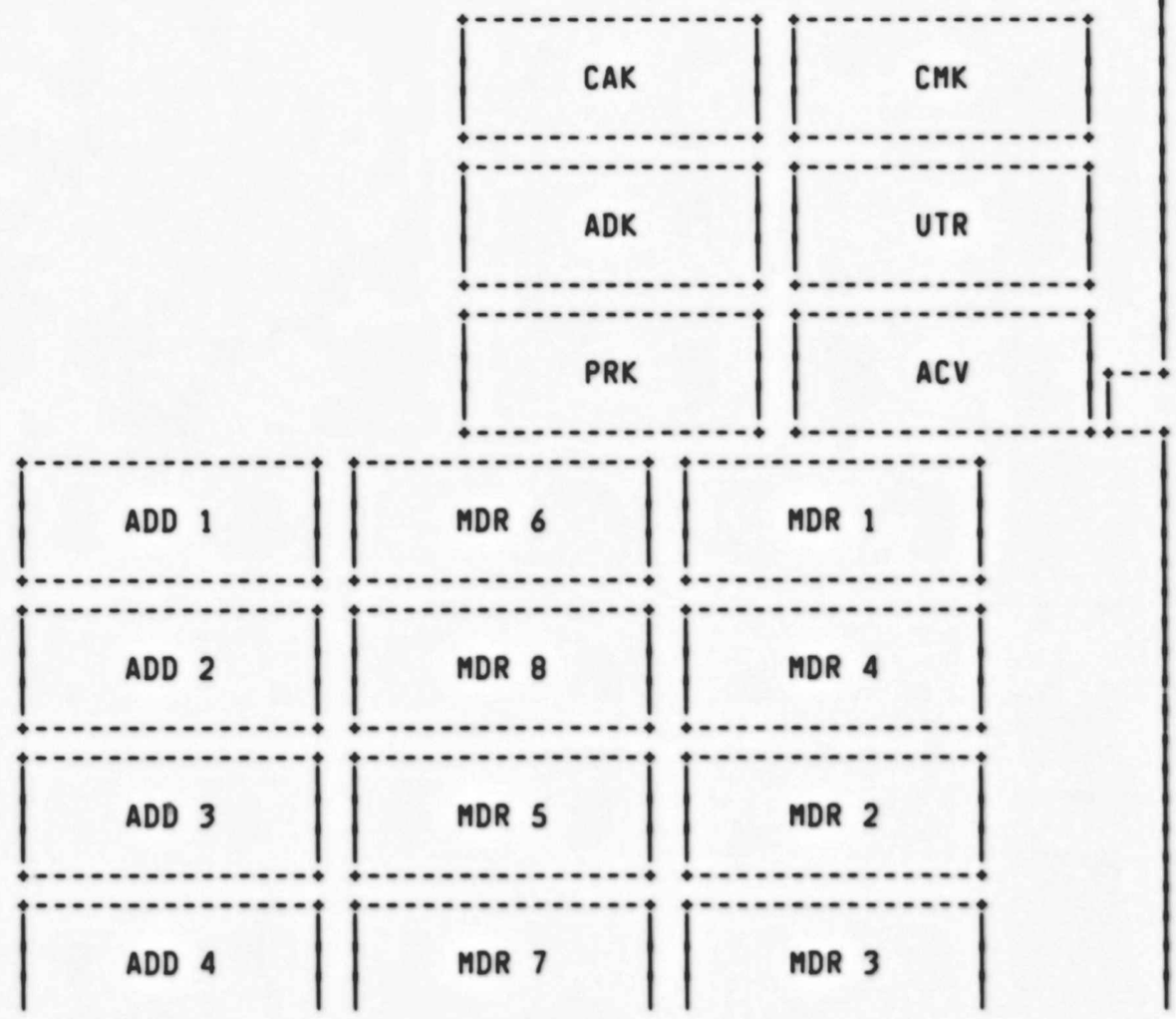
```
                0000   252 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0002 GATE ARRAY LOCATION"
```



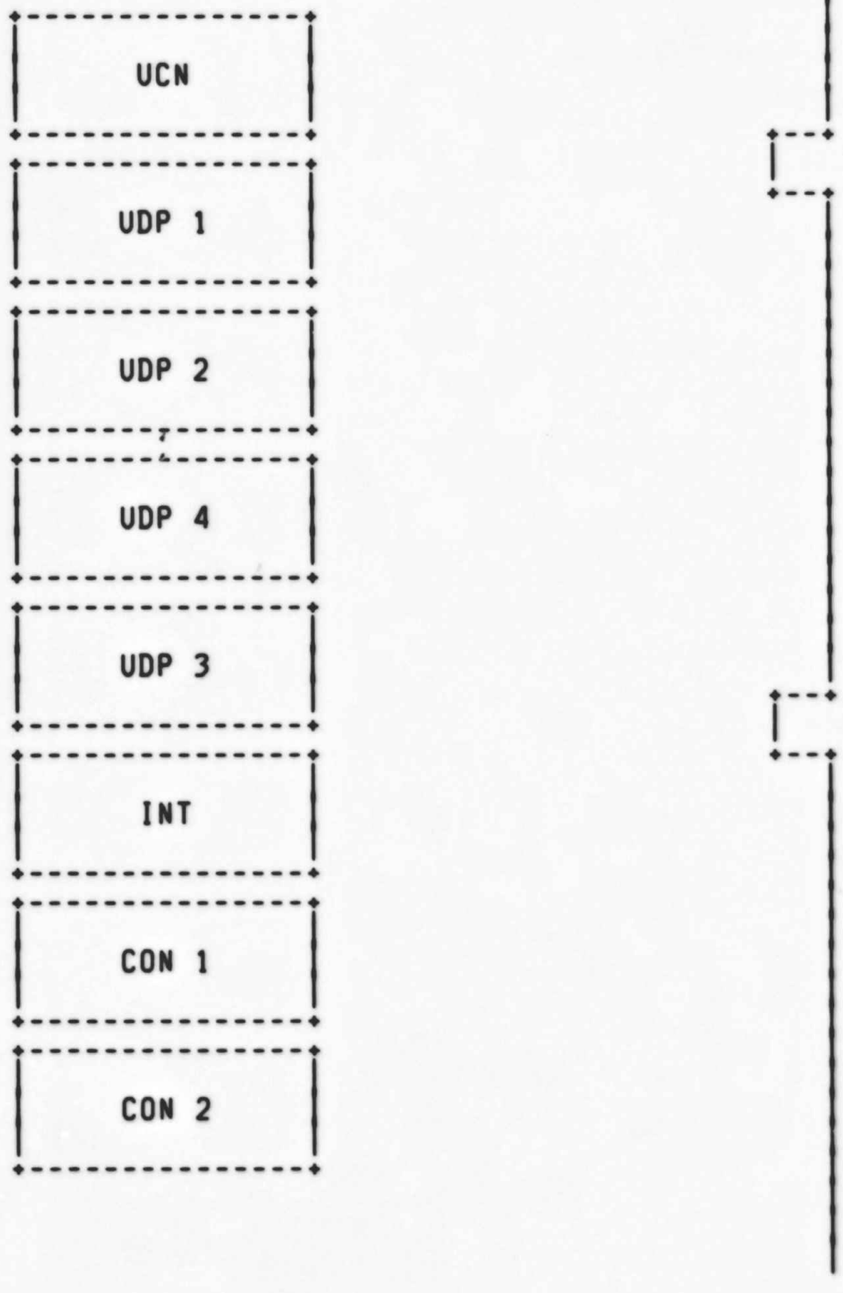
```
                0000   309 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0003 GATE ARRAY LOCATION"
```



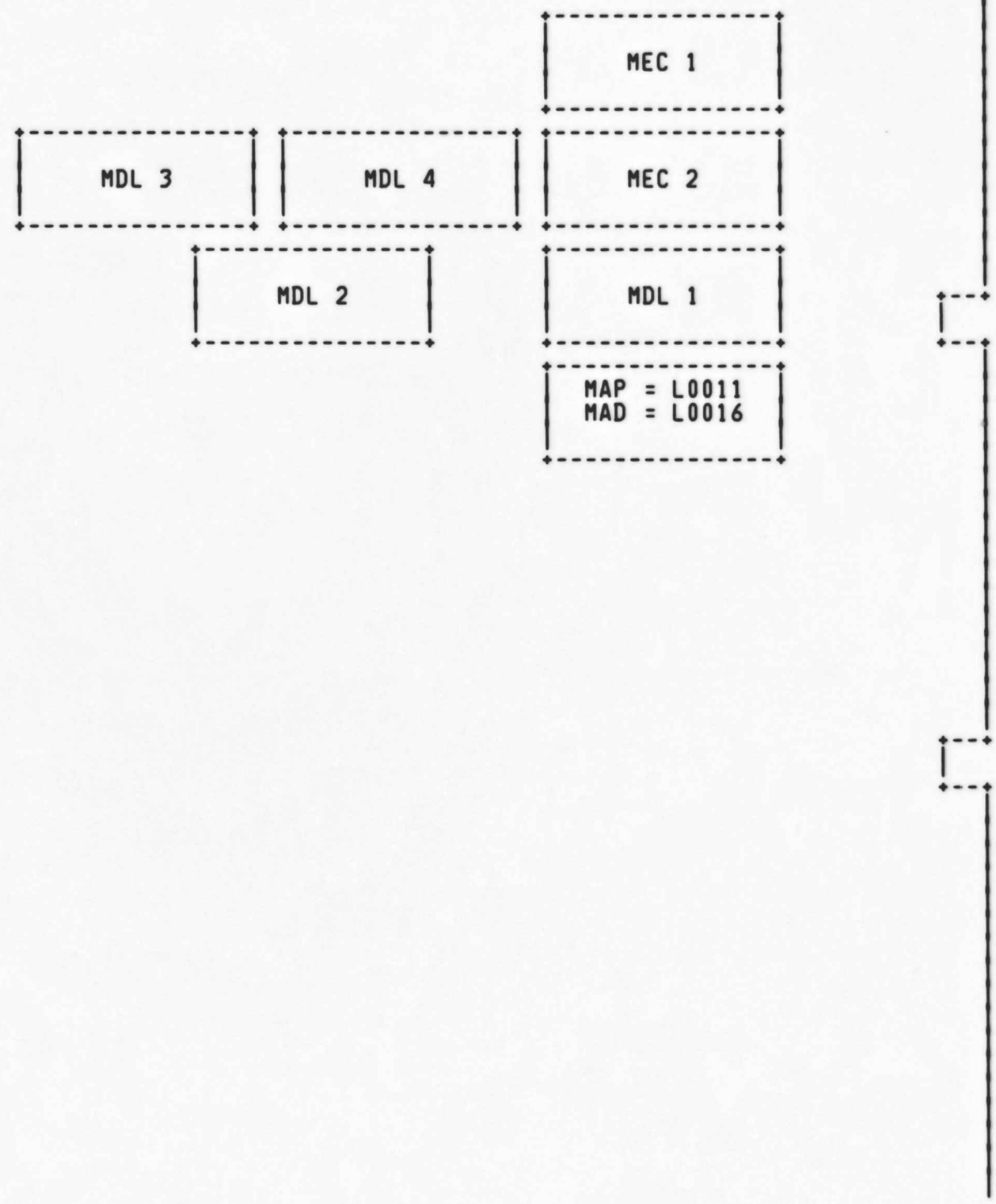
```
                0000   366 ;|              +---------------+ +---------------+ +---------------+          |
                0000   367 ;+-----------------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0004 GATE ARRAY LOCATION"
```



```
0000   424 ;|                                                                    |
0000   425 ;+-------------------------------------------------------------------+
```

```
                        .SBTTL  "       L0011 & L0016 GATE ARRAY LOCATION"
```



```
0000   482 ;|                                                                                |
0000   483 ;+--------------------------------------------------------------------------------+
```

```
                     0000   485          .SBTTL  ""
                     0000   486          .LIST   MC
                     0000   487          .LIBRARY        /UCOD$0:[VAX750]COMETMAC/
                     0000   488          .LIST   ME
                     0000   489          .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
00002C5E  0000                MONITOR_SIZE    =       ^X2C5E
00000400  0000                STACK_SIZE      =       1024
000007A2  0000                M$TEST_SIZE     =       14336 - STACK_SIZE - MONITOR_SIZE
          0000                                          ; MAXIMUM SIZE OF A TEST OVERLAY
000003E2  0000                M$TEST_SIZE_D   = M$TEST_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    491   .NLIST  ME
          0000    492   .LIST   MC
          0000    493   .SBTTL  ""
          0000    494   .NLIST  MC      ; SHUT-OFF LISTING FOR FIRST 'NEWTEST' CALL
```

```
                        .SBTTL  "TEST   082     Array 0 Memory Test Part 1
                      3 ;*******************************************************************************
                      4 ;++
                      5 ;
                      6 ; FUNCTIONAL DESCRIPTION:
                      7 ;
                      8 ;       This test will check the first array card writing first 0's then
                      9 ;       1's in ascending order.
                     10 ;
                     11 ;
                     12 ; TEST ALGORITHM:
                     13 ;
                     14 ;       1.  Announce that we are testing Array 0
                     15 ;       2.  Load u-code to initialize test conditions
                     16 ;       3.  Execute dcs program
                     17 ;               a.  Disable the cmi
                     18 ;               b.  Enable the cache
                     19 ;               c.  Set va to zero
                     20 ;               d.  Validate cache addresses 0 and 4
                     21 ;               e.  Load the pc to cause prefetch of addresses 0 and 4
                     22 ;               f.  Disable control P interrupts
                     23 ;               g.  Enable cmi and disable cache
                     24 ;               h.  Clear memory CSR0
                     25 ;               i.  Set memory csr1 to allow single bit errors
                     26 ;               j.  Read CSR2
                     27 ;               k.  Perform IRD1TEST to free interrupt pending logic
                     28 ;               L.  Set va to zero
                     29 ;       4.  Check low bits 1:0 of CSR2
                     30 ;               a.  If this is a 256kb array then mask in last address of
                     31 ;                   3FFFC then continue
                     32 ;               b.  If this is a 1mb array then mask in last address of FFFFC,
                     33 ;                   and change two micro words so they use RTEMP13 when
                     34 ;                   checking to see if at last address
                     35 ;               c.  If this is a 4mb array, then mask in last address of 3FFFFC,
                     36 ;                   change RTEMP13 to 3FFFFC, and change two microwords so
                     37 ;                   they use RTEMP13 when checking to see if at last address
                     38 ;       5.  Load index I with 2 which will be used for comparison of index K
                     39 ;       6.  Load u-code to write zeros to main memory
                     40 ;       7.  Load the first u-inst.
                     41 ;       8.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
                     42 ;           error section as follows:
                     43 ;               a.  First pass is from the writing of zero's throughout memory
                     44 ;               b.  Second pass is from the read-write-read throughout memory
                     45 ;       9.  Check to see if this is first pass or second pass, if first pass
                     46 ;           then goto 112$, if second pass then goto 110$
                     47 ;      10.  Set-up the rdm to watch for u-traps
                     48 ;      11.  Set the u-match halt for 1801
                     49 ;      12.  Start the cpu clock
                     50 ;               a.  Save the va in the rdm WH register
                     51 ;               b.  Write zero to the address in the va
                     52 ;               c.  Test va to see if at end of first array card
                     53 ;               d.  If not increment va and return to dcs address 0
                     54 ;      13.  Set-up watchdog timer with loop J to monitor above u-code
                     55 ;      14.  If loop J expires, stop the clock and force an error printout
                     56 ;      15.  If clock stops normally (u-match at 1801); test WH register for
```

```
001F     57 ;              end of first array
001F     58 ;        16.   If no error test cs address for not a u-trap
001F     59 ;        17.   Increment loop K for second pass
001F     60 ;        18.   Execute dcs code
001F     61 ;                   a.  Set va to zero
001F     62 ;        19.   Load the test u-code
001F     63 ;        20.   Load the first u-inst.
001F     64 ;        21.   Set-up the rdm to watch for u-traps
001F     65 ;        22.   Set the u-match to 1801
001F     66 ;        23.   Start the cpu clock
001F     67 ;                   a.  Save va in rdm WH register
001F     68 ;                   b.  Read address specified by va
001F     69 ;                   c.  Move data to RTEMP0
001F     70 ;                   d.  Test for a corrected data interrupt
001F     71 ;                   e.  Test that data is correct
001F     72 ;                   f.  Write one to the same location
001F     73 ;                   g.  Read location back
001F     74 ;                   h.  Save data in RTEMP0
001F     75 ;                   i.  Test for a corrected data interrupt
001F     76 ;                   j.  Test that data is correct
001F     77 ;                   k.  Test to see if at end of first array
001F     78 ;                   l.  Of not end of first array, increment the va by 4
001F     79 ;                   m.  Return to dcs address 0
001F     80 ;        24.   Set-up watch-dog timer with loop J to monitor above u-code
001F     81 ;        25.   If loop J expires, stop the clock and force an error printout
001F     82 ;        26.   If clock stops normally (u-match at 1801) test WH register
001F     83 ;              for end of first array.
001F     84 ;        27.   Test for machine checks (double bit error in memory)
001F     85 ;        28.   If not machine check test for incorrect data on first read
001F     86 ;        29.   If not bad data test for incorrect data on second read
001F     87 ;        30.   Test for corrected data interrupts
001F     88 ;        31.   Service the corrected data interrupt with program at dcs address 14
001F     89 ;                   a.  Save va in RTEMP7
001F     90 ;                   b.  Disable cmi, enable cache
001F     91 ;                   c.  Set va to zero
001F     92 ;                   d.  Validate cache address 0 and 4
001F     93 ;                   e.  Load pc to cause prefetch of addresses 0 and 4
001F     94 ;                   f.  Disable ^p interrupts
001F     95 ;                   g.  Read memory CSR0 to the rdm MHREG
001F     96 ;                   h.  Clear memory CSR0
001F     97 ;                   i.  Clear any interrupts pending
001F     98 ;                   j.  Perform an IRD1TEST to free the interrupt pending logic
001F     99 ;                   k.  Reload va with RTEMP7
001F    100 ;        32.   Continue running test micro code from step 17.
001F    101 ;        33.   Test for the end of the first array.
001F    102 ;        34.   When last address of array is found the test will end on endloop K
001F    103 ;              being expired
001F    104 ;        35.   The balance of the pseudo ops handle the various error conditions
001F    105 ;              (see error description)
001F    106 ;
001F    107 ;
001F    108 ; TEST PATTERNS:
001F    109 ;
001F    110 ;         First pass:  init memory to 0's
001F    111 ;        Second pass:  read 0's; write 1's; read 1's
```

```
                        001F    112 ;
                        001F    113 ;
                        001F    114 ; LOGIC DESCRIPTION:
                        001F    115 ;
                        001F    116 ;       My advice on this test is forget trying to troubleshoot with it.  It
                        001F    117 ;       is far to complex to be a usefull troubleshooting tool.  Rather, just
                        001F    118 ;       use the error printout to array and/or module swap till you find the
                        001F    119 ;       problem.  The first array is throughly tested only after running
                        001F    120 ;       all four tests in this series; together and in order.
                        001F    121 ;
                        001F    122 ;
                        001F    123 ; ASSUMPTIONS:
                        001F    124 ;
                        001F    125 ;       This test assumes the cmi is operational.
                        001F    126 ;
                        001F    127 ;
                        001F    128 ; ERROR DESCRIPTION:
                        001F    129 ;
                        001F    130 ;       First IFERROR:
                        001F    131 ;
                        001F    132 ;       EXPECTED LAST ADDRESS OF FIRST ARRAY
                        001F    133 ;       RECEIVED LAST ADDRESS
                        001F    134 ;       LOOP COUNT I
                        001F    135 ;       LOOP COUNT J
                        001F    136 ;       LOOP COUNT K
                        001F    137 ;       (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
                        001F    138 ;
                        001F    139 ;       Second IFERROR:
                        001F    140 ;
                        001F    141 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
                        001F    142 ;       MAIN MEMORY ADDRESS OF DATA ERROR
                        001F    143 ;       LOOP COUNT I
                        001F    144 ;       LOOP COUNT J
                        001F    145 ;       LOOP COUNT K
                        001F    146 ;       RECEIVED DATA (S/B 0'S OR 1'S)
                        001F    147 ;       (INDICATES ECC IS PROBABLE NOT WORKING)
                        001F    148 ;
                        001F    149 ;       Third IFERROR:
                        001F    150 ;
                        001F    151 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
                        001F    152 ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
                        001F    153 ;       LOOP COUNT I
                        001F    154 ;       LOOP COUNT J
                        001F    155 ;       LOOP COUNT K
                        001F    156 ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
                        001F    157 ;
                        001F    158 ;--
                        001F    159 ;******************************************************************************
                        001F    160
                        001F    161 ;------------------------------------------------------------------------------
                        001F    162 ;
                        001F    163 ; Initialize code
                        001F    164 ;
                        001F    165 ;------------------------------------------------------------------------------
                        001F    166
```

```
001F   167          INITIALIZE                                  ;init the cpu
0021   168          PRINT_S          30$                        ;announce testing array 0
0025   169          LOADDCS          40$,,0,<^X1C>              ;setup micro code
002C   170          FETCH            0                          ;start dcs program
0031   171          BRSTCLK          <^X1B>,INH                 ;end dcs program
0035   172          CMPREGMSK        WHREG,23$,,18$,,B          ;check array 0
0041   173          SKIP             100$,NOERROR               ;skip if 4mb module
0048   174          CMPREGMSK        WHREG,21$,,18$,,B,NE       ;check array 0 type
0054   175          SKIP             102$,ONERROR               ;skip if 1mb module
005B   176          MASK             4$,,19$,,L                 ;mask in address 3FFFC
0064   177          SKIP             108$                       ;go test
006B   178 ;
006B   179 ; This section will set up for a 4mb module
006B   180 ;
006B   181 100$:    MASK             4$,,24$,,L                 ;mask in address 3FFFFC
0074   182          LOADDCS          45$,,0,4                   ;R[TEMP13]_3FFFFC
007B   183          FETCH            0                          ;start dcs
0080   184          BRSTCLK          3                          ;end dcs
0084   185          SKIP             103$                       ;change uword to use R[TEMP13]
008B   186 ;
008B   187 ; This section will set-up for a 1mb module versus a 256kb module
008B   188 ;
008B   189 102$:    MASK             4$,,20$,,L                 ;mask in address FFFFC
0094   190 103$:    LDFIELD          22$,,B,55$,34,39           ;change rsrc fld for R[TEMP13]
00A5   191          LDFIELD          22$,,B,65$,34,39           ;change rsrc fld for R[TEMP13]
00B6   192
00B6   193 ;-------------------------------------------------------------------------------
00B6   194 ;
00B6   195 ; Write memory with 0's
00B6   196 ;
00B6   197 ;-------------------------------------------------------------------------------
00B6   198
00B6   199 108$:    LDINDEX          I,2                        ;dummy index I gets 2
00BD   200          LOADDCS          50$,,0,<^X0B>              ;test u-code
00C4   201          FETCH            0                          ;load first u-inst.
00C9   202          LOOP             K,1,2                      ;set index K for two passes
00D0   203          SKIP             110$,ONEQUAL,I,K           ;skip on second pass
00D7   204          SKIP             112$                       ;first pass, goto rdm set-up
00DE   205
00DE   206 ;-------------------------------------------------------------------------------
00DE   207 ;
00DE   208 ; Read the 0's from the previous write, check for valid data, write 1's, check
00DE   209 ; for valid data
00DE   210 ;
00DE   211 ;-------------------------------------------------------------------------------
00DE   212
00DE   213 110$:    FETCH            8                          ;reset va to zero
00E3   214          BRSTCLK          <^X0A>,INH                 ;...
00E7   215          LOADDCS          60$,,0,<^X2F>              ;test u-code
00EE   216          FETCH            0                          ;load first u-inst.
00F3   217 ;
00F3   218 ; This section is used for both the initial writing of 0's (first pass) then
00F3   219 ; for the read 0's-write 1's-read 1's (second pass)
00F3   220 ;
00F3   221 112$:    LOADREG          DCSCR,9$,,B                ;insure clock is off
```

```
00FB   222          LOADREG         CSAMR,2$,,W                ;load match address
0103   223          LOADREG         RDCTRL,7$,,B               ;clear master halt enable
010B   224          LOADREG         RDCTRL,1$,,B               ;watch for u-traps
0113   225          LOADREG         DCSCR,3$,,B                ;start the clock
011B   226          LOOP            J,1,8192                   ;watchdog timing loop
0122   227          CMPREGMSK       CLKDCS,7$,,8$,,B,          ;test for clock stopped
012E   228          SKIP            114$,ONERROR               ;skip when clock stops
0135   229          ENDLOOP         J                          ;end timing loop
0138   230          LOADREG         DCSCR,9$,,B                ;stop the clock
0140   231          CMPREG          WHREG,4$,,L,               ;force an error
0149   232          IFERROR         ,,<<MC0><MC1><MC2>>        ;program lost
0151   233          SKIP
0158   234
0158   235 ;-----------------------------------------------------------------------------
0158   236 ;
0158   237 ; Error evaluation code
0158   238 ;
0158   239 ;-----------------------------------------------------------------------------
0158   240
0158   241 114$:    CMPREGMSK       CSTRP,10$,,6$,,W,NE        ;test for machine check
0164   242          SKIP            140$,ONERROR               ;handle machine check
016B   243          CMPREGMSK       CLKDCS,11$,,12$,,B,NE      ;test for data errors
0177   244          SKIP            130$,ONERROR               ;incorrect data
017E   245          CMPREGMSK       CLKDCS,13$,,12$,,B,NE      ;test for data errors
018A   246          SKIP            130$,ONERROR               ;incorrect data
0191   247          CMPREGMSK       CLKDCS,14$,,12$,,B,NE      ;test for single bit error on 0
019D   248          SKIP            120$,ONERROR               ;log error
01A4   249          CMPREGMSK       CLKDCS,16$,,12$,,B,NE      ;test for single bit error on 1
01B0   250          SKIP            115$,ONERROR               ;log error
01B7   251          CMPREG          WHREG,4$,,L,               ;test for last address
01C0   252          SKIP            150$,NOERROR               ;go to next test
01C7   253          CMPREGMSK       CLKDCS,17$,,12$,,B         ;force and error
01D3   254          IFERROR                                    ;i think we're lost
01D8   255          SKIP                                       ;end test due to error
01DF   256 ;
01DF   257 ; Come here if the Second read caused corrected read data to be returned
01DF   258 ;
01DF   259 115$:    FETCH           <^X14>                     ;start dcs program
01E4   260          BRSTCLK         <^X2E>                     ;end dcs program
01E8   261          ERRLOG                                     ;log the error
01EA   262          SKIP            150$,ONERROR               ;quit after 64 errors
01F1   263          FETCH           <^X10>                     ;set up to continue testing
01F6   264          SKIP            112$                       ;test more memory
01FD   265 ;
01FD   266 ; Come here if the First read caused corrected read data to be returned
01FD   267 ;
01FD   268 120$:    FETCH           <^X14>                     ;start dcs program
0202   269          BRSTCLK         <^X2E>                     ;end dcs program
0206   270          ERRLOG                                     ;log the error
0208   271          SKIP            150$,ONERROR               ;quit after 64 errors
020F   272          FETCH           <^X07>                     ;set up to continue testing
0214   273          SKIP            112$                       ;test more memory
021B   274 ;
021B   275 ; Come here if the data read is not the data written
021B   276 ;
```

```
                                  021B  130$:   CMPREG          WHREG,15$,,L,          ;force an error
                                  0224          IFERROR         1,<<MEC><MDL>>,-
                                  0224          <<MC0><MC1><MC2><MA0><MA1><MA2>>       ;print va and data
                                  0231          SKIP                                   ;end test because of error
                                  0238  ;
                                  0238  ; Come here if there was a Machine check (Read Data Substitute)
                                  0238  ;
                                  0238  140$:   CMPREG          WHREG,15$,,L,          ;force an error
                                  0241          IFERROR         ,<<MEC><MDL>>,-
                                  0241          <<MC0><MC1><MC2><MA0><MA1><MA2>>       ;print va of machine check
                                  024E  ;-----------------------------------------------------------------------------
                                  024E  ;
                                  024E  ; This is where we exit the test if index K is equal to 2
                                  024E  ;
                                  024E  ;-----------------------------------------------------------------------------
                                  024E
                                  024E  150$:   ENDLOOP         K                      ;end pass loop
                                  0251          SKIP                                   ;go to next test
                                  0258
                                  0258
                                  0258  BEGIN_TEST_DATA
                                  0258
                                  0258          ;-----------------------------------------------------------------
                                  0258
                            28    0258  1$:     .BYTE   ^X28                           ;set to watch for u-traps
                          1801    0259  2$:     .WORD   ^X1801                         ;match address
                            11    025B  3$:     .BYTE   ^X11                           ;run clock
                      FFFFFFFF    025C  4$:     .LONG   ^XFFFFFFFF                     ;last address for first array
                                  0260                                                 ;will be masked here
                          3FFF    0260  6$:     .WORD   ^X3FFF                         ;mask for cmpregmsk pseudo op
                            00    0262  7$:     .BYTE   ^X00                           ;watch for clock stop
                            C0    0263  8$:     .BYTE   ^XC0                           ;mask for cmpregmsk pseudo op
                            60    0264  9$:     .BYTE   ^X60                           ;stop clock manually
                          0028    0265  10$:    .WORD   ^X0028                         ;machine check u-trap
                            09    0267  11$:    .BYTE   ^X09                           ;dcs halt on data error
                            3F    0268  12$:    .BYTE   ^X3F                           ;mask for cmpregmsk pseudo op
                            12    0269  13$:    .BYTE   ^X12                           ;dcs halt on data error
                            08    026A  14$:    .BYTE   ^X08                           ;halt single bit error on 0's
                      0F0F0F0F    026B  15$:    .LONG   ^X0F0F0F0F                     ;compare data to force errors
                            11    026F  16$:    .BYTE   ^X11                           ;halt single bit error on 1's
                            FF    0270  17$:    .BYTE   ^XFF                           ;used to force errors
                            03    0271  18$:    .BYTE   ^X03                           ;array 0 mask
                      0003FFFC    0272  19$:    .LONG   ^X0003FFFC                     ;256kb array 0 last address
                      000FFFFC    0276  20$:    .LONG   ^X000FFFFC                     ;1mb array 0 last address
                            02    027A  21$:    .BYTE   ^X02                           ;array 0 1mb (csr2 bits 0:1)
                            0D    027B  22$:    .BYTE   ^X0D                           ;rsrc = temp13
                            01    027C  23$:    .BYTE   ^X01                           ;array 0=4mb
                      003FFFFC    027D  24$:    .LONG   ^X003FFFFC                     ;4mb array 0 last address
41 20 47 4E 49 54 53 45 54 0A 0D 00'  0281  30$:    .ASCIC  <13><10>/TESTING ARRAY 0/<13><10>
          0A 0D 30 20 59 41 52 52     028D
                            13    0281
                                  0295                                                 ;message for start of test
                                  0295  40$:
U 00, 7700,C800,0364,0300,0470,1801  0295  ;% 00   NOP
U 01, 7700,8800,5BE4,0304,6870,1802  02A0  ;% 01   MEMSCAR_R[TEMP1]                    ;disable cmi
```

```
U 02. 7700.C800.5BE4.030C.6070.1803  02AB   334 ;* 02   MEMSCR_R[TEMP3]                                  ;...
U 03. 7700.8800.5BE4.0308.6870.1804  02B6   338 ;* 03   MEMSCAR_R[TEMP2]                                 ;turn on cache
U 04. 7700.C800.5BE4.03D8.6070.1805  02C1   342 ;* 04   MEMSCR_R[ZERO]                                   ;...
U 05. 7700.4800.5BE4.03D8.4A70.1806  02CC   346 ;* 05   VA_R[ZERO]                                      ;zero the va
U 06. 7700.4800.5BE4.03D8.5C80.1807  02D7   350 ;* 06   WRITE.PHY R[ZERO]                                ;validate cache address 0
U 07. 7700.C800.5BE4.03D8.5470.1808  02E2   354 ;* 07   WDR_UNROT(R[ZERO])                               ;...
U 08. 7700.C800.0364.0300.4470.1809  02ED   358 ;* 08   VA_VA+4                                         ;increment va by 4
U 09. 7700.C800.5BE4.03D8.5C80.180A  02F8   362 ;* 09   WRITE.PHY R[ZERO]                                ;validate cache address 4
U 0A. 7700.C800.5BE4.03D8.5470.180B  0303   366 ;* 0A   WDR_UNROT(R[ZERO])
U 0B. 7700.C800.5BE4.03D8.4870.180C  030E   370 ;* 0B   PC_R[ZERO]                                     ;cause prefetch
U 0C. 7700.0800.5BE4.032C.2470.180D  0319   374 ;* 0C   WCTRL/LOADCRAR,WB_R[TEMP11]                     ;disable ^p interrupts
U 0D. 7700.C800.5BE4.0330.2C70.180E  0324   378 ;* 0D   WCTRL/CONWRITE,WB_R[TEMP12]                    ;...
U 0E. 7700.0800.5BE4.0304.6870.180F  032F   382 ;* 0E   MEMSCAR_R[TEMP1]                                ;turn on cmi
U 0F. 7700.4800.5BE4.03D8.6070.1810  033A   386 ;* 0F   MEMSCR_R[ZERO]                                   ;...
U 10. 7700.0800.5BE4.0308.6870.1811  0345   390 ;* 10   MEMSCAR_R[TEMP2]                                ;disable cache
U 11. 7700.C800.5BE4.030C.6070.1812  0350   394 ;* 11   MEMSCR_R[TEMP3]                                  ;...
U 12. 7700.8800.5BE4.0320.4A70.1813  035B   398 ;* 12   VA_R[TEMP8]                                     ;CSR0 address to va
U 13. 7700.C800.5BE4.0324.5C80.1814  0366   402 ;* 13   WRITE.PHY R[TEMP9]                               ;clear CSR0
U 14. 7700.4800.0364.0300.4470.1815  0371   406 ;* 14   VA_VA+4                                         ;CSR1 address to va
U 15. 7700.4800.5BE4.0328.5C80.1816  037C   410 ;* 15   WRITE.PHY R[TEMP10]                              ;allow single bit errors
U 16. 7700.C800.0364.0300.4470.1817  0387   414 ;* 16   VA_VA+4                                         ;CSR2 address to va
U 17. 7700.C800.0364.0300.0400.1818  0392   418 ;* 17   READ.PHY                                        ;read CSR2
U 18. 6700.8812.5924.0300.0470.1819  039D   422 ;* 18   RDM_WB,WB_M[MDR]                                ;WHREG gets contents of CSR2
U 19. 7700.C800.0364.1700.0470.181A  03A8   426 ;* 19   IRDITEST                                       ;clear int pending line
U 1A. 7700.4800.5BE4.03D8.4A70.181B  03B3   430 ;* 1A   VA_R[ZERO]                                     ;zero the va register
U 1B. 7300.C800.0364.0300.0470.181C  03BE   434 ;* 1B   NOP,STOP.CLOCK                                 ;stop the cpu clock
                                     03C9   438 45$:
U 00. 7700.B800.7FE0.0001.8470.1801  03C9   439 ;* 00   LONLIT_[003FFFFC]                               ;last addr for 4mb
U 01. 7700.8868.5BE4.03D4.0470.1802  03D4   443 ;* 01   M[TEMP8]_R[LONLIT]                              ;
U 02. 7700.4848.5924.0334.0470.1803  03DF   447 ;* 02   R[TEMP13]_M[TEMP8]                              ;RTEMP13 = last addr 3FFFFC
U 03. 7300.C800.0364.0300.0470.1804  03EA   451 ;* 03   NOP,STOP.CLOCK                                 ;stop cpu clock
                                     03F5   455 50$:
U 00. 7700.C800.0364.0300.0470.1802  03F5   456 ;* 00   NOP,NEXT/1802
U 01. 6700.881B.5924.0300.0470.1802  0400   460 ;* 01   RDM_WB,WB_M[VA]                                 ;save va in rdm
U 02. 7700.8800.5BE4.0310.5C80.1803  040B   464 ;* 02   WRITE.PHY R[TEMP4]                               ;write zeros to memory
U 03. 7700.C800.0364.0300.0470.1804  0416   468 ;* 03   NOP                                            ;allow time for write
U 04. 7700.4800.0364.0300.0470.1805  0421   472 ;* 04   NOP                                            ;...
                                     042C   476 55$:
U 05. 7700.481B.0034.A318.0470.1800  042C   477 ;* 05   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     0437   481                                                        ;test for end of first array
U 06. 7700.4800.0364.0300.4470.1807  0437   482 ;* 06   VA_VA+4                                         ;increment va register
U 07. 7500.C800.0364.0300.0470.1808  0442   486 ;* 07   NOP,DCS.ADDR_0                                 ;return to address 0
U 08. 7700.4800.0364.0300.0470.1809  044D   490 ;* 08   NOP
U 09. 7700.4800.5BE4.03D8.4A70.180A  0458   494 ;* 09   VA_R[ZERO]                                     ;set va to zero
U 0A. 7300.C800.0364.0300.0470.180B  0463   498 ;* 0A   NOP,STOP.CLOCK                                 ;stop the cpu clock
                                     046E   502 60$:
U 00. 7700.C800.0364.0300.0470.1802  046E   503 ;* 00   NOP,NEXT/1802                                   ;allow time for memory
U 01. 7700.C800.0364.0300.0470.1802  0479   507 ;* 01   NOP                                            ;clock to start up.
U 02. 7700.4800.0364.0300.0470.1803  0484   511 ;* 02   NOP                                            ;...
U 03. 6700.881B.5924.0300.0470.1804  048F   515 ;* 03   RDM_WB,WB_M[VA]                                 ;save va in rdm
U 04. 7700.C800.0364.0300.0400.1805  049A   519 ;* 04   READ.PHY                                        ;read pattern back
U 05. 7700.4852.5924.0300.0470.1806  04A5   523 ;* 05   R[TEMP0]_M[MDR]                                ;save data in RTEMP0
U 06. 7700.C800.0364.AF00.0470.1801  04B0   527 ;* 06   BUT/INT-TIMSERV,NEXT/1801                      ;test for corrected data
U 07. 7700.4800.0034.A710.0470.1800  04BB   531 ;* 07   WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                     04C6   535                                                        ;test for good data
```

```
U 08. 7700.4800.0364.0300.0470.1809  04C6   536 ;% 08   NOP                                                ;allow time for memory clock
U 09. 7700.4800.0364.0300.0470.180A  04D1   540 ;% 09   NOP                                                ;to start up after single bit
U 0A. 7700.C800.0364.0300.0470.180B  04DC   544 ;% 0A   NOP                                                ;errors.
U 0B. 7700.4800.0364.0300.0470.180C  04E7   548 ;% 0B   NOP                                                ;...
U 0C. 7700.4800.5BE4.0314.5C80.180D  04F2   552 ;% 0C   WRITE.PHY R[TEMP5]                                  ;write ones to memory
U 0D. 7700.4800.0364.0300.0400.180E  04FD   556 ;% 0D   READ.PHY                                           ;read pattern back
U 0E. 7700.4852.5924.0300.0470.180F  0508   560 ;% 0E   R[TEMP0]_M[MDR]                                   ;save data in RTEMP0
U 0F. 7700.C800.0364.AF00.0470.1801  0513   564 ;% 0F   BUT/INT-TIMSERV,NEXT/1801                         ;test for corrected data
U 10. 7700.0800.0034.A714.0470.1800  051E   568 ;% 10   WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                     0529   572                                                           ;test for good data
                                     0529   573 65$:
U 11. 7700.481B.0034.A318.0470.1800  0529   574 ;% 11   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     0534   578                                                           ;test for end of first array
U 12. 7700.4800.0364.0300.4470.1813  0534   579 ;% 12   VA_VA+4                                           ;increment va
U 13. 7500.4800.0364.0300.0470.1814  053F   583 ;% 13   NOP,DCS.ADDR_0                                    ;return to dcs address 0
U 14. 7700.C800.0364.0300.0470.1815  054A   587 ;% 14   NOP
U 15. 7700.885B.5924.031C.0470.1816  0555   591 ;% 15   R[TEMP7]_M[VA]                                    ;save va for return
U 16. 7700.0800.5BE4.0304.6870.1817  0560   595 ;% 16   MEMSCAR_R[TEMP1]                                  ;disable cmi
U 17. 7700.C800.5BE4.030C.6070.1818  056B   599 ;% 17   MEMSCR_R[TEMP3]                                   ;...
U 18. 7700.8800.5BE4.0308.6870.1819  0576   603 ;% 18   MEMSCAR_R[TEMP2]                                  ;turn on cache
U 19. 7700.4800.5BE4.03D8.6070.181A  0581   607 ;% 19   MEMSCR_R[ZERO]                                    ;...
U 1A. 7700.4800.5BE4.03D8.4A70.181B  058C   611 ;% 1A   VA_R[ZERO]                                       ;zero the va
U 1B. 7700.4800.5BE4.03D8.5C80.181C  0597   615 ;% 1B   WRITE.PHY R[ZERO]                                 ;validate cache address 0
U 1C. 7700.4800.5BE4.03D8.5470.181D  05A2   619 ;% 1C   WDR_UNROT(R[ZERO])                                ;...
U 1D. 7700.C800.0364.0300.4470.181E  05AD   623 ;% 1D   VA_VA+4                                           ;increment va by 4
U 1E. 7700.4800.5BE4.03D8.5C80.181F  05B8   627 ;% 1E   WRITE.PHY R[ZERO]                                 ;validate cache address 4
U 1F. 7700.C800.5BE4.03D8.5470.1820  05C3   631 ;% 1F   WDR_UNROT(R[ZERO])
U 20. 7700.C800.5BE4.03D8.4870.1821  05CE   635 ;% 20   PC_R[ZERO]                                       ;cause prefetch
U 21. 7700.8800.5BE4.032C.2470.1822  05D9   639 ;% 21   WCTRL/LOADCRAR,WB_R[TEMP11]                      ;disable ^p interrupts
U 22. 7700.C800.5BE4.0330.2C70.1823  05E4   643 ;% 22   WCTRL/CONWRITE,WB_R[TEMP12]                      ;...
U 23. 7700.0800.5BE4.0304.6870.1824  05EF   647 ;% 23   MEMSCAR_R[TEMP1]                                  ;turn on cmi
U 24. 7700.4800.5BE4.03D8.6070.1825  05FA   651 ;% 24   MEMSCR_R[ZERO]                                    ;...
U 25. 7700.8800.5BE4.0308.6870.1826  0605   655 ;% 25   MEMSCAR_R[TEMP2]                                  ;disable cache
U 26. 7700.C800.5BE4.030C.6070.1827  0610   659 ;% 26   MEMSCR_R[TEMP3]                                   ;...
U 27. 7700.0800.5BE4.0320.4A70.1828  061B   663 ;% 27   VA_R[TEMP8]                                      ;address of CSR0
U 28. 7700.4800.0364.0300.0400.1829  0626   667 ;% 28   READ.PHY                                          ;read CSR0
U 29. 3700.8812.5924.0300.0470.182A  0631   671 ;% 29   WB_M[MDR],RDM_CMI                                ;MHREG gets CSR0
U 2A. 7700.C800.5BE4.0324.5C80.182B  063C   675 ;% 2A   WRITE.PHY R[TEMP9]                                ;clear CSR0
U 2B. 7700.C800.0364.7B00.0470.182C  0647   679 ;% 2B   BUT/UVCTR                                       ;clear any interrupts pending
U 2C. 7700.4800.0364.1700.0470.182D  0652   683 ;% 2C   IRD1TEST                                        ;free interrupt pending logic
U 2D. 7700.0800.5BE4.031C.4A70.182E  065D   687 ;% 2D   VA_R[TEMP7]                                      ;reset the va
U 2E. 7300.C800.0364.0300.0470.182F  0668   691 ;% 2E   NOP,STOP.CLOCK                                   ;stop the cpu clock
                                     0673   695
                                     0673   696 BEGIN_CPU_DATA
                                     0002   697
                                     0002   698 RTEMP_DATA
                                     0001   699
                         00000000    0001   700         .LONG   ^X00000000                              ;data pattern storage
                         0E000000    0005   701         .LONG   ^X0E000000                              ;reference cache only address
                         06000000    0009   702         .LONG   ^X06000000                              ;cache control register address
                         01000000    000D   703         .LONG   ^X01000000                              ;disable cache
                         00000000    0011   704         .LONG   ^X00000000                              ;test pattern 1
                         FFFFFFFF    0015   705         .LONG   ^XFFFFFFFF                              ;test pattern 2
                         0003FFFC    0019   706         .LONG   ^X0003FFFC                              ;last address for 256kb array 0
                         00000000    001D   707         .LONG   ^X00000000                              ;storage for va register
```

```
                    00F20000  0021   708          .LONG   ^X00F20000                      ;address of CSR0
                    E0000000  0025   709          .LONG   ^XE0000000                      ;clears CSR0
                    10000000  0029   710          .LONG   ^X10000000                      ;allow single bit error report
                    00C00000  002D   711          .LONG   ^X00C00000                      ;disable ^p interrupts
                    00400000  0031   712          .LONG   ^X00400000                      ;...
                    000FFFFC  0035   713          .LONG   ^X000FFFFC                      ;last address for 1mb array 0
                              0039   714
                              0039   715 END_CPU_DATA
                              0673   716
                              0673   717
                              0673   718 END_TEST_DATA
                              0673
                              0673       ;-------------------------------------------------------------------------
                              0673   719 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        .SBTTL  "TEST   083     Array 0 Memory Test Part 2
001F    721 ;*****************************************************************************
001F    722 ;++
001F    723 ;
001F    724 ; FUNCTIONAL DESCRIPTION:
001F    725 ;
001F    726 ;       This test will check the first array card writing first 1's then
001F    727 ;       0's in ascending order.
001F    728 ;
001F    729 ;
001F    730 ; TEST ALGORITHM:
001F    731 ;
001F    732 ;       1.  Load u-code to initialize test conditions
001F    733 ;       2.  Execute dcs program
001F    734 ;               a.  Disable the cmi
001F    735 ;               b.  Enable the cache
001F    736 ;               c.  Set va to zero
001F    737 ;               d.  Validate cache addresses 0 and 4
001F    738 ;               e.  Load the pc to cause prefetch of addresses 0 and 4
001F    739 ;               f.  Disable control P interrupts
001F    740 ;               g.  Enable cmi and disable cache
001F    741 ;               h.  Clear memory CSR0
001F    742 ;               i.  Set memory csr1 to allow single bit errors
001F    743 ;               j.  Read CSR2
001F    744 ;               k.  Perform IRD1TEST to free interrupt pending logic
001F    745 ;               L.  Set va to zero
001F    746 ;       3.  Check low bits 1:0 of CSR2
001F    747 ;               a.  If this is a 256kb array then mask in last address of
001F    748 ;                   3FFFC then continue
001F    749 ;               b.  If this is a 1mb array then mask in last address of FFFFC,
001F    750 ;                   and change two micro words so they use RTEMP13 when
001F    751 ;                   checking to see if at last address
001F    752 ;               c.  If this is a 4mb array, then mask in last address of 3FFFFC,
001F    753 ;                   change RTEMP13 to 3FFFFC, and change two microwords so
001F    754 ;                   they use RTEMP13 when checking to see if at last address
001F    755 ;       4.  Load index I with 2 which will be used for comparison of index K
001F    756 ;       5.  Load u-code to write ones to main memory
001F    757 ;       6.  Load the first u-inst.
001F    758 ;       7.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
001F    759 ;           error section as follows:
001F    760 ;               a.  First pass is from the writing of ones throughout memory
001F    761 ;               b.  Second pass is from the read-write-read throughout memory
001F    762 ;       8.  Check to see if this is first pass or second pass, if first pass
001F    763 ;           then goto 112$, if second pass then goto 110$
001F    764 ;       9.  Set-up the rdm to watch for u-traps
001F    765 ;       10. Set the u-match halt for 1801
001F    766 ;       11. Start the cpu clock
001F    767 ;               a.  Save the va in the rdm WH register
001F    768 ;               b.  Write ones to the address in the va
001F    769 ;               c.  Test va to see if at end of first array card
001F    770 ;               d.  If not increment va and return to dcs address 0
001F    771 ;       12. Set-up watchdog timer with loop J to monitor above u-code
001F    772 ;       13. If loop J expires, stop the clock and force an error printout
001F    773 ;       14. If clock stops normally (u-match at 1801); test WH register for
001F    774 ;           end of first array
```

```
001F   775 ;      15.  If no error test cs address for not a u-trap
001F   776 ;      16.  Increment loop K for second pass
001F   777 ;      17.  Execute dcs code
001F   778 ;               a.  Set va to zero
001F   779 ;      18.  Load the test u-code
001F   780 ;      19.  Load the first u-inst.
001F   781 ;      20.  Set-up the rdm to watch for u-traps
001F   782 ;      21.  Set the u-match to 1801
001F   783 ;      22.  Start the cpu clock
001F   784 ;               a.  Save va in rdm WH register
001F   785 ;               b.  Read address specified by va
001F   786 ;               c.  Move data to RTEMP0
001F   787 ;               d.  Test for a corrected data interrupt
001F   788 ;               e.  Test that data is correct
001F   789 ;               f.  Write zero to the same location
001F   790 ;               g.  Read location back
001F   791 ;               h.  Save data in RTEMP0
001F   792 ;               i.  Test for a corrected data interrupt
001F   793 ;               j.  Test that data is correct
001F   794 ;               k.  Test to see if at end of first array
001F   795 ;               l.  Increment the va by 4
001F   796 ;               m.  Return to dcs address 0
001F   797 ;      23.  Set-up watch-dog timer with loop J to monitor above u-code
001F   798 ;      24.  If loop J expires, stop the clock and force an error printout
001F   799 ;      25.  If clock stops normally (u-match at 1801) test WH register
001F   800 ;           for end of first array.
001F   801 ;      26.  Test for machine checks (double bit error in memory)
001F   802 ;      27.  If not machine check test for incorrect data on first read
001F   803 ;      28.  If not bad data test for incorrect data on second read
001F   804 ;      29.  Test for corrected data interrupts
001F   805 ;      30.  Service the corrected data interrupt with program at dcs address 14
001F   806 ;               a.  Save va in RTEMP7
001F   807 ;               b.  Disable cmi enable cache
001F   808 ;               c.  Set va to zero
001F   809 ;               d.  Validate cache address 0 and 4
001F   810 ;               e.  Load pc to cause prefetch of addresses 0 and 4
001F   811 ;               f.  Disable ^p interrupts
001F   812 ;               g.  Read memory CSR0 to the rdm MHREG
001F   813 ;               h.  Clear memory CSR0
001F   814 ;               i.  Clear any interrupts pending
001F   815 ;               j.  Perform an IRD1TEST to free the interrupt pending logic
001F   816 ;               k.  Reload va with RTEMP7
001F   817 ;      31.  Continue running test micro code from step 17.
001F   818 ;      32.  Test for the end of the first array.
001F   819 ;      33.  When last address of array is found the test will end on endloop K
001F   820 ;           being expired
001F   821 ;      34.  The balance of the pseudo ops handle the various error conditions
001F   822 ;           (see error description)
001F   823 ;
001F   824 ;
001F   825 ; TEST PATTERNS:
001F   826 ;
001F   827 ;        First pass:  init memory to 1's
001F   828 ;       Second pass:  read 1's; write 0's; read 0's
001F   829 ;
```

```
001F    830 ;
001F    831 ; LOGIC DESCRIPTION:
001F    832 ;
001F    833 ;       My advice on this test is forget trying to troubleshoot with it.  It
001F    834 ;       is far to complex to be a usefull troubleshooting tool.  Rather, just
001F    835 ;       use the error printout to array and/or module swap till you find the
001F    836 ;       problem.  The first array is throughly tested only after running
001F    837 ;       all four tests in this series; together and in order.
001F    838 ;
001F    839 ;
001F    840 ; ASSUMPTIONS:
001F    841 ;
001F    842 ;       This test assumes the cmi is operational.
001F    843 ;
001F    844 ;
001F    845 ; ERROR DESCRIPTION:
001F    846 ;
001F    847 ;       First IFERROR:
001F    848 ;
001F    849 ;       EXPECTED LAST ADDRESS OF FIRST ARRAY
001F    850 ;       RECEIVED LAST ADDRESS
001F    851 ;       LOOP COUNT I
001F    852 ;       LOOP COUNT J
001F    853 ;       LOOP COUNT K
001F    854 ;       (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
001F    855 ;
001F    856 ;       Second IFERROR:
001F    857 ;
001F    858 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F    859 ;       MAIN MEMORY ADDRESS OF DATA ERROR
001F    860 ;       LOOP COUNT I
001F    861 ;       LOOP COUNT J
001F    862 ;       LOOP COUNT K
001F    863 ;       RECEIVED DATA (S/B 0'S OR 1'S)
001F    864 ;       (INDICATES ECC IS PROBABLE NOT WORKING)
001F    865 ;
001F    866 ;       Third IFERROR:
001F    867 ;
001F    868 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F    869 ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
001F    870 ;       LOOP COUNT I
001F    871 ;       LOOP COUNT J
001F    872 ;       LOOP COUNT K
001F    873 ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
001F    874 ;
001F    875 ;--
001F    876 ;*****************************************************************************
001F    877
001F    878 ;-----------------------------------------------------------------------------
001F    879 ;
001F    880 ; Initialize code
001F    881 ;
001F    882 ;-----------------------------------------------------------------------------
001F    883
001F    884         INITIALIZE                                      ;init the cpu
```

```
0021   885          LOADDCS         40$,,0,<^X1C>           ;setup micro code
0028   886          FETCH           0                       ;start dcs program
002D   887          BRSTCLK         <^X1B>,INH              ;end dcs program
0031   888          CMPREGMSK       WHREG,23$,,18$,,B       ;check array 0
003D   889          SKIP            100$,NOERROR            ;skip if 4mb
0044   890          CMPREGMSK       WHREG,21$,,18$,,B,NE    ;check array 0 type
0050   891          SKIP            102$,ONERROR            ;skip if 1mb module
0057   892          MASK            4$,,19$,,L              ;mask in address 3FFFC
0060   893          SKIP            108$                    ;go test
0067   894
0067   895 ;
0067   896 ; This section will set up for a 4mb array
0067   897 ;
0067   898 100$:    MASK            4$,,24$,,L              ;mask in address 3FFFFC
0070   899          LOADDCS         45$,,0,4                ;R[TEMP13]_3FFFFC
0077   900          FETCH           0                       ;start dcs
007C   901          BRSTCLK         3                       ;end dcs
0080   902          SKIP            103$                    ;change uword to use R[TEMP13]
0087   903 ;
0087   904 ; This section will set-up for a 1mb module versus a 256kb module
0087   905 ;
0087   906 102$:    MASK            4$,,20$,,L              ;mask in address FFFFC
0090   907 103$:    LDFIELD         22$,,B,55$,34,39        ;change rsrc fld for R[TEMP13]
00A1   908          LDFIELD         22$,,B,65$,34,39        ;change rsrc fld for R[TEMP13]
00B2   909
00B2   910 ;-------------------------------------------------------------------------------
00B2   911 ; Write memory with 1's
00B2   912 ;
00B2   913 ;-------------------------------------------------------------------------------
00B2   914
00B2   915 108$:    LDINDEX         I,2                     ;dummy index I gets 2
00B9   916          LOADDCS         50$,,0,<^X0B>           ;test u-code
00C0   917          FETCH           0                       ;load first u-inst.
00C5   918          LOOP            K,1,2                   ;set index K for two passes
00CC   919          SKIP            110$,ONEQUAL,I,K        ;skip on second pass
00D3   920          SKIP            112$                    ;first pass, goto rdm set-up
00DA   921
00DA   922 ;-------------------------------------------------------------------------------
00DA   923 ;
00DA   924 ; Read the 1's from the previous write, check for valid data, write 0's, check
00DA   925 ; for valid data
00DA   926 ;
00DA   927 ;-------------------------------------------------------------------------------
00DA   928
00DA   929 110$:    FETCH           8                       ;reset va to zero
00DF   930          BRSTCLK         <^X0A>,INH              ;...
00E3   931          LOADDCS         60$,,0,<^X2F>           ;test u-code
00EA   932          FETCH           0                       ;load first u-inst.
00EF   933 ;
00EF   934 ; This section used for both the initial writing of 1's (first pass) then for
00EF   935 ; the read of 1's-write of 0's-read of 0's (second pass)
00EF   936 ;
00EF   937 112$:    LOADREG         DCSCR,9$,,B             ;insure clock is off
00F7   938          LOADREG         CSAMR,2$,,W             ;load match address
00FF   939          LOADREG         RDCTRL,7$,,B            ;clear master halt enable
```

```
0107   940          LOADREG         RDCTRL,1$,,B            ;watch for u-traps
010F   941          LOADREG         DCSCR,3$,,B             ;start the clock
0117   942          LOOP            J,1,8192                ;watchdog timing loop
011E   943          CMPREGMSK       CLKDCS,7$,,8$,,B,       ;test for clock stopped
012A   944          SKIP            114$,ONERROR            ;skip when clock stops
0131   945          ENDLOOP         J                       ;end timing loop
0134   946          LOADREG         DCSCR,9$,,B             ;stop the clock
013C   947          CMPREG          WHREG,4$,,L,            ;force an error
0145   948          IFERROR         ,,<<MC0><MC1><MC2>>     ;program lost
014D   949          SKIP
0154   950
0154   951 ;-------------------------------------------------------------------------------
0154   952 ;
0154   953 ; Error evaluation code
0154   954 ;
0154   955 ;-------------------------------------------------------------------------------
0154   956
0154   957 114$:    CMPREGMSK       CSTRP,10$,,6$,,W,NE     ;test for machine check
0160   958          SKIP            140$,ONERROR            ;handle machine check
0167   959          CMPREGMSK       CLKDCS,11$,,12$,,B,NE   ;test for data errors
0173   960          SKIP            130$,ONERROR            ;incorrect data
017A   961          CMPREGMSK       CLKDCS,13$,,12$,,B,NE   ;test for data errors
0186   962          SKIP            130$,ONERROR            ;incorrect data
018D   963          CMPREGMSK       CLKDCS,14$,,12$,,B,NE   ;test for single bit error on 0
0199   964          SKIP            120$,ONERROR            ;log error
01A0   965          CMPREGMSK       CLKDCS,16$,,12$,,B,NE   ;test for single bit error on 1
01AC   966          SKIP            115$,ONERROR            ;log error
01B3   967          CMPREG          WHREG,4$,,L,            ;test for last address
01BC   968          SKIP            150$,NOERROR            ;go to next test
01C3   969          CMPREGMSK       CLKDCS,17$,,12$,,B      ;force and error
01CF   970          IFERROR                                 ;i think we're lost
01D4   971          SKIP                                    ;end test due to error
01DB   972 ;
01DB   973 ; Come here if the Second read caused corrected read data to be returned
01DB   974 ;
01DB   975 115$:    FETCH           <^X14>                  ;start dcs program
01E0   976          BRSTCLK         <^X2E>                  ;end dcs program
01E4   977          ERRLOG                                  ;log the error
01E6   978          SKIP            150$,ONERROR            ;quit after 64 errors
01ED   979          FETCH           <^X10>                  ;set up to continue testing
01F2   980          SKIP            112$                    ;test more memory
01F9   981 ;
01F9   982 ; Come here if the First read caused corrected read data to be returned
01F9   983 ;
01F9   984 120$:    FETCH           <^X14>                  ;start dcs program
01FE   985          BRSTCLK         <^X2E>                  ;end dcs program
0202   986          ERRLOG                                  ;log the error
0204   987          SKIP            150$,ONERROR            ;quit after 64 errors
020B   988          FETCH           <^X07>                  ;set up to continue testing
0210   989          SKIP            112$                    ;test more memory
0217   990 ;
0217   991 ; Come here if the data read is not the data written
0217   992 ;
0217   993 130$:    CMPREG          WHREG,15$,,L,           ;force an error
0220   994          IFERROR         1,<<MEC><MDL>>,-
```

```
                     0220   995          <<MC0><MC1><MC2><MA0><MA1><MA2>>          ;print va and data
                     022D   996          SKIP                                      ;end test because of error
                     0234   997 ;
                     0234   998 ; Come here if there is a Machine check (Read Data Substitute)
                     0234   999 ;
                     0234  1000 140$:    CMPREG           WHREG,15$,,L,            ;force an error
                     023D  1001          IFERROR          ,<<MEC><MDL>>,-
                     023D  1002          <<MC0><MC1><MC2><MA0><MA1><MA2>>          ;print va of machine check
                     024A  1003 ;-------------------------------------------------------------------------
                     024A  1004 ;
                     024A  1005 ; This is where we exit the test if index K is equal to 2
                     024A  1006 ;
                     024A  1007 ;-------------------------------------------------------------------------
                     024A  1008
                     024A  1009 150$:    ENDLOOP          K
                     024D  1010          SKIP                                      ;go to next test
                     0254  1011
                     0254  1012
                     0254  1013 BEGIN_TEST_DATA
                     0254
                     0254          ;-------------------------------------------------------------------
                     0254  1014
                       28  0254  1015 1$:      .BYTE    ^X28                      ;set to watch for u-traps
                     1801  0255  1016 2$:      .WORD    ^X1801                    ;match address
                       11  0257  1017 3$:      .BYTE    ^X11                      ;run clock
                FFFFFFFF  0258  1018 4$:      .LONG    ^XFFFFFFFF                ;last address of first array
                     3FFF  025C  1019 6$:      .WORD    ^X3FFF                    ;mask for cmpregmsk pseudo op
                       00  025E  1020 7$:      .BYTE    ^X00                      ;watch for clock stop
                       C0  025F  1021 8$:      .BYTE    ^XC0                      ;mask for cmpregmsk pseudo op
                       60  0260  1022 9$:      .BYTE    ^X60                      ;stop clock manually
                     0028  0261  1023 10$:     .WORD    ^X0028                    ;machine check u-trap
                       09  0263  1024 11$:     .BYTE    ^X09                      ;dcs halt on data error
                       3F  0264  1025 12$:     .BYTE    ^X3F                      ;mask for cmpregmsk pseudo op
                       12  0265  1026 13$:     .BYTE    ^X12                      ;dcs halt on data error
                       08  0266  1027 14$:     .BYTE    ^X08                      ;halt single bit error on 0's
                 0F0F0F0F  0267  1028 15$:     .LONG    ^X0F0F0F0F                ;compare data to force errors
                       11  026B  1029 16$:     .BYTE    ^X11                      ;halt single bit error on 1's
                       FF  026C  1030 17$:     .BYTE    ^XFF                      ;used to force errors
                       03  026D  1031 18$:     .BYTE    ^X03                      ;array 0 mask
                 0003FFFC  026E  1032 19$:     .LONG    ^X0003FFFC                ;256kb array 0 last address
                 000FFFFC  0272  1033 20$:     .LONG    ^X000FFFFC                ;1mb array 0 last address
                       02  0276  1034 21$:     .BYTE    ^X02                      ;array 0 1mb (csr2 bits 0:1)
                       0D  0277  1035 22$:     .BYTE    ^X0D                      ;rsrc = temp13
                       01  0278  1036 23$:     .BYTE    ^X01                      ;array 0 = 4mb
                 003FFFFC  0279  1037 24$:     .LONG    ^X003FFFFC                ;4mb array 0 last address
                           027D  1038 40$:
U 00, 7700,C800,0364,0300,0470,1801  027D  1039 ;* 00   NOP
U 01, 7700,8800,5BE4,0304,6870,1802  0288  1043 ;* 01   MEMSCAR_R[TEMP1]           ;disable cmi
U 02, 7700,C800,5BE4,030C,6070,1803  0293  1047 ;* 02   MEMSCR_R[TEMP3]           ;...
U 03, 7700,8800,5BE4,0308,6870,1804  029E  1051 ;* 03   MEMSCAR_R[TEMP2]          ;turn on cache
U 04, 7700,C800,5BE4,03D8,6070,1805  02A9  1055 ;* 04   MEMSCR_R[ZERO]           ;...
U 05, 7700,4800,5BE4,03D8,4A70,1806  02B4  1059 ;* 05   VA_R[ZERO]               ;zero the va
U 06, 7700,4800,5BE4,03D8,5C80,1807  02BF  1063 ;* 06   WRITE.PHY R[ZERO]        ;validate cache address 0
U 07, 7700,C800,5BE4,03D8,5470,1808  02CA  1067 ;* 07   WDR_UNROT(R[ZERO])       ;...
U 08, 7700,C800,0364,0300,4470,1809  02D5  1071 ;* 08   VA_VA+4                   ;increment va by 4
```

```
U 09. 7700.C800.5BE4.03D8.5C80.180A  02E0  1075 ;% 09   WRITE.PHY R[ZERO]                                  ;validate cache address 4
U 0A. 7700.C800.5BE4.03D8.5470.180B  02EB  1079 ;% 0A   WDR_UNROT(R[ZERO])
U 0B. 7700.C800.5BE4.03D8.4870.180C  02F6  1083 ;% 0B   PC_R[ZERO]                                       ;cause prefetch
U 0C. 7700.0800.5BE4.032C.2470.180D  0301  1087 ;% 0C   WCTRL/LOADCRAR,WB_R[TEMP11]                      ;disable ^p interrupts
U 0D. 7700.C800.5BE4.0330.2C70.180E  030C  1091 ;% 0D   WCTRL/CONWRITE,WB_R[TEMP12]                     ;...
U 0E. 7700.0800.5BE4.0304.6870.180F  0317  1095 ;% 0E   MEMSCAR_R[TEMP1]                                ;turn on cmi
U 0F. 7700.4800.5BE4.03D8.6070.1810  0322  1099 ;% 0F   MEMSCR_R[ZERO]                                  ;...
U 10. 7700.0800.5BE4.0308.6870.1811  032D  1103 ;% 10   MEMSCAR_R[TEMP2]                                ;disable cache
U 11. 7700.C800.5BE4.030C.6070.1812  0338  1107 ;% 11   MEMSCR_R[TEMP3]                                 ;...
U 12. 7700.8800.5BE4.0320.4A70.1813  0343  1111 ;% 12   VA_R[TEMP8]                                     ;CSR0 address to va
U 13. 7700.C800.5BE4.0324.5C80.1814  034E  1115 ;% 13   WRITE.PHY R[TEMP9]                               ;clear CSR0
U 14. 7700.4800.0364.0300.4470.1815  0359  1119 ;% 14   VA_VA+4                                         ;CSR1 address to va
U 15. 7700.4800.5BE4.0328.5C80.1816  0364  1123 ;% 15   WRITE.PHY R[TEMP10]                              ;allow single bit errors
U 16. 7700.C800.0364.0300.4470.1817  036F  1127 ;% 16   VA_VA+4                                         ;CSR2 address to va
U 17. 7700.C800.0364.0300.0400.1818  037A  1131 ;% 17   READ.PHY                                        ;read CSR2
U 18. 6700.8812.5924.0300.0470.1819  0385  1135 ;% 18   RDM_WB,WB_M[MDR]                                ;WHREG gets contents of CSR2
U 19. 7700.C800.0364.1700.0470.181A  0390  1139 ;% 19   IRDITEST                                       ;clear int pending line
U 1A. 7700.4800.5BE4.03D8.4A70.181B  039B  1143 ;% 1A   VA_R[ZERO]                                     ;zero the va register
U 1B. 7300.C800.0364.0300.0470.181C  03A6  1147 ;% 1B   NOP,STOP.CLOCK                                  ;stop the cpu clock
                                     03B1  1151 45$:
U 00. 7700.B800.7FE0.0001.8470.1801  03B1  1152 ;% 00   LONLIT_[003FFFFC]                                ;last addr for 4mb
U 01. 7700.8868.5BE4.03D4.0470.1802  03BC  1156 ;% 01   M[TEMP8]_R[LONLIT]                               ;
U 02. 7700.4848.5924.0334.0470.1803  03C7  1160 ;% 02   R[TEMP13]_M[TEMP8]                               ;RTEMP13 = last addr 3FFFFC
U 03. 7300.C800.0364.0300.0470.1804  03D2  1164 ;% 03   NOP,STOP.CLOCK                                  ;stop cpu clock
                                     03DD  1168 50$:
U 00. 7700.C800.0364.0300.0470.1802  03DD  1169 ;% 00   NOP,NEXT/1802
U 01. 6700.881B.5924.0300.0470.1802  03E8  1173 ;% 01   RDM_WB,WB_M[VA]                                 ;save va in rdm
U 02. 7700.C800.5BE4.0314.5C80.1803  03F3  1177 ;% 02   WRITE.PHY R[TEMP5]                               ;write ones to memory
U 03. 7700.C800.0364.0300.0470.1804  03FE  1181 ;% 03   NOP                                            ;allow time for write
U 04. 7700.4800.0364.0300.0470.1805  0409  1185 ;% 04   NOP                                            ;...
                                     0414  1189 55$:
U 05. 7700.481B.0034.A318.0470.1800  0414  1190 ;% 05   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     041F  1194                                                       ;test for end of first array
U 06. 7700.4800.0364.0300.4470.1807  041F  1195 ;% 06   VA_VA+4                                         ;increment va register
U 07. 7500.C800.0364.0300.0470.1808  042A  1199 ;% 07   NOP,DCS.ADDR_0                                 ;return to address 0
U 08. 7700.4800.0364.0300.0470.1809  0435  1203 ;% 08   NOP
U 09. 7700.4800.5BE4.03D8.4A70.180A  0440  1207 ;% 09   VA_R[ZERO]                                     ;set va to zero
U 0A. 7300.C800.0364.0300.0470.180B  044B  1211 ;% 0A   NOP,STOP.CLOCK                                  ;stop the cpu clock
                                     0456  1215 60$:
U 00. 7700.C800.0364.0300.0470.1802  0456  1216 ;% 00   NOP,NEXT/1802                                   ;allow time for memory
U 01. 7700.C800.0364.0300.0470.1802  0461  1220 ;% 01   NOP                                            ;clock to start up.
U 02. 7700.4800.0364.0300.0470.1803  046C  1224 ;% 02   NOP                                            ;...
U 03. 6700.881B.5924.0300.0470.1804  0477  1228 ;% 03   RDM_WB,WB_M[VA]                                 ;save va in rdm
U 04. 7700.C800.0364.0300.0400.1805  0482  1232 ;% 04   READ.PHY                                        ;read pattern back
U 05. 7700.4852.5924.0300.0470.1806  048D  1236 ;% 05   R[TEMP0]_M[MDR]                                 ;save data in RTEMP0
U 06. 7700.C800.0364.AF00.0470.1801  0498  1240 ;% 06   BUT/INT-TIMSERV,NEXT/1801                      ;test for corrected data
U 07. 7700.0800.0034.A714.0470.1800  04A3  1244 ;% 07   WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                     04AE  1248                                                       ;test for good data
U 08. 7700.4800.0364.0300.0470.1809  04AE  1249 ;% 08   NOP                                            ;allow time for memory clock
U 09. 7700.4800.0364.0300.0470.180A  04B9  1253 ;% 09   NOP                                            ;to start up after single bit
U 0A. 7700.C800.0364.0300.0470.180B  04C4  1257 ;% 0A   NOP                                            ;errors.
U 0B. 7700.4800.0364.0300.0470.180C  04CF  1261 ;% 0B   NOP                                            ;...
U 0C. 7700.0800.5BE4.0310.5C80.180D  04DA  1265 ;% 0C   WRITE.PHY R[TEMP4]                               ;write zeros to memory
U 0D. 7700.4800.0364.0300.0400.180E  04E5  1269 ;% 0D   READ.PHY                                        ;read pattern back
U 0E. 7700.4852.5924.0300.0470.180F  04F0  1273 ;% 0E   R[TEMP0]_M[MDR]                                 ;save data in RTEMP0
```

```
U 0F. 7700.C800.0364.AF00.0470.1801  04FB  1277 ;* 0F   BUT/INT-TIMSERV,NEXT/1801                    ;test for corrected data
U 10. 7700.4800.0034.A710.0470.1800  0506  1281 ;* 10   WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                     0511  1285                                                     ;test for good data
                                     0511  1286 65$:
U 11. 7700.481B.0034.A318.0470.1800  0511  1287 ;* 11   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     051C  1291                                                     ;test for end of first array
U 12. 7700.4800.0364.0300.4470.1813  051C  1292 ;* 12   VA_VA+4                                     ;increment va
U 13. 7500.4800.0364.0300.0470.1814  0527  1296 ;* 13   NOP,DCS.ADDR_0                             ;return to dcs address 0
U 14. 7700.C800.0364.0300.0470.1815  0532  1300 ;* 14   NOP
U 15. 7700.885B.5924.031C.0470.1816  053D  1304 ;* 15   R[TEMP7]_M[VA]                              ;save va for return
U 16. 7700.0800.5BE4.0304.6870.1817  0548  1308 ;* 16   MEMSCAR_R[TEMP1]                           ;disable cmi
U 17. 7700.C800.5BE4.030C.6070.1818  0553  1312 ;* 17   MEMSCR_R[TEMP3]                            ;...
U 18. 7700.8800.5BE4.0308.6870.1819  055E  1316 ;* 18   MEMSCAR_R[TEMP2]                           ;turn on cache
U 19. 7700.4800.5BE4.03D8.6070.181A  0569  1320 ;* 19   MEMSCR_R[ZERO]                             ;...
U 1A. 7700.4800.5BE4.03D8.4A70.181B  0574  1324 ;* 1A   VA_R[ZERO]                                 ;zero the va
U 1B. 7700.4800.5BE4.03D8.5C80.181C  057F  1328 ;* 1B   WRITE.PHY R[ZERO]                           ;validate cache address 0
U 1C. 7700.4800.5BE4.03D8.5470.181D  058A  1332 ;* 1C   WDR_UNROT(R[ZERO])                         ;...
U 1D. 7700.C800.0364.0300.4470.181E  0595  1336 ;* 1D   VA_VA+4                                     ;increment va by 4
U 1E. 7700.4800.5BE4.03D8.5C80.181F  05A0  1340 ;* 1E   WRITE.PHY R[ZERO]                           ;validate cache address 4
U 1F. 7700.C800.5BE4.03D8.5470.1820  05AB  1344 ;* 1F   WDR_UNROT(R[ZERO])
U 20. 7700.C800.5BE4.03D8.4870.1821  05B6  1348 ;* 20   PC_R[ZERO]                                 ;cause prefetch
U 21. 7700.8800.5BE4.032C.2470.1822  05C1  1352 ;* 21   WCTRL/LOADCRAR,WB_R[TEMP11]                ;disable ^p interrupts
U 22. 7700.C800.5BE4.0330.2C70.1823  05CC  1356 ;* 22   WCTRL/CONWRITE,WB_R[TEMP12]                ;...
U 23. 7700.0800.5BE4.0304.6870.1824  05D7  1360 ;* 23   MEMSCAR_R[TEMP1]                           ;turn on cmi
U 24. 7700.4800.5BE4.03D8.6070.1825  05E2  1364 ;* 24   MEMSCR_R[ZERO]                             ;...
U 25. 7700.8800.5BE4.0308.6870.1826  05ED  1368 ;* 25   MEMSCAR_R[TEMP2]                           ;disable cache
U 26. 7700.C800.5BE4.030C.6070.1827  05F8  1372 ;* 26   MEMSCR_R[TEMP3]                            ;...
U 27. 7700.0800.5BE4.0320.4A70.1828  0603  1376 ;* 27   VA_R[TEMP8]                                ;address of CSR0
U 28. 7700.4800.0364.0300.0400.1829  060E  1380 ;* 28   READ.PHY                                   ;read CSR0
U 29. 3700.8812.5924.0300.0470.182A  0619  1384 ;* 29   WB_M[MDR],RDM_CMI                          ;MHREG gets CSR0
U 2A. 7700.C800.5BE4.0324.5C80.182B  0624  1388 ;* 2A   WRITE.PHY R[TEMP9]                         ;clear CSR0
U 2B. 7700.C800.0364.7B00.0470.182C  062F  1392 ;* 2B   BUT/UVCTR                                 ;clear any interrupts pending
U 2C. 7700.4800.0364.1700.0470.182D  063A  1396 ;* 2C   IRD1TEST                                  ;free interrupt pending logic
U 2D. 7700.0800.5BE4.031C.4A70.182E  0645  1400 ;* 2D   VA_R[TEMP7]                               ;reset the va
U 2E. 7300.C800.0364.0300.0470.182F  0650  1404 ;* 2E   NOP,STOP.CLOCK                            ;stop the cpu clock
                                     065B  1408
                                     065B  1409 BEGIN_CPU_DATA
                                     0002  1410
                                     0002  1411 RTEMP_DATA
                                     0001  1412
                          00000000   0001  1413          .LONG   ^X00000000                      ;data pattern storage
                          0E000000   0005  1414          .LONG   ^X0E000000                      ;reference cache only address
                          06000000   0009  1415          .LONG   ^X06000000                      ;cache control register address
                          01000000   000D  1416          .LONG   ^X01000000                      ;disable cache
                          00000000   0011  1417          .LONG   ^X00000000                      ;test pattern 1
                          FFFFFFFF   0015  1418          .LONG   ^XFFFFFFFF                      ;test pattern 2
                          0003FFFC   0019  1419          .LONG   ^X0003FFFC                      ;last address for 256kb array 0
                          00000000   001D  1420          .LONG   ^X00000000                      ;storage for va register
                          00F20000   0021  1421          .LONG   ^X00F20000                      ;address of CSR0
                          E0000000   0025  1422          .LONG   ^XE0000000                      ;clears CSR0
                          10000000   0029  1423          .LONG   ^X10000000                      ;allow single bit error report
                          00C00000   002D  1424          .LONG   ^X00C00000                      ;disable ^p interrupts
                          00400000   0031  1425          .LONG   ^X00400000                      ;...
                          000FFFFC   0035  1426          .LONG   ^X000FFFFC                      ;last address for 1mb array 0
                                     0039  1427
```

```
                     0039  1428 END_CPU_DATA
                     065B  1429
                     065B  1430
                     065B  1431 END_TEST_DATA
                     065B
                     065B       ;-----------------------------------------------------------------------
                     065B  1432 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        .SBTTL  "TEST   084     Array 0 Memory Test Part 3
001F   1434 ;*******************************************************************************
001F   1435 ;++
001F   1436 ;
001F   1437 ; FUNCTIONAL DESCRIPTION:
001F   1438 ;
001F   1439 ;       This test will check the first array card writing first 0's then
001F   1440 ;       1's in decending order.
001F   1441 ;
001F   1442 ;
001F   1443 ; TEST ALGORITHM:
001F   1444 ;
001F   1445 ;       1.  Load U-code to initialize test conditions
001F   1446 ;       2.  Execute dcs program
001F   1447 ;               a.  Disable the CMI
001F   1448 ;               b.  Enable the cache
001F   1449 ;               c.  Set VA to zero
001F   1450 ;               d.  Validate cache addresses 0 and 4
001F   1451 ;               e.  Load the PC to cause prefetch of addresses 0 and 4
001F   1452 ;               f.  Disable control P interrupts
001F   1453 ;               g.  Enable CMI and disable cache
001F   1454 ;               h.  Clear memory CSR0
001F   1455 ;               i.  Set memory CSR1 to allow single bit errors
001F   1456 ;               j.  Read CSR2
001F   1457 ;               k.  Perform IRD1TEST to free interrupt pending logic
001F   1458 ;               l.  Set VA to the last address of array of the first array as
001F   1459 ;                   if it were a 256kb module
001F   1460 ;       3.  Check low bits 1:0 of CSR2
001F   1461 ;               a.  If this is a 256kb array then continue because all
001F   1462 ;                   addresses default for this type module
001F   1463 ;               b.  If this is a 1mb array then do the following:
001F   1464 ;                       i.  Load different microword so VA gets loaded with
001F   1465 ;                           R[TEMP13] which contains last address for this
001F   1466 ;                           array
001F   1467 ;                      ii.  Change microword in memory to specify R[TEMP13]
001F   1468 ;                     iii.  Execute microword that was changed in the first
001F   1469 ;                           step
001F   1470 ;               c.  If this is a 4mb array, then do the following:
001F   1471 ;                       i.  Change RTEMP13 to 3FFFFC
001F   1472 ;                      ii.  Load different microword so VA gets loaded with
001F   1473 ;                           R[TEMP13] which contains last address for this
001F   1474 ;                           array
001F   1475 ;                     iii.  Change microword in memory to specify R[TEMP13]
001F   1476 ;                    iiii.  Execute microword that was changed in the second
001F   1477 ;                           step
001F   1478 ;       4.  Load index I with 2 which will be used for comparison of index K
001F   1479 ;       5.  Load U-code to write zeros to main memory
001F   1480 ;       6.  Load the first u-inst.
001F   1481 ;       7.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
001F   1482 ;           error section as follows:
001F   1483 ;               a.  First pass is from the writing of zeros throughout memory
001F   1484 ;               b.  Second pass is from the read-write-read throughout memory
001F   1485 ;       8.  Check to see if this is first pass or second pass, if first pass
001F   1486 ;           then goto 112$, if second pass then goto 110$
001F   1487 ;       9.  Set-up the rdm to watch for u-traps
```

```
001F  1488 ;      10.  Set the u-match halt for 1801
001F  1489 ;      11.  Start the cpu clock
001F  1490 ;               a.  Save the VA in the RDM WH register
001F  1491 ;               b.  Write zero to the address in the VA
001F  1492 ;               c.  Test VA to see if at end of first array card
001F  1493 ;               d.  If not decrement VA and return to DCS address 0
001F  1494 ;      12.  Set-up watchdog timer with loop J to monitor above U-code
001F  1495 ;      13.  If loop J expires, stop the clock and force an error printout
001F  1496 ;      14.  If clock stops normally (U-MATCH AT 1801); test WH register for
001F  1497 ;           end of first array
001F  1498 ;      15.  If no error test cs address for not a u-trap
001F  1499 ;      16.  Increment loop K for second pass
001F  1500 ;      17.  Execute DCS code
001F  1501 ;               a.  Set VA to last address of the array
001F  1502 ;      18.  Load the test U-code
001F  1503 ;      19.  Load the first U-inst.
001F  1504 ;      20.  Set-up the RDM to watch for u-traps
001F  1505 ;      21.  Set the u-match to 1801
001F  1506 ;      22.  Start the cpu clock
001F  1507 ;               a.  Save VA in RDM WH register
001F  1508 ;               b.  Read address specified by VA
001F  1509 ;               c.  Move data to RTEMP0
001F  1510 ;               d.  Test for a corrected data interrupt
001F  1511 ;               e.  Test that data is correct
001F  1512 ;               f.  Write ones to the same location
001F  1513 ;               g.  Read location back
001F  1514 ;               h.  Save data in RTEMP0
001F  1515 ;               i.  Test for a corrected data interrupt
001F  1516 ;               j.  Test that data is correct
001F  1517 ;               k.  Test to see if at end of first array
001F  1518 ;               l.  Decrement the VA by 4
001F  1519 ;               m.  Return to DCS address 0
001F  1520 ;      23.  Set-up watch-dog timer with loop J to monitor above U-code
001F  1521 ;      24.  If loop J expires, stop the clock and force an error printout
001F  1522 ;      25.  If clock stops normally (U-MATCH AT 1801) test WH register
001F  1523 ;           for end of array.
001F  1524 ;      26.  Test for machine checks (double bit error in memory)
001F  1525 ;      27.  If not machine check test for incorrect data on first read
001F  1526 ;      28.  If not bad data test for incorrect data on second read
001F  1527 ;      29.  Test for corrected data interrupts
001F  1528 ;      30.  Service the corrected data interrut with program at DCS address 14
001F  1529 ;               a.  Save VA in RTEMP7
001F  1530 ;               b.  Disable CMI enable cache
001F  1531 ;               c.  Set VA to zero
001F  1532 ;               d.  Validate cache address 0 and 4
001F  1533 ;               e.  Load PC to cause prefetch of addresses 0 and 4
001F  1534 ;               f.  Disable ^P interrupts
001F  1535 ;               g.  Read memory CSR0 to the RDM MHREG
001F  1536 ;               h.  Clear memory CSR0
001F  1537 ;               i.  Clear any interrupts pending
001F  1538 ;               j.  Perform an IRD1TEST to free the interrupt pending logic
001F  1539 ;               k.  Reload VA with RTEMP7
001F  1540 ;      31.  Continue running test micro code from step 17.
001F  1541 ;      32.  Test for the end of the first array.
001F  1542 ;      33.  When last address of array is found the test will end on ENDLOOP K
```

```
001F  1543 ;           being expired
001F  1544 ;      34.  The balance of the PSEUDO OPS handle the various error conditions
001F  1545 ;           (see error description)
001F  1546 ;
001F  1547 ;
001F  1548 ; TEST PATTERNS:
001F  1549 ;
001F  1550 ;       First pass:  Init memory to 0's
001F  1551 ;      Second pass:  read 0's; write 1's; read 1's
001F  1552 ;
001F  1553 ;
001F  1554 ; LOGIC DESCRIPTION:
001F  1555 ;
001F  1556 ;       My advice on this test is forget trying to troubleshoot with it.  It
001F  1557 ;       is far to complex to be a usefull troubleshooting tool.  Rather, just
001F  1558 ;       use the error printout to array and/or module swap till you find the
001F  1559 ;       problem.  The first array is throughly tested only after running
001F  1560 ;       all four tests in this series; together and in order.
001F  1561 ;
001F  1562 ;
001F  1563 ; ASSUMPTIONS:
001F  1564 ;
001F  1565 ;       This test assumes the CMI is operational.
001F  1566 ;
001F  1567 ;
001F  1568 ; ERROR DESCRIPTION:
001F  1569 ;
001F  1570 ;       First IFERROR:
001F  1571 ;
001F  1572 ;       EXPECTED LAST ADDRESS OF FIRST ARRAY
001F  1573 ;       RECEIVED LAST ADDRESS
001F  1574 ;       LOOP COUNT I
001F  1575 ;       LOOP COUNT J
001F  1576 ;       LOOP COUNT K
001F  1577 ;       (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
001F  1578 ;
001F  1579 ;       Second IFERROR:
001F  1580 ;
001F  1581 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  1582 ;       MAIN MEMORY ADDRESS OF DATA ERROR
001F  1583 ;       LOOP COUNT I
001F  1584 ;       LOOP COUNT J
001F  1585 ;       LOOP COUNT K
001F  1586 ;       RECEIVED DATA (S/B 0'S OR 1'S)
001F  1587 ;       (INDICATES ECC IS PROBABLE NOT WORKING)
001F  1588 ;
001F  1589 ;       Third IFERROR:
001F  1590 ;
001F  1591 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  1592 ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
001F  1593 ;       LOOP COUNT I
001F  1594 ;       LOOP COUNT J
001F  1595 ;       LOOP COUNT K
001F  1596 ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
001F  1597 ;
```

```
001F  1598 ;--
001F  1599 ;*******************************************************************************
001F  1600
001F  1601 ;-------------------------------------------------------------------------------
001F  1602 ;
001F  1603 ; Initialize code
001F  1604 ;
001F  1605 ;-------------------------------------------------------------------------------
001F  1606
001F  1607          INITIALIZE                                  ;init the cpu
0021  1608          LOADDCS         40$,,0,<^X1C>               ;setup micro code
0028  1609          FETCH           0                           ;start dcs program
002D  1610          BRSTCLK         <^X1B>,INH                  ;end dcs program
0031  1611          CMPREGMSK       WHREG,23$,,18$,,B           ;check array 0
003D  1612          SKIP            100$,NOERROR                ;skip if 4mb module
0044  1613          CMPREGMSK       WHREG,21$,,18$,,B,NE        ;check array 0 type
0050  1614          SKIP            102$,ONERROR                ;skip if 1mb module
0057  1615          SKIP            108$                        ;go test
005E  1616 ;
005E  1617 ; This section will set up for a 4mb array
005E  1618 ;
005E  1619 100$:    LOADDCS         45$,,0,4                    ;RTEMP13 = last addr 3FFFFC
0065  1620          FETCH           0                           ;start dcs
006A  1621          BRSTCLK         3                           ;end dcs
006E  1622 ;
006E  1623 ; This section will set-up for a 1mb module versus a 256kb module
006E  1624 ;
006E  1625 102$:    LOADDCS         70$,,<^X1A>,1               ;load VA with last addr. FFFFC
0075  1626          LDFIELD         22$,,B,55$,34,39            ;change RSRC fld for R[TEMP13]
0086  1627          FETCH           <^X1A>                      ;start dcs program
008B  1628          BRSTCLK         <^X1B>,INH                  ;end dcs program
008F  1629
008F  1630 ;-------------------------------------------------------------------------------
008F  1631 ;
008F  1632 ; Write memory with 0's
008F  1633 ;
008F  1634 ;-------------------------------------------------------------------------------
008F  1635
008F  1636 108$:    LDINDEX         I,2                         ;dummy index I gets 2
0096  1637          LOADDCS         50$,,0,<^X0B>               ;test u-code
009D  1638          FETCH           0                           ;load first u-inst.
00A2  1639          LOOP            K,1,2                       ;set index K for two passes
00A9  1640          SKIP            110$,ONEQUAL,I,K            ;skip on second pass
00B0  1641          SKIP            112$                        ;first pass, goto rdm set-up
00B7  1642
00B7  1643 ;-------------------------------------------------------------------------------
00B7  1644 ;
00B7  1645 ; Read the 0's from the previous write, check for valid data, write 1's, check
00B7  1646 ; for valid data
00B7  1647 ;
00B7  1648 ;-------------------------------------------------------------------------------
00B7  1649
00B7  1650 110$:    FETCH           8                           ;reset va to last address
00BC  1651          BRSTCLK         <^X0A>,INH                  ;...
00C0  1652          LOADDCS         60$,,0,<^X2F>               ;test u-code
```

```
00C7  1653          FETCH           0                       ;load first u-inst.
00CC  1654 ;
00CC  1655 ; This section is used for both the initial writing of 0's (first pass) then
00CC  1656 ; for the read of 0's-write of 1's-read of 1's (second pass)
00CC  1657 ;
00CC  1658 112$:    LOADREG         DCSCR,9$,,B             ;insure clock is off
00D4  1659          LOADREG         CSAMR,2$,,W             ;load match address
00DC  1660          LOADREG         RDCTRL,7$,,B            ;clear master halt enable
00E4  1661          LOADREG         RDCTRL,1$,,B            ;watch for u-traps
00EC  1662          LOADREG         DCSCR,3$,,B             ;start the clock
00F4  1663          LOOP            J,1,8192                ;watchdog timing loop
00FB  1664          CMPREGMSK       CLKDCS,7$,,8$,,B,       ;test for clock stopped
0107  1665          SKIP            114$,ONERROR            ;skip when clock stops
010E  1666          ENDLOOP         J                       ;end timing loop
0111  1667          LOADREG         DCSCR,9$,,B             ;stop the clock
0119  1668          CMPREG          WHREG,4$,,L,            ;force an error
0122  1669          IFERROR         ,,<<MC0><MC1><MC2>>     ;program lost
012A  1670          SKIP
0131  1671
0131  1672 ;-------------------------------------------------------------------------------
0131  1673 ;
0131  1674 ; Error evaluation code
0131  1675 ;
0131  1676 ;-------------------------------------------------------------------------------
0131  1677
0131  1678 114$:    CMPREGMSK       CSTRP,10$,,6$,,W,NE     ;test for machine check
013D  1679          SKIP            140$,ONERROR            ;handle machine check
0144  1680          CMPREGMSK       CLKDCS,11$,,12$,,B,NE   ;test for data errors
0150  1681          SKIP            130$,ONERROR            ;incorrect data
0157  1682          CMPREGMSK       CLKDCS,13$,,12$,,B,NE   ;test for data errors
0163  1683          SKIP            130$,ONERROR            ;incorrect data
016A  1684          CMPREGMSK       CLKDCS,14$,,12$,,B,NE   ;test for single bit error on 0
0176  1685          SKIP            120$,ONERROR            ;log error
017D  1686          CMPREGMSK       CLKDCS,16$,,12$,,B,NE   ;test for single bit error on 1
0189  1687          SKIP            115$,ONERROR            ;log error
0190  1688          CMPREG          WHREG,4$,,L,            ;test for last address
0199  1689          SKIP            150$,NOERROR            ;go to next test
01A0  1690          CMPREGMSK       CLKDCS,17$,,12$,,B      ;force and error
01AC  1691          IFERROR                                 ;i think we're lost
01B1  1692          SKIP                                    ;end test due to error
01B8  1693 ;
01B8  1694 ; Come here if the Second read caused corrected read data to be returned
01B8  1695 ;
01B8  1696 115$:    FETCH           <^X14>                  ;start dcs program
01BD  1697          BRSTCLK         <^X2E>                  ;end dcs program
01C1  1698          ERRLOG                                  ;log the error
01C3  1699          SKIP            150$,ONERROR            ;quit after 64 errors
01CA  1700          FETCH           <^X10>                  ;set up to continue testing
01CF  1701          SKIP            112$                    ;test more memory
01D6  1702 ;
01D6  1703 ; Come here if the First read caused corrected read data to be returned
01D6  1704 ;
01D6  1705 120$:    FETCH           <^X14>                  ;start dcs program
01DB  1706          BRSTCLK         <^X2E>                  ;end dcs program
01DF  1707          ERRLOG                                  ;log the error
```

```
                01E1  1708          SKIP            150$,ONERROR                ;quit after 64 errors
                01E8  1709          FETCH           <^X07>                      ;set up to continue testing
                01ED  1710          SKIP            112$                        ;test more memory
                01F4  1711 ;
                01F4  1712 ; Come here if the data read is not the data written
                01F4  1713 ;
                01F4  1714 130$:    CMPREG          WHREG,15$,,L,               ;force an error
                01FD  1715          IFERROR         1,<<MEC><MDL>>,-
                01FD  1716          <<MC0><MC1><MC2><MA0><MA1><MA2>>            ;print va and data
                020A  1717          SKIP                                        ;end test because of error
                0211  1718 ;
                0211  1719 ; Come here if there was a Machine check (Read Data Substitute)
                0211  1720 ;
                0211  1721 140$:    CMPREG          WHREG,15$,,L,               ;force an error
                021A  1722          IFERROR         ,<<MEC><MDL>>,-
                021A  1723          <<MC0><MC1><MC2><MA0><MA1><MA2>>            ;print va of machine check
                0227  1724 ;-------------------------------------------------------------------------------
                0227  1725 ;
                0227  1726 ; This is where we exit the test if index K is equal to 2
                0227  1727 ;
                0227  1728 ;-------------------------------------------------------------------------------
                0227  1729
                0227  1730 150$:    ENDLOOP         K
                022A  1731          SKIP                                        ;go to next test
                0231  1732
                0231  1733
                0231  1734 BEGIN_TEST_DATA
                0231
                0231      ;-------------------------------------------------------------------------------
                0231  1735
          28    0231  1736 1$:      .BYTE   ^X28                                ;set to watch for u-traps
        1801    0232  1737 2$:      .WORD   ^X1801                              ;match address
          11    0234  1738 3$:      .BYTE   ^X11                                ;run clock
    00000000    0235  1739 4$:      .LONG   ^X00000000                          ;last address of first array
        3FFF    0239  1740 6$:      .WORD   ^X3FFF                              ;mask for cmpregmsk pseudo op
          00    023B  1741 7$:      .BYTE   ^X00                                ;watch for clock stop
          C0    023C  1742 8$:      .BYTE   ^XC0                                ;mask for cmpregmsk pseudo op
          60    023D  1743 9$:      .BYTE   ^X60                                ;stop clock manually
        0028    023E  1744 10$:     .WORD   ^X0028                              ;machine check u-trap
          09    0240  1745 11$:     .BYTE   ^X09                                ;dcs halt on data error
          3F    0241  1746 12$:     .BYTE   ^X3F                                ;mask for cmpregmsk pseudo op
          12    0242  1747 13$:     .BYTE   ^X12                                ;dcs halt on data error
          08    0243  1748 14$:     .BYTE   ^X08                                ;halt single bit error on 0's
    0F0F0F0F    0244  1749 15$:     .LONG   ^X0F0F0F0F                          ;compare data to force errors
          11    0248  1750 16$:     .BYTE   ^X11                                ;halt single bit error on 1's
          FF    0249  1751 17$:     .BYTE   ^XFF                                ;used to force errors
          03    024A  1752 18$:     .BYTE   ^X03                                ;array 0 mask
          02    024B  1753 21$:     .BYTE   ^X02                                ;array 0 1mb (CSR2 bits 0:1)
          0D    024C  1754 22$:     .BYTE   ^X0D                                ;RSRC = TEMP13
          01    024D  1755 23$:     .BYTE   ^X01                                ;array 0 4mb
                024E  1756 40$:
U 00, 7700,C800,0364,0300,0470,1801   024E  1757 ;* 00    NOP
U 01, 7700,8800,5BE4,0304,6870,1802   0259  1761 ;* 01    MEMSCAR_R[TEMP1]                ;disable cmi
U 02, 7700,C800,5BE4,030C,6070,1803   0264  1765 ;* 02    MEMSCR_R[TEMP3]                 ;...
U 03, 7700,8800,5BE4,0308,6870,1804   026F  1769 ;* 03    MEMSCAR_R[TEMP2]                ;turn on cache
```

```
U 04. 7700,C800,5BE4,03D8,6070,1805  027A  1773 ;* 04   MEMSCR_R[ZERO]                                  ;...
U 05. 7700,4800,5BE4,03D8,4A70,1806  0285  1777 ;* 05   VA_R[ZERO]                                      ;zero the va
U 06. 7700,4800,5BE4,03D8,5C80,1807  0290  1781 ;* 06   WRITE.PHY R[ZERO]                               ;validate cache address 0
U 07. 7700,C800,5BE4,03D8,5470,1808  029B  1785 ;* 07   WDR_UNROT(R[ZERO])                             ;...
U 08. 7700,C800,0364,0300,4470,1809  02A6  1789 ;* 08   VA_VA+4                                        ;increment va by 4
U 09. 7700,C800,5BE4,03D8,5C80,180A  02B1  1793 ;* 09   WRITE.PHY R[ZERO]                               ;validate cache address 4
U 0A. 7700,C800,5BE4,03D8,5470,180B  02BC  1797 ;* 0A   WDR_UNROT(R[ZERO])
U 0B. 7700,C800,5BE4,03D8,4870,180C  02C7  1801 ;* 0B   PC_R[ZERO]                                     ;cause prefetch
U 0C. 7700,0800,5BE4,032C,2470,180D  02D2  1805 ;* 0C   WCTRL/LOADCRAR,WB_R[TEMP11]                    ;disable ^p interrupts
U 0D. 7700,C800,5BE4,0330,2C70,180E  02DD  1809 ;* 0D   WCTRL/CONWRITE,WB_R[TEMP12]                    ;...
U 0E. 7700,0800,5BE4,0304,6870,180F  02E8  1813 ;* 0E   MEMSCAR_R[TEMP1]                               ;turn on cmi
U 0F. 7700,4800,5BE4,03D8,6070,1810  02F3  1817 ;* 0F   MEMSCR_R[ZERO]                                ;...
U 10. 7700,0800,5BE4,0308,6870,1811  02FE  1821 ;* 10   MEMSCAR_R[TEMP2]                               ;disable cache
U 11. 7700,C800,5BE4,030C,6070,1812  0309  1825 ;* 11   MEMSCR_R[TEMP3]                                ;...
U 12. 7700,8800,5BE4,0320,4A70,1813  0314  1829 ;* 12   VA_R[TEMP8]                                    ;CSR0 address to va
U 13. 7700,C800,5BE4,0324,5C80,1814  031F  1833 ;* 13   WRITE.PHY R[TEMP9]                               ;clear CSR0
U 14. 7700,4800,0364,0300,4470,1815  032A  1837 ;* 14   VA_VA+4                                        ;CSR1 address to va
U 15. 7700,4800,5BE4,0328,5C80,1816  0335  1841 ;* 15   WRITE.PHY R[TEMP10]                              ;allow single bit errors
U 16. 7700,C800,0364,0300,4470,1817  0340  1845 ;* 16   VA_VA+4                                        ;CSR2 address to va
U 17. 7700,C800,0364,0300,0400,1818  034B  1849 ;* 17   READ.PHY                                       ;read CSR2
U 18. 6700,8812,5924,0300,0470,1819  0356  1853 ;* 18   RDM_WB,WB_M[MDR]                               ;WHREG gets contents of CSR2
U 19. 7700,C800,0364,1700,0470,181A  0361  1857 ;* 19   IRDITEST                                      ;clear int pending line
U 1A. 7700,4800,5BE4,0318,4A70,181B  036C  1861 ;* 1A   VA_R[TEMP6]                                    ;set VA to array 0 last address
U 1B. 7300,C800,0364,0300,0470,181C  0377  1865 ;* 1B   NOP,STOP.CLOCK                                 ;stop the cpu clock
                                     0382  1869 45$:
U 00. 7700,B800,7FE0,0001,8470,1801  0382  1870 ;* 00   LONLIT_[003FFFFC]                               ;last addr for 4mb
U 01. 7700,8868,5BE4,03D4,0470,1802  038D  1874 ;* 01   M[TEMP8]_R[LONLIT]                              ;
U 02. 7700,4848,5924,0334,0470,1803  0398  1878 ;* 02   R[TEMP13]_M[TEMP8]                             ;RTEMP13 = last addr 3FFFFC
U 03. 7300,C800,0364,0300,0470,1804  03A3  1882 ;* 03   NOP,STOP.CLOCK                                 ;stop cpu clock
                                     03AE  1886 50$:
U 00. 7700,C800,0364,0300,0470,1802  03AE  1887 ;* 00   NOP,NEXT/1802
U 01. 6700,881B,5924,0300,0470,1802  03B9  1891 ;* 01   RDM_WB,WB_M[VA]                                ;save va in rdm
U 02. 7700,8800,5BE4,0310,5C80,1803  03C4  1895 ;* 02   WRITE.PHY R[TEMP4]                              ;write zeros to memory
U 03. 7700,C800,0364,0300,0470,1804  03CF  1899 ;* 03   NOP                                           ;allow time for write
U 04. 7700,4800,0364,0300,0470,1805  03DA  1903 ;* 04   NOP                                           ;...
U 05. 7700,081B,0034,A310,0470,1800  03E5  1907 ;* 05   WB_M[VA].XOR.R[TEMP4],BUT/WX.EQ.0,NEXT/1800
                                     03F0  1911                                                       ;test for end of first array
U 06. 7700,D81B,C100,0302,4A70,1807  03F0  1912 ;* 06   VA_VA-ZLIT0[4]                                ;decrement va register
U 07. 7500,C800,0364,0300,0470,1808  03FB  1916 ;* 07   NOP,DCS.ADDR_0                                ;return to address 0
U 08. 7700,4800,0364,0300,0470,1809  0406  1920 ;* 08   NOP
                                     0411  1924 55$:
U 09. 7700,4800,5BE4,0318,4A70,180A  0411  1925 ;* 09   VA_R[TEMP6]                                    ;set va to array last address
U 0A. 7300,C800,0364,0300,0470,180B  041C  1929 ;* 0A   NOP,STOP.CLOCK                                 ;stop the cpu clock
                                     0427  1933 60$:
U 00. 7700,C800,0364,0300,0470,1802  0427  1934 ;* 00   NOP,NEXT/1802                                  ;allow time for memory
U 01. 7700,C800,0364,0300,0470,1802  0432  1938 ;* 01   NOP                                           ;clock to start up.
U 02. 7700,4800,0364,0300,0470,1803  043D  1942 ;* 02   NOP                                           ;...
U 03. 6700,881B,5924,0300,0470,1804  0448  1946 ;* 03   RDM_WB,WB_M[VA]                                ;save va in rdm
U 04. 7700,C800,0364,0300,0400,1805  0453  1950 ;* 04   READ.PHY                                       ;read pattern back
U 05. 7700,4852,5924,0300,0470,1806  045E  1954 ;* 05   R[TEMP0]_M[MDR]                               ;save data in RTEMP0
U 06. 7700,C800,0364,AF00,0470,1801  0469  1958 ;* 06   BUT/INT-TIMSERV,NEXT/1801                    ;test for corrected data
U 07. 7700,4800,0034,A710,0470,1800  0474  1962 ;* 07   WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                     047F  1966                                                       ;test for good data
U 08. 7700,4800,0364,0300,0470,1809  047F  1967 ;* 08   NOP                                           ;allow time for memory clock
U 09. 7700,4800,0364,0300,0470,180A  048A  1971 ;* 09   NOP                                           ;to start up after single bit
```

```
U 0A. 7700.C800.0364.0300.0470.180B  0495  1975 ;% 0A   NOP                                          ;errors.
U 0B. 7700.4800.0364.0300.0470.180C  04A0  1979 ;% 0B   NOP                                          ;...
U 0C. 7700.4800.5BE4.0314.5C80.180D  04AB  1983 ;% 0C   WRITE.PHY R[TEMP5]                            ;write ones to memory
U 0D. 7700.4800.0364.0300.0400.180E  04B6  1987 ;% 0D   READ.PHY                                     ;read pattern back
U 0E. 7700.4852.5924.0300.0470.180F  04C1  1991 ;% 0E   R[TEMP0]_M[MDR]                              ;save data in RTEMP0
U 0F. 7700.C800.0364.AF00.0470.1801  04CC  1995 ;% 0F   BUT/INT-TIMSERV,NEXT/1801                   ;test for corrected data
U 10. 7700.0800.0034.A714.0470.1800  04D7  1999 ;% 10   WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                     04E2  2003                                                   ;test for good data
U 11. 7700.081B.0034.A310.0470.1800  04E2  2004 ;% 11   WB_M[VA].XOR.R[TEMP4],BUT/WX.EQ.0,NEXT/1800
                                     04ED  2008                                                   ;test for end of first array
U 12. 7700.D81B.C100.0302.4A70.1813  04ED  2009 ;% 12   VA_VA-ZLIT0[4]                              ;decrement va
U 13. 7500.4800.0364.0300.0470.1814  04F8  2013 ;% 13   NOP,DCS.ADDR_0                               ;return to dcs address 0
U 14. 7700.C800.0364.0300.0470.1815  0503  2017 ;% 14   NOP
U 15. 7700.885B.5924.031C.0470.1816  050E  2021 ;% 15   R[TEMP7]_M[VA]                               ;save va for return
U 16. 7700.0800.5BE4.0304.6870.1817  0519  2025 ;% 16   MEMSCAR_R[TEMP1]                             ;disable cmi
U 17. 7700.C800.5BE4.030C.6070.1818  0524  2029 ;% 17   MEMSCR_R[TEMP3]                              ;...
U 18. 7700.8800.5BE4.0308.6870.1819  052F  2033 ;% 18   MEMSCAR_R[TEMP2]                             ;turn on cache
U 19. 7700.4800.5BE4.03D8.6070.181A  053A  2037 ;% 19   MEMSCR_R[ZERO]                               ;...
U 1A. 7700.4800.5BE4.03D8.4A70.181B  0545  2041 ;% 1A   VA_R[ZERO]                                  ;zero the va
U 1B. 7700.4800.5BE4.03D8.5C80.181C  0550  2045 ;% 1B   WRITE.PHY R[ZERO]                            ;validate cache address 0
U 1C. 7700.4800.5BE4.03D8.5470.181D  055B  2049 ;% 1C   WDR_UNROT(R[ZERO])                           ;...
U 1D. 7700.C800.0364.0300.4470.181E  0566  2053 ;% 1D   VA_VA+4                                     ;increment va by 4
U 1E. 7700.4800.5BE4.03D8.5C80.181F  0571  2057 ;% 1E   WRITE.PHY R[ZERO]                            ;validate cache address 4
U 1F. 7700.C800.5BE4.03D8.5470.1820  057C  2061 ;% 1F   WDR_UNROT(R[ZERO])
U 20. 7700.C800.5BE4.03D8.4870.1821  0587  2065 ;% 20   PC_R[ZERO]                                  ;cause prefetch
U 21. 7700.8800.5BE4.032C.2470.1822  0592  2069 ;% 21   WCTRL/LOADCRAR,WB_R[TEMP11]                 ;disable ^P interrupts
U 22. 7700.C800.5BE4.0330.2C70.1823  059D  2073 ;% 22   WCTRL/CONWRITE,WB_R[TEMP12]                 ;...
U 23. 7700.0800.5BE4.0304.6870.1824  05A8  2077 ;% 23   MEMSCAR_R[TEMP1]                             ;turn on cmi
U 24. 7700.4800.5BE4.03D8.6070.1825  05B3  2081 ;% 24   MEMSCR_R[ZERO]                               ;...
U 25. 7700.8800.5BE4.0308.6870.1826  05BE  2085 ;% 25   MEMSCAR_R[TEMP2]                             ;disable cache
U 26. 7700.C800.5BE4.030C.6070.1827  05C9  2089 ;% 26   MEMSCR_R[TEMP3]                              ;...
U 27. 7700.0800.5BE4.0320.4A70.1828  05D4  2093 ;% 27   VA_R[TEMP8]                                  ;address of CSR0
U 28. 7700.4800.0364.0300.0400.1829  05DF  2097 ;% 28   READ.PHY                                     ;read CSR0
U 29. 3700.8812.5924.0300.0470.182A  05EA  2101 ;% 29   WB_M[MDR],RDM_CMI                           ;MHREG gets CSR0
U 2A. 7700.C800.5BE4.0324.5C80.182B  05F5  2105 ;% 2A   WRITE.PHY R[TEMP9]                          ;clear CSR0
U 2B. 7700.C800.0364.7B00.0470.182C  0600  2109 ;% 2B   BUT/UVCTR                                  ;clear any interrupts pending
U 2C. 7700.4800.0364.1700.0470.182D  060B  2113 ;% 2C   IRD1TEST                                   ;free interrupt pending logic
U 2D. 7700.0800.5BE4.031C.4A70.182E  0616  2117 ;% 2D   VA_R[TEMP7]                                ;reset the va
U 2E. 7300.C800.0364.0300.0470.182F  0621  2121 ;% 2E   NOP,STOP.CLOCK                              ;stop the cpu clock
                                     062C  2125 70$:
U 1A. 7700.0800.5BE4.0334.4A70.181B  062C  2126 ;% 1A   VA_R[TEMP13]                                ;va gets array last addr FFFFC
                                     0637  2130
                                     0637  2131 BEGIN_CPU_DATA
                                     0002  2132
                                     0002  2133 RTEMP_DATA
                                     0001  2134
                          00000000   0001  2135         .LONG   ^X00000000                       ;data pattern storage
                          0E000000   0005  2136         .LONG   ^X0E000000                       ;reference cache only address
                          06000000   0009  2137         .LONG   ^X06000000                       ;cache control register address
                          01000000   000D  2138         .LONG   ^X01000000                       ;disable cache
                          00000000   0011  2139         .LONG   ^X00000000                       ;test pattern 1
                          FFFFFFFF   0015  2140         .LONG   ^XFFFFFFFF                       ;test pattern 2
                          0003FFFC   0019  2141         .LONG   ^X0003FFFC                       ;last address for 256kb array 0
                          00000000   001D  2142         .LONG   ^X00000000                       ;storage for va register
                          00F20000   0021  2143         .LONG   ^X00F20000                       ;address of CSR0
```

```
                    E0000000  0025  2144          .LONG   ^XE0000000                      ;clears CSR0
                    10000000  0029  2145          .LONG   ^X10000000                      ;allow single bit error report
                    00C00000  002D  2146          .LONG   ^X00C00000                      ;disable ^P interrupts
                    00400000  0031  2147          .LONG   ^X00400000                      ;...
                    000FFFFC  0035  2148          .LONG   ^X000FFFFC                      ;last address for 1mb array 0
                              0039  2149
                              0039  2150 END_CPU_DATA
                              0637  2151
                              0637  2152
                              0637  2153 END_TEST_DATA
                              0637
                              0637      ;-------------------------------------------------------------------------
                              0637  2154 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
001F                    .SBTTL  "TEST   085     Array 0 Memory Test Part 4
001F  2156 ;*****************************************************************************
001F  2157 ;++
001F  2158 ;
001F  2159 ; FUNCTIONAL DESCRIPTION:
001F  2160 ;
001F  2161 ;       This test will check the first array card writing first 1's then
001F  2162 ;       0's in decending order.
001F  2163 ;
001F  2164 ;
001F  2165 ; TEST ALGORITHM:
001F  2166 ;
001F  2167 ;       1.  Load U-code to initialize test conditions
001F  2168 ;       2.  Execute dcs program
001F  2169 ;               a.  Disable the CMI
001F  2170 ;               b.  Enable the cache
001F  2171 ;               c.  Set VA to zero
001F  2172 ;               d.  Validate cache addresses 0 and 4
001F  2173 ;               e.  Load the PC to cause prefetch of addresses 0 and 4
001F  2174 ;               f.  Disable control P interrupts
001F  2175 ;               g.  Enable CMI and disable cache
001F  2176 ;               h.  Clear memory CSR0
001F  2177 ;               i.  Set memory CSR1 to allow single bit errors
001F  2178 ;               j.  Read CSR2
001F  2179 ;               k.  Perform IRD1TEST to free interrupt pending logic
001F  2180 ;               l.  Set VA to last address of array
001F  2181 ;       3.  Check low bits 1:0 of CSR2
001F  2182 ;               a.  If this is a 256kb array then continue because all
001F  2183 ;                   addresses default for this type module
001F  2184 ;               b.  If this is a 1mb array then do the following:
001F  2185 ;                       i.  Load different microword so VA gets loaded with
001F  2186 ;                           R[TEMP13] which contains last address for this
001F  2187 ;                           array
001F  2188 ;                      ii.  Change microword in memory to specify R[TEMP13]
001F  2189 ;                     iii.  Execute microword that was changed in the first
001F  2190 ;                           step
001F  2191 ;               c.  If this is a 4mb array, then do the following:
001F  2192 ;                       i.  Change RTEMP13 to 3FFFFC
001F  2193 ;                      ii.  Load different microword so VA gets loaded with
001F  2194 ;                           R[TEMP13] which contains the last address for
001F  2195 ;                           this array
001F  2196 ;                     iii.  Change microword in memory to specify R[TEMP13]
001F  2197 ;                    iiii.  Execute microword that was changed in the first
001F  2198 ;                           step
001F  2199 ;       4.  Load index I with 2 which will be used for comparison of index K
001F  2200 ;       5.  Load U-code to write ones to main memory
001F  2201 ;       6.  Load the first u-inst.
001F  2202 ;       7.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
001F  2203 ;           error section as follows:
001F  2204 ;               a.  First pass is from the writing of ones throughout memory
001F  2205 ;               b.  Second pass is from the read-write-read throughout memory
001F  2206 ;       8.  Check to see if this is first pass or second pass, if first pass
001F  2207 ;           then goto 112$, if second pass then goto 110$
001F  2208 ;       9.  Set-up the rdm to watch for u-traps
001F  2209 ;      10.  Set the u-match halt for 1801
```

```
001F  2210 ;      11.  Start the cpu clock
001F  2211 ;               a.  Save the VA in the RDM WH register
001F  2212 ;               b.  Write ones to the address in the VA
001F  2213 ;               c.  Test VA to see if at end of first array card
001F  2214 ;               d.  If not decrement VA and return to DCS address 0
001F  2215 ;      12.  Set-up watchdog timer with loop J to monitor above U-code
001F  2216 ;      13.  If loop J expires, stop the clock and force an error printout
001F  2217 ;      14.  If clock stops normally (U-MATCH AT 1801); test WH register for
001F  2218 ;           end of first array
001F  2219 ;      15.  If no error test cs address for not a u-trap
001F  2220 ;      16.  Increment loop K for second pass
001F  2221 ;      17.  Execute DCS code
001F  2222 ;               a.  Set VA to last address of the array
001F  2223 ;      18.  Load the test U-code
001F  2224 ;      19.  Load the first U-inst.
001F  2225 ;      20.  Set-up the RDM to watch for u-traps
001F  2226 ;      21.  Set the u-match to 1801
001F  2227 ;      22.  Start the cpu clock
001F  2228 ;               a.  Save VA in RDM WH register
001F  2229 ;               b.  Read address specified by VA
001F  2230 ;               c.  Move data to RTEMP0
001F  2231 ;               d.  Test for a corrected data interrupt
001F  2232 ;               e.  Test that data is correct
001F  2233 ;               f.  Write zero to the same location
001F  2234 ;               g.  Read location back
001F  2235 ;               h.  Save data in RTEMP0
001F  2236 ;               i.  Test for a corrected data interrupt
001F  2237 ;               j.  Test that data is correct
001F  2238 ;               k.  Test to see if at end of first array
001F  2239 ;               l.  Decrement the VA by 4
001F  2240 ;               m.  Return to DCS address 0
001F  2241 ;      23.  Set-up watch-dog timer with loop J to monitor above U-code
001F  2242 ;      24.  If loop J expires, stop the clock and force an error printout
001F  2243 ;      25.  If clock stops normally (U-MATCH AT 1801) test WH register
001F  2244 ;           for end of array.
001F  2245 ;      26.  Test for machine checks (double bit error in memory)
001F  2246 ;      27.  If not machine check test for incorrect data on first read
001F  2247 ;      28.  If not bad data test for incorrect data on second read
001F  2248 ;      29.  Test for corrected data interrupts
001F  2249 ;      30.  Service the corrected data interrut with program at DCS address 14
001F  2250 ;               a.  Save VA in RTEMP7
001F  2251 ;               b.  Disable CMI enable cache
001F  2252 ;               c.  Set VA to zero
001F  2253 ;               d.  Validate cache address 0 and 4
001F  2254 ;               e.  Load PC to cause prefetch of addresses 0 and 4
001F  2255 ;               f.  Disable ^P interrupts
001F  2256 ;               g.  Read memory CSR0 to the RDM MHREG
001F  2257 ;               h.  Clear memory CSR0
001F  2258 ;               i.  Clear any interrupts pending
001F  2259 ;               j.  Perform an IRD1TEST to free the interrupt pending logic
001F  2260 ;               k.  Reload VA with RTEMP7
001F  2261 ;      31.  Continue running test micro code from step 17.
001F  2262 ;      32.  Test for the end of the first array.
001F  2263 ;      33.  When last address of array is found the test will end on ENDLOOP K
001F  2264 ;           being expired
```

```
001F  2265 ;       34.  When this test is completed (ENDLOOP K expired) then print message
001F  2266 ;            'FINISHED ARRAY 0' as this is the last test of 4 tests that check
001F  2267 ;            array 0
001F  2268 ;       35.  The balance of the PSEUDO OPS handle the various error conditions
001F  2269 ;            (see error description)
001F  2270 ;
001F  2271 ;
001F  2272 ; TEST PATTERNS:
001F  2273 ;
001F  2274 ;       First pass:  Init memory to 1's
001F  2275 ;      Second pass:  read 1's; write 0's; read 0's
001F  2276 ;
001F  2277 ;
001F  2278 ; LOGIC DESCRIPTION:
001F  2279 ;
001F  2280 ;       My advice on this test is forget trying to troubleshoot with it.  It
001F  2281 ;       is far to complex to be a usefull troubleshooting tool.  Rather, just
001F  2282 ;       use the error printout to array and/or module swap till you find the
001F  2283 ;       problem.  The first array is throughly tested only after running
001F  2284 ;       all four tests in this series; together and in order.
001F  2285 ;
001F  2286 ;
001F  2287 ; ASSUMPTIONS:
001F  2288 ;
001F  2289 ;       This test assumes the CMI is operational.
001F  2290 ;
001F  2291 ;
001F  2292 ; ERROR DESCRIPTION:
001F  2293 ;
001F  2294 ;       First IFERROR:
001F  2295 ;
001F  2296 ;       EXPECTED LAST ADDRESS OF FIRST ARRAY
001F  2297 ;       RECEIVED LAST ADDRESS
001F  2298 ;       LOOP COUNT I
001F  2299 ;       LOOP COUNT J
001F  2300 ;       LOOP COUNT K
001F  2301 ;       (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
001F  2302 ;
001F  2303 ;       Second IFERROR:
001F  2304 ;
001F  2305 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  2306 ;       MAIN MEMORY ADDRESS OF DATA ERROR
001F  2307 ;       LOOP COUNT I
001F  2308 ;       LOOP COUNT J
001F  2309 ;       LOOP COUNT K
001F  2310 ;       RECEIVED DATA (S/B 0'S OR 1'S)
001F  2311 ;       (INDICATES ECC IS PROBABLE NOT WORKING)
001F  2312 ;
001F  2313 ;       Third IFERROR:
001F  2314 ;
001F  2315 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  2316 ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
001F  2317 ;       LOOP COUNT I
001F  2318 ;       LOOP COUNT J
001F  2319 ;       LOOP COUNT K
```

```
001F  2320 ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
001F  2321 ;
001F  2322 ;--
001F  2323 ;*******************************************************************************
001F  2324
001F  2325 ;-------------------------------------------------------------------------------
001F  2326 ;
001F  2327 ; Initialize code
001F  2328 ;
001F  2329 ;-------------------------------------------------------------------------------
001F  2330
001F  2331          INITIALIZE                                  ;init the cpu
0021  2332          LOADDCS         40$,,0,<^X1C>               ;setup micro code
0028  2333          FETCH           0                           ;start dcs program
002D  2334          BRSTCLK         <^X1B>,INH                  ;end dcs program
0031  2335          CMPREGMSK       WHREG,23$,,18$,,B           ;check array 0
003D  2336          SKIP            100$,NOERROR                ;skip if 4mb
0044  2337          CMPREGMSK       WHREG,21$,,18$,,B,NE        ;check array 0 type
0050  2338          SKIP            102$,ONERROR                ;skip if 1mb module
0057  2339          SKIP            108$                        ;go test
005E  2340 ;
005E  2341 ; This section wil set up for a 4mb array
005E  2342 ;
005E  2343 100$:    LOADDCS         45$,,0,4                    ;RTEMP13 = last addr 3FFFFC
0065  2344          FETCH           0                           ;start dcs
006A  2345          BRSTCLK         3                           ;end dcs
006E  2346 ;
006E  2347 ; This section will set-up for a 1mb module versus a 256kb module
006E  2348 ;
006E  2349 102$:    LOADDCS         70$,,<^X1A>,1               ;load VA with last address
0075  2350          LDFIELD         22$,,B,55$,34,39            ;change RSRC fld for R[TEMP13]
0086  2351          FETCH           <^X1A>                      ;start dcs program
008B  2352          BRSTCLK         <^X1B>,INH                  ;end dcs program
008F  2353
008F  2354 ;-------------------------------------------------------------------------------
008F  2355 ;
008F  2356 ; Write memory memory with 1's
008F  2357 ;
008F  2358 ;-------------------------------------------------------------------------------
008F  2359
008F  2360 108$:    LDINDEX         I,2                         ;dummy index I gets 2
0096  2361          LOADDCS         50$,,0,<^X0B>               ;test u-code
009D  2362          FETCH           0                           ;load first u-inst.
00A2  2363          LOOP            K,1,2                       ;set index K for two passes
00A9  2364          SKIP            110$,ONEQUAL,I,K            ;skip on second pass
00B0  2365          SKIP            112$                        ;first pass, goto rdm set-up
00B7  2366
00B7  2367 ;-------------------------------------------------------------------------------
00B7  2368 ;
00B7  2369 ; Read the 1's fro the previous write, check for valid data, wirte 0's, check
00B7  2370 ; for valid data
00B7  2371 ;
00B7  2372 ;-------------------------------------------------------------------------------
00B7  2373
00B7  2374 110$:    FETCH           8                           ;reset va to last address
```

```
00BC  2375          BRSTCLK         <^X0A>,INH              ;...
00C0  2376          LOADDCS         60$,,0,<^X2F>           ;test u-code
00C7  2377          FETCH           0                       ;load first u-inst.
00CC  2378 ;
00CC  2379 ; This section is used for both the initial writing of 1's (first pass) then
00CC  2380 ; for the read of 1's-write of 0's-read of 0's (second pass)
00CC  2381 ;
00CC  2382 112$:    LOADREG         DCSCR,9$,,B             ;insure clock is off
00D4  2383          LOADREG         CSAMR,2$,,W             ;load match address
00DC  2384          LOADREG         RDCTRL,7$,,B            ;clear master halt enable
00E4  2385          LOADREG         RDCTRL,1$,,B            ;watch for u-traps
00EC  2386          LOADREG         DCSCR,3$,,B             ;start the clock
00F4  2387          LOOP            J,1,8192                ;watchdog timing loop
00FB  2388          CMPREGMSK       CLKDCS,7$,,8$,,B,       ;test for clock stopped
0107  2389          SKIP            114$,ONERROR            ;skip when clock stops
010E  2390          ENDLOOP         J                       ;end timing loop
0111  2391          LOADREG         DCSCR,9$,,B             ;stop the clock
0119  2392          CMPREG          WHREG,4$,,L,            ;force an error
0122  2393          IFERROR         ,,<<MC0><MC1><MC2>>     ;program lost
012A  2394          SKIP
0131  2395
0131  2396 ;-------------------------------------------------------------------------------
0131  2397 ;
0131  2398 ; Error evaluation code
0131  2399 ;
0131  2400 ;-------------------------------------------------------------------------------
0131  2401
0131  2402 114$:    CMPREGMSK       CSTRP,10$,,6$,,W,NE     ;test for machine check
013D  2403          SKIP            140$,ONERROR            ;handle machine check
0144  2404          CMPREGMSK       CLKDCS,11$,,12$,,B,NE   ;test for data errors
0150  2405          SKIP            130$,ONERROR            ;incorrect data
0157  2406          CMPREGMSK       CLKDCS,13$,,12$,,B,NE   ;test for data errors
0163  2407          SKIP            130$,ONERROR            ;incorrect data
016A  2408          CMPREGMSK       CLKDCS,14$,,12$,,B,NE   ;test for single bit error on 0
0176  2409          SKIP            120$,ONERROR            ;log error
017D  2410          CMPREGMSK       CLKDCS,16$,,12$,,B,NE   ;test for single bit error on 1
0189  2411          SKIP            115$,ONERROR            ;log error
0190  2412          CMPREG          WHREG,4$,,L,            ;test for last address
0199  2413          SKIP            150$,NOERROR            ;go to next test
01A0  2414          CMPREGMSK       CLKDCS,17$,,12$,,B      ;force and error
01AC  2415          IFERROR                                 ;i think we're lost
01B1  2416          SKIP                                    ;end test due to error
01B8  2417 ;
01B8  2418 ; Come here if the Second read caused corrected read data to be returned
01B8  2419 ;
01B8  2420 115$:    FETCH           <^X14>                  ;start dcs program
01BD  2421          BRSTCLK         <^X2E>                  ;end dcs program
01C1  2422          ERRLOG                                  ;log the error
01C3  2423          SKIP            150$,ONERROR            ;quit after 64 errors
01CA  2424          FETCH           <^X10>                  ;set up to continue testing
01CF  2425          SKIP            112$                    ;test more memory
01D6  2426 ;
01D6  2427 ; Come here if the First read caused corrected read data to be returned
01D6  2428 ;
01D6  2429 120$:    FETCH           <^X14>                  ;start dcs program
```

```
                01DB  2430          BRSTCLK         <^X2E>                  ;end dcs program
                01DF  2431          ERRLOG                                  ;log the error
                01E1  2432          SKIP            150$,ONERROR            ;quit after 64 errors
                01E8  2433          FETCH           <^X07>                  ;set up to continue testing
                01ED  2434          SKIP            112$                    ;test more memory
                01F4  2435 ;
                01F4  2436 ; Come here if the data read is not the data written
                01F4  2437 ;
                01F4  2438 130$:    CMPREG          WHREG,15$,,L,           ;force an error
                01FD  2439          IFERROR         1,<<MEC><MDL>>,-
                01FD  2440          <<MC0><MC1><MC2><MA0><MA1><MA2>>        ;print va and data
                020A  2441          SKIP                                    ;end test because of error
                0211  2442 ;
                0211  2443 ; Come here if there was a Machine check (Read Data Substitute)
                0211  2444 ;
                0211  2445 140$:    CMPREG          WHREG,15$,,L,           ;force an error
                021A  2446          IFERROR         ,<<MEC><MDL>>,-
                021A  2447          <<MC0><MC1><MC2><MA0><MA1><MA2>>        ;print va of machine check
                0227  2448 ;-------------------------------------------------------------------------
                0227  2449 ;
                0227  2450 ; This is where we exit the test if index K is equal to 2.  Also we want to
                0227  2451 ; dump the error log and print message 'FINISHED ARRAY 0'.
                0227  2452 ;
                0227  2453 ;-------------------------------------------------------------------------
                0227  2454 150$:    ENDLOOP         K
                022A  2455          DUMPLOG                                 ;dump the crd's for array 0
                022C  2456          PRINT_S         30$                     ;print finished message
                0230  2457          SKIP                                    ;go to next test
                0237  2458
                0237  2459
                0237  2460 BEGIN_TEST_DATA
                0237
                0237        ;-------------------------------------------------------------------
                0237  2461
                  28  0237  2462 1$:      .BYTE   ^X28                    ;set to watch for u-traps
                1801  0238  2463 2$:      .WORD   ^X1801                  ;match address
                  11  023A  2464 3$:      .BYTE   ^X11                    ;run clock
            00000000  023B  2465 4$:      .LONG   ^X00000000              ;last address of first array
                3FFF  023F  2466 6$:      .WORD   ^X3FFF                  ;mask for cmpregmsk pseudo op
                  00  0241  2467 7$:      .BYTE   ^X00                    ;watch for clock stop
                  C0  0242  2468 8$:      .BYTE   ^XC0                    ;mask for cmpregmsk pseudo op
                  60  0243  2469 9$:      .BYTE   ^X60                    ;stop clock manually
                0028  0244  2470 10$:     .WORD   ^X0028                  ;machine check u-trap
                  09  0246  2471 11$:     .BYTE   ^X09                    ;dcs halt on data error
                  3F  0247  2472 12$:     .BYTE   ^X3F                    ;mask for cmpregmsk pseudo op
                  12  0248  2473 13$:     .BYTE   ^X12                    ;dcs halt on data error
                  08  0249  2474 14$:     .BYTE   ^X08                    ;halt single bit error on 0's
            0F0F0F0F  024A  2475 15$:     .LONG   ^X0F0F0F0F              ;compare data to force errors
                  11  024E  2476 16$:     .BYTE   ^X11                    ;halt single bit error on 1's
                  FF  024F  2477 17$:     .BYTE   ^XFF                    ;used to force errors
                  03  0250  2478 18$:     .BYTE   ^X03                    ;array 0 mask
                  02  0251  2479 21$:     .BYTE   ^X02                    ;array 0 1mb (CSR2 bits 0:1)
                  0D  0252  2480 22$:     .BYTE   ^X0D                    ;RSRC = TEMP13
                  01  0253  2481 23$:     .BYTE   ^X01                    ;array 0 4mb
20 44 45 48 53 49 4E 49 46 0A 0D 00' 0254  2482 30$:     .ASCIC  <13><10>/FINISHED ARRAY 0/<13><10>
```

```
         0A 0D 30 20 59 41 52 52 41  0260
                                 14  0254
                                     0269  2483 40$:
U 00. 7700.C800.0364.0300.0470.1801  0269  2484 ;* 00   NOP
U 01. 7700.8800.5BE4.0304.6870.1802  0274  2488 ;* 01   MEMSCAR_R[TEMP1]                                  ;disable cmi
U 02. 7700.C800.5BE4.030C.6070.1803  027F  2492 ;* 02   MEMSCR_R[TEMP3]                                   ;...
U 03. 7700.8800.5BE4.0308.6870.1804  028A  2496 ;* 03   MEMSCAR_R[TEMP2]                                  ;turn on cache
U 04. 7700.C800.5BE4.03D8.6070.1805  0295  2500 ;* 04   MEMSCR_R[ZERO]                                    ;...
U 05. 7700.4800.5BE4.03D8.4A70.1806  02A0  2504 ;* 05   VA_R[ZERO]                                       ;zero the va
U 06. 7700.4800.5BE4.03D8.5C80.1807  02AB  2508 ;* 06   WRITE.PHY R[ZERO]                                 ;validate cache address 0
U 07. 7700.C800.5BE4.03D8.5470.1808  02B6  2512 ;* 07   WDR_UNROT(R[ZERO])                                ;...
U 08. 7700.C800.0364.0300.4470.1809  02C1  2516 ;* 08   VA_VA+4                                          ;increment va by 4
U 09. 7700.C800.5BE4.03D8.5C80.180A  02CC  2520 ;* 09   WRITE.PHY R[ZERO]                                 ;validate cache address 4
U 0A. 7700.C800.5BE4.03D8.5470.180B  02D7  2524 ;* 0A   WDR_UNROT(R[ZERO])
U 0B. 7700.C800.5BE4.03D8.4870.180C  02E2  2528 ;* 0B   PC_R[ZERO]                                      ;cause prefetch
U 0C. 7700.0800.5BE4.032C.2470.180D  02ED  2532 ;* 0C   WCTRL/LOADCRAR,WB_R[TEMP11]                      ;disable ^p interrupts
U 0D. 7700.C800.5BE4.0330.2C70.180E  02F8  2536 ;* 0D   WCTRL/CONWRITE,WB_R[TEMP12]                     ;...
U 0E. 7700.0800.5BE4.0304.6870.180F  0303  2540 ;* 0E   MEMSCAR_R[TEMP1]                                 ;turn on cmi
U 0F. 7700.4800.5BE4.03D8.6070.1810  030E  2544 ;* 0F   MEMSCR_R[ZERO]                                   ;...
U 10. 7700.0800.5BE4.0308.6870.1811  0319  2548 ;* 10   MEMSCAR_R[TEMP2]                                 ;disable cache
U 11. 7700.C800.5BE4.030C.6070.1812  0324  2552 ;* 11   MEMSCR_R[TEMP3]                                  ;...
U 12. 7700.8800.5BE4.0320.4A70.1813  032F  2556 ;* 12   VA_R[TEMP8]                                     ;CSR0 address to va
U 13. 7700.C800.5BE4.0324.5C80.1814  033A  2560 ;* 13   WRITE.PHY R[TEMP9]                               ;clear CSR0
U 14. 7700.4800.0364.0300.4470.1815  0345  2564 ;* 14   VA_VA+4                                          ;CSR1 address to va
U 15. 7700.4800.5BE4.0328.5C80.1816  0350  2568 ;* 15   WRITE.PHY R[TEMP10]                              ;allow single bit errors
U 16. 7700.C800.0364.0300.4470.1817  035B  2572 ;* 16   VA_VA+4                                          ;CSR2 address to va
U 17. 7700.C800.0364.0300.0400.1818  0366  2576 ;* 17   READ.PHY                                        ;read CSR2
U 18. 6700.8812.5924.0300.0470.1819  0371  2580 ;* 18   RDM_WB,WB_M[MDR]                                 ;WHREG gets contents of CSR2
U 19. 7700.C800.0364.1700.0470.181A  037C  2584 ;* 19   IRDITEST                                        ;clear int pending line
U 1A. 7700.4800.5BE4.0318.4A70.181B  0387  2588 ;* 1A   VA_R[TEMP6]                                     ;set VA to last addr of array 0
U 1B. 7300.C800.0364.0300.0470.181C  0392  2592 ;* 1B   NOP,STOP.CLOCK                                  ;stop the cpu clock
                                     039D  2596 45$:
U 00. 7700.B800.7FE0.0001.8470.1801  039D  2597 ;* 00   LONLIT_[003FFFFC]                                ;last addr for 4mb
U 01. 7700.8868.5BE4.03D4.0470.1802  03A8  2601 ;* 01   M[TEMP8]_R[LONLIT]                               ;
U 02. 7700.4848.5924.0334.0470.1803  03B3  2605 ;* 02   R[TEMP13]_M[TEMP8]                              ;RTEMP13 = last addr 3FFFFC
U 03. 7300.C800.0364.0300.0470.1804  03BE  2609 ;* 03   NOP,STOP.CLOCK                                  ;stop cpu clock
                                     03C9  2613 50$:
U 00. 7700.C800.0364.0300.0470.1802  03C9  2614 ;* 00   NOP,NEXT/1802
U 01. 6700.881B.5924.0300.0470.1802  03D4  2618 ;* 01   RDM_WB,WB_M[VA]                                  ;save va in rdm
U 02. 7700.C800.5BE4.0314.5C80.1803  03DF  2622 ;* 02   WRITE.PHY R[TEMP5]                               ;write ones to memory
U 03. 7700.C800.0364.0300.0470.1804  03EA  2626 ;* 03   NOP                                            ;allow time for write
U 04. 7700.4800.0364.0300.0470.1805  03F5  2630 ;* 04   NOP                                            ;...
U 05. 7700.081B.0034.A310.0470.1800  0400  2634 ;* 05   WB_M[VA].XOR.R[TEMP4],BUT/WX.EQ.0,NEXT/1800
                                     040B  2638                                                       ;test for end of first array
U 06. 7700.D81B.C100.0302.4A70.1807  040B  2639 ;* 06   VA_VA-ZLIT0[4]                                 ;decrement va register
U 07. 7500.C800.0364.0300.0470.1808  0416  2643 ;* 07   NOP,DCS.ADDR_0                                 ;return to address 0
U 08. 7700.4800.0364.0300.0470.1809  0421  2647 ;* 08   NOP
                                     042C  2651 55$:
U 09. 7700.4800.5BE4.0318.4A70.180A  042C  2652 ;* 09   VA_R[TEMP6]                                     ;set va to array last address
U 0A. 7300.C800.0364.0300.0470.180B  0437  2656 ;* 0A   NOP,STOP.CLOCK                                  ;stop the cpu clock
                                     0442  2660 60$:
U 00. 7700.C800.0364.0300.0470.1802  0442  2661 ;* 00   NOP,NEXT/1802                                   ;allow time for memory
U 01. 7700.C800.0364.0300.0470.1802  044D  2665 ;* 01   NOP                                            ;clock to start up.
U 02. 7700.4800.0364.0300.0470.1803  0458  2669 ;* 02   NOP                                            ;...
U 03. 6700.881B.5924.0300.0470.1804  0463  2673 ;* 03   RDM_WB,WB_M[VA]                                  ;save va in rdm
```

```
U 04. 7700.C800.0364.0300.0400.1805  046E  2677 ;% 04   READ.PHY                                          ;read pattern back
U 05. 7700.4852.5924.0300.0470.1806  0479  2681 ;% 05   R[TEMP0]_M[MDR]                                   ;save data in RTEMP0
U 06. 7700.C800.0364.AF00.0470.1801  0484  2685 ;% 06   BUT/INT-TIMSERV,NEXT/1801                        ;test for corrected data
U 07. 7700.0800.0034.A714.0470.1800  048F  2689 ;% 07   WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                     049A  2693                                                         ;test for good data
U 08. 7700.4800.0364.0300.0470.1809  049A  2694 ;% 08   NOP                                             ;allow time for memory clock
U 09. 7700.4800.0364.0300.0470.180A  04A5  2698 ;% 09   NOP                                             ;to start up after single bit
U 0A. 7700.C800.0364.0300.0470.180B  04B0  2702 ;% 0A   NOP                                             ;errors.
U 0B. 7700.4800.0364.0300.0470.180C  04BB  2706 ;% 0B   NOP                                             ;...
U 0C. 7700.0800.5BE4.0310.5C80.180D  04C6  2710 ;% 0C   WRITE.PHY R[TEMP4]                               ;write zero's to memory
U 0D. 7700.4800.0364.0300.0400.180E  04D1  2714 ;% 0D   READ.PHY                                          ;read pattern back
U 0E. 7700.4852.5924.0300.0470.180F  04DC  2718 ;% 0E   R[TEMP0]_M[MDR]                                   ;save data in RTEMP0
U 0F. 7700.C800.0364.AF00.0470.1801  04E7  2722 ;% 0F   BUT/INT-TIMSERV,NEXT/1801                        ;test for corrected data
U 10. 7700.4800.0034.A710.0470.1800  04F2  2726 ;% 10   WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                     04FD  2730                                                         ;test for good data
U 11. 7700.081B.0034.A310.0470.1800  04FD  2731 ;% 11   WB_M[VA].XOR.R[TEMP4],BUT/WX.EQ.0,NEXT/1800
                                     0508  2735                                                         ;test for end of first array
U 12. 7700.D81B.C100.0302.4A70.1813  0508  2736 ;% 12   VA_VA-ZLIT0[4]                                   ;decrement va
U 13. 7500.4800.0364.0300.0470.1814  0513  2740 ;% 13   NOP,DCS.ADDR_0                                   ;return to dcs address 0
U 14. 7700.C800.0364.0300.0470.1815  051E  2744 ;% 14   NOP
U 15. 7700.885B.5924.031C.0470.1816  0529  2748 ;% 15   R[TEMP7]_M[VA]                                   ;save va for return
U 16. 7700.0800.5BE4.0304.6870.1817  0534  2752 ;% 16   MEMSCAR_R[TEMP1]                                ;disable cmi
U 17. 7700.C800.5BE4.030C.6070.1818  053F  2756 ;% 17   MEMSCR_R[TEMP3]                                 ;...
U 18. 7700.8800.5BE4.0308.6870.1819  054A  2760 ;% 18   MEMSCAR_R[TEMP2]                                ;turn on cache
U 19. 7700.4800.5BE4.03D8.6070.181A  0555  2764 ;% 19   MEMSCR_R[ZERO]                                  ;...
U 1A. 7700.4800.5BE4.03D8.4A70.181B  0560  2768 ;% 1A   VA_R[ZERO]                                      ;zero the va
U 1B. 7700.4800.5BE4.03D8.5C80.181C  056B  2772 ;% 1B   WRITE.PHY R[ZERO]                                ;validate cache address 0
U 1C. 7700.4800.5BE4.03D8.5470.181D  0576  2776 ;% 1C   WDR_UNROT(R[ZERO])                               ;...
U 1D. 7700.C800.0364.0300.4470.181E  0581  2780 ;% 1D   VA_VA+4                                         ;increment va by 4
U 1E. 7700.4800.5BE4.03D8.5C80.181F  058C  2784 ;% 1E   WRITE.PHY R[ZERO]                                ;validate cache address 4
U 1F. 7700.C800.5BE4.03D8.5470.1820  0597  2788 ;% 1F   WDR_UNROT(R[ZERO])
U 20. 7700.C800.5BE4.03D8.4870.1821  05A2  2792 ;% 20   PC_R[ZERO]                                      ;cause prefetch
U 21. 7700.8800.5BE4.032C.2470.1822  05AD  2796 ;% 21   WCTRL/LOADCRAR,WB_R[TEMP11]                     ;disable ^P interrupts
U 22. 7700.C800.5BE4.0330.2C70.1823  05B8  2800 ;% 22   WCTRL/CONWRITE,WB_R[TEMP12]                     ;...
U 23. 7700.0800.5BE4.0304.6870.1824  05C3  2804 ;% 23   MEMSCAR_R[TEMP1]                                ;turn on cmi
U 24. 7700.4800.5BE4.03D8.6070.1825  05CE  2808 ;% 24   MEMSCR_R[ZERO]                                  ;...
U 25. 7700.8800.5BE4.0308.6870.1826  05D9  2812 ;% 25   MEMSCAR_R[TEMP2]                                ;disable cache
U 26. 7700.C800.5BE4.030C.6070.1827  05E4  2816 ;% 26   MEMSCR_R[TEMP3]                                 ;...
U 27. 7700.0800.5BE4.0320.4A70.1828  05EF  2820 ;% 27   VA_R[TEMP8]                                     ;address of CSR0
U 28. 7700.4800.0364.0300.0400.1829  05FA  2824 ;% 28   READ.PHY                                        ;read CSR0
U 29. 3700.8812.5924.0300.0470.182A  0605  2828 ;% 29   WB_M[MDR],RDM_CMI                               ;MHREG gets CSR0
U 2A. 7700.C800.5BE4.0324.5C80.182B  0610  2832 ;% 2A   WRITE.PHY R[TEMP9]                              ;clear CSR0
U 2B. 7700.C800.0364.7B00.0470.182C  061B  2836 ;% 2B   BUT/UVCTR                                      ;clear any interrupts pending
U 2C. 7700.4800.0364.1700.0470.182D  0626  2840 ;% 2C   IRD1TEST                                       ;free interrupt pending logic
U 2D. 7700.0800.5BE4.031C.4A70.182E  0631  2844 ;% 2D   VA_R[TEMP7]                                     ;reset the va
U 2E. 7300.C800.0364.0300.0470.182F  063C  2848 ;% 2E   NOP,STOP.CLOCK                                  ;stop the cpu clock
                                     0647  2852 70$:
U 1A. 7700.0800.5BE4.0334.4A70.181B  0647  2853 ;% 1A   VA_R[TEMP13]                                    ;va gets array last addr FFFFC
                                     0652  2857
                                     0652  2858 BEGIN_CPU_DATA
                                     0002  2859
                                     0002  2860 RTEMP_DATA
                                     0001  2861
                         00000000    0001  2862         .LONG   ^X00000000                            ;data pattern storage
                         0E000000    0005  2863         .LONG   ^X0E000000                            ;reference cache only address
```

```
                    06000000  0009  2864          .LONG   ^X06000000                      ;cache control register address
                    01000000  000D  2865          .LONG   ^X01000000                      ;disable cache
                    00000000  0011  2866          .LONG   ^X00000000                      ;test pattern 1
                    FFFFFFFF  0015  2867          .LONG   ^XFFFFFFFF                      ;test pattern 2
                    0003FFFC  0019  2868          .LONG   ^X0003FFFC                      ;last address for 256kb array 0
                    00000000  001D  2869          .LONG   ^X00000000                      ;storage for va register
                    00F20000  0021  2870          .LONG   ^X00F20000                      ;address of CSR0
                    E0000000  0025  2871          .LONG   ^XE0000000                      ;clears CSR0
                    10000000  0029  2872          .LONG   ^X10000000                      ;allow single bit error report
                    00C00000  002D  2873          .LONG   ^X00C00000                      ;disable ^P interrupts
                    00400000  0031  2874          .LONG   ^X00400000                      ;...
                    000FFFFC  0035  2875          .LONG   ^X000FFFFC                      ;last address for 1mb array 0
                              0039  2876
                              0039  2877 END_CPU_DATA
                              0652  2878
                              0652  2879
                              0652  2880 END_TEST_DATA
                              0652
                              0652      ;-------------------------------------------------------------------------
                              0652  2881 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        .SBTTL  "TEST   086     Array 1 Memory Test Part 1
001F   2883 ;*******************************************************************************
001F   2884 ;++
001F   2885 ;
001F   2886 ; FUNCTIONAL DESCRIPTION:
001F   2887 ;
001F   2888 ;       This test will check the second array card writing first 0's then
001F   2889 ;       1's in ascending order.
001F   2890 ;
001F   2891 ;
001F   2892 ; TEST ALGORITHM:
001F   2893 ;
001F   2894 ;       1.  Load u-code to initialize test conditions
001F   2895 ;       2.  Load shared u-code that will validate cache address 0 and 4,
001F   2896 ;           disable Control_P interrupts, and clear CSR0.
001F   2897 ;       3.  Execute shared dcs program
001F   2898 ;               a.  Disable the cmi
001F   2899 ;               b.  Enable the cache
001F   2900 ;               c.  Set va to zero
001F   2901 ;               d.  Validate cache addresses 0 and 4
001F   2902 ;               e.  Load the pc to cause prefetch of addresses 0 and 4
001F   2903 ;               f.  Disable control P interrupts
001F   2904 ;               g.  Enable cmi and disable cache
001F   2905 ;               h.  Clear memory CSR0
001F   2906 ;               i.  Perform IRD1TEST to free interrupt pending logic
001F   2907 ;       4.  Execute init dcs program
001F   2908 ;               a.  Set memory csr1 to a!low single bit errors
001F   2909 ;               b.  Read CSR2
001F   2910 ;               c.  Perform IRD1TEST to free interrupt pending logic
001F   2911 ;       5.  Check bits 3:2 of CSR2 to see if there is a second array present in
001F   2912 ;           the system.  If not present then go to next test.
001F   2913 ;       6.  Announce that we are testing array 1 (second array)
001F   2914 ;       7.  Check bits 1:0 of CSR2 to determine the array 0 size
001F   2915 ;       8.  If array 0 is 256kb then mask address 7FFFC and go test (test data
001F   2916 ;           defaults to test a 256kb module for array 1 assuming array 0 is a
001F   2917 ;           256kb as would have to be to follow configuration specs).
001F   2918 ;       9.  If array 0 is a 1mb module then:
001F   2919 ;               a.  Set array one starting address to 100000
001F   2920 ;               b.  Check for array 1 size
001F   2921 ;               c.  If array 1 is a 1mb module then:
001F   2922 ;                       i.  Set array 1 last address to 1FFFFC
001F   2923 ;                      ii.  Execute DCS code to set R[TEMP6] to 1FFFFC
001F   2924 ;                     iii.  Go test
001F   2925 ;               d.  If array 1 is a 256kb module then:
001F   2926 ;                       i.  Set array 1 last address to 13FFFC
001F   2927 ;                      ii.  Execute DCS code to set R[TEMP6] to 13FFFC
001F   2928 ;                     iii.  Go test
001F   2929 ;      10.  If array 0 is 4mb, then:
001F   2930 ;               a. Set array 1 starting address to 400000
001F   2931 ;               b. Check for array 1 size
001F   2932 ;               c. If array 1 is 4mb then:
001F   2933 ;                       i. Set array 1 last address to 7FFFFC
001F   2934 ;                      ii. Execute dcs code to set R[TEMP6] to 7FFFFC
001F   2935 ;                     iii. Go test
001F   2936 ;               d. If array 1 is 1mb, then:
```

```
001F  2937 ;                          i. Set array 1 last address to 4FFFFC
001F  2938 ;                         ii. Execute dcs code to set R[TEMP6] to 4FFFFC
001F  2939 ;                        iii. Go test
001F  2940 ;      11.  Load index I with 2 which will be used for comparison of index K
001F  2941 ;      12.  Load u-code to write zeros to main memory
001F  2942 ;      13.  Load the first u-inst.
001F  2943 ;      14.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
001F  2944 ;           error section as follows:
001F  2945 ;               a.  First pass is from the writing of zero's throughout memory
001F  2946 ;               b.  Second pass is from the read-write-read throughout memory
001F  2947 ;      15.  Check to see if this is first pass or second pass, if first pass
001F  2948 ;           then goto 112$, if second pass then goto 110$
001F  2949 ;      16.  Set-up the rdm to watch for u-traps
001F  2950 ;      17.  Set the u-match halt for 1801
001F  2951 ;      18.  Start the cpu clock
001F  2952 ;               a.  Save the va in the rdm WH register
001F  2953 ;               b.  Write zero to the address in the va
001F  2954 ;               c.  Test va to see if at end of second array card
001F  2955 ;               d.  If not increment va and return to dcs address 0
001F  2956 ;      19.  Set-up watchdog timer with loop J to monitor above u-code
001F  2957 ;      20.  If loop J expires, stop the clock and force an error printout
001F  2958 ;      21.  If clock stops normally (u-match at 1801); test WH register for
001F  2959 ;           end of second array
001F  2960 ;      22.  If no error test cs address for not a u-trap
001F  2961 ;      23.  Increment loop K for second pass
001F  2962 ;      24.  Execute dcs code
001F  2963 ;               a.  Set va to the starting address for array 1
001F  2964 ;      25.  Load the test u-code
001F  2965 ;      26.  Load the first u-inst.
001F  2966 ;      27.  Set-up the rdm to watch for u-traps
001F  2967 ;      28.  Set the u-match to 1801
001F  2968 ;      29.  Start the cpu clock
001F  2969 ;               a.  Save va in rdm WH register
001F  2970 ;               b.  Read address specified by va
001F  2971 ;               c.  Move data to RTEMP0
001F  2972 ;               d.  Test for a corrected data interrupt
001F  2973 ;               e.  Test that data is correct
001F  2974 ;               f.  Write one to the same location
001F  2975 ;               g.  Read location back
001F  2976 ;               h.  Save data in RTEMP0
001F  2977 ;               i.  Test for a corrected data interrupt
001F  2978 ;               j.  Test that data is correct
001F  2979 ;               k.  Test to see if at end of second array
001F  2980 ;               l.  Of not end of second array, increment the va by 4
001F  2981 ;               m.  Return to dcs address 0
001F  2982 ;      30.  Set-up watch-dog timer with loop J to monitor above u-code
001F  2983 ;      31.  If loop J expires, stop the clock and force an error printout
001F  2984 ;      32.  If clock stops normally (u-match at 1801) test WH register
001F  2985 ;           for end of second array.
001F  2986 ;      33.  Test for machine checks (double bit error in memory)
001F  2987 ;      34.  If not machine check test for incorrect data on first read
001F  2988 ;      35.  If not bad data test for incorrect data on second read
001F  2989 ;      36.  Test for corrected data interrupts
001F  2990 ;      37.  Service the corrected data interrupt with program at dcs address 14
001F  2991 ;               a.  Save va in RTEMP7
```

```
001F  2992 ;                b.  Disable cmi, enable cache
001F  2993 ;                c.  Set va to zero
001F  2994 ;                d.  Validate cache address 0 and 4
001F  2995 ;                e.  Load pc to cause prefetch of addresses 0 and 4
001F  2996 ;                f.  Disable ^p interrupts
001F  2997 ;                g.  Read memory CSR0 to the rdm MHREG
001F  2998 ;                h.  Clear memory CSR0
001F  2999 ;                i.  Clear any interrupts pending
001F  3000 ;                j.  Perform an IRD1TEST to free the interrupt pending logic
001F  3001 ;                k.  Reload va with RTEMP7
001F  3002 ;          38.  Continue running test micro code from step 17.
001F  3003 ;          39.  Test for the end of the second array.
001F  3004 ;          40.  When last address of array is found the test will end on endloop K
001F  3005 ;               being expired
001F  3006 ;          41.  The balance of the pseudo ops handle the various error conditions
001F  3007 ;               (see error description)
001F  3008 ;
001F  3009 ;
001F  3010 ; TEST PATTERNS:
001F  3011 ;
001F  3012 ;           First pass:  init memory to 0's
001F  3013 ;          Second pass:  read 0's; write 1's; read 1's
001F  3014 ;
001F  3015 ;
001F  3016 ; LOGIC DESCRIPTION:
001F  3017 ;
001F  3018 ;           My advice on this test is forget trying to troubleshoot with it.  It
001F  3019 ;           is far to complex to be a usefull troubleshooting tool.  Rather, just
001F  3020 ;           use the error printout to array and/or module swap till you find the
001F  3021 ;           problem.  The second array is throughly tested only after running
001F  3022 ;           all four tests in this series; together and in order.
001F  3023 ;
001F  3024 ;
001F  3025 ; ASSUMPTIONS:
001F  3026 ;
001F  3027 ;           This test assumes the cmi is operational.
001F  3028 ;
001F  3029 ;
001F  3030 ; ERROR DESCRIPTION:
001F  3031 ;
001F  3032 ;           First IFERROR:
001F  3033 ;
001F  3034 ;           EXPECTED LAST ADDRESS OF FIRST ARRAY
001F  3035 ;           RECEIVED LAST ADDRESS
001F  3036 ;           LOOP COUNT I
001F  3037 ;           LOOP COUNT J
001F  3038 ;           LOOP COUNT K
001F  3039 ;           (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
001F  3040 ;
001F  3041 ;           Second IFERROR:
001F  3042 ;
001F  3043 ;           NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  3044 ;           MAIN MEMORY ADDRESS OF DATA ERROR
001F  3045 ;           LOOP COUNT I
001F  3046 ;           LOOP COUNT J
```

```
                     ;       LOOP COUNT K
                     ;       RECEIVED DATA (S/B 0'S OR 1'S)
                     ;       (INDICATES ECC IS PROBABLE NOT WORKING)
                     ;
                     ;       Third IFERROR:
                     ;
                     ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
                     ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
                     ;       LOOP COUNT I
                     ;       LOOP COUNT J
                     ;       LOOP COUNT K
                     ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
                     ;
                     ;--
                     ;*******************************************************************************

                     ;-------------------------------------------------------------------------------
                     ;
                     ; Initialize code
                     ;
                     ;-------------------------------------------------------------------------------

001F 3069            INITIALIZE                                      ;init the cpu
0021 3070            LOADDCS         40$,,0,<^X10>                   ;set-up init micro code
0028 3071            LOADDCS         65$,,<^X11>,<^X1B>              ;set-up shared micro code
002F 3072            FETCH           <^X11>                          ;execute shared micro code
0034 3073            BRSTCLK         <^X2B>                          ;end shard micro code
0038 3074            FETCH           0                               ;execute init micro code
003D 3075            BRSTCLK         <^X0A>,INH                      ;end init micro code
0041 3076            CMPREGMSK       WHREG,23$,,25$,,B               ;check for array 1 present
004D 3077            SKIP            155$,NOERROR                    ;skip if nothing there (exit)
0054 3078            PRINT_S         30$                             ;announce testing array 1
0058 3079            CMPREGMSK       WHREG,31$,,18$,,B               ;check array 0
0064 3080            SKIP            90$,NOERROR                     ;skip if 4mb
006B 3081            CMPREGMSK       WHREG,21$,,18$,,B,NE            ;check array 0 type
0077 3082            SKIP            100$,ONERROR                    ;skip if 1mb module
007E 3083            MASK            4$,,19$,,L                      ;array 1 last address = 7FFFC
0087 3084            SKIP            108$                            ;go test
008E 3085 ;
008E 3086 ; This section of code handles the case where array 0 is 4mb. It will also
008E 3087 ; check the size of array 1.
008E 3088 ;
008E 3089 90$:       LDFIELD         32$,,L,57$,31,62,LON            ;array 1 start address = 400000
009F 3090            LOADDCS         79$,,<^X08>,1                   ;array 1 start address = 400000
00A6 3091            FETCH           <^X08>                          ;start dcs
00AB 3092            BRSTCLK         <^X0A>,INH                      ;end dcs
00AF 3093            CMPREGMSK       WHREG,33$,,25$,,B               ;check array 1
00BB 3094            SKIP            95$,ONERROR                     ;skip if 1mb
00C2 3095            LOADDCS         82$,,<^X0C>,1                   ;array 1 last address = 7FFFFC
00C9 3096            MASK            4$,,35$,,L                      ;array 1 last address = 7FFFFC
00D2 3097            SKIP            102$                            ;execute changed dcs
00D9 3098 95$:       LOADDCS         84$,,<^X0C>,1                   ;array 1 last address = 4FFFFC
00E0 3099            MASK            4$,,37$,,L                      ;array 1 last address = 4FFFFC
00E9 3100            SKIP            102$                            ;execute changed dcs
00F0 3101 ;
```

```
00F0  3102 ; This section of code will handle the case where array 0 (first array slot) is
00F0  3103 ; a 1mb array module.  It will also check the size of array 1
00F0  3104 ;
00F0  3105 100$:   LDFIELD          24$,,L,57$,31,62,LON    ;array 1 start address = 100000
0101  3106         LOADDCS          75$,,<^X08>,1           ;array 1 srart address = 100000
0108  3107         FETCH            <^X08>                  ;start dcs program
010D  3108         BRSTCLK          <^X0A>,INH              ;end dcs program
0111  3109         CMPREGMSK        WHREG,26$,,25$,,B       ;check array 1 size
011D  3110         SKIP             104$,ONERROR            ;skip if 256kb module
0124  3111         MASK             4$,,20$,,L              ;array 1 last address = 1FFFFC
012D  3112 ;
012D  3113 ; Come here to execute the altered dcs and set up last address.
012D  3114 ;
012D  3115 102$:   FETCH            <^X0B>                  ;start dcs program
0132  3116         BRSTCLK          <^X0F>,INH              ;end dcs program
0136  3117         SKIP             108$                    ;go test
013D  3118 ;
013D  3119 ; Come here if array 1 is a 256kb module.  This section will skip back to 102$
013D  3120 ; to execute altered dcs to establish the correct last address.
013D  3121 ;
013D  3122 104$:   LOADDCS          77$,,<^X0C>,1           ;array 1 last address = 13FFFC
0144  3123         MASK             4$,,27$,,L              ;array 1 last address = 13FFFC
014D  3124         SKIP             102$                    ;go execute changed dcs
0154  3125
0154  3126 ;-------------------------------------------------------------------------------
0154  3127 ;
0154  3128 ; Write memory with 0's
0154  3129 ;
0154  3130 ;-------------------------------------------------------------------------------
0154  3131
0154  3132 108$:   LDINDEX          I,2                     ;dummy index I gets 2
015B  3133         LOADDCS          50$,,0,<^X0B>           ;test u-code
0162  3134         FETCH            0                       ;load first u-inst.
0167  3135         LOOP             K,1,2                   ;set index K for two passes
016E  3136         SKIP             110$,ONEQUAL,I,K        ;skip on second pass
0175  3137         SKIP             112$                    ;first pass, goto rdm set-up
017C  3138
017C  3139 ;-------------------------------------------------------------------------------
017C  3140 ;
017C  3141 ; Read the 0's from the previous write, check for valid data, write 1's, check
017C  3142 ; for valid data
017C  3143 ;
017C  3144 ;-------------------------------------------------------------------------------
017C  3145
017C  3146 110$:   FETCH            7                       ;reset va to starting address
0181  3147         BRSTCLK          <^X0A>,INH              ;...
0185  3148         LOADDCS          60$,,0,<^X11>           ;test u-code
018C  3149         FETCH            0                       ;load first u-inst.
0191  3150 ;
0191  3151 ; This section is used for both the initial writing of 0's (first pass) then
0191  3152 ; for the read 0's-write 1's-read 1's (second pass)
0191  3153 ;
0191  3154 112$:   LOADREG          DCSCR,9$,,B             ;insure clock is off
0199  3155         LOADREG          CSAMR,2$,,W             ;load match address
01A1  3156         LOADREG          RDCTRL,7$,,B            ;clear master halt enable
```

```
01A9  3157          LOADREG         RDCTRL,1$,,B            ;watch for u-traps
01B1  3158          LOADREG         DCSCR,3$,,B             ;start the clock
01B9  3159          LOOP            J,1,8192                ;watchdog timing loop
01C0  3160          CMPREGMSK       CLKDCS,7$,,8$,,B,       ;test for clock stopped
01CC  3161          SKIP            114$,ONERROR            ;skip when clock stops
01D3  3162          ENDLOOP         J                       ;end timing loop
01D6  3163          LOADREG         DCSCR,9$,,B             ;stop the clock
01DE  3164          CMPREG          WHREG,4$,,L,            ;force an error
01E7  3165          IFERROR         ,,<<MC0><MC1><MC2>>     ;program lost
01EF  3166          SKIP
01F6  3167
01F6  3168 ;-------------------------------------------------------------------------------
01F6  3169 ;
01F6  3170 ; Error evaluation code
01F6  3171 ;
01F6  3172 ;-------------------------------------------------------------------------------
01F6  3173
01F6  3174 114$:    CMPREGMSK       CSTRP,10$,,6$,,W,NE     ;test for machine check
0202  3175          SKIP            140$,ONERROR            ;handle machine check
0209  3176          CMPREGMSK       CLKDCS,11$,,12$,,B,NE   ;test for data errors
0215  3177          SKIP            130$,ONERROR            ;incorrect data
021C  3178          CMPREGMSK       CLKDCS,13$,,12$,,B,NE   ;test for data errors
0228  3179          SKIP            130$,ONERROR            ;incorrect data
022F  3180          CMPREGMSK       CLKDCS,14$,,12$,,B,NE   ;test for single bit error on 0
023B  3181          SKIP            120$,ONERROR            ;log error
0242  3182          CMPREGMSK       CLKDCS,16$,,12$,,B,NE   ;test for single bit error on 1
024E  3183          SKIP            115$,ONERROR            ;log error
0255  3184          CMPREG          WHREG,4$,,L,            ;test for last address
025E  3185          SKIP            150$,NOERROR            ;go to next test
0265  3186          CMPREGMSK       CLKDCS,17$,,12$,,B      ;force and error
0271  3187          IFERROR                                 ;i think we're lost
0276  3188          SKIP                                    ;end test due to error
027D  3189 ;
027D  3190 ; Come here if the Second read caused corrected read data to be returned
027D  3191 ;
027D  3192 115$:    FETCH           <^X11>                  ;start dcs program
0282  3193          BRSTCLK         <^X2B>                  ;end dcs program
0286  3194          ERRLOG                                  ;log the error
0288  3195          SKIP            150$,ONERROR            ;quit after 64 errors
028F  3196          FETCH           <^X0D>                  ;set up to continue testing
0294  3197          SKIP            112$                    ;test more memory
029B  3198 ;
029B  3199 ; Come here if the First read caused corrected read data to be returned
029B  3200 ;
029B  3201 120$:    FETCH           <^X11>                  ;start dcs program
02A0  3202          BRSTCLK         <^X2B>                  ;end dcs program
02A4  3203          ERRLOG                                  ;log the error
02A6  3204          SKIP            150$,ONERROR            ;quit after 64 errors
02AD  3205          FETCH           <^X06>                  ;set up to continue testing
02B2  3206          SKIP            112$                    ;test more memory
02B9  3207 ;
02B9  3208 ; Come here if the data read is not the data written
02B9  3209 ;
02B9  3210 130$:    CMPREG          WHREG,15$,,L,           ;force an error
02C2  3211          IFERROR         1,<<MEC><MDL>>,-
```

```
                    02C2  3212          <<MC0><MC1><MC2><MA0><MA1><MA2>>         ;print va and data
                    02CF  3213          SKIP                                    ;end test because of error
                    02D6  3214 ;
                    02D6  3215 ; Come here if there was a Machine check (Read Data Substitute)
                    02D6  3216 ;
                    02D6  3217 140$:    CMPREG          WHREG,15$,,L,           ;force an error
                    02DF  3218          IFERROR         ,<<MEC><MDL>>,-
                    02DF  3219          <<MC0><MC1><MC2><MA0><MA1><MA2>>         ;print va of machine check
                    02EC  3220 ;-------------------------------------------------------------------------
                    02EC  3221 ;
                    02EC  3222 ; This is where we exit the test if index K is equal to 2
                    02EC  3223 ;
                    02EC  3224 ;-------------------------------------------------------------------------
                    02EC  3225
                    02EC  3226 150$:    ENDLOOP         K                       ;end pass loop
                    02EF  3227 155$:    SKIP                                    ;go to next test
                    02F6  3228
                    02F6  3229
                    02F6  3230 BEGIN_TEST_DATA
                    02F6
                    02F6        ;-------------------------------------------------------------------
                    02F6  3231
                 28 02F6  3232 1$:      .BYTE   ^X28                            ;set to watch for u-traps
               1801 02F7  3233 2$:      .WORD   ^X1801                          ;match address
                 11 02F9  3234 3$:      .BYTE   ^X11                            ;run clock
          FFFFFFFF 02FA  3235 4$:      .LONG   ^XFFFFFFFF                      ;last address for second array
                    02FE  3236                                                  ;will be masked here
               3FFF 02FE  3237 6$:      .WORD   ^X3FFF                          ;mask for cmpregmsk pseudo op
                 00 0300  3238 7$:      .BYTE   ^X00                            ;watch for clock stop
                 C0 0301  3239 8$:      .BYTE   ^XC0                            ;mask for cmpregmsk pseudo op
                 60 0302  3240 9$:      .BYTE   ^X60                            ;stop clock manually
               0028 0303  3241 10$:     .WORD   ^X0028                          ;machine check u-trap
                 08 0305  3242 11$:     .BYTE   ^X08                            ;dcs halt on data error
                 3F 0306  3243 12$:     .BYTE   ^X3F                            ;mask for cmpregmsk pseudo op
                 0F 0307  3244 13$:     .BYTE   ^X0F                            ;dcs halt on data error
                 07 0308  3245 14$:     .BYTE   ^X07                            ;halt single bit error on 0's
          0F0F0F0F 0309  3246 15$:     .LONG   ^X0F0F0F0F                      ;compare data to force errors
                 0E 030D  3247 16$:     .BYTE   ^X0E                            ;halt single bit error on 1's
                 FF 030E  3248 17$:     .BYTE   ^XFF                            ;used to force errors
                 03 030F  3249 18$:     .BYTE   ^X03                            ;array 0 mask
          0007FFFC 0310  3250 19$:     .LONG   ^X0007FFFC                      ;256kb array 1 last address
          001FFFFC 0314  3251 20$:     .LONG   ^X001FFFFC                      ;1mb array 1 last address
                 02 0318  3252 21$:     .BYTE   ^X02                            ;array 0 1mb (csr2 bits 0:1)
                 00 0319  3253 23$:     .BYTE   ^X00                            ;slot empty compare data
          00100000 031A  3254 24$:     .LONG   ^X00100000                      ;array 1 start address
                 0C 031E  3255 25$:     .BYTE   ^X0C                            ;array 1 mask
                 08 031F  3256 26$:     .BYTE   ^X08                            ;array 1 1mb (csr2 bits 3:2)
          0013FFFC 0320  3257 27$:     .LONG   ^X0013FFFC                      ;array 1 = 256kb end address
41 20 47 4E 49 54 53 45 54 0A 0D 00' 0324  3258 30$:     .ASCIC  <13><10>/TESTING ARRAY 1/<13><10>
            0A 0D 31 20 59 41 52 52  0330
                 13  0324
                    0338  3259                                                 ;message for start of test
                 01 0338  3260 31$:     .BYTE   ^X01                            ;array 0 4mb
          00400000 0339  3261 32$:     .LONG   ^X00400000                      ;array 1 start address
                 04 033D  3262 33$:     .BYTE   ^X04                            ;array 1 4mb
```

```
                        007FFFFC  033E  3263 35$:    .LONG   ^X007FFFFC                          ;4mb array 1 last address
                        004FFFFC  0342  3264 37$:    .LONG   ^X004FFFFC                          ;1mb array 1 last address
                                  0346  3265 40$:
U 00. 7700.C800.0364.0300.0470.1801   0346  3266 ;% 00   NOP
U 01. 7700.8800.5BE4.0320.4A70.1802   0351  3270 ;% 01   VA_R[TEMP8]                             ;CSR0 address to va
U 02. 7700.C800.0364.0300.4470.1803   035C  3274 ;% 02   VA_VA+4                                 ;CSR1 address to va
U 03. 7700.4800.5BE4.0328.5C80.1804   0367  3278 ;% 03   WRITE.PHY R[TEMP10]                     ;allow single bit errors
U 04. 7700.C800.0364.0300.4470.1805   0372  3282 ;% 04   VA_VA+4                                 ;CSR2 address to va
U 05. 7700.C800.0364.0300.0400.1806   037D  3286 ;% 05   READ.PHY                                ;read CSR2
U 06. 6700.8812.5924.0300.0470.1807   0388  3290 ;% 06   RDM_WB,WB_M[MDR]                        ;WHREG gets contents of CSR2
U 07. 7700.C800.0364.1700.0470.1808   0393  3294 ;% 07   IRDITEST                                ;clear int pending line
U 08. 7700.B800.7FFD.FFFF.8470.1809   039E  3298 ;% 08   LONLIT_[00040000]                       ;array starting address
U 09. 7700.4800.5BE4.03D4.4A70.180A   03A9  3302 ;% 09   VA_R[LONLIT]                           ;VA gets starting address
U 0A. 7300.C800.0364.0300.0470.180B   03B4  3306 ;% 0A   NOP,STOP.CLOCK                         ;stop the cpu clock
U 0B. 7700.4800.0364.0300.0470.180C   03BF  3310 ;% 0B   NOP
U 0C. 7700.F800.7FF0.0001.8470.180D   03CA  3314 ;% 0C   LONLIT_[001FFFFC]                       ;array 1 ending address
U 0D. 7700.8868.5BE4.03D4.0470.180E   03D5  3318 ;% 0D   M[TEMP8]_R[LONLIT]                     ;that gets setup in R[TEMP6]
U 0E. 7700.0848.5924.0318.0470.180F   03E0  3322 ;% 0E   R[TEMP6]_M[TEMP8]                     ;...
U 0F. 7300.C800.0364.0300.0470.1810   03EB  3326 ;% 0F   NOP,STOP.CLOCK
                                  03F6  3330 50$:
U 00. 7700.C800.0364.0300.0470.1802   03F6  3331 ;% 00   NOP,NEXT/1802
U 01. 6700.881B.5924.0300.0470.1802   0401  3335 ;% 01   RDM_WB,WB_M[VA]                         ;save va in rdm
U 02. 7700.8800.5BE4.0310.5C80.1803   040C  3339 ;% 02   WRITE.PHY R[TEMP4]                     ;write zeros to memory
U 03. 7700.4800.0364.0300.0470.9804   0417  3343 ;% 03   NOP,CLOCK.EXT                         ;allow time for write
U 04. 7700.481B.0034.A318.0470.1800   0422  3347 ;% 04   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                  042D  3351                                                  ;test for end of second array
U 05. 7700.C800.0364.0300.4470.1806   042D  3352 ;% 05   VA_VA+4                                 ;increment va register
U 06. 7500.C800.0364.0300.0470.1807   0438  3356 ;% 06   NOP,DCS.ADDR_0                         ;return to address 0
U 07. 7700.C800.0364.0300.0470.1808   0443  3360 ;% 07   NOP
                                  044E  3364 57$:
U 08. 7700.B800.7FFD.FFFF.8470.1809   044E  3365 ;% 08   LONLIT_[00040000]                       ;array starting address
U 09. 7700.4800.5BE4.03D4.4A70.180A   0459  3369 ;% 09   VA_R[LONLIT]                           ;VA gets starting address
U 0A. 7300.C800.0364.0300.0470.180B   0464  3373 ;% 0A   NOP,STOP.CLOCK                         ;stop the cpu clock
                                  046F  3377 60$:
U 00. 7700.4800.0364.0300.0470.9802   046F  3378 ;% 00   NOP,CLOCK.EXT,NEXT/1802                ;allow time for memory
U 01. 7700.C800.0364.0300.0470.1802   047A  3382 ;% 01   NOP                                     ;clock to start up.
U 02. 6700.081B.5924.0300.0470.1803   0485  3386 ;% 02   RDM_WB,WB_M[VA]                         ;save va in rdm
U 03. 7700.4800.0364.0300.0400.1804   0490  3390 ;% 03   READ.PHY                                ;read pattern back
U 04. 7700.4852.5924.0300.0470.1805   049B  3394 ;% 04   R[TEMP0]_M[MDR]                        ;save data in RTEMP0
U 05. 7700.C800.0364.AF00.0470.1801   04A6  3398 ;% 05   BUT/INT-TIMSERV,NEXT/1801             ;test for corrected data
U 06. 7700.4800.0034.A710.0470.1800   04B1  3402 ;% 06   WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                  04BC  3406                                                  ;test for good data
U 07. 7700.4800.0364.0300.0470.9808   04BC  3407 ;% 07   NOP,CLOCK.EXT                         ;allow time for memory clock
U 08. 7700.C800.0364.0300.0470.9809   04C7  3411 ;% 08   NOP,CLOCK.EXT                         ;to start up after single bit
U 09. 7700.C800.5BE4.0314.5C80.180A   04D2  3415 ;% 09   WRITE.PHY R[TEMP5]                     ;write ones to memory
U 0A. 7700.4800.0364.0300.0400.180B   04DD  3419 ;% 0A   READ.PHY                                ;read pattern back
U 0B. 7700.4852.5924.0300.0470.180C   04E8  3423 ;% 0B   R[TEMP0]_M[MDR]                        ;save data in RTEMP0
U 0C. 7700.C800.0364.AF00.0470.1801   04F3  3427 ;% 0C   BUT/INT-TIMSERV,NEXT/1801             ;test for corrected data
U 0D. 7700.0800.0034.A714.0470.1800   04FE  3431 ;% 0D   WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                  0509  3435                                                  ;test for good data
U 0E. 7700.481B.0034.A318.0470.1800   0509  3436 ;% 0E   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                  0514  3440                                                  ;test for end of second array
U 0F. 7700.4800.0364.0300.4470.1810   0514  3441 ;% 0F   VA_VA+4                                 ;increment va
U 10. 7500.4800.0364.0300.0470.1811   051F  3445 ;% 10   NOP,DCS.ADDR_0                         ;return to dcs address 0
                                  052A  3449 ;
```

```
                                052A  3450 ; This dcs code here is shared code.  It will be used by both the init code and
                                052A  3451 ; the error code when a single bit error is detected and needs to be logged
                                052A  3452 ;
                                052A  3453 65$:
U 11. 7700.4800.0364.0300.0470.1812   052A  3454 ;% 11   NOP
U 12. 7700.885B.5924.031C.0470.1813   0535  3458 ;% 12   R[TEMP7]_M[VA]                          ;save va for return
U 13. 7700.0800.5BE4.0304.6870.1814   0540  3462 ;% 13   MEMSCAR_R[TEMP1]                       ;disable cmi
U 14. 7700.4800.5BE4.030C.6070.1815   054B  3466 ;% 14   MEMSCR_R[TEMP3]                        ;...
U 15. 7700.8800.5BE4.0308.6870.1816   0556  3470 ;% 15   MEMSCAR_R[TEMP2]                       ;turn on cache
U 16. 7700.C800.5BE4.03D8.6070.1817   0561  3474 ;% 16   MEMSCR_R[ZERO]                         ;...
U 17. 7700.4800.5BE4.03D8.4A70.1818   056C  3478 ;% 17   VA_R[ZERO]                            ;zero the va
U 18. 7700.4800.5BE4.03D8.5C80.1819   0577  3482 ;% 18   WRITE.PHY R[ZERO]                      ;validate cache address 0
U 19. 7700.C800.5BE4.03D8.5470.181A   0582  3486 ;% 19   WDR_UNROT(R[ZERO])                    ;...
U 1A. 7700.C800.0364.0300.4470.181B   058D  3490 ;% 1A   VA_VA+4                               ;increment va by 4
U 1B. 7700.4800.5BE4.03D8.5C80.181C   0598  3494 ;% 1B   WRITE.PHY R[ZERO]                      ;validate cache address 4
U 1C. 7700.4800.5BE4.03D8.5470.181D   05A3  3498 ;% 1C   WDR_UNROT(R[ZERO])
U 1D. 7700.C800.5BE4.03D8.4870.181E   05AE  3502 ;% 1D   PC_R[ZERO]                            ;cause prefetch
U 1E. 7700.0800.5BE4.032C.2470.181F   05B9  3506 ;% 1E   WCTRL/LOADCRAR,WB_R[TEMP11]           ;disable ^p interrupts
U 1F. 7700.C800.5BE4.0330.2C70.1820   05C4  3510 ;% 1F   WCTRL/CONWRITE,WB_R[TEMP12]           ;...
U 20. 7700.0800.5BE4.0304.6870.1821   05CF  3514 ;% 20   MEMSCAR_R[TEMP1]                       ;turn on cmi
U 21. 7700.C800.5BE4.03D8.6070.1822   05DA  3518 ;% 21   MEMSCR_R[ZERO]                         ;...
U 22. 7700.8800.5BE4.0308.6870.1823   05E5  3522 ;% 22   MEMSCAR_R[TEMP2]                       ;disable cache
U 23. 7700.C800.5BE4.030C.6070.1824   05F0  3526 ;% 23   MEMSCR_R[TEMP3]                        ;...
U 24. 7700.8800.5BE4.0320.4A70.1825   05FB  3530 ;% 24   VA_R[TEMP8]                           ;address of CSR0
U 25. 7700.4800.0364.0300.0400.1826   0606  3534 ;% 25   READ.PHY                             ;read CSR0
U 26. 3700.0812.5924.0300.0470.1827   0611  3538 ;% 26   WB_M[MDR],RDM_CMI                    ;MHREG gets CSR0
U 27. 7700.C800.5BE4.0324.5C80.1828   061C  3542 ;% 27   WRITE.PHY R[TEMP9]                    ;clear CSR0
U 28. 7700.C800.0364.7B00.0470.1829   0627  3546 ;% 28   BUT/UVCTR                            ;clear any interrupts pending
U 29. 7700.C800.0364.1700.0470.182A   0632  3550 ;% 29   IRD1TEST                             ;free interrupt pending logic
U 2A. 7700.0800.5BE4.031C.4A70.182B   063D  3554 ;% 2A   VA_R[TEMP7]                          ;reset the va
U 2B. 7300.C800.0364.0300.0470.182C   0648  3558 ;% 2B   NOP,STOP.CLOCK                       ;stop the cpu clock
                                0653  3562 75$:                                            ;array 0 = 1mb yeilds
U 08. 7700.B800.7FF7.FFFF.8470.1809   0653  3563 ;% 08   LONLIT_[00100000]                    ;array 1 starting address
                                065E  3567 77$:                                            ;array 0 = 1mb with array 1 =
U 0C. 7700.F800.7FF6.0001.8470.180D   065E  3568 ;% 0C   LONLIT_[0013FFFC]                    ;256kb, array 1 ending address
                                0669  3572 79$:                                            ;array 0 = 4mb yields
U 08. 7700.B800.7FDF.FFFF.8470.1809   0669  3573 ;% 08   LONLIT_[00400000]                    ;array 1 starting address
                                0674  3577 82$:                                            ;array 0 = 4mb with array 1 =
U 0C. 7700.F800.7FC0.0001.8470.180D   0674  3578 ;% 0C   LONLIT_[007FFFFC]                    ;4mb, array 1 last address
                                067F  3582 84$:                                            ;array 0 = 4mb with array 1 =
U 0C. 7700.F800.7FD8.0001.8470.180D   067F  3583 ;% 0C   LONLIT_[004FFFFC]                    ;1mb, array 1 last adress
                                068A  3587
                                068A  3588 BEGIN_CPU_DATA
                                0002  3589
                                0002  3590 RTEMP_DATA
                                0001  3591
                    00000000    0001  3592          .LONG   ^X00000000                     ;data pattern storage
                    0E000000    0005  3593          .LONG   ^X0E000000                     ;reference cache only address
                    06000000    0009  3594          .LONG   ^X06000000                     ;cache control register address
                    01000000    000D  3595          .LONG   ^X01000000                     ;disable cache
                    00000000    0011  3596          .LONG   ^X00000000                     ;test pattern 1
                    FFFFFFFF    0015  3597          .LONG   ^XFFFFFFFF                     ;test pattern 2
                    0007FFFC    0019  3598          .LONG   ^X0007FFFC                     ;last address for 256kb array 0
                    00000000    001D  3599          .LONG   ^X00000000                     ;storage for va register
                    00F20000    0021  3600          .LONG   ^X00F20000                     ;address of CSR0
```

```
                    E0000000  0025  3601          .LONG   ^XE0000000                      ;clears CSR0
                    10000000  0029  3602          .LONG   ^X10000000                      ;allow single bit error report
                    00C00000  002D  3603          .LONG   ^X00C00000                      ;disable ^p interrupts
                    00400000  0031  3604          .LONG   ^X00400000                      ;...
                              0035  3605
                              0035  3606 END_CPU_DATA
                              068A  3607
                              068A  3608
                              068A  3609 END_TEST_DATA
                              068A
                              068A        ;-------------------------------------------------------------------------
                              068A  3610 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        .SBTTL  "TEST   087     Array 1 Memory Test Part 2
001F  3612 ;*******************************************************************************
001F  3613 ;++
001F  3614 ;
001F  3615 ; FUNCTIONAL DESCRIPTION:
001F  3616 ;
001F  3617 ;       This test will check the second array card writing first 1's then
001F  3618 ;       0's in ascending order.
001F  3619 ;
001F  3620 ;
001F  3621 ; TEST ALGORITHM:
001F  3622 ;
001F  3623 ;       1.  Load u-code to initialize test conditions
001F  3624 ;       2.  Load shared u-code that will validate cache address 0 and 4,
001F  3625 ;           disable Control_P interrupt, and clear CSR0
001F  3626 ;       3.  Execute dcs program
001F  3627 ;               a.  Disable the cmi
001F  3628 ;               b.  Enable the cache
001F  3629 ;               c.  Set va to zero
001F  3630 ;               d.  Validate cache addresses 0 and 4
001F  3631 ;               e.  Load the pc to cause prefetch of addresses 0 and 4
001F  3632 ;               f.  Disable control P interrupts
001F  3633 ;               g.  Enable cmi and disable cache
001F  3634 ;               h.  Clear memory CSR0
001F  3635 ;               i.  Perform IRD1TEST to free interrupt pending logic
001F  3636 ;       4.  Execute init dcs program
001F  3637 ;               a.  Set memory csr1 to allow single bit errors
001F  3638 ;               b.  Read CSR2
001F  3639 ;               c.  Perform IRD1TEST to free interrupt pending logic
001F  3640 ;       5.  Check bits 3:2 of CSR2 to see if there is a second array present in
001F  3641 ;           the system.  If not present then go to next test.
001F  3642 ;       6.  Check bits 1:0 of CSR2 to determine the array 0 size
001F  3643 ;       7.  If array 0 is 256kb then mask address 7FFFC and go test (test data
001F  3644 ;           defaults to test a 256kb module for array 1 assuming array 0 is a
001F  3645 ;           256kb as would have to be to follow configuration specs).
001F  3646 ;       8.  If array 0 is a 1mb module then:
001F  3647 ;               a.  Set array one starting address to 100000
001F  3648 ;               b.  Check for array 1 size
001F  3649 ;               c.  If array 1 is a 1mb module then:
001F  3650 ;                       i.  Set array 1 last address to 1FFFFC
001F  3651 ;                      ii.  Execute DCS code to set R[TEMP6] to 1FFFFC
001F  3652 ;                     iii.  Go test
001F  3653 ;               d.  If array 1 is a 256kb module then:
001F  3654 ;                       i.  Set array 1 last address to 13FFFC
001F  3655 ;                      ii.  Execute DCS code to set R[TEMP6] to 13FFFC
001F  3656 ;                     iii.  Go test
001F  3657 ;       9.  If array 0 is 4mb, then:
001F  3658 ;               a. Set array 1 starting address to 400000
001F  3659 ;               b. Check for array 1 size
001F  3660 ;               c. If array 1 is 4mb then:
001F  3661 ;                       i. Set array 1 last address to 7FFFFC
001F  3662 ;                      ii. Execute dcs code to set R[TEMP6] to 7FFFFC
001F  3663 ;                     iii. Go test
001F  3664 ;               d. If array 1 is 1mb, then:
001F  3665 ;                       i. Set array 1 last address to 4FFFFC
```

```
001F  3666 ;                        ii. Execute dcs code to set R[TEMP6] to 4FFFFC
001F  3667 ;                       iii. Go test
001F  3668 ;       10.  Load index I with 2 which will be used for comparison of index K
001F  3669 ;       11.  Load u-code to write ones to main memory
001F  3670 ;       12.  Load the first u-inst.
001F  3671 ;       13.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
001F  3672 ;            error section as follows:
001F  3673 ;                a.  First pass is from the writing of ones throughout memory
001F  3674 ;                b.  Second pass is from the read-write-read throughout memory
001F  3675 ;       14.  Check to see if this is first pass or second pass, if first pass
001F  3676 ;            then goto 112$, if second pass then goto 110$
001F  3677 ;       15.  Set-up the rdm to watch for u-traps
001F  3678 ;       16.  Set the u-match halt for 1801
001F  3679 ;       17.  Start the cpu clock
001F  3680 ;                a.  Save the va in the rdm WH register
001F  3681 ;                b.  Write ones to the address in the va
001F  3682 ;                c.  Test va to see if at end of second array card
001F  3683 ;                d.  If not increment va and return to dcs address 0
001F  3684 ;       18.  Set-up watchdog timer with loop J to monitor above u-code
001F  3685 ;       19.  If loop J expires, stop the clock and force an error printout
001F  3686 ;       20.  If clock stops normally (u-match at 1801); test WH register for
001F  3687 ;            end of second array
001F  3688 ;       21.  If no error test cs address for not a u-trap
001F  3689 ;       22.  Increment loop K for second pass
001F  3690 ;       23.  Execute dcs code
001F  3691 ;                a.  Set va to the starting address for array 1
001F  3692 ;       24.  Load the test u-code
001F  3693 ;       25.  Load the first u-inst.
001F  3694 ;       26.  Set-up the rdm to watch for u-traps
001F  3695 ;       27.  Set the u-match to 1801
001F  3696 ;       28.  Start the cpu clock
001F  3697 ;                a.  Save va in rdm WH register
001F  3698 ;                b.  Read address specified by va
001F  3699 ;                c.  Move data to RTEMP0
001F  3700 ;                d.  Test for a corrected data interrupt
001F  3701 ;                e.  Test that data is correct
001F  3702 ;                f.  Write zero to the same location
001F  3703 ;                g.  Read location back
001F  3704 ;                h.  Save data in RTEMP0
001F  3705 ;                i.  Test for a corrected data interrupt
001F  3706 ;                j.  Test that data is correct
001F  3707 ;                k.  Test to see if at end of second array
001F  3708 ;                l.  Increment the va by 4
001F  3709 ;                m.  Return to dcs address 0
001F  3710 ;       29.  Set-up watch-dog timer with loop J to monitor above u-code
001F  3711 ;       30.  If loop J expires, stop the clock and force an error printout
001F  3712 ;       31.  If clock stops normally (u-match at 1801) test WH register
001F  3713 ;            for end of second array.
001F  3714 ;       32.  Test for machine checks (double bit error in memory)
001F  3715 ;       33.  If not machine check test for incorrect data on first read
001F  3716 ;       34.  If not bad data test for incorrect data on second read
001F  3717 ;       35.  Test for corrected data interrupts
001F  3718 ;       36.  Service the corrected data interrupt with program at dcs address 14
001F  3719 ;                a.  Save va in RTEMP7
001F  3720 ;                b.  Disable cmi enable cache
```

```
001F  3721 ;                c.  Set va to zero
001F  3722 ;                d.  Validate cache address 0 and 4
001F  3723 ;                e.  Load pc to cause prefetch of addresses 0 and 4
001F  3724 ;                f.  Disable ^p interrupts
001F  3725 ;                g.  Read memory CSR0 to the rdm MHREG
001F  3726 ;                h.  Clear memory CSR0
001F  3727 ;                i.  Clear any interrupts pending
001F  3728 ;                j.  Perform an IRD1TEST to free the interrupt pending logic
001F  3729 ;                k.  Reload va with RTEMP7
001F  3730 ;           37.  Continue running test micro code from step 17.
001F  3731 ;           38.  Test for the end of the second array.
001F  3732 ;           39.  When last address of array is found the test will end on endloop K
001F  3733 ;                being expired
001F  3734 ;           40.  The balance of the pseudo ops handle the various error conditions
001F  3735 ;                (see error description)
001F  3736 ;
001F  3737 ;
001F  3738 ; TEST PATTERNS:
001F  3739 ;
001F  3740 ;            First pass:  init memory to 1's
001F  3741 ;           Second pass:  read 1's; write 0's; read 0's
001F  3742 ;
001F  3743 ;
001F  3744 ; LOGIC DESCRIPTION:
001F  3745 ;
001F  3746 ;           My advice on this test is forget trying to troubleshoot with it.  It
001F  3747 ;           is far to complex to be a usefull troubleshooting tool.  Rather, just
001F  3748 ;           use the error printout to array and/or module swap till you find the
001F  3749 ;           problem.  The second array is throughly tested only after running
001F  3750 ;           all four tests in this series; together and in order.
001F  3751 ;
001F  3752 ;
001F  3753 ; ASSUMPTIONS:
001F  3754 ;
001F  3755 ;           This test assumes the cmi is operational.
001F  3756 ;
001F  3757 ;
001F  3758 ; ERROR DESCRIPTION:
001F  3759 ;
001F  3760 ;           First IFERROR:
001F  3761 ;
001F  3762 ;           EXPECTED LAST ADDRESS OF FIRST ARRAY
001F  3763 ;           RECEIVED LAST ADDRESS
001F  3764 ;           LOOP COUNT I
001F  3765 ;           LOOP COUNT J
001F  3766 ;           LOOP COUNT K
001F  3767 ;           (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
001F  3768 ;
001F  3769 ;           Second IFERROR:
001F  3770 ;
001F  3771 ;           NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  3772 ;           MAIN MEMORY ADDRESS OF DATA ERROR
001F  3773 ;           LOOP COUNT I
001F  3774 ;           LOOP COUNT J
001F  3775 ;           LOOP COUNT K
```

```
001F  3776 ;       RECEIVED DATA (S/B 0'S OR 1'S)
001F  3777 ;       (INDICATES ECC IS PROBABLE NOT WORKING)
001F  3778 ;
001F  3779 ;       Third IFERROR:
001F  3780 ;
001F  3781 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  3782 ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
001F  3783 ;       LOOP COUNT I
001F  3784 ;       LOOP COUNT J
001F  3785 ;       LOOP COUNT K
001F  3786 ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
001F  3787 ;
001F  3788 ;--
001F  3789 ;*******************************************************************************
001F  3790
001F  3791 ;-------------------------------------------------------------------------------
001F  3792 ;
001F  3793 ; Initialize code
001F  3794 ;
001F  3795 ;-------------------------------------------------------------------------------
001F  3796
001F  3797          INITIALIZE                                  ;init the cpu
0021  3798          LOADDCS         40$,,0,<^X1C>               ;setup init micro code
0028  3799          LOADDCS         65$,,<^X11>,<^X1B>          ;set-up shared micro code
002F  3800          FETCH           <^X11>                      ;execute shared micro code
0034  3801          BRSTCLK         <^X2B>,INH                  ;end shared micro code
0038  3802          FETCH           0                           ;start dcs program
003D  3803          BRSTCLK         <^X0A>,INH                  ;end dcs program
0041  3804          CMPREGMSK       WHREG,23$,,25$,,B           ;check array 1 type
004D  3805          SKIP            155$,NOERROR                ;skip if nothing there (exit)
0054  3806          CMPREGMSK       WHREG,31$,,18$,,B           ;check array 0
0060  3807          SKIP            90$,NOERROR                 ;skip if 4mb
0067  3808          CMPREGMSK       WHREG,21$,,18$,,B,NE        ;check array 0 type
0073  3809          SKIP            100$,ONERROR                ;skip if 1mb module
007A  3810          MASK            4$,,19$,,L                  ;array 1 last address = 7FFFC
0083  3811          SKIP            108$                        ;go test
008A  3812 ;
008A  3813 ; This section of code handles the case where array 0 is 4mb. It will also
008A  3814 ; check the size of array 1.
008A  3815 ;
008A  3816 90$:     LDFIELD         32$,,L,57$,31,62,LON        ;array 1 start address = 400000
009B  3817          LOADDCS         79$,,<^X08>,1               ;array 1 start address = 400000
00A2  3818          FETCH           <^X08>                      ;start dcs
00A7  3819          BRSTCLK         <^X0A>,INH                  ;end dcs
00AB  3820          CMPREGMSK       WHREG,33$,,25$,,B           ;check array 1
00B7  3821          SKIP            95$,ONERROR                 ;skip if 1mb
00BE  3822          LOADDCS         82$,,<^X0C>,1               ;array 1 last address = 7FFFFC
00C5  3823          MASK            4$,,35$,,L                  ;array 1 last address = 7FFFFC
00CE  3824          SKIP            102$                        ;execute changed dcs
00D5  3825 95$:     LOADDCS         84$,,<^X0C>,1               ;array 1 last address = 4FFFFC
00DC  3826          MASK            4$,,37$,,L                  ;array 1 last address = 4FFFFC
00E5  3827          SKIP            102$                        ;execute changed dcs
00EC  3828 ;
00EC  3829 ; This section of code will handle the case where array 0 (first array slot) is
00EC  3830 ; a 1mb array module.  It will also check the size of array 1
```

```
00EC  3831 ;
00EC  3832 100$:   LDFIELD         24$,,L,57$,31,62,LON    ;array 1 start address = 100000
00FD  3833         LOADDCS         75$,,<^X08>,1           ;array 1 start address = 100000
0104  3834         FETCH           <^X08>                  ;start dcs program
0109  3835         BRSTCLK         <^X0A>,INH              ;end dcs program
010D  3836         CMPREGMSK       WHREG,26$,,25$,,B       ;check array 1 size
0119  3837         SKIP            104$,ONERROR            ;skip if 256kb module
0120  3838         MASK            4$,,20$,,L              ;array 1 last address = 1FFFFC
0129  3839 ;
0129  3840 ; Come here to execute the altered dcs and set up last address
0129  3841 ;
0129  3842 102$:   FETCH           <^X0B>                  ;start dcs program
012E  3843         BRSTCLK         <^X0F>,INH              ;end dcs program
0132  3844         SKIP            108$                    ;go test
0139  3845 ;
0139  3846 ; Come here if array 1is a 256kb module.  This section will skip back to 102$
0139  3847 ; to execute altered dcs to establish the correct last address.
0139  3848 ;
0139  3849 104$:   LOADDCS         77$,,<^X0C>,1           ;array 1 last address = 13FFFC
0140  3850         MASK            4$,,27$,,L              ;array 1 last address = 13FFFC
0149  3851         SKIP            102$                    ;go execute changed dcs
0150  3852
0150  3853 ;-------------------------------------------------------------------------------
0150  3854 ; Write memory with 1's
0150  3855 ;
0150  3856 ;-------------------------------------------------------------------------------
0150  3857
0150  3858 108$:   LDINDEX         I,2                     ;dummy index I gets 2
0157  3859         LOADDCS         50$,,0,<^X0B>           ;test u-code
015E  3860         FETCH           0                       ;load first u-inst.
0163  3861         LOOP            K,1,2                   ;set index K for two passes
016A  3862         SKIP            110$,ONEQUAL,I,K        ;skip on second pass
0171  3863         SKIP            112$                    ;first pass, goto rdm set-up
0178  3864
0178  3865 ;-------------------------------------------------------------------------------
0178  3866 ;
0178  3867 ; Read the 1's from the previous write, check for valid data, write 0's, check
0178  3868 ; for valid data
0178  3869 ;
0178  3870 ;-------------------------------------------------------------------------------
0178  3871
0178  3872 110$:   FETCH           7                       ;reset va starting address
017D  3873         BRSTCLK         <^X0A>,INH              ;...
0181  3874         LOADDCS         60$,,0,<^X11>           ;test u-code
0188  3875         FETCH           0                       ;load first u-inst.
018D  3876 ;
018D  3877 ; This section used for both the initial writing of 1's (first pass) then for
018D  3878 ; the read of 1's-write of 0's-read of 0's (second pass)
018D  3879 ;
018D  3880 112$:   LOADREG         DCSCR,9$,,B             ;insure clock is off
0195  3881         LOADREG         CSAMR,2$,,W             ;load match address
019D  3882         LOADREG         RDCTRL,7$,,B            ;clear master halt enable
01A5  3883         LOADREG         RDCTRL,1$,,B            ;watch for u-traps
01AD  3884         LOADREG         DCSCR,3$,,B             ;start the clock
01B5  3885         LOOP            J,1,8192                ;watchdog timing loop
```

```
01BC  3886          CMPREGMSK       CLKDCS,7$,,8$,,B,       ;test for clock stopped
01C8  3887          SKIP            114$,ONERROR            ;skip when clock stops
01CF  3888          ENDLOOP         J                       ;end timing loop
01D2  3889          LOADREG         DCSCR,9$,,B             ;stop the clock
01DA  3890          CMPREG          WHREG,4$,,L,            ;force an error
01E3  3891          IFERROR         ,,<<MC0><MC1><MC2>>     ;program lost
01EB  3892          SKIP
01F2  3893
01F2  3894 ;-------------------------------------------------------------------------------
01F2  3895 ;
01F2  3896 ; Error evaluation code
01F2  3897 ;
01F2  3898 ;-------------------------------------------------------------------------------
01F2  3899
01F2  3900 114$:    CMPREGMSK       CSTRP,10$,,6$,,W,NE     ;test for machine check
01FE  3901          SKIP            140$,ONERROR            ;handle machine check
0205  3902          CMPREGMSK       CLKDCS,11$,,12$,,B,NE   ;test for data errors
0211  3903          SKIP            130$,ONERROR            ;incorrect data
0218  3904          CMPREGMSK       CLKDCS,13$,,12$,,B,NE   ;test for data errors
0224  3905          SKIP            130$,ONERROR            ;incorrect data
022B  3906          CMPREGMSK       CLKDCS,14$,,12$,,B,NE   ;test for single bit error on 0
0237  3907          SKIP            120$,ONERROR            ;log error
023E  3908          CMPREGMSK       CLKDCS,16$,,12$,,B,NE   ;test for single bit error on 1
024A  3909          SKIP            115$,ONERROR            ;log error
0251  3910          CMPREG          WHREG,4$,,L,            ;test for last address
025A  3911          SKIP            150$,NOERROR            ;go to next test
0261  3912          CMPREGMSK       CLKDCS,17$,,12$,,B      ;force and error
026D  3913          IFERROR                                 ;i think we're lost
0272  3914          SKIP                                    ;end test due to error
0279  3915 ;
0279  3916 ; Come here if the Second read caused corrected read data to be returned
0279  3917 ;
0279  3918 115$:    FETCH           <^X11>                  ;start dcs program
027E  3919          BRSTCLK         <^X2B>                  ;end dcs program
0282  3920          ERRLOG                                  ;log the error
0284  3921          SKIP            150$,ONERROR            ;quit after 64 errors
028B  3922          FETCH           <^X0D>                  ;set up to continue testing
0290  3923          SKIP            112$                    ;test more memory
0297  3924 ;
0297  3925 ; Come here if the First read caused corrected read data to be returned
0297  3926 ;
0297  3927 120$:    FETCH           <^X11>                  ;start dcs program
029C  3928          BRSTCLK         <^X2B>                  ;end dcs program
02A0  3929          ERRLOG                                  ;log the error
02A2  3930          SKIP            150$,ONERROR            ;quit after 64 errors
02A9  3931          FETCH           <^X06>                  ;set up to continue testing
02AE  3932          SKIP            112$                    ;test more memory
02B5  3933 ;
02B5  3934 ; Come here if the data read is not the data written
02B5  3935 ;
02B5  3936 130$:    CMPREG          WHREG,15$,,L,           ;force an error
02BE  3937          IFERROR         1,<<MEC><MDL>>,-
02BE  3938          <<MC0><MC1><MC2><MA0><MA1><MA2>>        ;print va and data
02CB  3939          SKIP                                    ;end test because of error
02D2  3940 ;
```

```
                    02D2  3941 ; Come here if there is a Machine check (Read Data Substitute)
                    02D2  3942 ;
                    02D2  3943 140$:   CMPREG          WHREG,15$,,L,           ;force an error
                    02DB  3944         IFERROR         ,<<MEC><MDL>>,-
                    02DB  3945         <<MC0><MC1><MC2><MA0><MA1><MA2>>        ;print va of machine check
                    02E8  3946 ;-------------------------------------------------------------------------------
                    02E8  3947 ;
                    02E8  3948 ; This is where we exit the test if index K is equal to 2
                    02E8  3949 ;
                    02E8  3950 ;-------------------------------------------------------------------------------
                    02E8  3951
                    02E8  3952 150$:   ENDLOOP         K
                    02EB  3953 155$:   SKIP                                    ;go to next test
                    02F2  3954
                    02F2  3955
                    02F2  3956 BEGIN_TEST_DATA
                    02F2
                    02F2       ;-------------------------------------------------------------------
                    02F2  3957
              28    02F2  3958 1$:     .BYTE   ^X28                            ;set to watch for u-traps
            1801    02F3  3959 2$:     .WORD   ^X1801                          ;match address
              11    02F5  3960 3$:     .BYTE   ^X11                            ;run clock
        FFFFFFFF    02F6  3961 4$:     .LONG   ^XFFFFFFFF                      ;last address of second array
            3FFF    02FA  3962 6$:     .WORD   ^X3FFF                          ;mask for cmpregmsk pseudo op
              00    02FC  3963 7$:     .BYTE   ^X00                            ;watch for clock stop
              C0    02FD  3964 8$:     .BYTE   ^XC0                            ;mask for cmpregmsk pseudo op
              60    02FE  3965 9$:     .BYTE   ^X60                            ;stop clock manually
            0028    02FF  3966 10$:    .WORD   ^X0028                          ;machine check u-trap
              08    0301  3967 11$:    .BYTE   ^X08                            ;dcs halt on data error
              3F    0302  3968 12$:    .BYTE   ^X3F                            ;mask for cmpregmsk pseudo op
              0F    0303  3969 13$:    .BYTE   ^X0F                            ;dcs halt on data error
              07    0304  3970 14$:    .BYTE   ^X07                            ;halt single bit error on 0's
        0F0F0F0F    0305  3971 15$:    .LONG   ^X0F0F0F0F                      ;compare data to force errors
              0E    0309  3972 16$:    .BYTE   ^X0E                            ;halt single bit error on 1's
              FF    030A  3973 17$:    .BYTE   ^XFF                            ;used to force errors
              03    030B  3974 18$:    .BYTE   ^X03                            ;array 0 mask
        0007FFFC    030C  3975 19$:    .LONG   ^X0007FFFC                      ;256kb array 1 last address
        001FFFFC    0310  3976 20$:    .LONG   ^X001FFFFC                      ;1mb array 1 last address
              02    0314  3977 21$:    .BYTE   ^X02                            ;array 0 1mb (csr2 bits 0:1)
              00    0315  3978 23$:    .BYTE   ^X00                            ;slot empty compare data
        00100000    0316  3979 24$:    .LONG   ^X00100000                      ;array 1 starting address
              0C    031A  3980 25$:    .BYTE   ^X0C                            ;array 1 mask
              08    031B  3981 26$:    .BYTE   ^X08                            ;arrat 1 1mb (csr2 bits 3:2)
        0013FFFC    031C  3982 27$:    .LONG   ^X0013FFFC                      ;array 1 = 256kb last address
              01    0320  3983 31$:    .BYTE   ^X01                            ;array 0 4mb
        00400000    0321  3984 32$:    .LONG   ^X00400000                      ;array 1 start address
              04    0325  3985 33$:    .BYTE   ^X04                            ;array 1 4mb
        007FFFFC    0326  3986 35$:    .LONG   ^X007FFFFC                      ;4mb array 1 last address
        004FFFFC    032A  3987 37$:    .LONG   ^X004FFFFC                      ;1mb array 1 last address
                    032E  3988 40$:
U 00, 7700,C800,0364,0300,0470,1801 032E  3989 ;* 00    NOP
U 01, 7700,8800,5BE4,0320,4A70,1802 0339  3993 ;* 01    VA_R[TEMP8]                     ;CSR0 address to va
U 02, 7700,C800,0364,0300,4470,1803 0344  3997 ;* 02    VA_VA+4                         ;CSR1 address to va
U 03, 7700,4800,5BE4,0328,5C80,1804 034F  4001 ;* 03    WRITE.PHY R[TEMP10]             ;allow single bit errors
U 04, 7700,C800,0364,0300,4470,1805 035A  4005 ;* 04    VA_VA+4                         ;CSR2 address to va
```

```
U 05. 7700.C800.0364.0300.0400.1806  0365  4009 ;% 05   READ.PHY                                          ;read CSR2
U 06. 6700.8812.5924.0300.0470.1807  0370  4013 ;% 06   RDM_WB,WB_M[MDR]                                  ;WHREG gets contents of CSR2
U 07. 7700.C800.0364.1700.0470.1808  037B  4017 ;% 07   IRDITEST                                         ;clear int pending line
U 08. 7700.B800.7FFD.FFFF.8470.1809  0386  4021 ;% 08   LONLIT_[00040000]                                ;array starting address
U 09. 7700.4800.5BE4.03D4.4A70.180A  0391  4025 ;% 09   VA_R[LONLIT]                                     ;VA gets starting address
U 0A. 7300.C800.0364.0300.0470.180B  039C  4029 ;% 0A   NOP,STOP.CLOCK                                   ;stop the cpu clock
U 0B. 7700.4800.0364.0300.0470.180C  03A7  4033 ;% 0B   NOP
U 0C. 7700.F800.7FF0.0001.8470.180D  03B2  4037 ;% 0C   LONLIT_[001FFFFC]                                ;array 1 ending address
U 0D. 7700.8868.5BE4.03D4.0470.180E  03BD  4041 ;% 0D   M[TEMP8]_R[LONLIT]                               ;that gets setup in R[TEMP6]
U 0E. 7700.0848.5924.0318.0470.180F  03C8  4045 ;% 0E   R[TEMP6]_M[TEMP8]
U 0F. 7300.C800.0364.0300.0470.1810  03D3  4049 ;% 0F   NOP,STOP.CLOCK
                                     03DE  4053 50$:
U 00. 7700.C800.0364.0300.0470.1802  03DE  4054 ;% 00   NOP,NEXT/1802
U 01. 6700.881B.5924.0300.0470.1802  03E9  4058 ;% 01   RDM_WB,WB_M[VA]                                  ;save va in rdm
U 02. 7700.C800.5BE4.0314.5C80.1803  03F4  4062 ;% 02   WRITE.PHY R[TEMP5]                               ;write ones to memory
U 03. 7700.4800.0364.0300.0470.9804  03FF  4066 ;% 03   NOP,CLOCK.EXT                                    ;allow time for write
U 04. 7700.481B.0034.A318.0470.1800  040A  4070 ;% 04   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     0415  4074                                                         ;test for end of second array
U 05. 7700.C800.0364.0300.4470.1806  0415  4075 ;% 05   VA_VA+4                                         ;increment va register
U 06. 7500.C800.0364.0300.0470.1807  0420  4079 ;% 06   NOP,DCS.ADDR_0                                  ;return to address 0
U 07. 7700.C800.0364.0300.0470.1808  042B  4083 ;% 07   NOP
                                     0436  4087 57$:
U 08. 7700.B800.7FFD.FFFF.8470.1809  0436  4088 ;% 08   LONLIT_[00040000]                                ;array starting address
U 09. 7700.4800.5BE4.03D4.4A70.180A  0441  4092 ;% 09   VA_R[LONLIT]                                     ;VA gets starting address
U 0A. 7300.C800.0364.0300.0470.180B  044C  4096 ;% 0A   NOP,STOP.CLOCK                                   ;stop the cpu clock
                                     0457  4100 60$:
U 00. 7700.4800.0364.0300.0470.9802  0457  4101 ;% 00   NOP,CLOCK.EXT,NEXT/1802                          ;allow time for memory
U 01. 7700.C800.0364.0300.0470.1802  0462  4105 ;% 01   NOP                                             ;clock to start up.
U 02. 6700.081B.5924.0300.0470.1803  046D  4109 ;% 02   RDM_WB,WB_M[VA]                                  ;save va in rdm
U 03. 7700.4800.0364.0300.0400.1804  0478  4113 ;% 03   READ.PHY                                         ;read pattern back
U 04. 7700.4852.5924.0300.0470.1805  0483  4117 ;% 04   R[TEMP0]_M[MDR]                                  ;save data in RTEMP0
U 05. 7700.C800.0364.AF00.0470.1801  048E  4121 ;% 05   BUT/INT-TIMSERV,NEXT/1801                       ;test for corrected data
U 06. 7700.0800.0034.A714.0470.1800  0499  4125 ;% 06   WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                     04A4  4129                                                         ;test for good data
U 07. 7700.4800.0364.0300.0470.9808  04A4  4130 ;% 07   NOP,CLOCK.EXT                                    ;allow time for memory clock
U 08. 7700.C800.0364.0300.0470.9809  04AF  4134 ;% 08   NOP,CLOCK.EXT                                    ;to start up after single bit
U 09. 7700.8800.5BE4.0310.5C80.180A  04BA  4138 ;% 09   WRITE.PHY R[TEMP4]                               ;write zeros to memory
U 0A. 7700.4800.0364.0300.0400.180B  04C5  4142 ;% 0A   READ.PHY                                         ;read pattern back
U 0B. 7700.4852.5924.0300.0470.180C  04D0  4146 ;% 0B   R[TEMP0]_M[MDR]                                  ;save data in RTEMP0
U 0C. 7700.C800.0364.AF00.0470.1801  04DB  4150 ;% 0C   BUT/INT-TIMSERV,NEXT/1801                       ;test for corrected data
U 0D. 7700.4800.0034.A710.0470.1800  04E6  4154 ;% 0D   WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                     04F1  4158                                                         ;test for good data
U 0E. 7700.481B.0034.A318.0470.1800  04F1  4159 ;% 0E   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     04FC  4163                                                         ;test for end of second array
U 0F. 7700.4800.0364.0300.4470.1810  04FC  4164 ;% 0F   VA_VA+4                                         ;increment va
U 10. 7500.4800.0364.0300.0470.1811  0507  4168 ;% 10   NOP,DCS.ADDR_0                                  ;return to dcs address 0
                                     0512  4172 ;
                                     0512  4173 ; This dcs code here is shared code.  It will be used by both the init code and
                                     0512  4174 ; the error code when a single bit error is detected and needs to be logged
                                     0512  4175 ;
                                     0512  4176 65$:
U 11. 7700.4800.0364.0300.0470.1812  0512  4177 ;% 11   NOP
U 12. 7700.885B.5924.031C.0470.1813  051D  4181 ;% 12   R[TEMP7]_M[VA]                                  ;save va for return
U 13. 7700.0800.5BE4.0304.6870.1814  0528  4185 ;% 13   MEMSCAR_R[TEMP1]                                ;disable cmi
U 14. 7700.4800.5BE4.030C.6070.1815  0533  4189 ;% 14   MEMSCR_R[TEMP3]                                 ;...
```

```
U 15. 7700.8800.5BE4.0308.6870.1816  053E  4193 ;* 15   MEMSCAR_R[TEMP2]                    ;turn on cache
U 16. 7700.C800.5BE4.03D8.6070.1817  0549  4197 ;* 16   MEMSCR_R[ZERO]                     ;...
U 17. 7700.4800.5BE4.03D8.4A70.1818  0554  4201 ;* 17   VA_R[ZERO]                        ;zero the va
U 18. 7700.4800.5BE4.03D8.5C80.1819  055F  4205 ;* 18   WRITE.PHY R[ZERO]                 ;validate cache address 0
U 19. 7700.C800.5BE4.03D8.5470.181A  056A  4209 ;* 19   WDR_UNROT(R[ZERO])               ;...
U 1A. 7700.C800.0364.0300.4470.181B  0575  4213 ;* 1A   VA_VA+4                          ;increment va by 4
U 1B. 7700.4800.5BE4.03D8.5C80.181C  0580  4217 ;* 1B   WRITE.PHY R[ZERO]                 ;validate cache address 4
U 1C. 7700.4800.5BE4.03D8.5470.181D  058B  4221 ;* 1C   WDR_UNROT(R[ZERO])
U 1D. 7700.C800.5BE4.03D8.4870.181E  0596  4225 ;* 1D   PC_R[ZERO]                       ;cause prefetch
U 1E. 7700.0800.5BE4.032C.2470.181F  05A1  4229 ;* 1E   WCTRL/LOADCRAR,WB_R[TEMP11]      ;disable ^p interrupts
U 1F. 7700.C800.5BE4.0330.2C70.1820  05AC  4233 ;* 1F   WCTRL/CONWRITE,WB_R[TEMP12]     ;...
U 20. 7700.0800.5BE4.0304.6870.1821  05B7  4237 ;* 20   MEMSCAR_R[TEMP1]                 ;turn on cmi
U 21. 7700.C800.5BE4.03D8.6070.1822  05C2  4241 ;* 21   MEMSCR_R[ZERO]                   ;...
U 22. 7700.8800.5BE4.0308.6870.1823  05CD  4245 ;* 22   MEMSCAR_R[TEMP2]                 ;disable cache
U 23. 7700.C800.5BE4.030C.6070.1824  05D8  4249 ;* 23   MEMSCR_R[TEMP3]                  ;...
U 24. 7700.8800.5BE4.0320.4A70.1825  05E3  4253 ;* 24   VA_R[TEMP8]                      ;address of CSR0
U 25. 7700.4800.0364.0300.0400.1826  05EE  4257 ;* 25   READ.PHY                         ;read CSR0
U 26. 3700.0812.5924.0300.0470.1827  05F9  4261 ;* 26   WB_M[MDR],RDM_CMI               ;MHREG gets CSR0
U 27. 7700.C800.5BE4.0324.5C80.1828  0604  4265 ;* 27   WRITE.PHY R[TEMP9]              ;clear CSR0
U 28. 7700.C800.0364.7B00.0470.1829  060F  4269 ;* 28   BUT/UVCTR                       ;clear any interrupts pending
U 29. 7700.C800.0364.1700.0470.182A  061A  4273 ;* 29   IRD1TEST                        ;free interrupt pending logic
U 21. 7700.0800.5BE4.031C.4A70.1822  0625  4277 ;* 21   VA_R[TEMP7]                     ;reset the va
U 2B. 7300.C800.0364.0300.0470.182C  0630  4281 ;* 2B   NOP,STOP.CLOCK                  ;stop the cpu clock
                                     063B  4285 75$:                                  ;array 0 = 1mb yields
U 08. 7700.B800.7FF7.FFFF.8470.1809  063B  4286 ;* 08   LONLIT_[00100000]              ;array 1 starting address
                                     0646  4290 77$:                                  ;array 0 = 1mb with array 1 =
U 0C. 7700.F800.7FF6.0001.8470.180D  0646  4291 ;* 0C   LONLIT_[0013FFFC]              ;256kb, array 1 ending address
                                     0651  4295 79$:                                  ;array 0 = 4mb yields
U 08. 7700.B800.7FDF.FFFF.8470.1809  0651  4296 ;* 08   LONLIT_[00400000]              ;array 1 starting address
                                     065C  4300 82$:                                  ;array 0 = 4mb with array 1 =
U 0C. 7700.F800.7FC0.0001.8470.180D  065C  4301 ;* 0C   LONLIT_[007FFFFC]              ;4mb, array 1 last address
                                     0667  4305 84$:                                  ;array 0 = 4mb with array 1 =
U 0C. 7700.F800.7FD8.0001.8470.180D  0667  4306 ;* 0C   LONLIT_[004FFFFC]              ;1mb, array 1 last adress
                                     0672  4310
                                     0672  4311 BEGIN_CPU_DATA
                                     0002  4312
                                     0002  4313 RTEMP_DATA
                                     0001  4314
                         00000000    0001  4315         .LONG   ^X00000000           ;data pattern storage
                         0E000000    0005  4316         .LONG   ^X0E000000           ;reference cache only address
                         06000000    0009  4317         .LONG   ^X06000000           ;cache control register address
                         01000000    000D  4318         .LONG   ^X01000000           ;disable cache
                         00000000    0011  4319         .LONG   ^X00000000           ;test pattern 1
                         FFFFFFFF    0015  4320         .LONG   ^XFFFFFFFF           ;test pattern 2
                         0007FFFC    0019  4321         .LONG   ^X0007FFFC           ;last address for 256kb array 0
                         00000000    001D  4322         .LONG   ^X00000000           ;storage for va register
                         00F20000    0021  4323         .LONG   ^X00F20000           ;address of CSR0
                         E0000000    0025  4324         .LONG   ^XE0000000           ;clears CSR0
                         10000000    0029  4325         .LONG   ^X10000000           ;allow single bit error report
                         00C00000    002D  4326         .LONG   ^X00C00000           ;disable ^p interrupts
                         00400000    0031  4327         .LONG   ^X00400000           ;...
                                     0035  4328
                                     0035  4329 END_CPU_DATA
                                     0672  4330
                                     0672  4331 END_TEST_DATA
```

```
                                0672
                                0672        ;-----------------------------------------------------------------------
                                0672  4332 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        001F        .SBTTL  "TEST   088     Array 1 Memory Test Part 3
                        001F  4334 ;*******************************************************************************
                        001F  4335 ;++
                        001F  4336 ;
                        001F  4337 ; FUNCTIONAL DESCRIPTION:
                        001F  4338 ;
                        001F  4339 ;       This test will check the second array card writing first 0's then
                        001F  4340 ;       1's in decending order.
                        001F  4341 ;
                        001F  4342 ;
                        001F  4343 ; TEST ALGORITHM:
                        001F  4344 ;
                        001F  4345 ;       1.  Load U-code to initialize test conditions
                        001F  4346 ;       2.  Load shared u-code that will validate cache address 0 and 4,
                        001F  4347 ;           disable Control_P interrupts, and clear CSR0
                        001F  4348 ;       3.  Execute dcs program
                        001F  4349 ;               a.  Disable the CMI
                        001F  4350 ;               b.  Enable the cache
                        001F  4351 ;               c.  Set VA to zero
                        001F  4352 ;               d.  Validate cache addresses 0 and 4
                        001F  4353 ;               e.  Load the PC to cause prefetch of addresses 0 and 4
                        001F  4354 ;               f.  Disable control P interrupts
                        001F  4355 ;               g.  Enable CMI and disable cache
                        001F  4356 ;               h.  Clear memory CSR0
                        001F  4357 ;               i.  Perform IRD1TEST to free interrupt pending logic
                        001F  4358 ;       4.  Execute init dcs program
                        001F  4359 ;               a.  Set memory CSR1 to allow single bit errors
                        001F  4360 ;               b.  Read CSR2
                        001F  4361 ;               c.  Perform IRD1TEST to free interrupt pending logic
                        001F  4362 ;       5.  Check bits 3:2 of CSR2 to see if there is a second array present in
                        001F  4363 ;           the system.  If not present then go to next test.
                        001F  4364 ;       6.  Check bits 1:0 of CSR2 to determine the array 0 size
                        001F  4365 ;       7.  If array 0 is 256kb then mask address 40000 and go test (test data
                        001F  4366 ;           defaults to test a 256kb module for array 1 assuming array 0 is a
                        001F  4367 ;           256kb as would have to be to follow confiruration specs).
                        001F  4368 ;       8.  If array 0 is a 1mb module then:
                        001F  4369 ;               a.  Set array 1 starting address to 100000
                        001F  4370 ;               b.  Set R[TEMP6] to 100000
                        001F  4371 ;               c.  Check for array 1 size
                        001F  4372 ;               d.  If array 1 is a 1mb module then:
                        001F  4373 ;                       i.  Set array 1 last address to 1FFFFC
                        001F  4374 ;                      ii.  Execute DCs code to set VA to 1FFFFC
                        001F  4375 ;                     iii.  Go test
                        001F  4376 ;               e.  If array 1 is a 256kb module then:
                        001F  4377 ;                       i.  Set array 1 last address to 13FFFC
                        001F  4378 ;                      ii.  Execute DCS code to set VA to 13FFFC
                        001F  4379 ;                     iii.  Go test
                        001F  4380 ;       9.  If array 0 is 4mb, then:
                        001F  4381 ;               a. Set array 1 starting address to 400000
                        001F  4382 ;               b. Set R[TEMP6] to 400000
                        001F  4383 ;               c. Check array 1 size
                        001F  4384 ;               d. If array 1 is 4mb, then:
                        001F  4385 ;                       i. Set array 1 last address to 7FFFFC
                        001F  4386 ;                      ii. Execute dcs code to set VA to 7FFFFC
                        001F  4387 ;                     iii. Go test
```

```
001F  4388 ;                    e. If array 1 is 1mb, then:
001F  4389 ;                         i. Set array 1 last address to 4FFFFC
001F  4390 ;                        ii. Execute dcs code to set VA to 4FFFFC
001F  4391 ;                       iii. Go test
001F  4392 ;       10.  Load index I with 2 which will be used for comparison of index K
001F  4393 ;       11.  Load U-code to write zeros to main memory
001F  4394 ;       12.  Load the first u-inst.
001F  4395 ;       13.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
001F  4396 ;            error section as follows:
001F  4397 ;                a.  First pass is from the writing of zeros throughout memory
001F  4398 ;                b.  Second pass is from the read-write-read throughout memory
001F  4399 ;       14.  Check to see if this is first pass or second pass, if first pass
001F  4400 ;            then goto 112$, if second pass then goto 110$
001F  4401 ;       15.  Set-up the rdm to watch for u-traps
001F  4402 ;       16.  Set the u-match halt for 1801
001F  4403 ;       17.  Start the cpu clock
001F  4404 ;                a.  Save the VA in the RDM WH register
001F  4405 ;                b.  Write zero to the address in the VA
001F  4406 ;                c.  Test VA to see if at end of second array card
001F  4407 ;                d.  If not decrement VA and return to DCS address 0
001F  4408 ;       18.  Set-up watchdog timer with loop J to monitor above U-code
001F  4409 ;       19.  If loop J expires, stop the clock and force an error printout
001F  4410 ;       20.  If clock stops normally (U-MATCH AT 1801); test WH register for
001F  4411 ;            end of second array
001F  4412 ;       21.  If no error test cs address for not a u-trap
001F  4413 ;       22.  Increment loop K for second pass
001F  4414 ;       23.  Execute DCS code
001F  4415 ;                a.  Set VA to last address of array 1
001F  4416 ;       24.  Load the test U-code
001F  4417 ;       25.  Load the first U-inst.
001F  4418 ;       26.  Set-up the RDM to watch for u-traps
001F  4419 ;       27.  Set the u-match to 1801
001F  4420 ;       28.  Start the cpu clock
001F  4421 ;                a.  Save VA in RDM WH register
001F  4422 ;                b.  Read address specified by VA
001F  4423 ;                c.  Move data to RTEMP0
001F  4424 ;                d.  Test for a corrected data interrupt
001F  4425 ;                e.  Test that data is correct
001F  4426 ;                f.  Write ones to the same location
001F  4427 ;                g.  Read location back
001F  4428 ;                h.  Save data in RTEMP0
001F  4429 ;                i.  Test for a corrected data interrupt
001F  4430 ;                j.  Test that data is correct
001F  4431 ;                k.  Test to see if at end of second array
001F  4432 ;                l.  Decrement the VA by 4
001F  4433 ;                m.  Return to DCS address 0
001F  4434 ;       29.  Set-up watch-dog timer with loop J to monitor above U-code
001F  4435 ;       30.  If loop J expires, stop the clock and force an error printout
001F  4436 ;       31.  If clock stops normally (U-MATCH AT 1801) test WH register
001F  4437 ;            for end of array.
001F  4438 ;       32.  Test for machine checks (double bit error in memory)
001F  4439 ;       33.  If not machine check test for incorrect data on first read
001F  4440 ;       34.  If not bad data test for incorrect data on second read
001F  4441 ;       35.  Test for corrected data interrupts
001F  4442 ;       36.  Service the corrected data interrut with program at DCS address 14
```

```
001F  4443 ;                a.  Save VA in RTEMP7
001F  4444 ;                b.  Disable CMI enable cache
001F  4445 ;                c.  Set VA to zero
001F  4446 ;                d.  Validate cache address 0 and 4
001F  4447 ;                e.  Load PC to cause prefetch of addresses 0 and 4
001F  4448 ;                f.  Disable ^P interrupts
001F  4449 ;                g.  Read memory CSR0 to the RDM MHREG
001F  4450 ;                h.  Clear memory CSR0
001F  4451 ;                i.  Clear any interrupts pending
001F  4452 ;                j.  Perform an IRD1TEST to free the interrupt pending logic
001F  4453 ;                k.  Reload VA with RTEMP7
001F  4454 ;          37.  Continue running test micro code from step 17.
001F  4455 ;          38.  Test for the end of the second array.
001F  4456 ;          39.  When last address of array is found the test will end on ENDLOOP K
001F  4457 ;               being expired
001F  4458 ;          40.  The balance of the PSEUDO OPS handle the various error conditions
001F  4459 ;               (see error description)
001F  4460 ;
001F  4461 ;
001F  4462 ; TEST PATTERNS:
001F  4463 ;
001F  4464 ;           First pass:  Init memory to 0's
001F  4465 ;          Second pass:  read 0's; write 1's; read 1's
001F  4466 ;
001F  4467 ;
001F  4468 ; LOGIC DESCRIPTION:
001F  4469 ;
001F  4470 ;          My advice on this test is forget trying to troubleshoot with it.  It
001F  4471 ;          is far to complex to be a usefull troubleshooting tool.  Rather, just
001F  4472 ;          use the error printout to array and/or module swap till you find the
001F  4473 ;          problem.  The second array is throughly tested only after running
001F  4474 ;          all four tests in this series; together and in order.
001F  4475 ;
001F  4476 ;
001F  4477 ; ASSUMPTIONS:
001F  4478 ;
001F  4479 ;          This test assumes the CMI is operational.
001F  4480 ;
001F  4481 ;
001F  4482 ; ERROR DESCRIPTION:
001F  4483 ;
001F  4484 ;          First IFERROR:
001F  4485 ;
001F  4486 ;          EXPECTED LAST ADDRESS OF FIRST ARRAY
001F  4487 ;          RECEIVED LAST ADDRESS
001F  4488 ;          LOOP COUNT I
001F  4489 ;          LOOP COUNT J
001F  4490 ;          LOOP COUNT K
001F  4491 ;          (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
001F  4492 ;
001F  4493 ;          Second IFERROR:
001F  4494 ;
001F  4495 ;          NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  4496 ;          MAIN MEMORY ADDRESS OF DATA ERROR
001F  4497 ;          LOOP COUNT I
```

```
001F  4498 ;       LOOP COUNT J
001F  4499 ;       LOOP COUNT K
001F  4500 ;       RECEIVED DATA (S/B 0'S OR 1'S)
001F  4501 ;       (INDICATES ECC IS PROBABLE NOT WORKING)
001F  4502 ;
001F  4503 ;       Third IFERROR:
001F  4504 ;
001F  4505 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  4506 ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
001F  4507 ;       LOOP COUNT I
001F  4508 ;       LOOP COUNT J
001F  4509 ;       LOOP COUNT K
001F  4510 ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
001F  4511 ;
001F  4512 ;--
001F  4513 ;*******************************************************************************
001F  4514
001F  4515 ;-------------------------------------------------------------------------------
001F  4516 ;
001F  4517 ; Initialize code
001F  4518 ;
001F  4519 ;-------------------------------------------------------------------------------
001F  4520
001F  4521         INITIALIZE                                  ;init the cpu
0021  4522         LOADDCS         40$,,0,<^X1C>               ;setup init micro code
0028  4523         LOADDCS         65$,,<^X11>,<^X1B>          ;set-up shared micro code
002F  4524         FETCH           <^X11>                      ;execute shared micro code
0034  4525         BRSTCLK         <^X2B>,INH                  ;end shared micro code
0038  4526         FETCH           0                           ;start dcs program
003D  4527         BRSTCLK         <^X0A>,INH                  ;end dcs program
0041  4528         CMPREGMSK       WHREG,23$,,25$,,B           ;check array 1 type
004D  4529         SKIP            155$,NOERROR                ;skip if nothing there
0054  4530         CMPREGMSK       WHREG,28$,,18$,,B           ;check array 0 size
0060  4531         SKIP            90$,NOERROR                 ;skip if 4mb module
0067  4532         CMPREGMSK       WHREG,21$,,18$,,B,NE        ;check array 0 type
0073  4533         SKIP            100$,ONERROR                ;skip if 1mb module
007A  4534         MASK            4$,,19$,,L                  ;array 1 last address = 40000
0083  4535         SKIP            108$                        ;go test
008A  4536 ;
008A  4537 ; This section of code handles the case where array 0 is 4mb. It will also
008A  4538 ; check the size of array 1
008A  4539 ;
008A  4540 90$:    MASK            4$,,29$,,L                  ;array 1 starting addr = 400000
0093  4541         LOADDCS         80$,,<^X0C>,1               ;array 1 starting addr = 400000
009A  4542         FETCH           <^X0B>                      ;start dcs
009F  4543         BRSTCLK         <^X0F>,INH                  ;end dcs
00A3  4544         CMPREGMSK       WHREG,30$,,25$,,B           ;check array 1 size
00AF  4545         SKIP            95$,ONERROR                 ;skip if 1mb
00B6  4546         LDFIELD         32$,,L,57$,31,62,LON        ;last address = 7FFFFC
00C7  4547         LOADDCS         82$,,<^X08>,1               ;last address = 7FFFFC
00CE  4548         SKIP            102$                        ;execute changed dcs
00D5  4549 95$:    LDFIELD         34$,,L,57$,31,62,LON        ;last address = 4FFFFC
00E6  4550         LOADDCS         84$,,<^X08>,1               ;last address = 4FFFFC
00ED  4551         SKIP            102$                        ;execute changed dcs
00F4  4552 ;
```

```
00F4  4553 ; This section of code will handle the case where array 0 (first array slot) is
00F4  4554 ; a 1mb array module.  It will also check the size of array 1
00F4  4555 ;
00F4  4556 100$:   MASK            4$,,24$,,L              ;array 1 starting addr = 100000
00FD  4557         FETCH           <^X0B>                  ;start dcs program
0102  4558         BRSTCLK         <^X0F>,INH              ;end dcs program
0106  4559         CMPREGMSK       WHREG,26$,,25$,,B       ;check array 1 size
0112  4560         SKIP            104$,ONERROR            ;skip if array 1 is 256kb
0119  4561         LDFIELD         20$,,L,57$,31,62,LON    ;array 1 last address = 1FFFFC
012A  4562         LOADDCS         75$,,<^X08>,1           ;array 1 last address = 1FFFFC
0131  4563 ;
0131  4564 ; Come here to execute the altered dcs and set up last address
0131  4565 ;
0131  4566 102$:   FETCH           <^X08>                  ;start dcs program
0136  4567         BRSTCLK         <^X0A>,INH              ;end dcs program
013A  4568         SKIP            108$                    ;go test
0141  4569 ;
0141  4570 ; Come here if array 1 is a 256kb module.  This section will skip back to 102$
0141  4571 ; to execute altered dcs to establish the correct last address
0141  4572 ;
0141  4573 104$:   LDFIELD         27$,,L,57$,31,62,LON    ;array 1 last address = 13FFFC
0152  4574         LOADDCS         77$,,<^X08>,1           ;array 1 last address = 13FFFC
0159  4575         SKIP            102$                    ;go execute changed dcs
0160  4576
0160  4577 ;-------------------------------------------------------------------------------
0160  4578 ;
0160  4579 ; Write memory with 0's
0160  4580 ;
0160  4581 ;-------------------------------------------------------------------------------
0160  4582
0160  4583 108$:   LDINDEX         I,2                     ;dummy index I gets 2
0167  4584         LOADDCS         50$,,0,<^X0B>           ;test u-code
016E  4585         FETCH           0                       ;load first u-inst.
0173  4586         LOOP            K,1,2                   ;set index K for two passes
017A  4587         SKIP            110$,ONEQUAL,I,K        ;skip on second pass
0181  4588         SKIP            112$                    ;first pass, goto rdm set-up
0188  4589
0188  4590 ;-------------------------------------------------------------------------------
0188  4591 ;
0188  4592 ; Read the 0's from the previous write, check for valid data, write 1's, check
0188  4593 ; for valid data
0188  4594 ;
0188  4595 ;-------------------------------------------------------------------------------
0188  4596
0188  4597 110$:   FETCH           7                       ;reset va to array 1 last addr
018D  4598         BRSTCLK         <^X0A>,INH              ;...
0191  4599         LOADDCS         60$,,0,<^X11>           ;test u-code
0198  4600         FETCH           0                       ;load first u-inst.
019D  4601 ;
019D  4602 ; This section is used for both the initial writing of 0's (first pass) then
019D  4603 ; for the read of 0's-write of 1's-read of 1's (second pass)
019D  4604 ;
019D  4605 112$:   LOADREG         DCSCR,9$,,B             ;insure clock is off
01A5  4606         LOADREG         CSAMR,2$,,W             ;load match address
01AD  4607         LOADREG         RDCTRL,7$,,B            ;clear master halt enable
```

```
01B5  4608          LOADREG         RDCTRL,1$,,B            ;watch for u-traps
01BD  4609          LOADREG         DCSCR,3$,,B             ;start the clock
01C5  4610          LOOP            J,1,8192                ;watchdog timing loop
01CC  4611          CMPREGMSK       CLKDCS,7$,,8$,,B,       ;test for clock stopped
01D8  4612          SKIP            114$,ONERROR            ;skip when clock stops
01DF  4613          ENDLOOP         J                       ;end timing loop
01E2  4614          LOADREG         DCSCR,9$,,B             ;stop the clock
01EA  4615          CMPREG          WHREG,4$,,L,            ;force an error
01F3  4616          IFERROR         ,,<<MC0><MC1><MC2>>     ;program lost
01FB  4617          SKIP
0202  4618
0202  4619 ;-------------------------------------------------------------------------------
0202  4620 ;
0202  4621 ; Error evaluation code
0202  4622 ;
0202  4623 ;-------------------------------------------------------------------------------
0202  4624
0202  4625 114$:    CMPREGMSK       CSTRP,10$,,6$,,W,NE     ;test for machine check
020E  4626          SKIP            140$,ONERROR            ;handle machine check
0215  4627          CMPREGMSK       CLKDCS,11$,,12$,,B,NE   ;test for data errors
0221  4628          SKIP            130$,ONERROR            ;incorrect data
0228  4629          CMPREGMSK       CLKDCS,13$,,12$,,B,NE   ;test for data errors
0234  4630          SKIP            130$,ONERROR            ;incorrect data
023B  4631          CMPREGMSK       CLKDCS,14$,,12$,,B,NE   ;test for single bit error on 0
0247  4632          SKIP            120$,ONERROR            ;log error
024E  4633          CMPREGMSK       CLKDCS,16$,,12$,,B,NE   ;test for single bit error on 1
025A  4634          SKIP            115$,ONERROR            ;log error
0261  4635          CMPREG          WHREG,4$,,L,            ;test for last address
026A  4636          SKIP            150$,NOERROR            ;go to next test
0271  4637          CMPREGMSK       CLKDCS,17$,,12$,,B      ;force and error
027D  4638          IFERROR                                 ;i think we're lost
0282  4639          SKIP                                    ;end test due to error
0289  4640 ;
0289  4641 ; Come here if the Second read caused corrected read data to be returned
0289  4642 ;
0289  4643 115$:    FETCH           <^X11>                  ;start dcs program
028E  4644          BRSTCLK         <^X2B>                  ;end dcs program
0292  4645          ERRLOG                                  ;log the error
0294  4646          SKIP            150$,ONERROR            ;quit after 64 errors
029B  4647          FETCH           <^X0D>                  ;set up to continue testing
02A0  4648          SKIP            112$                    ;test more memory
02A7  4649 ;
02A7  4650 ; Come here if the First read caused corrected read data to be returned
02A7  4651 ;
02A7  4652 120$:    FETCH           <^X11>                  ;start dcs program
02AC  4653          BRSTCLK         <^X2B>                  ;end dcs program
02B0  4654          ERRLOG                                  ;log the error
02B2  4655          SKIP            150$,ONERROR            ;quit after 64 errors
02B9  4656          FETCH           <^X06>                  ;set up to continue testing
02BE  4657          SKIP            112$                    ;test more memory
02C5  4658 ;
02C5  4659 ; Come here if the data read is not the data written
02C5  4660 ;
02C5  4661 130$:    CMPREG          WHREG,15$,,L,           ;force an error
02CE  4662          IFERROR         1,<<MEC><MDL>>,-
```

```
                02CE  4663          <<MC0><MC1><MC2><MA0><MA1><MA2>>         ;print va and data
                02DB  4664          SKIP                                    ;end test because of error
                02E2  4665 ;
                02E2  4666 ; Come here if there was a Machine check (Read Data Substitute)
                02E2  4667 ;
                02E2  4668 140$:    CMPREG          WHREG,15$,,L,           ;force an error
                02EB  4669          IFERROR         ,<<MEC><MDL>>,-
                02EB  4670          <<MC0><MC1><MC2><MA0><MA1><MA2>>        ;print va of machine check
                02F8  4671 ;-------------------------------------------------------------------------
                02F8  4672 ;
                02F8  4673 ; This is where we exit the test if index K is equal to 2
                02F8  4674 ;
                02F8  4675 ;-------------------------------------------------------------------------
                02F8  4676
                02F8  4677 150$:    ENDLOOP         K
                02FB  4678 155$:    SKIP                                    ;go to next test
                0302  4679
                0302  4680
                0302  4681 BEGIN_TEST_DATA
                0302
                0302        ;-------------------------------------------------------------------
                0302  4682
          28    0302  4683 1$:      .BYTE   ^X28                            ;set to watch for u-traps
        1801    0303  4684 2$:      .WORD   ^X1801                          ;match address
          11    0305  4685 3$:      .BYTE   ^X11                            ;run clock
    FFFFFFFF    0306  4686 4$:      .LONG   ^XFFFFFFFF                      ;starting address for array 1
        3FFF    030A  4687 6$:      .WORD   ^X3FFF                          ;mask for cmpregmsk pseudo op
          00    030C  4688 7$:      .BYTE   ^X00                            ;watch for clock stop
          C0    030D  4689 8$:      .BYTE   ^XC0                            ;mask for cmpregmsk pseudo op
          60    030E  4690 9$:      .BYTE   ^X60                            ;stop clock manually
        0028    030F  4691 10$:     .WORD   ^X0028                          ;machine check u-trap
          08    0311  4692 11$:     .BYTE   ^X08                            ;dcs halt on data error
          3F    0312  4693 12$:     .BYTE   ^X3F                            ;mask for cmpregmsk pseudo op
          0F    0313  4694 13$:     .BYTE   ^X0F                            ;dcs halt on data error
          07    0314  4695 14$:     .BYTE   ^X07                            ;halt single bit error on 0's
    0F0F0F0F    0315  4696 15$:     .LONG   ^X0F0F0F0F                      ;compare data to force errors
          0E    0319  4697 16$:     .BYTE   ^X0E                            ;halt single bit error on 1's
          FF    031A  4698 17$:     .BYTE   ^XFF                            ;used to force errors
          03    031B  4699 18$:     .BYTE   ^X03                            ;array 0 mask
    00040000    031C  4700 19$:     .LONG   ^X00040000                      ;array 1 starting address
    001FFFFC    0320  4701 20$:     .LONG   ^X001FFFFC                      ;array 1 ending address - 1mb
          02    0324  4702 21$:     .BYTE   ^X02                            ;array 0 256kb (CSR2 bits 0:1)
          00    0325  4703 23$:     .BYTE   ^X00                            ;slot empty compare data
    00100000    0326  4704 24$:     .LONG   ^X00100000                      ;array 1 starting address
          0C    032A  4705 25$:     .BYTE   ^X0C                            ;array 1 mask
          08    032B  4706 26$:     .BYTE   ^X08                            ;array 1 1mb (csr2 bits 3:2)
    0013FFFC    032C  4707 27$:     .LONG   ^X0013FFFC                      ;array 1 ending address - 256kb
          01    0330  4708 28$:     .BYTE   ^X01                            ;array 0 4mb
    00400000    0331  4709 29$:     .LONG   ^X00400000                      ;array 1 start addr
          04    0335  4710 30$:     .BYTE   ^X04                            ;array 1 4mb
    007FFFFC    0336  4711 32$:     .LONG   ^X007FFFFC                      ;array 1 last address - 4mb
    004FFFFC    033A  4712 34$:     .LONG   ^X004FFFFC                      ;array 1 last address - 1mb
                033E  4713 40$:
U 00, 7700,C800,0364,0300,0470,1801 033E  4714 ;% 00   NOP
U 01, 7700,8800,5BE4,0320,4A70,1802 0349  4718 ;% 01   VA_R[TEMP8]                                ;CSR0 address to va
```

```
U 02. 7700,C800,0364,0300,4470,1803  0354  4722 ;% 02   VA_VA+4                                          ;CSR1 address to va
U 03. 7700,4800,5BE4,0328,5C80,1804  035F  4726 ;% 03   WRITE.PHY R[TEMP10]                               ;allow single bit errors
U 04. 7700,C800,0364,0300,4470,1805  036A  4730 ;% 04   VA_VA+4                                          ;CSR2 address to va
U 05. 7700,C800,0364,0300,0400,1806  0375  4734 ;% 05   READ.PHY                                         ;read CSR2
U 06. 6700,8812,5924,0300,0470,1807  0380  4738 ;% 06   RDM_WB,WB_M[MDR]                                ;WHREG gets contents of CSR2
U 07. 7700,C800,0364,1700,0470,1808  038B  4742 ;% 07   IRDITEST                                        ;clear int pending line
U 08. 7700,7800,7FFC,0001,8470,1809  0396  4746 ;% 08   LONLIT_[0007FFFC]                               ;array 1 ending address
U 09. 7700,4800,5BE4,03D4,4A70,180A  03A1  4750 ;% 09   VA_R[LONLIT]                                    ;VA gets ending address
U 0A. 7300,C800,0364,0300,0470,180B  03AC  4754 ;% 0A   NOP,STOP.CLOCK                                  ;stop the cpu clock
U 0B. 7700,4800,0364,0300,0470,180C  03B7  4758 ;% 0B   NOP
U 0C. 7700,3800,7FF7,FFFF,8470,180D  03C2  4762 ;% 0C   LONLIT_[00100000]                               ;array 1 starting address
U 0D. 7700,8868,5BE4,03D4,0470,180E  03CD  4766 ;% 0D   M[TEMP8]_R[LONLIT]                             ;store in M[TEMP8]
U 0E. 7700,0848,5924,0318,0470,180F  03D8  4770 ;% 0E   R[TEMP6]_M[TEMP8]                              ;R[TEMP6] gets start addr.
U 0F. 7300,C800,0364,0300,0470,1810  03E3  4774 ;% 0F   NOP,STOP.CLOCK
                                     03EE  4778 50$:
U 00. 7700,C800,0364,0300,0470,1802  03EE  4779 ;% 00   NOP,NEXT/1802
U 01. 6700,881B,5924,0300,0470,1802  03F9  4783 ;% 01   RDM_WB,WB_M[VA]                                 ;save va in rdm
U 02. 7700,8800,5BE4,0310,5C80,1803  0404  4787 ;% 02   WRITE.PHY R[TEMP4]                               ;write zeros to memory
U 03. 7700,4800,0364,0300,0470,9804  040F  4791 ;% 03   NOP,CLOCK.EXT                                   ;allow time for write
U 04. 7700,481B,0034,A318,0470,1800  041A  4795 ;% 04   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     0425  4799                                                        ;test for end of second array
U 05. 7700,581B,C100,0302,4A70,1806  0425  4800 ;% 05   VA_VA-ZLIT0[4]                                 ;decrement va register
U 06. 7500,C800,0364,0300,0470,1807  0430  4804 ;% 06   NOP,DCS.ADDR_0                                 ;return to address 0
U 07. 7700,C800,0364,0300,0470,1808  043B  4808 ;% 07   NOP
                                     0446  4812 57$:
U 08. 7700,7800,7FFC,0001,8470,1809  0446  4813 ;% 08   LONLIT_[0007FFFC]                               ;array 1 ending address
U 09. 7700,4800,5BE4,03D4,4A70,180A  0451  4817 ;% 09   VA_R[LONLIT]                                    ;VA gets array 1 ending address
U 0A. 7300,C800,0364,0300,0470,180B  045C  4821 ;% 0A   NOP,STOP.CLOCK                                  ;stop the cpu clock
                                     0467  4825 60$:
U 00. 7700,4800,0364,0300,0470,9802  0467  4826 ;% 00   NOP,CLOCK.EXT,NEXT/1802                         ;allow time for memory
U 01. 7700,C800,0364,0300,0470,1802  0472  4830 ;% 01   NOP                                             ;clock to start up.
U 02. 6700,081B,5924,0300,0470,1803  047D  4834 ;% 02   RDM_WB,WB_M[VA]                                 ;save va in rdm
U 03. 7700,4800,0364,0300,0400,1804  0488  4838 ;% 03   READ.PHY                                         ;read pattern back
U 04. 7700,4852,5924,0300,0470,1805  0493  4842 ;% 04   R[TEMP0]_M[MDR]                                 ;save data in RTEMP0
U 05. 7700,C800,0364,AF00,0470,1801  049E  4846 ;% 05   BUT/INT-TIMSERV,NEXT/1801                      ;test for corrected data
U 06. 7700,4800,0034,A710,0470,1800  04A9  4850 ;% 06   WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                     04B4  4854                                                        ;test for good data
U 07. 7700,4800,0364,0300,0470,9808  04B4  4855 ;% 07   NOP,CLOCK.EXT                                   ;allow time for memory clock
U 08. 7700,C800,0364,0300,0470,9809  04BF  4859 ;% 08   NOP,CLOCK.EXT                                   ;to start up after single bit
U 09. 7700,C800,5BE4,0314,5C80,180A  04CA  4863 ;% 09   WRITE.PHY R[TEMP5]                               ;write ones to memory
U 0A. 7700,4800,0364,0300,0400,180B  04D5  4867 ;% 0A   READ.PHY                                         ;read pattern back
U 0B. 7700,4852,5924,0300,0470,180C  04E0  4871 ;% 0B   R[TEMP0]_M[MDR]                                 ;save data in RTEMP0
U 0C. 7700,C800,0364,AF00,0470,1801  04EB  4875 ;% 0C   BUT/INT-TIMSERV,NEXT/1801                      ;test for corrected data
U 0D. 7700,0800,0034,A714,0470,1800  04F6  4879 ;% 0D   WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                     0501  4883                                                        ;test for good data
U 0E. 7700,481B,0034,A318,0470,1800  0501  4884 ;% 0E   WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                     050C  4888                                                        ;test for end of second array
U 0F. 7700,D81B,C100,0302,4A70,1810  050C  4889 ;% 0F   VA_VA-ZLIT0[4]                                 ;decrement va
U 10. 7500,4800,0364,0300,0470,1811  0517  4893 ;% 10   NOP,DCS.ADDR_0                                 ;return to dcs address 0
                                     0522  4897 ;
                                     0522  4898 ; This dcs code here is shared code.  It will be used by both the init code and
                                     0522  4899 ; the error code when a single bit error is detected and needs to be logged
                                     0522  4900 ;
                                     0522  4901 65$:
U 11. 7700,4800,0364,0300,0470,1812  0522  4902 ;% 11   NOP
```

```
U 12. 7700,885B,5924,031C,0470,1813  052D  4906 ;% 12   R[TEMP7]_M[VA]                    ;save va for return
U 13. 7700,0800,5BE4,0304,6870,1814  0538  4910 ;% 13   MEMSCAR_R[TEMP1]                 ;disable cmi
U 14. 7700,4800,5BE4,030C,6070,1815  0543  4914 ;% 14   MEMSCR_R[TEMP3]                  ;...
U 15. 7700,8800,5BE4,0308,6870,1816  054E  4918 ;% 15   MEMSCAR_R[TEMP2]                 ;turn on cache
U 16. 7700,C800,5BE4,03D8,6070,1817  0559  4922 ;% 16   MEMSCR_R[ZERO]                   ;...
U 17. 7700,4800,5BE4,03D8,4A70,1818  0564  4926 ;% 17   VA_R[ZERO]                      ;zero the va
U 18. 7700,4800,5BE4,03D8,5C80,1819  056F  4930 ;% 18   WRITE.PHY R[ZERO]               ;validate cache address 0
U 19. 7700,C800,5BE4,03D8,5470,181A  057A  4934 ;% 19   WDR_UNROT(R[ZERO])             ;...
U 1A. 7700,C800,0364,0300,4470,181B  0585  4938 ;% 1A   VA_VA+4                        ;increment va by 4
U 1B. 7700,4800,5BE4,03D8,5C80,181C  0590  4942 ;% 1B   WRITE.PHY R[ZERO]               ;validate cache address 4
U 1C. 7700,4800,5BE4,03D8,5470,181D  059B  4946 ;% 1C   WDR_UNROT(R[ZERO])
U 1D. 7700,C800,5BE4,03D8,4870,181E  05A6  4950 ;% 1D   PC_R[ZERO]                     ;cause prefetch
U 1E. 7700,0800,5BE4,032C,2470,181F  05B1  4954 ;% 1E   WCTRL/LOADCRAR,WB_R[TEMP11]    ;disable ^P interrupts
U 1F. 7700,C800,5BE4,0330,2C70,1820  05BC  4958 ;% 1F   WCTRL/CONWRITE,WB_R[TEMP12]   ;...
U 20. 7700,0800,5BE4,0304,6870,1821  05C7  4962 ;% 20   MEMSCAR_R[TEMP1]               ;turn on cmi
U 21. 7700,C800,5BE4,03D8,6070,1822  05D2  4966 ;% 21   MEMSCR_R[ZERO]                 ;...
U 22. 7700,8800,5BE4,0308,6870,1823  05DD  4970 ;% 22   MEMSCAR_R[TEMP2]               ;disable cache
U 23. 7700,C800,5BE4,030C,6070,1824  05E8  4974 ;% 23   MEMSCR_R[TEMP3]               ;...
U 24. 7700,8800,5BE4,0320,4A70,1825  05F3  4978 ;% 24   VA_R[TEMP8]                   ;address of CSR0
U 25. 7700,4800,0364,0300,0400,1826  05FE  4982 ;% 25   READ.PHY                     ;read CSR0
U 26. 3700,0812,5924,0300,0470,1827  0609  4986 ;% 26   WB_M[MDR],RDM_CMI            ;MHREG gets CSR0
U 27. 7700,C800,5BE4,0324,5C80,1828  0614  4990 ;% 27   WRITE.PHY R[TEMP9]           ;clear CSR0
U 28. 7700,C800,0364,7B00,0470,1829  061F  4994 ;% 28   BUT/UVCTR                    ;clear any interrupts pending
U 29. 7700,C800,0364,1700,0470,182A  062A  4998 ;% 29   IRD1TEST                     ;free interrupt pending logic
U 2A. 7700,0800,5BE4,031C,4A70,182B  0635  5002 ;% 2A   VA_R[TEMP7]                   ;reset the va
U 2B. 7300,C800,0364,0300,0470,182C  0640  5006 ;% 2B   NOP,STOP.CLOCK               ;stop the cpu clock
                                     064B  5010 75$:
U 08. 7700,7800,7FF0,0001,8470,1809  064B  5011 ;% 08   LONLIT_[001FFFFC]            ;array 1 ending address - 1mb
                                     0656  5015 77$:
U 08. 7700,7800,7FF6,0001,8470,1809  0656  5016 ;% 08   LONLIT_[0013FFFC]            ;array 1 ending address - 256kb
                                     0661  5020 80$:
U 0C. 7700,3800,7FDF,FFFF,8470,180D  0661  5021 ;% 0C   LONLIT_[00400000]            ;array 1 starting address
                                     066C  5025 82$:
U 08. 7700,7800,7FC0,0001,8470,1809  066C  5026 ;% 08   LONLIT_[007FFFFC]            ;array 1 last address - 4mb
                                     0677  5030 84$:
U 08. 7700,7800,7FD8,0001,8470,1809  0677  5031 ;% 08   LONLIT_[004FFFFC]            ;array 1 last address - 1mb
                                     0682  5035 BEGIN_CPU_DATA
                                     0002  5036
                                     0002  5037 RTEMP_DATA
                                     0001  5038
                        00000000     0001  5039          .LONG   ^X00000000        ;data pattern storage
                        0E000000     0005  5040          .LONG   ^X0E000000        ;reference cache only address
                        06000000     0009  5041          .LONG   ^X06000000        ;cache control register address
                        01000000     000D  5042          .LONG   ^X01000000        ;disable cache
                        00000000     0011  5043          .LONG   ^X00000000        ;test pattern 1
                        FFFFFFFF     0015  5044          .LONG   ^XFFFFFFFF        ;test pattern 2
                        00040000     0019  5045          .LONG   ^X00040000        ;last address for 256kb array 0
                        00000000     001D  5046          .LONG   ^X00000000        ;storage for va register
                        00F20000     0021  5047          .LONG   ^X00F20000        ;address of CSR0
                        E0000000     0025  5048          .LONG   ^XE0000000        ;clears CSR0
                        10000000     0029  5049          .LONG   ^X10000000        ;allow single bit error report
                        00C00000     002D  5050          .LONG   ^X00C00000        ;disable ^P interrupts
                        00400000     0031  5051          .LONG   ^X00400000        ;...
                                     0035  5052
                                     0035  5053 END_CPU_DATA
```

```
ZZ-ECKAC-8.0    V08.00                    "TEST 088 Array 27-FEB-1986      Fiche 4  Frame F7        Sequence 701
ECKAC                          VAX-11/750 MIC MICRO DIAGNOSTICS          27-FEB-1986 11:42:33  VAX/VMS Macro V04-00        Page  8
V08.00                         "TEST 088 Array 1 Memory Test Part 3      27-FEB-1986 11:42:08  [VAX750.MIC]ECKAC4.MAR;1          (1

                                    0682  5054
                                    0682  5055
                                    0682  5056 END_TEST_DATA
                                    0682
                                    0682       ;-----------------------------------------------------------------------
                                    0682  5057 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        001F        .SBTTL  "TEST   089     Array 1 Memory Test Part 4
                        001F  5059 ;*******************************************************************************
                        001F  5060 ;++
                        001F  5061 ;
                        001F  5062 ; FUNCTIONAL DESCRIPTION:
                        001F  5063 ;
                        001F  5064 ;       This test will check the second array card writing first 1's then
                        001F  5065 ;       0's in decending order.
                        001F  5066 ;
                        001F  5067 ;
                        001F  5068 ; TEST ALGORITHM:
                        001F  5069 ;
                        001F  5070 ;       1.  Load U-code to initialize test conditions
                        001F  5071 ;       2.  Load shared u-code that will validate cache address 0 and 4,
                        001F  5072 ;           disable Control_P interrupts, and clear CSR0
                        001F  5073 ;       3.  Execute dcs program
                        001F  5074 ;               a.  Disable the CMI
                        001F  5075 ;               b.  Enable the cache
                        001F  5076 ;               c.  Set VA to zero
                        001F  5077 ;               d.  Validate cache addresses 0 and 4
                        001F  5078 ;               e.  Load the PC to cause prefetch of addresses 0 and 4
                        001F  5079 ;               f.  Disable control P interrupts
                        001F  5080 ;               g.  Enable CMI and disable cache
                        001F  5081 ;               h.  Clear memory CSR0
                        001F  5082 ;               i.  Perform IRD1TEST to free interrupt pending logic
                        001F  5083 ;       4.  Execute init dcs program
                        001F  5084 ;               a.  Set memory CSR1 to allow single bit errors
                        001F  5085 ;               b.  Read CSR2
                        001F  5086 ;               c.  Perform IRD1TEST to free interrupt pending logic
                        001F  5087 ;       5.  Check bits 3:2 of CSR2 to see if there is a second array present in
                        001F  5088 ;           the system.  If not present then go to the next test.
                        001F  5089 ;       6.  Check bits 1:0 of CSR2 to determine the array 0 size
                        001F  5090 ;       7.  If array 0 is 256kb then mask address 40000 and go test (test data
                        001F  5091 ;           defaults to test a 256kb module for array 1 assuming array 0 is a
                        001F  5092 ;           256kb as would have to be to follow configuration specs).
                        001F  5093 ;       8.  If array 0 is a 1mb module then:
                        001F  5094 ;               a.  Set array 1 starting address to 100000
                        001F  5095 ;               b.  Set R[TEMP6] to 100000
                        001F  5096 ;               c.  Check for array 1 size
                        001F  5097 ;               d.  If array 1 is a 1mb module then:
                        001F  5098 ;                       i.  Set array 1 last address to 1FFFFC
                        001F  5099 ;                      ii.  Execute DCS code to set VA to 1FFFFC
                        001F  5100 ;                     iii.  Go test
                        001F  5101 ;               e.  If array 1 is a 256kb module then:
                        001F  5102 ;                       i.  Set array 1 last address to 13FFFC
                        001F  5103 ;                      ii.  Execute DCS code to set VA to 13FFFC
                        001F  5104 ;                     iii.  Go test
                        001F  5105 ;       9.  If array 0 is 4mb, then:
                        001F  5106 ;               a. Set array 1 starting address to 400000
                        001F  5107 ;               b. Set R[TEMP6] to 400000
                        001F  5108 ;               c. Check array 1 size
                        001F  5109 ;               d. If array 1 is 4mb, then:
                        001F  5110 ;                       i. Set array 1 last address to 7FFFFC
                        001F  5111 ;                      ii. Execute dcs code to set VA to 7FFFFC
                        001F  5112 ;                     iii. Go test
```

```
001F  5113 ;                    e. If array 1 is 1mb, then:
001F  5114 ;                            i. Set array 1 last address to 4FFFFC
001F  5115 ;                           ii. Execute dcs code to set VA to 4FFFFC
001F  5116 ;                          iii. Go test
001F  5117 ;       10.  Load index I with 2 which will be used for comparison of index K
001F  5118 ;       11.  Load U-code to write ones to main memory
001F  5119 ;       12.  Load the first u-inst.
001F  5120 ;       13.  Set-up a loop (K) of 2 for two passes through code at 112$ and the
001F  5121 ;            error section as follows:
001F  5122 ;                a.  First pass is from the writing of ones throughout memory
001F  5123 ;                b.  Second pass is from the read-write-read throughout memory
001F  5124 ;       14.  Check to see if this is first pass or second pass, if first pass
001F  5125 ;            then goto 112$, if second pass then goto 110$
001F  5126 ;       15.  Set-up the rdm to watch for u-traps
001F  5127 ;       16.  Set the u-match halt for 1801
001F  5128 ;       17.  Start the cpu clock
001F  5129 ;                a.  Save the VA in the RDM WH register
001F  5130 ;                b.  Write ones to the address in the VA
001F  5131 ;                c.  Test VA to see if at end of second array card
001F  5132 ;                d.  If not decrement VA and return to DCS address 0
001F  5133 ;       18.  Set-up watchdog timer with loop J to monitor above U-code
001F  5134 ;       19.  If loop J expires, stop the clock and force an error printout
001F  5135 ;       20.  If clock stops normally (U-MATCH AT 1801); test WH register for
001F  5136 ;            end of second array
001F  5137 ;       21.  If no error test cs address for not a u-trap
001F  5138 ;       22.  Increment loop K for second pass
001F  5139 ;       23.  Execute DCS code
001F  5140 ;                a.  Set VA to last address of array 1
001F  5141 ;       24.  Load the test U-code
001F  5142 ;       25.  Load the first U-inst.
001F  5143 ;       26.  Set-up the RDM to watch for u-traps
001F  5144 ;       27.  Set the u-match to 1801
001F  5145 ;       28.  Start the cpu clock
001F  5146 ;                a.  Save VA in RDM WH register
001F  5147 ;                b.  Read address specified by VA
001F  5148 ;                c.  Move data to RTEMP0
001F  5149 ;                d.  Test for a corrected data interrupt
001F  5150 ;                e.  Test that data is correct
001F  5151 ;                f.  Write zero to the same location
001F  5152 ;                g.  Read location back
001F  5153 ;                h.  Save data in RTEMP0
001F  5154 ;                i.  Test for a corrected data interrupt
001F  5155 ;                j.  Test that data is correct
001F  5156 ;                k.  Test to see if at end of second array
001F  5157 ;                l.  Decrement the VA by 4
001F  5158 ;                m.  Return to DCS address 0
001F  5159 ;       29.  Set-up watch-dog timer with loop J to monitor above U-code
001F  5160 ;       30.  If loop J expires, stop the clock and force an error printout
001F  5161 ;       31.  If clock stops normally (U-MATCH AT 1801) test WH register
001F  5162 ;            for end of array.
001F  5163 ;       32.  Test for machine checks (double bit error in memory)
001F  5164 ;       33.  If not machine check test for incorrect data on first read
001F  5165 ;       34.  If not bad data test for incorrect data on second read
001F  5166 ;       35.  Test for corrected data interrupts
001F  5167 ;       36.  Service the corrected data interrut with program at DCS address 14
```

```
001F  5168 ;                a.  Save VA in RTEMP7
001F  5169 ;                b.  Disable CMI enable cache
001F  5170 ;                c.  Set VA to zero
001F  5171 ;                d.  Validate cache address 0 and 4
001F  5172 ;                e.  Load PC to cause prefetch of addresses 0 and 4
001F  5173 ;                f.  Disable ^P interrupts
001F  5174 ;                g.  Read memory CSR0 to the RDM MHREG
001F  5175 ;                h.  Clear memory CSR0
001F  5176 ;                i.  Clear any interrupts pending
001F  5177 ;                j.  Perform an IRD1TEST to free the interrupt pending logic
001F  5178 ;                k.  Reload VA with RTEMP7
001F  5179 ;          37.  Continue running test micro code from step 17.
001F  5180 ;          38.  Test for the end of the second array.
001F  5181 ;          39.  When last address of array is found the test will end on ENDLOOP K
001F  5182 ;               being expired
001F  5183 ;          40.  When this test is completed (ENDLOOP K expired) then print message
001F  5184 ;               'FINISHED ARRAY 1' as this is the last test of 4 tests that check
001F  5185 ;               array 1
001F  5186 ;          41.  The balance of the PSEUDO OPS handle the various error conditions
001F  5187 ;               (see error description)
001F  5188 ;
001F  5189 ;
001F  5190 ; TEST PATTERNS:
001F  5191 ;
001F  5192 ;           First pass:  Init memory to 1's
001F  5193 ;          Second pass:  read 1's; write 0's; read 0's
001F  5194 ;
001F  5195 ;
001F  5196 ; LOGIC DESCRIPTION:
001F  5197 ;
001F  5198 ;          My advice on this test is forget trying to troubleshoot with it.  It
001F  5199 ;          is far to complex to be a usefull troubleshooting tool.  Rather, just
001F  5200 ;          use the error printout to array and/or module swap till you find the
001F  5201 ;          problem.  The second array is throughly tested only after running
001F  5202 ;          all four tests in this series; together and in order.
001F  5203 ;
001F  5204 ;
001F  5205 ; ASSUMPTIONS:
001F  5206 ;
001F  5207 ;          This test assumes the CMI is operational.
001F  5208 ;
001F  5209 ;
001F  5210 ; ERROR DESCRIPTION:
001F  5211 ;
001F  5212 ;          First IFERROR:
001F  5213 ;
001F  5214 ;          EXPECTED LAST ADDRESS OF FIRST ARRAY
001F  5215 ;          RECEIVED LAST ADDRESS
001F  5216 ;          LOOP COUNT I
001F  5217 ;          LOOP COUNT J
001F  5218 ;          LOOP COUNT K
001F  5219 ;          (INDICATES PROGRAM WAS LOST TRYING TO READ 0'S AND WRITE BACK 1'S)
001F  5220 ;
001F  5221 ;          Second IFERROR:
001F  5222 ;
```

```
001F  5223 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  5224 ;       MAIN MEMORY ADDRESS OF DATA ERROR
001F  5225 ;       LOOP COUNT I
001F  5226 ;       LOOP COUNT J
001F  5227 ;       LOOP COUNT K
001F  5228 ;       RECEIVED DATA (S/B 0'S OR 1'S)
001F  5229 ;       (INDICATES ECC IS PROBABLE NOT WORKING)
001F  5230 ;
001F  5231 ;       Third IFERROR:
001F  5232 ;
001F  5233 ;       NOT SIGNIFICANT (USED TO FORCE AN ERROR PRINTOUT)
001F  5234 ;       MAIN MEMORY ADDRESS OF DOUBLE BIT ERROR
001F  5235 ;       LOOP COUNT I
001F  5236 ;       LOOP COUNT J
001F  5237 ;       LOOP COUNT K
001F  5238 ;       (INDICATES ECC DETECTED A DOUBLE BIT ERROR AND FORCED A MACHINE CHECK)
001F  5239 ;
001F  5240 ;--
001F  5241 ;*****************************************************************************
001F  5242
001F  5243 ;-----------------------------------------------------------------------------
001F  5244 ;
001F  5245 ; Initialize code
001F  5246 ;
001F  5247 ;-----------------------------------------------------------------------------
001F  5248
001F  5249          INITIALIZE                                  ;init the cpu
0021  5250          LOADDCS         40$,,0,<^X1C>               ;setup init micro code
0028  5251          LOADDCS         65$,,<^X11>,<^X1B>          ;set-up shared micro code
002F  5252          FETCH           <^X11>                      ;execute shared micro code
0034  5253          BRSTCLK         <^X2B>,INH                  ;end shared micro code
0038  5254          FETCH           0                           ;start dcs program
003D  5255          BRSTCLK         <^X0A>,INH                  ;end dcs program
0041  5256          CMPREGMSK       WHREG,23$,,25$,,B           ;check array 1 type
004D  5257          SKIP            155$,NOERROR                ;skip if array 1 not present
0054  5258          CMPREGMSK       WHREG,28$,,18$,,B           ;check array 0 size
0060  5259          SKIP            90$,NOERROR                 ;skip if 4mb
0067  5260          CMPREGMSK       WHREG,21$,,18$,,B,NE        ;check array 0 size
0073  5261          SKIP            100$,ONERROR                ;skip if array 0 is = 1mb
007A  5262          MASK            4$,,19$,,L                  ;array 1 last address = 40000
0083  5263          SKIP            108$                        ;go test
008A  5264 ;
008A  5265 ; This section of code handles the case where array 0 is 4mb. It will also
008A  5266 ; check the size of array 1
008A  5267 ;
008A  5268 90$:     MASK            4$,,29$,,L                  ;array 1 starting addr = 400000
0093  5269          LOADDCS         80$,,<^X0C>,1               ;array 1 starting addr = 400000
009A  5270          FETCH           <^X0B>                      ;start dcs
009F  5271          BRSTCLK         <^X0F>,INH                  ;end dcs
00A3  5272          CMPREGMSK       WHREG,36$,,25$,,B           ;check array 1 size
00AF  5273          SKIP            95$,ONERROR                 ;skip if 1mb
00B6  5274          LDFIELD         32$,,L,57$,31,62,LON        ;last address = 7FFFFC
00C7  5275          LOADDCS         82$,,<^X08>,1               ;last address = 7FFFFC
00CE  5276          SKIP            102$                        ;execute changed dcs
00D5  5277 95$:     LDFIELD         34$,,L,57$,31,62,LON        ;last address = 4FFFFC
```

```
00E6  5278              LOADDCS         84$,,<^X08>,1           ;last address = 4FFFFC
00ED  5279              SKIP            102$                    ;execute changed dcs
00F4  5280 ;
00F4  5281 ; This section of code will handle the case where array 0 (first array slot) is
00F4  5282 ; a 1mb array module.  It will also check the size of array 1
00F4  5283 ;
00F4  5284 100$:        MASK            4$,,24$,,L              ;array 1 starting addr = 100000
00FD  5285              FETCH           <^X0B>                  ;start dcs program
0102  5286              BRSTCLK         <^X0F>,INH              ;end dcs program
0106  5287              CMPREGMSK       WHREG,26$,,25$,,B       ;check array 1 size
0112  5288              SKIP            104$,ONERROR            ;skip if array 1 is 256kb
0119  5289              LDFIELD         20$,,L,57$,31,62,LON    ;array 1 last address = 1FFFFC
012A  5290              LOADDCS         75$,,<^X08>,1           ;array 1 last address = 1FFFFC
0131  5291 ;
0131  5292 ; Come here to execute the altered dcs and set up last address
0131  5293 ;
0131  5294 102$:        FETCH           <^X08>                  ;start dcs program
0136  5295              BRSTCLK         <^X0A>,INH              ;end dcs program
013A  5296              SKIP            108$                    ;go test
0141  5297 ;
0141  5298 ; Come here if array 1 is a 256kb module.  This section will skip back to 102$
0141  5299 ; to execute altered dcs to establish the correct last address.
0141  5300 ;
0141  5301 104$:        LDFIELD         27$,,L,57$,31,62,LON    ;array 1 last address = 13FFFC
0152  5302              LOADDCS         77$,,<^X08>,1           ;array 1 last address = 13FFFC
0159  5303              SKIP            102$                    ;go execute changed dcs
0160  5304
0160  5305 ;-------------------------------------------------------------------------------
0160  5306 ;
0160  5307 ; Write memory memory with 1's
0160  5308 ;
0160  5309 ;-------------------------------------------------------------------------------
0160  5310
0160  5311 108$:        LDINDEX         I,2                     ;dummy index I gets 2
0167  5312              LOADDCS         50$,,0,<^X0B>           ;test u-code
016E  5313              FETCH           0                       ;load first u-inst.
0173  5314              LOOP            K,1,2                   ;set index K for two passes
017A  5315              SKIP            110$,ONEQUAL,I,K        ;skip on second pass
0181  5316              SKIP            112$                    ;first pass, goto rdm set-up
0188  5317
0188  5318 ;-------------------------------------------------------------------------------
0188  5319 ;
0188  5320 ; Read the 1's fro the previous write, check for valid data, wirte 0's, check
0188  5321 ; for valid data
0188  5322 ;
0188  5323 ;-------------------------------------------------------------------------------
0188  5324
0188  5325 110$:        FETCH           7                       ;reset va to last address
018D  5326              BRSTCLK         <^X0A>,INH              ;...
0191  5327              LOADDCS         60$,,0,<^X11>           ;test u-code
0198  5328              FETCH           0                       ;load first u-inst.
019D  5329 ;
019D  5330 ; This section is used for both the initial writing of 1's (first pass) then
019D  5331 ; for the read of 1's-write of 0's-read of 0's (second pass)
019D  5332 ;
```

```
019D  5333 112$:   LOADREG         DCSCR,9$,,B              ;insure clock is off
01A5  5334         LOADREG         CSAMR,2$,,W              ;load match address
01AD  5335         LOADREG         RDCTRL,7$,,B             ;clear master halt enable
01B5  5336         LOADREG         RDCTRL,1$,,B             ;watch for u-traps
01BD  5337         LOADREG         DCSCR,3$,,B              ;start the clock
01C5  5338         LOOP            J,1,8192                 ;watchdog timing loop
01CC  5339         CMPREGMSK       CLKDCS,7$,,8$,,B,        ;test for clock stopped
01D8  5340         SKIP            114$,ONERROR             ;skip when clock stops
01DF  5341         ENDLOOP         J                        ;end timing loop
01E2  5342         LOADREG         DCSCR,9$,,B              ;stop the clock
01EA  5343         CMPREG          WHREG,4$,,L,             ;force an error
01F3  5344         IFERROR         ,,<<MC0><MC1><MC2>>      ;program lost
01FB  5345         SKIP
0202  5346
0202  5347 ;-------------------------------------------------------------------------------
0202  5348 ;
0202  5349 ; Error evaluation code
0202  5350 ;
0202  5351 ;-------------------------------------------------------------------------------
0202  5352
0202  5353 114$:   CMPREGMSK       CSTRP,10$,,6$,,W,NE      ;test for machine check
020E  5354         SKIP            140$,ONERROR             ;handle machine check
0215  5355         CMPREGMSK       CLKDCS,11$,,12$,,B,NE    ;test for data errors
0221  5356         SKIP            130$,ONERROR             ;incorrect data
0228  5357         CMPREGMSK       CLKDCS,13$,,12$,,B,NE    ;test for data errors
0234  5358         SKIP            130$,ONERROR             ;incorrect data
023B  5359         CMPREGMSK       CLKDCS,14$,,12$,,B,NE    ;test for single bit error on 0
0247  5360         SKIP            120$,ONERROR             ;log error
024E  5361         CMPREGMSK       CLKDCS,16$,,12$,,B,NE    ;test for single bit error on 1
025A  5362         SKIP            115$,ONERROR             ;log error
0261  5363         CMPREG          WHREG,4$,,L,             ;test for last address
026A  5364         SKIP            150$,NOERROR             ;go to next test
0271  5365         CMPREGMSK       CLKDCS,17$,,12$,,B       ;force and error
027D  5366         IFERROR                                  ;i think we're lost
0282  5367         SKIP                                     ;end test due to error
0289  5368 ;
0289  5369 ; Come here if the Second read caused corrected read data to be returned
0289  5370 ;
0289  5371 115$:   FETCH           <^X11>                   ;start dcs program
028E  5372         BRSTCLK         <^X2B>                   ;end dcs program
0292  5373         ERRLOG                                   ;log the error
0294  5374         SKIP            150$,ONERROR             ;quit after 64 errors
029B  5375         FETCH           <^X0D>                   ;set up to continue testing
02A0  5376         SKIP            112$                     ;test more memory
02A7  5377 ;
02A7  5378 ; Come here if the First read caused corrected read data to be returned
02A7  5379 ;
02A7  5380 120$:   FETCH           <^X11>                   ;start dcs program
02AC  5381         BRSTCLK         <^X2B>                   ;end dcs program
02B0  5382         ERRLOG                                   ;log the error
02B2  5383         SKIP            150$,ONERROR             ;quit after 64 errors
02B9  5384         FETCH           <^X06>                   ;set up to continue testing
02BE  5385         SKIP            112$                     ;test more memory
02C5  5386 ;
02C5  5387 ; Come here if the data read is not the data written
```

```
                02C5  5388 ;
                02C5  5389 130$:    CMPREG          WHREG,15$,,L,           ;force an error
                02CE  5390          IFERROR         1,<<MEC><MDL>>,-
                02CE  5391          <<MC0><MC1><MC2><MA0><MA1><MA2>>        ;print va and data
                02DB  5392          SKIP                                    ;end test because of error
                02E2  5393 ;
                02E2  5394 ; Come here if there was a Machine check (Read Data Substitute)
                02E2  5395 ;
                02E2  5396 140$:    CMPREG          WHREG,15$,,L,           ;force an error
                02EB  5397          IFERROR         ,<<MEC><MDL>>,-
                02EB  5398          <<MC0><MC1><MC2><MA0><MA1><MA2>>        ;print va of machine check
                02F8  5399 ;-------------------------------------------------------------------------
                02F8  5400 ;
                02F8  5401 ; This is where we exit the test if index K is equal to 2.  Also we want to
                02F8  5402 ; dump the error log and print message 'FINISHED ARRAY 1'.
                02F8  5403 ;
                02F8  5404 ;-------------------------------------------------------------------------
                02F8  5405 150$:    ENDLOOP         K
                02FB  5406          DUMPLOG                                 ;dump the crd's for array 0
                02FD  5407          PRINT_S         30$                     ;print finished message
                0301  5408 155$:    SKIP                                    ;go to next test
                0308  5409
                0308  5410
                0308  5411 BEGIN_TEST_DATA
                0308
                0308          ;-----------------------------------------------------------------
                0308  5412
              28  0308  5413 1$:      .BYTE   ^X28                            ;set to watch for u-traps
            1801  0309  5414 2$:      .WORD   ^X1801                          ;match address
              11  030B  5415 3$:      .BYTE   ^X11                            ;run clock
        FFFFFFFF  030C  5416 4$:      .LONG   ^XFFFFFFFF                      ;starting address for array 1
            3FFF  0310  5417 6$:      .WORD   ^X3FFF                          ;mask for cmpregmsk pseudo op
              00  0312  5418 7$:      .BYTE   ^X00                            ;watch for clock stop
              C0  0313  5419 8$:      .BYTE   ^XC0                            ;mask for cmpregmsk pseudo op
              60  0314  5420 9$:      .BYTE   ^X60                            ;stop clock manually
            0028  0315  5421 10$:     .WORD   ^X0028                          ;machine check u-trap
              08  0317  5422 11$:     .BYTE   ^X08                            ;dcs halt on data error
              3F  0318  5423 12$:     .BYTE   ^X3F                            ;mask for cmpregmsk pseudo op
              0F  0319  5424 13$:     .BYTE   ^X0F                            ;dcs halt on data error
              07  031A  5425 14$:     .BYTE   ^X07                            ;halt single bit error on 0's
        0F0F0F0F  031B  5426 15$:     .LONG   ^X0F0F0F0F                      ;compare data to force errors
              0E  031F  5427 16$:     .BYTE   ^X0E                            ;halt single bit error on 1's
              FF  0320  5428 17$:     .BYTE   ^XFF                            ;used to force errors
              03  0321  5429 18$:     .BYTE   ^X03                            ;array 0 mask
        00040000  0322  5430 19$:     .LONG   ^X00040000                      ;array 1 starting address
        001FFFFC  0326  5431 20$:     .LONG   ^X001FFFFC                      ;array 1 ending address - 1mb
              02  032A  5432 21$:     .BYTE   ^X02                            ;array 0 256kb (CSR2 bits 0:1)
              00  032B  5433 23$:     .BYTE   ^X00                            ;slot empty compare data
        00100000  032C  5434 24$:     .LONG   ^X00100000                      ;array 1 starting address
              0C  0330  5435 25$:     .BYTE   ^X0C                            ;array 1 mask
              08  0331  5436 26$:     .BYTE   ^X08                            ;array 1 1mb (csr2 bits 3:2)
        0013FFFC  0332  5437 27$:     .LONG   ^X0013FFFC                      ;array 1 ending address - 256kb
              01  0336  5438 28$:     .BYTE   ^X01                            ;array 0 4mb
        00400000  0337  5439 29$:     .LONG   ^X00400000                      ;array 1 start addr
20 44 45 48 53 49 4E 49 46 0A 0D 00' 033B  5440 30$:     .ASCIC  <13><10>/FINISHED ARRAY 1/<13><10>
```

```
        0A 0D 31 20 59 41 52 52 41  0347
                                14  033B
                                    0350  5441
                                    0350  5442 32$:    .LONG   ^X007FFFFC                      ;message to print when finished
                           007FFFFC 0350  5442          .LONG   ^X007FFFFC                      ;array 1 last address - 4mb
                           004FFFFC 0354  5443 34$:     .LONG   ^X004FFFFC                      ;array 1 last address - 1mb
                                 04 0358  5444 36$:     .BYTE   ^X04                            ;array 1 4mb
                                    0359  5445 40$:
U 00, 7700,C800,0364,0300,0470,1801 0359  5446 ;% 00    NOP
U 01, 7700,8800,5BE4,0320,4A70,1802 0364  5450 ;% 01    VA_R[TEMP8]                             ;CSR0 address to va
U 02, 7700,C800,0364,0300,4470,1803 036F  5454 ;% 02    VA_VA+4                                 ;CSR1 address to va
U 03, 7700,4800,5BE4,0328,5C80,1804 037A  5458 ;% 03    WRITE.PHY R[TEMP10]                     ;allow single bit errors
U 04, 7700,C800,0364,0300,4470,1805 0385  5462 ;% 04    VA_VA+4                                 ;CSR2 address to va
U 05, 7700,C800,0364,0300,0400,1806 0390  5466 ;% 05    READ.PHY                                ;read CSR2
U 06, 6700,8812,5924,0300,0470,1807 039B  5470 ;% 06    RDM_WB,WB_M[MDR]                        ;WHREG gets contents of CSR2
U 07, 7700,C800,0364,1700,0470,1808 03A6  5474 ;% 07    IRDITEST                                ;clear int pending line
U 08, 7700,7800,7FFC,0001,8470,1809 03B1  5478 ;% 08    LONLIT_[0007FFFC]                       ;array 1 ending address
U 09, 7700,4800,5BE4,03D4,4A70,180A 03BC  5482 ;% 09    VA_R[LONLIT]                            ;VA gets ending address
U 0A, 7300,C800,0364,0300,0470,180B 03C7  5486 ;% 0A    NOP,STOP.CLOCK                          ;stop the cpu clock
U 0B, 7700,4800,0364,0300,0470,180C 03D2  5490 ;% 0B    NOP
U 0C, 7700,3800,7FF7,FFFF,8470,180D 03DD  5494 ;% 0C    LONLIT_[00100000]                       ;array 1 starting address
U 0D, 7700,8868,5BE4,03D4,0470,180E 03E8  5498 ;% 0D    M[TEMP8]_R[LONLIT]                      ;store in M[TEMP8]
U 0E, 7700,0848,5924,0318,0470,180F 03F3  5502 ;% 0E    R[TEMP6]_M[TEMP8]                       ;R[TEMP6] gets starting address
U 0F, 7300,C800,0364,0300,0470,1810 03FE  5506 ;% 0F    NOP,STOP.CLOCK
                                    0409  5510 50$:
U 00, 7700,C800,0364,0300,0470,1802 0409  5511 ;% 00    NOP,NEXT/1802
U 01, 6700,881B,5924,0300,0470,1802 0414  5515 ;% 01    RDM_WB,WB_M[VA]                         ;save va in rdm
U 02, 7700,C800,5BE4,0314,5C80,1803 041F  5519 ;% 02    WRITE.PHY R[TEMP5]                      ;write ones to memory
U 03, 7700,4800,0364,0300,0470,9804 042A  5523 ;% 03    NOP,CLOCK.EXT                           ;allow time for write
U 04, 7700,481B,0034,A318,0470,1800 0435  5527 ;% 04    WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
                                    0440  5531                                              ;test for end of second array
U 05, 7700,581B,C100,0302,4A70,1806 0440  5532 ;% 05    VA_VA-ZLIT0[4]                          ;decrement va register
U 06, 7500,C800,0364,0300,0470,1807 044B  5536 ;% 06    NOP,DCS.ADDR_0                          ;return to address 0
U 07, 7700,C800,0364,0300,0470,1808 0456  5540 ;% 07    NOP
                                    0461  5544 57$:
U 08, 7700,7800,7FFC,0001,8470,1809 0461  5545 ;% 08    LONLIT_[0007FFFC]                       ;array 1 ending address
U 09, 7700,4800,5BE4,03D4,4A70,180A 046C  5549 ;% 09    VA_R[LONLIT]                            ;VA gets array 1 ending address
U 0A, 7300,C800,0364,0300,0470,180B 0477  5553 ;% 0A    NOP,STOP.CLOCK                          ;stop the cpu clock
                                    0482  5557 60$:
U 00, 7700,4800,0364,0300,0470,9802 0482  5558 ;% 00    NOP,CLOCK.EXT,NEXT/1802                 ;allow time for memory
U 01, 7700,C800,0364,0300,0470,1802 048D  5562 ;% 01    NOP                                     ;clock to start up.
U 02, 6700,081B,5924,0300,0470,1803 0498  5566 ;% 02    RDM_WB,WB_M[VA]                         ;save va in rdm
U 03, 7700,4800,0364,0300,0400,1804 04A3  5570 ;% 03    READ.PHY                                ;read pattern back
U 04, 7700,4852,5924,0300,0470,1805 04AE  5574 ;% 04    R[TEMP0]_M[MDR]                         ;save data in RTEMP0
U 05, 7700,C800,0364,AF00,0470,1801 04B9  5578 ;% 05    BUT/INT-TIMSERV,NEXT/1801               ;test for corrected data
U 06, 7700,0800,0034,A714,0470,1800 04C4  5582 ;% 06    WB_M[TEMP0].XOR.R[TEMP5],BUT/WX.NE.0,NEXT/1800
                                    04CF  5586                                              ;test for good data
U 07, 7700,4800,0364,0300,0470,9808 04CF  5587 ;% 07    NOP,CLOCK.EXT                           ;allow time for memory clock
U 08, 7700,C800,0364,0300,0470,9809 04DA  5591 ;% 08    NOP,CLOCK.EXT                           ;to start up after single bit
U 09, 7700,8800,5BE4,0310,5C80,180A 04E5  5595 ;% 09    WRITE.PHY R[TEMP4]                      ;write zero's to memory
U 0A, 7700,4800,0364,0300,0400,180B 04F0  5599 ;% 0A    READ.PHY                                ;read pattern back
U 0B, 7700,4852,5924,0300,0470,180C 04FB  5603 ;% 0B    R[TEMP0]_M[MDR]                         ;save data in RTEMP0
U 0C, 7700,C800,0364,AF00,0470,1801 0506  5607 ;% 0C    BUT/INT-TIMSERV,NEXT/1801               ;test for corrected data
U 0D, 7700,4800,0034,A710,0470,1800 0511  5611 ;% 0D    WB_M[TEMP0].XOR.R[TEMP4],BUT/WX.NE.0,NEXT/1800
                                    051C  5615                                              ;test for good data
U 0E, 7700,481B,0034,A318,0470,1800 051C  5616 ;% 0E    WB_M[VA].XOR.R[TEMP6],BUT/WX.EQ.0,NEXT/1800
```

```
                                0527  5620                                                    ;test for end of second array
U 0F. 7700,D81B,C100,0302,4A70,1810  0527  5621 ;% 0F   VA_VA-ZLIT0[4]                        ;decrement va
U 10. 7500,4800,0364,0300,0470,1811  0532  5625 ;% 10   NOP,DCS.ADDR_0                        ;return to dcs address 0
                                053D  5629 ;
                                053D  5630 ; This dcs code here is shared code.  It will be used by both the init code and
                                053D  5631 ; the error code when a single bit error is detected and needs to be logged
                                053D  5632 ;
                                053D  5633 65$:
U 11. 7700,4800,0364,0300,0470,1812  053D  5634 ;% 11   NOP
U 12. 7700,885B,5924,031C,0470,1813  0548  5638 ;% 12   R[TEMP7]_M[VA]                         ;save va for return
U 13. 7700,0800,5BE4,0304,6870,1814  0553  5642 ;% 13   MEMSCAR_R[TEMP1]                      ;disable cmi
U 14. 7700,4800,5BE4,030C,6070,1815  055E  5646 ;% 14   MEMSCR_R[TEMP3]                       ;...
U 15. 7700,8800,5BE4,0308,6870,1816  0569  5650 ;% 15   MEMSCAR_R[TEMP2]                      ;turn on cache
U 16. 7700,C800,5BE4,03D8,6070,1817  0574  5654 ;% 16   MEMSCR_R[ZERO]                        ;...
U 17. 7700,4800,5BE4,03D8,4A70,1818  057F  5658 ;% 17   VA_R[ZERO]                           ;zero the va
U 18. 7700,4800,5BE4,03D8,5C80,1819  058A  5662 ;% 18   WRITE.PHY R[ZERO]                     ;validate cache address 0
U 19. 7700,C800,5BE4,03D8,5470,181A  0595  5666 ;% 19   WDR_UNROT(R[ZERO])                   ;...
U 1A. 7700,C800,0364,0300,4470,181B  05A0  5670 ;% 1A   VA_VA+4                              ;increment va by 4
U 1B. 7700,4800,5BE4,03D8,5C80,181C  05AB  5674 ;% 1B   WRITE.PHY R[ZERO]                     ;validate cache address 4
U 1C. 7700,4800,5BE4,03D8,5470,181D  05B6  5678 ;% 1C   WDR_UNROT(R[ZERO])
U 1D. 7700,C800,5BE4,03D8,4870,181E  05C1  5682 ;% 1D   PC_R[ZERO]                           ;cause prefetch
U 1E. 7700,0800,5BE4,032C,2470,181F  05CC  5686 ;% 1E   WCTRL/LOADCRAR,WB_R[TEMP11]          ;disable ^P interrupts
U 1F. 7700,C800,5BE4,0330,2C70,1820  05D7  5690 ;% 1F   WCTRL/CONWRITE,WB_R[TEMP12]         ;...
U 20. 7700,0800,5BE4,0304,6870,1821  05E2  5694 ;% 20   MEMSCAR_R[TEMP1]                      ;turn on cmi
U 21. 7700,C800,5BE4,03D8,6070,1822  05ED  5698 ;% 21   MEMSCR_R[ZERO]                       ;...
U 22. 7700,8800,5BE4,0308,6870,1823  05F8  5702 ;% 22   MEMSCAR_R[TEMP2]                      ;disable cache
U 23. 7700,C800,5BE4,030C,6070,1824  0603  5706 ;% 23   MEMSCR_R[TEMP3]                       ;...
U 24. 7700,8800,5BE4,0320,4A70,1825  060E  5710 ;% 24   VA_R[TEMP8]                          ;address of CSR0
U 25. 7700,4800,0364,0300,0400,1826  0619  5714 ;% 25   READ.PHY                             ;read CSR0
U 26. 3700,0812,5924,0300,0470,1827  0624  5718 ;% 26   WB_M[MDR],RDM_CMI                    ;MHREG gets CSR0
U 27. 7700,C800,5BE4,0324,5C80,1828  062F  5722 ;% 27   WRITE.PHY R[TEMP9]                   ;clear CSR0
U 28. 7700,C800,0364,7B00,0470,1829  063A  5726 ;% 28   BUT/UVCTR                           ;clear any interrupts pending
U 29. 7700,C800,0364,1700,0470,182A  0645  5730 ;% 29   IRD1TEST                            ;free interrupt pending logic
U 2A. 7700,0800,5BE4,031C,4A70,182B  0650  5734 ;% 2A   VA_R[TEMP7]                         ;reset the va
U 2B. 7300,C800,0364,0300,0470,182C  065B  5738 ;% 2B   NOP,STOP.CLOCK                      ;stop the cpu clock
                                0666  5742 75$:
U 08. 7700,7800,7FF0,0001,8470,1809  0666  5743 ;% 08   LONLIT_[001FFFFC]                    ;array 1 ending address - 1mb
                                0671  5747 77$:
U 08. 7700,7800,7FF6,0001,8470,1809  0671  5748 ;% 08   LONLIT_[0013FFFC]                    ;array 1 ending address - 256kb
                                067C  5752 80$:
U 0C. 7700,3800,7FDF,FFFF,8470,180D  067C  5753 ;% 0C   LONLIT_[00400000]                    ;array 1 starting address
                                0687  5757 82$:
U 08. 7700,7800,7FC0,0001,8470,1809  0687  5758 ;% 08   LONLIT_[007FFFFC]                    ;array 1 last address - 4mb
                                0692  5762 84$:
U 08. 7700,7800,7FD8,0001,8470,1809  0692  5763 ;% 08   LONLIT_[004FFFFC]                    ;array 1 last address - 1mb
                                069D  5767
                                069D  5768 BEGIN_CPU_DATA
                                0002  5769
                                0002  5770 RTEMP_DATA
                                0001  5771
                    00000000    0001  5772          .LONG   ^X00000000                    ;data pattern storage
                    0E000000    0005  5773          .LONG   ^X0E000000                    ;reference cache only address
                    06000000    0009  5774          .LONG   ^X06000000                    ;cache control register address
                    01000000    000D  5775          .LONG   ^X01000000                    ;disable cache
                    00000000    0011  5776          .LONG   ^X00000000                    ;test pattern 1
```

```
                    FFFFFFFF  0015  5777          .LONG   ^XFFFFFFFF                      ;test pattern 2
                    00040000  0019  5778          .LONG   ^X00040000                      ;last address for 256kb array 0
                    00000000  001D  5779          .LONG   ^X00000000                      ;storage for va register
                    00F20000  0021  5780          .LONG   ^X00F20000                      ;address of CSR0
                    E0000000  0025  5781          .LONG   ^XE0000000                      ;clears CSR0
                    10000000  0029  5782          .LONG   ^X10000000                      ;allow single bit error report
                    00C00000  002D  5783          .LONG   ^X00C00000                      ;disable ^P interrupts
                    00400000  0031  5784          .LONG   ^X00400000                      ;...
                              0035  5785
                              0035  5786 END_CPU_DATA
                              069D  5787
                              069D  5788
                              069D  5789 END_TEST_DATA
                              069D
                              069D        ;-----------------------------------------------------------------------
                              069D  5790 ENDTEST
%MACRO-W-GENWRN, Generated WARNING:  TEST MAY BE TOO LARGE WITH DEBUG MONITOR;
```

```
                        .SBTTL  "TEST   08A     Test Main Memory Byte Mask
001F   5792 ;*******************************************************************************
001F   5793 ;++
001F   5794 ;
001F   5795 ; FUNCTIONAL DESCRIPTION:
001F   5796 ;
001F   5797 ;       THIS TEST WILL VERIFY THAT THE BYTE MASK FUNCTION OF MAIN MEMORY IS
001F   5798 ;       OPERATIONAL.
001F   5799 ;
001F   5800 ;
001F   5801 ; TEST ALGORITHM:
001F   5802 ;
001F   5803 ;       1.  CREATE A TEST LOOP OF 8
001F   5804 ;       2.  LET THE CPU CLOCK FREE RUN
001F   5805 ;       3.  CLEAR THE CMICTL REGISTER
001F   5806 ;       4.  WRITE ALL ZEROS FROM THE RDM TO ADDRESS 0
001F   5807 ;       5.  SELECT A BYTE MASK AND LOAD INTO THE MA REGISTER
001F   5808 ;       6.  WRITE ALL ONES FROM THE RDM TO ADDRESS 0
001F   5809 ;       7.  READ ADDRESS ZERO BACK TO THE RDM
001F   5810 ;       8.  TEST THE MH REGISTER FOR THE CORRECT RESULTS
001F   5811 ;       9.  IF NO ERROR GO TO STEP 3 FOR THE REMAINING TEST LOOPS
001F   5812 ;
001F   5813 ;
001F   5814 ; TEST PATTERNS:
001F   5815 ;
001F   5816 ;       ITERATION       BYTE MASK       RESULTS
001F   5817 ;           1           18000000        000000FF
001F   5818 ;           2           28000000        0000FF00
001F   5819 ;           3           48000000        00FF0000
001F   5820 ;           4           88000000        FF000000
001F   5821 ;           5           E8000000        FFFFFF00
001F   5822 ;           6           D8000000        FFFF00FF
001F   5823 ;           7           B8000000        FF00FFFF
001F   5824 ;           8           78000000        00FFFFFF
001F   5825 ;
001F   5826 ;
001F   5827 ; LOGIC DESCRIPTION:
001F   5828 ;
001F   5829 ;       THIS TEST WRITES VARIOUS BYTE MASK PATTERNS FROM THE RDM TO THE CPU
001F   5830 ;       MAIN MEMORY.  ANY FAILURES SHOULD BE CONFINED TO THE MDL GATE ARRAY
001F   5831 ;       AND/OR THE CMC MODULE.
001F   5832 ;
001F   5833 ;
001F   5834 ; ASSUMPTIONS:
001F   5835 ;
001F   5836 ;       THIS TEST ASSUMES THE CMI BUS IS NOT HUNG BY ANOTHER DEVICE
001F   5837 ;
001F   5838 ;
001F   5839 ; ERROR DESCRIPTION:
001F   5840 ;
001F   5841 ;       EXPECTED MEMORY DATA
001F   5842 ;       RECEIVED MEMORY DATA
001F   5843 ;       LOOP COUNT I
001F   5844 ;
001F   5845 ;--
```

```
                     001F  5846 ;*****************************************************************************
                     001F  5847
                     001F  5848
                     001F  5849
                     001F  5850          INITIALIZE                                  ;INIT THE CPU
                     0021  5851          LOOP            I,1,8                       ;TEST LOOP OF 8
                     0028  5852          ERRLOOP                                     ;ERROR LOOP POINT
                     002A  5853          FETCH           0                           ;START DCS PROGRAM
                     002F  5854          LOADREG         DCSCR,1$,,B                 ;START THE CPU CLOCK
                     0037  5855          LOADREG         CMICTL,2$,,B                ;CLEAR THE CMICTL REGISTER
                     003F  5856          LOADREG         MAREG,3$,,L                 ;WRITE COMMAND TO ADDRESS ZERO
                     0047  5857          LOADREG         MDREG,4$,,L                 ;DATA FOR WRITE
                     004F  5858          LOADREG         CMICTL,5$,,B                ;PERFORM WRITE
                     0057  5859          LOADREG         MAREG,6$,I,L                ;LOAD BYTE MASK PATTERN
                     005F  5860          LOADREG         MDREG,7$,,L                 ;ALL ONES DATA
                     0067  5861          LOADREG         CMICTL,5$,,B                ;PERFORM WRITE
                     006F  5862          LOADREG         MAREG,8$,,L                 ;READ COMMAND TO ADDRESS ZERO
                     0077  5863          LOADREG         CMICTL,9$,,B                ;READ RESULTS BACK
                     007F  5864          LOADREG         DCSCR,10$,,B                ;STOP THE CLOCK
                     0087  5865          CMPREG          MHREG,11$,I,L               ;TEST RESULTS
                     0090  5866          IFERROR         ,<<MDL>>,<<MC0><MC1><MC2>> ;INCORRECT BYTE MASK
                     0099  5867          ENDLOOP         I                           ;END TEST LOOP
                     009C  5868          SKIP                                        ;GO TO NEXT TEST
                     00A3  5869
                     00A3  5870
                     00A3  5871 BEGIN_TEST_DATA
                     00A3
                     00A3               ;-------------------------------------------------------------------
                     00A3  5872
                     00A3  5873
                  00 00A3  5874 1$:      .BYTE   ^X00                                ;RUN CPU CLOCK
                  01 00A4  5875 2$:      .BYTE   ^X01                                ;CLEAR CMICTL REGISTER
          F8000000 00A5  5876 3$:      .LONG   ^XF8000000                          ;WRITE TO ADDRESS ZERO
          00000000 00A9  5877 4$:      .LONG   ^X00000000                          ;WRITE DATA
                28 00AD  5878 5$:      .BYTE   ^X28                                ;PERFORM WRITE
          18000000 00AE  5879 6$:      .LONG   ^X18000000                          ;BYTE MASK TABLE
          28000000 00B2  5880          .LONG   ^X28000000
          48000000 00B6  5881          .LONG   ^X48000000
          88000000 00BA  5882          .LONG   ^X88000000
          E8000000 00BE  5883          .LONG   ^XE8000000
          D8000000 00C2  5884          .LONG   ^XD8000000
          B8000000 00C6  5885          .LONG   ^XB8000000
          78000000 00CA  5886          .LONG   ^X78000000
          FFFFFFFF 00CE  5887 7$:      .LONG   ^XFFFFFFFF                          ;WRITE DATA
          00000000 00D2  5888 8$:      .LONG   ^X00000000                          ;READ COMMAND TO ADDRESS ZERO
                48 00D6  5889 9$:      .BYTE   ^X48                                ;PERFORM READ
                60 00D7  5890 10$:     .BYTE   ^X60                                ;STOP THE CPU CLOCK
          000000FF 00D8  5891 11$:     .LONG   ^X000000FF                          ;RESULTS TABLE
          0000FF00 00DC  5892          .LONG   ^X0000FF00
          00FF0000 00E0  5893          .LONG   ^X00FF0000
          FF000000 00E4  5894          .LONG   ^XFF000000
          FFFFFF00 00E8  5895          .LONG   ^XFFFFFF00
          FFFF00FF 00EC  5896          .LONG   ^XFFFF00FF
          FF00FFFF 00F0  5897          .LONG   ^XFF00FFFF
          00FFFFFF 00F4  5898          .LONG   ^X00FFFFFF
```

```
                                00F8  5899
                                00F8  5900
                                00F8  5901 BEGIN_CPU_DATA
                                0002  5902
                                0002  5903 DCS_DATA
                                0001  5904
U 00, 7700,C800,0364,0300,0470,1801  0001  5905 ;% 00   NOP                                    ;CODE TO SPIN CLOCK ON
U 01, 7500,C800,0364,0300,0470,1802  000C  5909 ;% 01   NOP,DCS.ADDR_0
                                0017  5913
                                0017  5914 END_CPU_DATA
                                00F8  5915
                                00F8  5916
                                00F8  5917 END_TEST_DATA
                                00F8
                                00F8       ;-----------------------------------------------------------------------
                                00F8  5918 ENDTEST
```

```
                        .SBTTL  "TEST   08B     Test CPU Byte Mask
                5920 ;*******************************************************************************
                5921 ;++
                5922 ;
                5923 ; FUNCTIONAL DESCRIPTION:
                5924 ;
                5925 ;       THIS TEST CHECKS THE BYTE MASK FUNCTION OF CPU.
                5926 ;
                5927 ;
                5928 ; TEST ALGORITHM:
                5929 ;
                5930 ;       SUBTEST #1
                5931 ;       1.  CREATE A TEST LOOP OF 4
                5932 ;       2.  SELECT A TEST ADDRESS AND LOAD INTO THE WD REGISTER
                5933 ;       3.  EXECUTE DCS PROGRAM
                5934 ;               a.  ENABLE CMI
                5935 ;               b.  DISABLE CACHE
                5936 ;               c.  MOVE TEST ADDRESS TO VA REGISTER
                5937 ;               d.  WRITE ZERO TO THE TEST ADDRESS
                5938 ;               e.  WRITE ONES TO THE TEST ADDRESS WITH SIZE EQUAL BYTE
                5939 ;               f.  READ THE RESULTS BACK TO THE RDM WH REGISTER
                5940 ;       4.  TEST THE WH REGISTER FOR THE CORRECT TEST PATTERN
                5941 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST ADDRESSES
                5942 ;
                5943 ;       SUBTEST #2
                5944 ;       1.  SAME AS SUBTEST #1 EXCEPT SIZE EQUALS WORD
                5945 ;
                5946 ;
                5947 ; TEST PATTERNS:
                5948 ;
                5949 ;       ITERATION       TEST ADDRESS    RESULTS SUB#1   RESULTS SUB#2
                5950 ;           1             00000000        000000FF        0000FFFF
                5951 ;           2             00000001        0000FF00        00FFFF00
                5952 ;           3             00000002        00FF0000        FFFF0000
                5953 ;           4             00000003        FF000000            N/A
                5954 ;
                5955 ;
                5956 ; LOGIC DESCRIPTION:
                5957 ;
                5958 ;       THIS TEST VERIFIES THAT THE CML GATE ARRAY CAN GENERATE THE CORRECT
                5959 ;       BYTE MASK BASED ON THE ADDRESS AND SIZE.  SINCE THE BYTE MASK OF THE
                5960 ;       MEMORY CONTROLLER HAS BEEN TESTED FROM THE RDM, ANY FAILURES SHOULD
                5961 ;       BE CONFINED TO THE CML GATE ARRAY AND/OR THE MIC MODULE.
                5962 ;
                5963 ;
                5964 ; ASSUMPTIONS:
                5965 ;
                5966 ;       THIS TEST ASSUMES THAT THE PREVIOUS CMI/MEMORY CONTROLLER TESTS HAVE
                5967 ;       RUN SUCCESSFULLY.
                5968 ;
                5969 ;
                5970 ; ERROR DESCRIPTION:
                5971 ;
                5972 ;       EXPECTED MAIN MEMORY CONTENTS
                5973 ;       RECEIVED MAIN MEMORY CONTENTS
```

```
                        ;       LOOP COUNT I
                        ;
                        ;--
                        ;*****************************************************************************
                        ;/////////////////////////////////////////////////////////////////////////////
                        ;+
                        ;SUBTEST #1
                        ;
                        ;       TEST MASK WITH SIZE EQUAL BYTE
                        ;-
0017  5984              SUBTEST                                         ;SUBTEST #1
001A
001A                    ;/////////////////////////////////////////////////////////////////////////////
001A  5985              INITIALIZE                                      ;INIT THE CPU
001C  5986              EXPUTRAP                                        ;POSSIBLE FAILURE MODE
001E  5987              LOADDCS         3$,,7,1                         ;MODIFY U-CODE FOR SUBTEST
0025  5988              LOOP            I,1,4                           ;TEST LOOP OF 4
002C  5989              LOADREG         WDREG,1$,I,L                    ;TEST ADDRESS TO WD REGISTER
0034  5990              ERRLOOP                                         ;ERROR LOOP POINT
0036  5991              FETCH           0                               ;START DCS PROGRAM
003B  5992              BRSTCLK         <^X0A>                          ;END DCS PROGRAM
003F  5993              CMPREG          WHREG,2$,I,L                    ;TEST FOR CORRECT PATTERN
0048  5994              IFERROR         ,<<CML>>                        ;INCORRECT BYTE MASK
004E  5995              ENDLOOP         I                               ;END TEST LOOP
0051  5996
0051  5997
0051  5998
0051  5999  ;/////////////////////////////////////////////////////////////////////////////
0051  6000  ;+
0051  6001  ;SUBTEST #2
0051  6002  ;
0051  6003  ;       TEST MASK WITH SIZE EQUAL WORD
0051  6004  ;-
0051  6005          SUBTEST                                             ;SUBTEST #2
0054
0054            ;/////////////////////////////////////////////////////////////////////////////
0054  6006          INITIALIZE                                          ;INIT THE CPU
0056  6007          EXPUTRAP                                            ;POSSIBLE FAILURE MODE
0058  6008          LOADDCS         4$,,7,1                             ;MODIFY U-CODE FOR SUBTEST
005F  6009          LOOP            I,1,3                               ;TEST LOOP OF 3
0066  6010          LOADREG         WDREG,1$,I,L                        ;LOAD TEST ADDRESS TO WD REG
006E  6011          ERRLOOP                                             ;ERROR LOOP POINT
0070  6012          FETCH           0                                   ;START DCS PROGRAM
0075  6013          BRSTCLK         <^X0A>                              ;END DCS PROGRAM
0079  6014          CMPREG          WHREG,5$,I,L,                       ;TEST FOR CORRECT DATA
0082  6015          IFERROR         ,<<CML>>,                           ;INCORRECT MASK
0088  6016          ENDLOOP         I                                   ;END TEST LOOP
008B  6017          SKIP                                                ;GO TO NEXT TEST
0092  6018
0092  6019
0092  6020
0092  6021  BEGIN_TEST_DATA
0092
0092            ;-------------------------------------------------------------------------
0092  6022
```

```
                          00000000  0092  6023 1$:     .LONG   ^X00000000                      ;TEST ADDRESSES FOR VA REGISTER
                          00000001  0096  6024         .LONG   ^X00000001
                          00000002  009A  6025         .LONG   ^X00000002
                          00000003  009E  6026         .LONG   ^X00000003
                          000000FF  00A2  6027 2$:     .LONG   ^X000000FF                      ;RESULTS TABLE SUBTEST #1
                          0000FF00  00A6  6028         .LONG   ^X0000FF00
                          00FF0000  00AA  6029         .LONG   ^X00FF0000
                          FF000000  00AE  6030         .LONG   ^XFF000000
                                    00B2  6031 3$:
U 07. 7701,8800,5BE4,0010,5D80,1808 00B2  6032 ;% 07   WRITE,SIZE[BYTE],WB_R[TEMP4]           ;U-CODE SUBTEST #1
                                    00BD  6036 4$:
U 07. 7701,C800,5BE4,0110,5D80,1808 00BD  6037 ;% 07   WRITE,SIZE[WORD],WB_R[TEMP4]           ;U-CODE SUBTEST #2
                          0000FFFF  00C8  6041 5$:     .LONG   ^X0000FFFF                      ;RESULTS TABLE SUBTEST #2
                          00FFFF00  00CC  6042         .LONG   ^X00FFFF00
                          FFFF0000  00D0  6043         .LONG   ^XFFFF0000
                                    00D4  6044
                                    00D4  6045
                                    00D4  6046
                                    00D4  6047 BEGIN_CPU_DATA
                                    0002  6048
                                    0002  6049 DCS_DATA
                                    0001  6050
U 00. 7700,C800,0364,0300,0470,1801 0001  6051 ;% 00   NOP
U 01. 7700,C800,5BE4,0300,6870,1802 000C  6055 ;% 01   MEMSCAR_R[TEMP0]                       ;TURN CMI ON
U 02. 7700,8800,5BE4,0304,6070,1803 0017  6059 ;% 02   MEMSCR_R[TEMP1]
U 03. 7700,8800,5BE4,0308,6870,1804 0022  6063 ;% 03   MEMSCAR_R[TEMP2]                       ;TURN CACHE OFF
U 04. 7700,C800,5BE4,030C,6070,1805 002D  6067 ;% 04   MEMSCR_R[TEMP3]
U 05. 5700,4800,0364,0300,4A70,1806 0038  6071 ;% 05   VA_RDM                                ;TEST ADDRESS TO VA REGISTER
U 06. 7700,4800,5BE4,03D8,5C80,1807 0043  6075 ;% 06   WRITE.PHY R[ZERO]                      ;ZERO THE TEST ADDRESS
U 07. 7701,8800,5BE4,0010,5D80,1808 004E  6079 ;% 07   WRITE,SIZE[BYTE],WB_R[TEMP4]           ;PERFORM BYTE WRITE
U 08. 7700,C800,0364,0300,0510,1809 0059  6083 ;% 08   READ.LONG                             ;READ BACK RESULTS
U 09. 6700,0812,5924,0300,0470,180A 0064  6087 ;% 09   RDM_WB,WB_M[MDR]                      ;PUT RESULTS IN WH REGISTER
U 0A. 7300,C800,0364,0300,0470,180B 006F  6091 ;% 0A   NOP,STOP.CLOCK                        ;STOP THE CPU CLOCK
                                    007A  6095
                                    007A  6096 RTEMP_DATA
                                    0001  6097
                          0E000000  0001  6098         .LONG   ^X0E000000                      ;REFERENCE CACHE ONLY ADDRESS
                          00000000  0005  6099         .LONG   ^X00000000                      ;TURN CMI ON
                          06000000  0009  6100         .LONG   ^X06000000                      ;CACHE CONTROL REGISTER ADDRESS
                          01000000  000D  6101         .LONG   ^X01000000                      ;TURN CACHE OFF
                          FFFFFFFF  0011  6102         .LONG   ^XFFFFFFFF                      ;WRITE DATA
                                    0015  6103
                                    0015  6104 END_CPU_DATA
                                    00D4  6105
                                    00D4  6106
                                    00D4  6107 END_TEST_DATA
                                    00D4
                                    00D4          ;-----------------------------------------------------------------------------
                                    00D4  6108 ENDTEST
```

```
0031          .SBTTL  "TEST   08C     Test Byte Mask with Write Second Micro Order
0031  6110 ;*******************************************************************************
0031  6111 ;++
0031  6112 ;
0031  6113 ; FUNCTIONAL DESCRIPTION:
0031  6114 ;
0031  6115 ;       THIS TEST CHECKS THE OPERATION OF THE BYTE MASK WHILE PERFORMING
0031  6116 ;       A WRITE SECOND MICRO ORDER.
0031  6117 ;
0031  6118 ;
0031  6119 ; TEST ALGORITHM:
0031  6120 ;
0031  6121 ;       SUBTEST #1
0031  6122 ;       1.  CREATE A TEST LOOP OF 4
0031  6123 ;       2.  LOAD A TEST ADDRESS INTO THE WD REGISTER
0031  6124 ;       3.  EXECUTE DCS PROGRAM
0031  6125 ;               a.  ENABLE CMI
0031  6126 ;               b.  DISABLE CACHE
0031  6127 ;               c.  LOAD VA REGISTER WITH ZERO
0031  6128 ;               d.  WRITE ZERO'S TO ADDRESS ZERO
0031  6129 ;               e.  PERFORM ANOTHER WRITE TO ADDRESS ZERO TO SET SIZE LATCH
0031  6130 ;                   TO BYTE.
0031  6131 ;               f.  LOAD THE TEST ADDRESS INTO THE VA REGISTER
0031  6132 ;               g.  PERFORM A WRITE SECOND WITH SIZE EQUAL LONG
0031  6133 ;               h.  READ RESULTS BACK TO THE RDM WH REGISTER
0031  6134 ;       4.  TEST THE WH REGISTER FOR THE CORRECT RESULTS
0031  6135 ;       5.  IF NO ERROR GO TO STEP 2 FOR THE REMAINING TEST ADDRESSES
0031  6136 ;
0031  6137 ;       SUBTEST #2
0031  6138 ;       1.  SAME AS SUBTEST #1 EXCEPT SIZE LATCH IS SET TO WORD
0031  6139 ;
0031  6140 ;       SUBTEST #3
0031  6141 ;       1.  SAME AS SUBTEST #1 EXCEPT SIZE LATCH IS SET TO LONG
0031  6142 ;
0031  6143 ;
0031  6144 ; TEST PATTERNS:
0031  6145 ;
0031  6146 ;       ITERATION    TEST ADDRESS    RESULTS 1    RESULTS 2    RESULTS 3
0031  6147 ;           1          00000000      000000FF     000000FF     000000FF
0031  6148 ;           2          00000001      000000FF     000000FF     000000FF
0031  6149 ;           3          00000002      000000FF     000000FF     0000FFFF
0031  6150 ;           4          00000003      000000FF     000000FF     00FFFFFF
0031  6151 ;
0031  6152 ;
0031  6153 ; LOGIC DESCRIPTION:
0031  6154 ;
0031  6155 ;       THIS TEST ATTEMPTS TO CHECK SOME SPECIAL GATING WITHIN THE CML GATE
0031  6156 ;       ARRAY THAT IS ONLY USED DURING A WRITE SECOND MICRO ORDER.  ANY
0031  6157 ;       FAILURES OF THE WRITE SIZE LATCH (FLIP-FLOP INSIDE CML) OR WRITE
0031  6158 ;       SECOND GATING WILL BE DETECTED BY THIS TEST.
0031  6159 ;
0031  6160 ;
0031  6161 ; ASSUMPTIONS:
0031  6162 ;
0031  6163 ;       THIS TEST ASSUMES THE CMI IS OPERATIONAL.
```

```
0031  6164 ;
0031  6165 ;
0031  6166 ; ERROR DESCRIPTION:
0031  6167 ;
0031  6168 ;       EXPECTED MAIN MEMORY DATA
0031  6169 ;       RECEIVED MAIN MEMORY DATA
0031  6170 ;       LOOP COUNT I
0031  6171 ;
0031  6172 ;--
0031  6173 ;*******************************************************************************
0031  6174 ;///////////////////////////////////////////////////////////////////////////////
0031  6175 ;+
0031  6176 ;SUBTEST #1
0031  6177 ;
0031  6178 ;       WRITE SECOND BYTE
0031  6179 ;-
0031  6180         SUBTEST                                         ;SUBTEST #1
0034
0034        ;////////////////////////////////////////////////////////////////////////
0034  6181         INITIALIZE                                      ;INIT THE CPU
0036  6182         LOADDCS         3$,,7,1                         ;MODIFY U-CODE FOR SUBTEST
003D  6183         LOOP            I,1,4                           ;TEST LOOP OF 4
0044  6184         LOADREG         WDREG,1$,I,L                    ;LOAD TEST ADDRESS INTO WD REG
004C  6185         ERRLOOP                                         ;ERROR LOOP POINT
004E  6186         FETCH           0                               ;START DCS PROGRAM
0053  6187         BRSTCLK         <^X0C>                          ;END DCS PROGRAM
0057  6188         CMPREG          WHREG,2$,,L,                    ;TEST FOR CORRECT DATA
0060  6189         IFERROR         ,<<CML>>,                       ;INCORRECT BYTE MASK
0066  6190         ENDLOOP         I                               ;END TEST LOOP
0069  6191
0069  6192
0069  6193
0069  6194 ;///////////////////////////////////////////////////////////////////////////////
0069  6195 ;+
0069  6196 ;SUBTEST #2
0069  6197 ;
0069  6198 ;       WRITE SECOND WORD
0069  6199 ;-
0069  6200         SUBTEST                                         ;SUBTEST #2
006C
006C        ;////////////////////////////////////////////////////////////////////////
006C  6201         INITIALIZE                                      ;INIT THE CPU
006E  6202         LOADDCS         4$,,7,1                         ;MODIFY U-CODE FOR SUBTEST
0075  6203         LOOP            I,1,4                           ;TEST LOOP OF 4
007C  6204         LOADREG         WDREG,1$,I,L                    ;LOAD TEST ADDRESS INTO WD REG
0084  6205         ERRLOOP                                         ;ERROR LOOP POINT
0086  6206         FETCH           0                               ;START DCS PROGRAM
008B  6207         BRSTCLK         <^X0C>                          ;END DCS PROGRAM
008F  6208         CMPREG          WHREG,6$,,L,                    ;TEST FOR CORRECT DATA
0098  6209         IFERROR         ,<<CML>>,                       ;INCORRECT BYTE MASK
009E  6210         ENDLOOP         I                               ;END TEST LOOP
00A1  6211
00A1  6212
00A1  6213 ;///////////////////////////////////////////////////////////////////////////////
00A1  6214 ;+
```

```
                        00A1  6215 ;SUBTEST #3
                        00A1  6216 ;
                        00A1  6217 ;       WRITE SECOND LONG
                        00A1  6218 ;-
                        00A1  6219         SUBTEST                                          ;SUBTEST #3
                        00A4
                        00A4               ;////////////////////////////////////////////////////////////////////////////
                        00A4  6220         INITIALIZE                                       ;INIT THE CPU
                        00A6  6221         LOADDCS          5$,,7,1                         ;MODIFY U-CODE FOR SUBTEST
                        00AD  6222         LOOP             I,1,4                           ;TEST LOOP OF 4
                        00B4  6223         LOADREG          WDREG,1$,I,L                    ;LOAD TEST ADDRESS INTO WD REG
                        00BC  6224         ERRLOOP                                          ;ERROR LOOP POINT
                        00BE  6225         FETCH            0                               ;START DCS PROGRAM
                        00C3  6226         BRSTCLK          <^X0C>                          ;END DCS PROGRAM
                        00C7  6227         CMPREG           WHREG,7$,I,L,                   ;TEST FOR CORRECT DATA
                        00D0  6228         IFERROR          ,<<CML>>,                       ;INCORRECT BYTE MASK
                        00D6  6229         ENDLOOP          I                               ;END TEST LOOP
                        00D9  6230         SKIP                                             ;GO TO NEXT TEST
                        00E0  6231
                        00E0  6232
                        00E0  6233
                        00E0  6234 BEGIN_TEST_DATA
                        00E0               ;-------------------------------------------------------------------------
                        00E0  6235
                  00000000  00E0  6236 1$:      .LONG    ^X00000000                       ;TEST ADDRESS TABLE
                  00000001  00E4  6237         .LONG    ^X00000001
                  00000002  00E8  6238         .LONG    ^X00000002
                  00000003  00EC  6239         .LONG    ^X00000003
                  000000FF  00F0  6240 2$:      .LONG    ^X000000FF                       ;RESULTS SUBTEST #1
                            00F4  6241 3$:
U 07, 7701,C800,5BE4,00D8,5D80,1808 00F4  6242 ;% 07    WRITE,SIZE[BYTE],WB_R[ZERO]      ;U-CODE SUBTEST #1
                            00FF  6246 4$:
U 07, 7701,8800,5BE4,01D8,5D80,1808 00FF  6247 ;% 07    WRITE,SIZE[WORD],WB_R[ZERO]      ;U-CODE SUBTEST #2
                            010A  6251 5$:
U 07, 7701,8800,5BE4,02D8,5D80,1808 010A  6252 ;% 07    WRITE,SIZE[LONG],WB_R[ZERO]      ;U-CODE SUBTEST #3
                  000000FF  0115  6256 6$:      .LONG    ^X000000FF                       ;RESULTS SUBTEST #2
                  000000FF  0119  6257 7$:      .LONG    ^X000000FF                       ;RESULTS SUBTEST #3
                  000000FF  011D  6258         .LONG    ^X000000FF
                  0000FFFF  0121  6259         .LONG    ^X0000FFFF
                  00FFFFFF  0125  6260         .LONG    ^X00FFFFFF
                            0129  6261
                            0129  6262
                            0129  6263
                            0129  6264 BEGIN_CPU_DATA
                            0002  6265
                            0002  6266 DCS_DATA
                            0001  6267
U 00, 7700,C800,0364,0300,0470,1801 0001  6268 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  6272 ;% 01    MEMSCAR_R[TEMP0]                 ;TURN CMI ON
U 02, 7700,8800,5BE4,0304,6070,1803 0017  6276 ;% 02    MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,6870,1804 0022  6280 ;% 03    MEMSCAR_R[TEMP2]                 ;TURN CACHE OFF
U 04, 7700,C800,5BE4,030C,6070,1805 002D  6284 ;% 04    MEMSCR_R[TEMP3]
U 05, 7700,4800,5BE4,03D8,4A70,1806 0038  6288 ;% 05    VA_R[ZERO]                       ;ZERO VA REGISTER
U 06, 7701,8800,5BE4,02D8,5590,1807 0043  6292 ;% 06    WRITE,LONG,SIZE[LONG],WB_R[ZERO] ;ZERO THE TEST LOCATION
```

```
U 07. 7701,C800,5BE4,00D8,5D80,1808  004E  6296 ;% 07   WRITE,SIZE[BYTE],WB_R[ZERO]                ;SET WRITE SIZE LATCH
U 08. 5700,4800,0364,0300,4A70,1809  0059  6300 ;% 08   VA_RDM                                    ;TEST ADDRESS TO VA REGISTER
U 09. 7701,C800,5BE4,0210,54A0,180A  0064  6304 ;% 09   WRITE.SECOND,WDR_UNROT(R[TEMP4]),SIZE[LONG]
                                     006F  6308                                                  ;PERFORM WRITE SECOND
U 0A. 7700,4800,0364,0300,0510,180B  006F  6309 ;% 0A   READ.LONG                                 ;READ RESULTS BACK
U 0B. 6700,0812,5924,0300,0470,180C  007A  6313 ;% 0B   RDM_WB,WB_M[MDR]                          ;MOVE RESULTS TO WH REGISTER
U 0C. 7300,C800,0364,0300,0470,180D  0085  6317 ;% 0C   NOP,STOP.CLOCK                            ;STOP THE CPU CLOCK
                                     0090  6321
                                     0090  6322 RTEMP_DATA
                                     0001  6323
                          0E000000   0001  6324          .LONG   ^X0E000000                     ;REFERENCE CACHE ONLY ADDRESS
                          00000000   0005  6325          .LONG   ^X00000000                     ;TURN CMI ON
                          06000000   0009  6326          .LONG   ^X06000000                     ;CACHE CONTROL REGISTER ADDRESS
                          01000000   000D  6327          .LONG   ^X01000000                     ;TURN CACHE OFF
                          FFFFFFFF   0011  6328          .LONG   ^XFFFFFFFF                     ;TEST PATTERN
                                     0015  6329
                                     0015  6330 END_CPU_DATA
                                     0129  6331
                                     0129  6332
                                     0129  6333 END_TEST_DATA
                                     0129
                                     0129        ;-------------------------------------------------------------------------
                                     0129  6334 ENDTEST
```

```
001C        .SBTTL  "TEST   08D     Test Read Lock Function
001C  6336 ;*******************************************************************************
001C  6337 ;++
001C  6338 ;
001C  6339 ; FUNCTIONAL DESCRIPTION:
001C  6340 ;
001C  6341 ;       This test checks the operation of the read lock bus micro order and
001C  6342 ;       the read lock timeout counter.
001C  6343 ;
001C  6344 ;
001C  6345 ; Test Algorithm:
001C  6346 ;
001C  6347 ;       1.  Load u-code to initialize the test conditions
001C  6348 ;       2.  Execute dcs program
001C  6349 ;               a.  Enable the cmi
001C  6350 ;               b.  Disable cache
001C  6351 ;       3.  Load u-code to free run the cpu clock
001C  6352 ;       4.  Load the first u-inst.
001C  6353 ;       5.  Start the cpu clock
001C  6354 ;       6.  Clear the CMICTL register
001C  6355 ;       7.  Load a Read Lock command into the RDM MA register
001C  6356 ;       8.  Execute a CMI cycle
001C  6357 ;       9.  Stop the cpu clock
001C  6358 ;      10.  Load test u-code into DCS
001C  6359 ;      11.  Inusre the clocks are off.
001C  6360 ;      12.  Clear Master Stop Enable.
001C  6361 ;      13.  Set-up RDCTRL register to watch for micro trap.
001C  6362 ;      14.  Start the clocks.
001C  6363 ;      15.  Set-up a watch dog timing loop.  The watch dog timer will catch the
001C  6364 ;           case where RLTO never takes place.  The following micro code gets
001C  6365 ;           executed while the clock is running:
001C  6366 ;               a.  Zero RTEMP7
001C  6367 ;               b.  Zero the VA register
001C  6368 ;               c.  Put 1 in the Q register
001C  6369 ;               d.  Perform a read lock and branch to DCS address 0
001C  6370 ;               e.  Loop on address zero and one while incrementing RTEMP7
001C  6371 ;      16.  If watch dog loop expires then force an error
001C  6372 ;      17.  Test the CSTRP register for a machine check
001C  6373 ;      18.  If no error execute u-code starting at DCS address 8
001C  6374 ;               a.  Read the Write Vector Occurred register to the RDM
001C  6375 ;      19.  Test the WHREG to see if the Write Vector Occurred bit is set
001C  6376 ;      20.  If no error execute u-code starting at DCS address D
001C  6377 ;               a.  Read the Bus Error register to the RDM
001C  6378 ;      21.  Test the WHREG to see if the Non-Existant Memory bit is set
001C  6379 ;      22.  If no error execute u-code starting at DCS address 11
001C  6380 ;               a.  Read the Machine Check register to the RDM
001C  6381 ;      23.  Test the WHREG to see if the Bus Error bit is set
001C  6382 ;      24.  If no error execute u-code starting at DCS address 16
001C  6383 ;               a.  Read RTEMP7 to the RDM
001C  6384 ;      25.  Check for value of FF(hex) in WHREG ( this value is for the CMK
001C  6385 ;           gate array).
001C  6386 ;               a.  If the value is equal to FF
001C  6387 ;                       Then skip to next test
001C  6388 ;                   Else skip to 100$ to check against CML values
001C  6389 ;      26.  (100$) Check for value of 1FE(hex) in WHREG ( this is one of the
```

```
001C  6390 ;           two good values that the CML-RLTO circuit can timeout at).
001C  6391 ;               a.  If the value is equal to 1FE
001C  6392 ;                       Then skip to next test
001C  6393 ;                   Else skip to 110$ to check against second CML value
001C  6394 ;      27.  (110$) Check for value of 1FD(hex) in WHREG ( this is the second of
001C  6395 ;           two good values).
001C  6396 ;               a.  If the value is equal to 1FD
001C  6397 ;                       Then skip to next test
001C  6398 ;                   Else flag an error
001C  6399 ;
001C  6400 ;
001C  6401 ; TEST PATTERNS:
001C  6402 ;
001C  6403 ;       None
001C  6404 ;
001C  6405 ;
001C  6406 ; LOGIC DESCRIPTION:
001C  6407 ;
001C  6408 ;       This test checks the operation of the read lock micro order by first
001C  6409 ;       performing a read lock from the RDM and then performing a second
001C  6410 ;       read lock form the CPU.  The second read lock should cause
001C  6411 ;       the cpu read lock timer to time out and initiate a machine check.
001C  6412 ;       The write vector occurred, non-existatnt memory, and bus error bits
001C  6413 ;       should all be set in the appropriate status registers.  The majority
001C  6414 ;       of logic tested is in the CML gate array.
001C  6415 ;
001C  6416 ;
001C  6417 ; ASSUMPTIONS:
001C  6418 ;
001C  6419 ;       This test assumes the CMI is operational.
001C  6420 ;
001C  6421 ;
001C  6422 ; ERROR DESCRIPTION:
001C  6423 ;
001C  6424 ;       First IFERROR:
001C  6425 ;
001C  6426 ;       Expected machine check address
001C  6427 ;       Received CS address
001C  6428 ;
001C  6429 ;       Second IFERROR:
001C  6430 ;
001C  6431 ;       Expected contents of write vector occurred register
001C  6432 ;       Received contents of write vector occurred register
001C  6433 ;
001C  6434 ;       Third IFERROR:
001C  6435 ;
001C  6436 ;       Expected contents of bus error register
001C  6437 ;       Received contents of bus error register
001C  6438 ;
001C  6439 ;       Forth IFERROR:
001C  6440 ;
001C  6441 ;       Expected contents of machine check error register
001C  6442 ;       Received contents of machine check error register
001C  6443 ;
001C  6444 ;       Fifth IFERROR:
```

```
001C  6445 ;
001C  6446 ;       Expected contents of RTEMP7
001C  6447 ;       Received contents of RTEMP7
001C  6448 ;
001C  6449 ;--
001C  6450 ;*******************************************************************************
001C  6451
001C  6452
001C  6453          INITIALIZE                              ;init the cpu
001E  6454          EXPUTRAP                                ;expect a u-trap
0020  6455 ;
0020  6456 ; Turn the CMI on and Cache off.
0020  6457 ;
0020  6458          LOADDCS         10$,,0,6                ;code to enable cmi
0027  6459          FETCH           0                       ;start dcs program
002C  6460          BRSTCLK         5,INH                   ;end dcs program
0030  6461          LOADDCS         11$,,0,2                ;code to run cpu clock
0037  6462          FETCH           0                       ;load first u-inst.
003C  6463 ;
003C  6464 ; Perform a Read Lock Function on the CMI from the RDM
003C  6465 ;
003C  6466          LOADREG         DCSCR,2$,,B             ;start the cpu clock
0044  6467          LOADREG         CMICTL,1$,,B            ;clear the cmictl register
004C  6468          LOADREG         MAREG,3$,,L             ;load read lock command
0054  6469          LOADREG         CMICTL,4$,,B            ;perform the cmi cycle
005C  6470          LOADREG         DCSCR,5$,,B             ;stop the cpu clock
0064  6471          LOADDCS         12$,,0,<^X19>           ;code for test
006B  6472          FETCH           2                       ;set-up first DCS address
0070  6473 ;
0070  6474 ; Here we manually turn the clock on and start dcs up.  This is done after the
0070  6475 ; first Read lock function has taken place from the RDM.  If the Watch dog
0070  6476 ; timing loop expires then force an error.  If it doesn't then check to see
0070  6477 ; if RLTO was correctly detected.
0070  6478 ;
0070  6479          LOADREG         DCSCR,18$,,B            ;insure that clock is off
0078  6480          LOADREG         RDCTRL,20$,,B           ;clear the master stop enable
0080  6481          LOADREG         RDCTRL,21$,,B           ;watch for microtrap
0088  6482          LOADREG         DCSCR,22$,,B            ;start the clocks
0090  6483          LOOP            I,1,1024                ;start watch dog counter
0097  6484          CMPREGMSK       CLKDCS,23$,,24$,,B,     ;check for clock stopped
00A3  6485          SKIP            80$,ONERROR             ;skip on clocks stopped
00AA  6486          ENDLOOP         I                       ;increment watch dog counter
00AD  6487          LOADREG         DCSCR,18$,,B            ;stop the clocks
00B5  6488          SKIP            90$                     ;go force an error
00BC  6489 ;
00BC  6490 ; Check the contents of the CSTRP, Write Vector Occurred, Bus Error, and
00BC  6491 ; Machine Check registers.
00BC  6492 ;
00BC  6493 80$:
00BC  6494          CMPREGMSK       CSTRP,6$,,7$,,W,        ;test for machine check
00C8  6495          IFERROR         ,<<CML><UTR>>,          ;no read lock timeout
00CF  6496          FETCH           9                       ;start dcs program
00D4  6497          BRSTCLK         <^X0D>,INH              ;end dcs program
00D8  6498          CMPREGMSK       WHREG,8$,,9$,,L,        ;test for write vector occurred
00E4  6499          IFERROR         ,<<CML><ADK>>,          ;write vector not set
```

```
                    00EB  6500          FETCH           <^X0E>                  ;start dcs program
                    00F0  6501          BRSTCLK         <^X11>                  ;end dcs program
                    00F4  6502          CMPREGMSK       WHREG,13$,,9$,,L,       ;test for nonexistant memory
                    0100  6503          IFERROR         ,<<UTR>>,               ;bit not set
                    0106  6504          FETCH           <^X12>                  ;start dcs program
                    010B  6505          BRSTCLK         <^X16>                  ;end dcs program
                    010F  6506          CMPREGMSK       WHREG,13$,,9$,,L,       ;test for bus error
                    011B  6507          IFERROR         ,<<UTR>>,               ;bit not set
                    0121  6508 ;
                    0121  6509 ; If we got this far then the last thing to do is to check the value for the
                    0121  6510 ; RLTO that was generated in RTEMP7.  This value could be one of three values,
                    0121  6511 ; depending on the gate array installed (CML or CMK).  For the CMK there is
                    0121  6512 ; only one value - FF(hex).  For the CML there could be two different values -
                    0121  6513 ; 1FE or 1FD ( The first time the test is run is will be 1FE but if the test
                    0121  6514 ; is ran continiously, then the value could come out as 1FD because of timing).
                    0121  6515 ;
                    0121  6516 90$:
                    0121  6517          FETCH           <^X17>                  ;start dcs program
                    0126  6518          BRSTCLK         <^X19>                  ;end dcs program
                    012A  6519          CMPREG          WHREG,14$,,L,           ;RLTO count = FF (CMK)
                    0133  6520          SKIP            100$,ONERROR            ;skip if no
                    013A  6521          SKIP
                    0141  6522 ;
                    0141  6523 ; Come here to check and see if the count is equal to 1FE
                    0141  6524 ;
                    0141  6525 100$:
                    0141  6526          CMPREG          WHREG,15$,,L,           ;RLTO count = 1FE (CML)
                    014A  6527          SKIP            110$,ONERROR            ;skip if no
                    0151  6528          SKIP
                    0158  6529 ;
                    0158  6530 ; Come here to check and see if the count is equal to 1FD
                    0158  6531 ;
                    0158  6532 110$:
                    0158  6533          CMPREG          WHREG,16$,,L,           ;RLTO count = 1FD (CML)
                    0161  6534          IFERROR         ,<<CML>>,               ;Callout error - CML
                    0167  6535          SKIP
                    016E  6536
                    016E  6537 BEGIN_TEST_DATA
                    016E
                    016E        ;-------------------------------------------------------------------------
                    016E  6538
                      01  016E  6539 1$:     .BYTE   ^X01                    ;clear cmictl register
                      00  016F  6540 2$:     .BYTE   ^X00                    ;start the cpu clock
               02000000   0170  6541 3$:     .LONG   ^X02000000              ;read lock command
                      48  0174  6542 4$:     .BYTE   ^X48                    ;perform the read lock
                      60  0175  6543 5$:     .BYTE   ^X60                    ;stop the cpu clock
                    0028  0176  6544 6$:     .WORD   ^X0028                  ;machine check vector
                    3FFF  0178  6545 7$:     .WORD   ^X3FFF                  ;mask for cmpregmsk pseudo op
               01000000   017A  6546 8$:     .LONG   ^X01000000              ;write vector occured
               0F000000   017E  6547 9$:     .LONG   ^X0F000000              ;mask for cmpregmsk pseudo op
                          0182  6548 10$:
U 00, 7700,C800,0364,0300,0470,1801   0182  6549 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802   018D  6553 ;% 01   MEMSCAR_R[TEMP0]          ;turn cmi on
U 02, 7700,8800,5BE4,0304,6070,1803   0198  6557 ;% 02   MEMSCR_R[TEMP1]
U 03, 7700,8800,5BE4,0308,6870,1804   01A3  6561 ;% 03   MEMSCAR_R[TEMP2]          ;turn cache off
```

```
U 04. 7700,C800,5BE4,030C,6070,1805  01AE  6565 ;* 04   MEMSCR_R[TEMP3]
U 05. 7300,4800,0364,0300,0470,1806  01B9  6569 ;* 05   NOP,STOP.CLOCK                          ;stop the cpu clock
                                     01C4  6573 11$:
U 00. 7700,C800,0364,0300,0470,1801  01C4  6574 ;* 00   NOP                                   ;code to run clock
U 01. 7500,C800,0364,0300,0470,1802  01CF  6578 ;* 01   NOP,DCS.ADDR_0
                                     01DA  6582 12$:
U 00. 7700,8840,0390,031C,0470,1801  01DA  6583 ;* 00   R[TEMP7]_RB+Q                          ;increment RTEMP7
U 01. 7500,8840,0390,031C,0470,1802  01E5  6587 ;* 01   R[TEMP7]_RB+Q,DCS.ADDR_0               ;increment RTEMP7
U 02. 7700,4800,0364,0300,0470,1803  01F0  6591 ;* 02   NOP
U 03. 7700,8840,5B70,031C,0470,1804  01FB  6595 ;* 03   R[TEMP7]_0                           ;zero RTEMP7
U 04. 7700,4800,5BE4,03D8,4A70,1805  0206  6599 ;* 04   VA_R[ZERO]                           ;zero the va register
U 05. 7700,B800,7FFF,FFFF,0470,1806  0211  6603 ;* 05   LONLIT_[00000001]                    ;move 1 to q register
U 06. 7700,8800,5BE5,03D4,0470,1807  021C  6607 ;* 06   Q_R[LONLIT]
U 07. 7501,8800,0364,0200,0530,1808  0227  6611 ;* 07   READ.MOD.LOCK,SIZE[LONG],DCS.ADDR_0  ;perform read lock
U 08. 7300,4800,0364,0300,0470,1809  0232  6615 ;* 08   NOP,STOP.CLOCK                        ;stop the cpu clock
U 09. 7700,4800,0364,0300,0470,180A  023D  6619 ;* 09   NOP
U 0A. 7700,8800,5BE4,0310,6870,180B  0248  6623 ;* 0A   MEMSCAR_R[TEMP4]                     ;read write vector register
U 0B. 6700,4800,0364,0300,6470,180C  0253  6627 ;* 0B   RDM_WB,WCTRL/MEMSCR                  ;move register to rdm
U 0C. 7700,4800,5BE4,03D8,6070,180D  025E  6631 ;* 0C   MEMSCR_R[ZERO]                      ;clear the register
U 0D. 7300,C800,0364,0300,0470,180E  0269  6635 ;* 0D   NOP,STOP.CLOCK                      ;stop the cpu clock
U 0E. 7700,4800,0364,0300,0470,180F  0274  6639 ;* 0E   NOP
U 0F. 7700,C800,5BE4,0314,6870,1810  027F  6643 ;* 0F   MEMSCAR_R[TEMP5]                     ;read bus error register
U 10. 6700,4800,0364,0300,6470,1811  028A  6647 ;* 10   RDM_WB,WCTRL/MEMSCR
U 11. 7300,4800,0364,0300,0470,1812  0295  6651 ;* 11   NOP,STOP.CLOCK
U 12. 7700,C800,0364,0300,0470,1813  02A0  6655 ;* 12   NOP
U 13. 7700,4800,5BE4,0318,6870,1814  02AB  6659 ;* 13   MEMSCAR_R[TEMP6]                     ;read machine check register
U 14. 6700,C800,0364,0300,6470,1815  02B6  6663 ;* 14   RDM_WB,WCTRL/MEMSCR
U 15. 7700,4800,5BE4,03D8,6070,1816  02C1  6667 ;* 15   MEMSCR_R[ZERO]                      ;clear machine check register
U 16. 7300,4800,0364,0300,0470,1817  02CC  6671 ;* 16   NOP,STOP.CLOCK
U 17. 7700,4800,0364,0300,0470,1818  02D7  6675 ;* 17   NOP
U 18. 6700,8800,5BE4,031C,0470,1819  02E2  6679 ;* 18   RDM_WB,WB_R[TEMP7]                   ;read timing to rdm
U 19. 7300,C800,0364,0300,0470,181A  02ED  6683 ;* 19   NOP,STOP.CLOCK
                          08000000   02F8  6687 13$:     .LONG   ^X08000000                ;compare data
                          000000FF   02FC  6688 14$:     .LONG   ^X000000FF                ;timing for read lock timeout
                          000001FE   0300  6689 15$:     .LONG   ^X000001FE                ;these two longwords are
                          000001FD   0304  6690 16$:     .LONG   ^X000001FD                ;possible counts for RLTO (CML)
                                60   0308  6691 18$:     .BYTE   ^X60                      ;value for turning clock off
                                00   0309  6692 20$:     .BYTE   ^X00                      ;value for clearing MSTP
                                28   030A  6693 21$:     .BYTE   ^X28                      ;value for watching micro traps
                                10   030B  6694 22$:     .BYTE   ^X10                      ;value used to turn clock on
                                00   030C  6695 23$:     .BYTE   ^X00                      ;value to check for clock run
                                C0   030D  6696 24$:     .BYTE   ^XC0                      ;mask used in clock run compare
                                     030E  6697
                                     030E  6698
                                     030E  6699 BEGIN_CPU_DATA
                                     0002  6700
                                     0002  6701 RTEMP_DATA
                                     0001  6702
                          0E000000   0001  6703          .LONG   ^X0E000000                ;reference cache only address
                          00000000   0005  6704          .LONG   ^X00000000                ;turn cmi on
                          06000000   0009  6705          .LONG   ^X06000000                ;cache control register address
                          01000000   000D  6706          .LONG   ^X01000000                ;disable cache
                          02000000   0011  6707          .LONG   ^X02000000                ;write vector occurred reg add
                          09000000   0015  6708          .LONG   ^X09000000                ;bus error register address
                          08000000   0019  6709          .LONG   ^X08000000                ;machine check register address
```

```
001D  6710
001D  6711 END_CPU_DATA
030E  6712
030E  6713
030E  6714 END_TEST_DATA
030E
030E       ;-----------------------------------------------------------------------
030E  6715 ENDTEST
```

```
0020          .SBTTL  "TEST   08E     Test Invalidate Cache Logic
0020  6717 ;*******************************************************************************
0020  6718 ;++
0020  6719 ;
0020  6720 ; FUNCTIONAL DESCRIPTION:
0020  6721 ;
0020  6722 ;       THIS TEST WILL CHECK THE ABILITY OF THE CPU TO DETECT CMI
0020  6723 ;       WRITES TO MAIN MEMORY BY ANOTHER DEVICE.  THE CPU SHOULD
0020  6724 ;       INVALIDATE THE APPROPRIATE HIT LOCATIONS IN CACHE.
0020  6725 ;
0020  6726 ;
0020  6727 ; TEST ALGORITHM:
0020  6728 ;
0020  6729 ;       1.  CREATE A LOOP OF 6 ITERATIONS
0020  6730 ;       2.  LOAD MICROCODE TO VALIDATE CACHE
0020  6731 ;       3.  LOAD WD REGISTER WITH TEST ADDRESS
0020  6732 ;       4.  EXECUTE DCS PROGRAM
0020  6733 ;               a.  TURN CMI ON
0020  6734 ;               b.  LOAD TEST ADDRESS INTO VA REGISTER
0020  6735 ;               c.  PERFORM A WRITE TO MAIN MEMORY
0020  6736 ;       5.  MODIFY DCS MICROCODE
0020  6737 ;       6.  SET UP THE RDM REGISTERS TO PERFORM A WRITE TO THE TEST ADDRESS
0020  6738 ;       7.  EXECUTE DCS PROGRAM
0020  6739 ;               a.  TURN OFF CMI
0020  6740 ;               b.  LOAD TEST ADDRESS INTO VA REGISTER
0020  6741 ;               c.  PERFORM A READ OPERATION
0020  6742 ;               d.  READ CACHE ERROR REGISTER TO RDM
0020  6743 ;       8.  TEST CONTENTS OF WH REGISTER FOR A CACHE MISS
0020  6744 ;       9.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST ADDRESSES
0020  6745 ;
0020  6746 ;
0020  6747 ; TEST PATTERNS:
0020  6748 ;
0020  6749 ;       ITERATION       TEST ADDRESS
0020  6750 ;           1             00AAAAAA
0020  6751 ;           2             00555555
0020  6752 ;           3             00333333
0020  6753 ;           4             000F0F0F
0020  6754 ;           5             0000FF00
0020  6755 ;           6             000000FF
0020  6756 ;
0020  6757 ;
0020  6758 ; LOGIC DESCRIPTION:
0020  6759 ;
0020  6760 ;       THIS TEST UNFORTUNATLY INVOLVES A GREAT DEAL OF LOGIC.  THE CML GATE
0020  6761 ;       ARRAY DETECTS THAT A CMI WRITE IS GOING ON AND GENERATES A SIGNAL
0020  6762 ;       CALLED "SNAPSHOT CMI".  THIS SIGNAL RUNS TO CAK, ADK, PRK, AND MDR TO
0020  6763 ;       CAUSE THE CACHE TO INVALIDATE SHOULD THERE BE A HIT.
0020  6764 ;
0020  6765 ;
0020  6766 ; ASSUMPTIONS:
0020  6767 ;
0020  6768 ;       THIS TEST ASSUMES THE WBUS, MBUS, CMI AND RDM ARE OPERATIONAL.
0020  6769 ;
0020  6770 ;
```

```
                     6771 ; ERROR DESCRIPTION:
                     6772 ;
                     6773 ;       EXPECTED CACHE ERROR REGISTER CONTENTS
                     6774 ;       RECEIVED CACHE ERROR REGISTER CONTENTS
                     6775 ;       LOOP COUNT
                     6776 ;
                     6777 ;--
                     6778 ;*****************************************************************************
                     6779
                     6780
                     6781
0020                 6782         INITIALIZE                                 ;INIT THE CPU
0022                 6783         LOOP            I,1,6                      ;CREATE A TEST LOOP OF 6
0029                 6784         LOADDCS         1$,,0,6                    ;U-CODE TO VALIDATE CACHE
0030                 6785         LOADREG         WDREG,2$,I,L               ;TEST ADDRESS TO WD REGISTER
0038                 6786         FETCH           0                          ;START DCS PROGRAM
003D                 6787         BRSTCLK         5                          ;END DCS PROGRAM
0041                 6788         LOADDCS         3$,,0,<^X0A>               ;TEST U-CODE
0048                 6789         ERRLOOP                                    ;ERROR LOOP POINT
004A                 6790         LOADREG         RDCTRL,4$,,B               ;CLEAR MASTER HALT ENABLE
0052                 6791         LOADREG         RDCTRL,5$,,B               ;SET MASTER HALT & TRAP HALT
005A                 6792         FETCH           0                          ;LOAD FIRST DCS INSTRUCTION
005F                 6793         LOADREG         DCSCR,6$,,B                ;START THE CPU CLOCK
0067                 6794         LOADREG         CMICTL,7$,,B               ;CLEAR CMICTL REGISTER
006F                 6795         LOADREG         MAREG,8$,I,L               ;LOAD ADDRESS & COMMAND
0077                 6796         LOADREG         MDREG,4$,,L                ;LOAD ZERO DATA
007F                 6797         LOADREG         CMICTL,9$,,B               ;PERFORM MAIN MEMORY WRITE
0087                 6798         LOADREG         DCSCR,10$,,B               ;STOP THE CPU CLOCK
008F                 6799         FETCH           2                          ;START DCS PROGRAM
0094                 6800         BRSTCLK         9,INH                      ;END DCS PROGRAM
0098                 6801         CMPREGMSK       WHREG,11$,,12$,,L,         ;TEST FOR CACHE MISS
00A4                 6802         IFERROR         ,<<CML><CAK><ADK><PRK><MDR>>,
00AE                 6803         ENDLOOP         I                          ;END TEST LOOP
00B1                 6804         SKIP                                       ;GO TO NEXT TEST
00B8                 6805
00B8                 6806
00B8                 6807
00B8                 6808 BEGIN_TEST_DATA
00B8                 6809
00B8                 6810 1$:
U 00, 7700,C800,0364,0300,0470,1801  00B8  6811 ;* 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802  00C3  6815 ;* 01    MEMSCAR_R[TEMP0]                   ;TURN CMI ON
U 02, 7700,C800,5BE4,03D8,6070,1803  00CE  6819 ;* 02    MEMSCR_R[ZERO]
U 03, 5700,C800,0364,0300,4A70,1804  00D9  6823 ;* 03    VA_RDM                             ;TEST ADDRESS TO VA REGISTER
U 04, 7700,C800,5BE4,03D8,5C80,1805  00E4  6827 ;* 04    WRITE.PHY R[ZERO]                  ;VALIDATE THE LOCATION IN CACHE
U 05, 7300,4800,0364,0300,0470,1806  00EF  6831 ;* 05    NOP,STOP.CLOCK                     ;STOP THE CPU CLOCK
                        00AAAAAA     00FA  6835 2$:       .LONG   ^X00AAAAAA             ;TEST ADDRESSES
                        00555555     00FE  6836           .LONG   ^X00555555
                        00333333     0102  6837           .LONG   ^X00333333
                        000F0F0F     0106  6838           .LONG   ^X000F0F0F
                        0000FF00     010A  6839           .LONG   ^X0000FF00
                        000000FF     010E  6840           .LONG   ^X000000FF
                                     0112  6841 3$:
```

```
U 00. 7700,C800,0364,0300,0470,1801  0112  6842 ;% 00   NOP
U 01. 7500,C800,0364,0300,0470,1802  011D  6846 ;% 01   NOP,DCS.ADDR_0                    ;ALLOW CPU CLOCK TO SPIN
U 02. 7700,4800,0364,0300,0470,1803  0128  6850 ;% 02   NOP
U 03. 7700,C800,5BE4,0300,6870,1804  0133  6854 ;% 03   MEMSCAR_R[TEMP0]                 ;SHUT OFF CMI
U 04. 7700,8800,5BE4,0308,6070,1805  013E  6858 ;% 04   MEMSCR_R[TEMP2]
U 05. 5700,4800,0364,0300,4A70,1806  0149  6862 ;% 05   VA_RDM                           ;LOAD TEST ADDRESS INTO VA REG
U 06. 7700,4800,0364,0300,0400,1807  0154  6866 ;% 06   READ.PHY                        ;PERFORM READ OPERATION
U 07. 7700,8800,5BE4,0304,6870,1808  015F  6870 ;% 07   MEMSCAR_R[TEMP1]               ;TEST FOR CACHE MISS
U 08. 6700,4800,0364,0300,6470,1809  016A  6874 ;% 08   RDM_WB,WCTRL/MEMSCR
U 09. 7300,4800,0364,0300,0470,180A  0175  6878 ;% 09   NOP,STOP.CLOCK                   ;STOP THE CPU CLOCK
                                       00  0180  6882 4$:     .BYTE   ^X00              ;USED TO ZERO REGISTERS
                                       28  0181  6883 5$:     .BYTE   ^X28              ;SET MASTER HALT & TRAP HALT
                                       10  0182  6884 6$:     .BYTE   ^X10              ;START THE CPU CLOCK
                                       01  0183  6885 7$:     .BYTE   ^X01              ;CLEAR CMICTL REGISTER
                                 F8AAAAAA  0184  6886 8$:     .LONG   ^XF8AAAAAA        ;ADDRESS & COMMAND
                                 F8555555  0188  6887         .LONG   ^XF8555555
                                 F8333333  018C  6888         .LONG   ^XF8333333
                                 F80F0F0F  0190  6889         .LONG   ^XF80F0F0F
                                 F800FF00  0194  6890         .LONG   ^XF800FF00
                                 F80000FF  0198  6891         .LONG   ^XF80000FF
                                       28  019C  6892 9$:     .BYTE   ^X28              ;START CMI CYCLE
                                       70  019D  6893 10$:    .BYTE   ^X70              ;STOP THE CPU CLOCK
                                 00000000  019E  6894 11$:    .LONG   ^X00000000        ;TEST FOR CACHE MISS
                                 0F000000  01A2  6895 12$:    .LONG   ^X0F000000        ;MASK FOR CMPREGMSK PSEUDO OP
                                           01A6  6896
                                           01A6  6897
                                           01A6  6898
                                           01A6  6899 BEGIN_CPU_DATA
                                           0002  6900
                                           0002  6901 RTEMP_DATA
                                           0001  6902
                                 0E000000  0001  6903         .LONG   ^X0E000000        ;REFERENCE CACHE ONLY REG ADD
                                 04000000  0005  6904         .LONG   ^X04000000        ;CACHE ERROR REGISTER ADDRESS
                                 01000000  0009  6905         .LONG   ^X01000000        ;TURN CMI OFF
                                           000D  6906
                                           000D  6907 END_CPU_DATA
                                           01A6  6908
                                           01A6  6909
                                           01A6  6910 END_TEST_DATA
                                           01A6
                                           01A6       ;-----------------------------------------------------------------------
                                           01A6  6911 ENDTEST
```

```
0047        .SBTTL  "TEST   08F     Test Bus Field NOP, READ, and READ.PHY Micro Orders to Mai -
cont> n Memory

0047  6913 ;*******************************************************************************
0047  6914 ;++
0047  6915 ;
0047  6916 ; FUNCTIONAL DESCRIPTION:
0047  6917 ;
0047  6918 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE FOLLOWING
0047  6919 ;       'BUS" FIELD MICRO ORDERS USING MAIN MEMORY:
0047  6920 ;               1.  NOP
0047  6921 ;               2.  READ
0047  6922 ;               3.  READ.PHY
0047  6923 ;
0047  6924 ;
0047  6925 ; TEST ALGORITHM:
0047  6926 ;
0047  6927 ;       1.  CREATE A TEST LOOP OF 3 TO SELECT BUS MICRO ORDERS
0047  6928 ;       2.  LOAD MICRO ORDER BEING TESTED INTO DCS
0047  6929 ;       3.  EXECUTE DCS PROGRAM
0047  6930 ;               a.  TURN MME OFF
0047  6931 ;               b.  TURN CMI ON
0047  6932 ;               c.  TURN CACHE OFF
0047  6933 ;               e.  LOAD 0'S INTO VA REGISTER
0047  6934 ;               f.  WRITE 1'S TO ADDRESS 0 OF MAIN MEMORY
0047  6935 ;               g.  LOAD 0'S INTO THE MDR REGISTER
0047  6936 ;               h.  PERFORM THE SELECTED BUS FUNCTION
0047  6937 ;               i.  MOVE THE CONTENTS OF THE MDR REGISTER TO RDM WH REGISTER
0047  6938 ;       4.  TEST THE WH REGISTER FOR THE CORRECT DATA
0047  6939 ;       5.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
0047  6940 ;
0047  6941 ;
0047  6942 ; TEST PATTERNS:
0047  6943 ;
0047  6944 ;       ITERATION       FUNCTION
0047  6945 ;
0047  6946 ;           1           BUS/NOP
0047  6947 ;           2           BUS/READ
0047  6948 ;           3           BUS/READ.PHY
0047  6949 ;
0047  6950 ;
0047  6951 ; LOGIC DESCRIPTION:
0047  6952 ;
0047  6953 ;       THIS TEST CHECKS THE ABILITY OF THE CPU TO PERFORM THREE
0047  6954 ;       OF THE BASIC BUS MICRO ORDERS TO MAIN MEMORY.  THE MDR GATE ARRAY
0047  6955 ;       IS THE BASIC DATA PATH AND IS A GOOD FIRST CHOICE TO REPLACE.  NEXT
0047  6956 ;       BEST GUESS IS CAK OR CML.
0047  6957 ;
0047  6958 ;
0047  6959 ; ASSUMPTIONS:
0047  6960 ;
0047  6961 ;       THIS TEST ASSUMES THAT THE CMI HAS BEEN TESTED FROM THE RDM
0047  6962 ;
0047  6963 ;
0047  6964 ; ERROR DESCRIPTION:
0047  6965 ;
0047  6966 ;       EXPECTED MDR DATA
```

```
                                0047  6967 ;       RECEIVED MDR DATA
                                0047  6968 ;       LOOP COUNT I
                                0047  6969 ;
                                0047  6970 ;--
                                0047  6971 ;*******************************************************************************
                                0047  6972 ;///////////////////////////////////////////////////////////////////////////////
                                0047  6973
                                0047  6974         INITIALIZE                                      ;INIT CPU
                                0049  6975         LOOP            I,1,3                           ;CREATE A TEST LOOP
                                0050  6976         LOADDCS         1$,I,<^X0A>,1                   ;MODIFY DCS ADDRESS 0A
                                0057  6977         ERRLOOP                                         ;ERROR LOOP
                                0059  6978         FETCH           0                               ;START DCS PROGRAM
                                005E  6979         BRSTCLK         <^X0C>                          ;END DCS PROGRAM
                                0062  6980         CMPREG          WHREG,2$,I,L                    ;COMPARE RESULTS
                                006B  6981         IFERROR         ,<<MDR><CML><CAK>>,<<MC0><MC1><MC2>>;FAILING ARRAYS
                                0076  6982         ENDLOOP         I                               ;END TEST LOOP
                                0079  6983         SKIP                                            ;GO TO NEXT TEST
                                0080  6984
                                0080  6985 BEGIN_TEST_DATA
                                0080
                                0080      ;-------------------------------------------------------------------
                                0080  6986
                                0080  6987 1$:
U 0A, 7701,8800,0364,0200,0470,180B  0080  6988 ;% 0A   BUS/NOP,SIZE[LONG]                         ;DCS ADD 08 LOOP 1
U 0A, 7701,8800,0364,0200,0500,180B  008B  6992 ;% 0A   BUS/READ,SIZE[LONG]                       ;DCS ADD 08 LOOP 2
U 0A, 7700,4800,0364,0300,0400,180B  0096  6996 ;% 0A   BUS/READ.PHY                              ;DCS ADD 08 LOOP 3
                                     00A1  7000
                        00000000     00A1  7001 2$:      .LONG   ^X00000000                      ;RESULTS LOOP 1
                        FFFFFFFF     00A5  7002          .LONG   ^XFFFFFFFF                      ;RESULTS LOOP 2
                        FFFFFFFF     00A9  7003          .LONG   ^XFFFFFFFF                      ;RESULTS LOOP 3
                                     00AD  7004
                                     00AD  7005 BEGIN_CPU_DATA
                                     0002  7006
                                     0002  7007 DCS_DATA
                                     0001  7008
U 00, 7700,C800,0364,0300,0470,1801  0001  7009 ;% 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802  000C  7013 ;% 01   MEMSCAR_R[TEMP0]                          ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803  0017  7017 ;% 02   MEMSCR_R[TEMP1]                           ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804  0022  7021 ;% 03   MEMSCAR_R[TEMP2]                          ;REF CACHE ONLY ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805  002D  7025 ;% 04   MEMSCR_R[TEMP3]                           ;TURN CMI ON
U 05, 7700,0800,5BE4,0310,6870,1806  0038  7029 ;% 05   MEMSCAR_R[TEMP4]                          ;CACHE CONT REG ADD TO MEMSCAR
U 06, 7700,4800,5BE4,0314,6070,1807  0043  7033 ;% 06   MEMSCR_R[TEMP5]                           ;TURN CACHE OFF
U 07, 7700,8800,5BE4,0304,4A70,1808  004E  7037 ;% 07   VA_R[TEMP1]                               ;0'S TO VA REGISTER
U 08, 7700,C800,5BE4,0318,5C80,1809  0059  7041 ;% 08   WRITE.PHY R[TEMP6]                        ;1'S TO ADDRESS 0
U 09, 7700,0800,5BE4,0304,4670,180A  0064  7045 ;% 09   WB_R[TEMP1],WCTRL/MDR_WB                 ;0'S TO MDR REGISTER
U 0A, 7701,8800,0364,0200,0470,180B  006F  7049 ;% 0A   BUS/NOP,SIZE[LONG]                       ;PERFORM BUS FUNCTION
U 0B, 6700,0812,5924,0300,0470,180C  007A  7053 ;% 0B   RDM_WB,WB_M[MDR]                         ;READ MDR TO RDM
U 0C, 7300,C800,0364,0300,0470,180D  0085  7057 ;% 0C   NOP,STOP.CLOCK                           ;STOP CPU CLOCK
                                     0090  7061
                                     0090  7062 RTEMP_DATA
                                     0001  7063
                        00000000     0001  7064          .LONG   ^X00000000                      ;PHY/VIR REG ADDRESS
                        00000000     0005  7065          .LONG   ^X00000000                      ;DATA
                        0E000000     0009  7066          .LONG   ^X0E000000                      ;REF CACHE ONLY ADDRESS
                        00000000     000D  7067          .LONG   ^X00000000                      ;TURN CMI ON
```

```
06000000  0011  7068          .LONG   ^X06000000                      ;CACHE CONT REG ADDRESS
01000000  0015  7069          .LONG   ^X01000000                      ;TURN CACHE OFF
FFFFFFFF  0019  7070          .LONG   ^XFFFFFFFF                      ;TEST PATTERN
          001D  7071
          001D  7072 CACHE_DATA
          0001  7073
AAAAAAAA  0001  7074          .LONG   ^XAAAAAAAA                      ;CACHE ADD 0
          0005  7075
          0005  7076 END_CPU_DATA
          00AD  7077
          00AD  7078 END_TEST_DATA
          00AD
          00AD        ;-------------------------------------------------------------------------
          00AD  7079
          00AD  7080 ENDTEST
```

```
                        003A        .SBTTL  "TEST   090     Test Bus Field READ.LNG Micro Order using Main Memory
                        003A  7082 ;*******************************************************************************
                        003A  7083 ;++
                        003A  7084 ;
                        003A  7085 ; FUNCTIONAL DESCRIPTION:
                        003A  7086 ;
                        003A  7087 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE "BUS/READ.LNG"
                        003A  7088 ;       MICRO ORDER USING MAIN MEMORY.
                        003A  7089 ;
                        003A  7090 ;
                        003A  7091 ; TEST ALGORITHM:
                        003A  7092 ;
                        003A  7093 ;       1.  CREATE A TEST LOOP TO SELECT TEST PATTERNS
                        003A  7094 ;       2.  PUT SELECTED TEST PATTERN INTO RDM WD REGISTER
                        003A  7095 ;       3.  EXECUTE DCS PROGRAM
                        003A  7096 ;               a.  TURN MME OFF
                        003A  7097 ;               b.  TURN CMI ON
                        003A  7098 ;               c.  TURN CACHE OFF
                        003A  7099 ;               d.  PUT 3 INTO THE VA REGISTER
                        003A  7100 ;               e.  WRITE TEST PATTERN TO MAIN MEMORY
                        003A  7101 ;               f.  PERFORM BUS/READ.LNG
                        003A  7102 ;               g.  MOVE READ DATA TO RDM WH REGISTER
                        003A  7103 ;       4.  TEST CSTRP FOR ANY UNEXPECTED MICRO TRAPS
                        003A  7104 ;       5.  TEST WH REGISTER FOR CORRECT DATA
                        003A  7105 ;       6.  IF NO ERROR GO TO STEP 2 FOR REMAINING TEST PATTERNS
                        003A  7106 ;
                        003A  7107 ;
                        003A  7108 ; TEST PATTERNS:
                        003A  7109 ;
                        003A  7110 ;       ITERATION       PATTERN
                        003A  7111 ;
                        003A  7112 ;           1           AAAAAAAA
                        003A  7113 ;           2           55555555
                        003A  7114 ;           3           33333333
                        003A  7115 ;           4           0F0F0F0F
                        003A  7116 ;           5           00FF00FF
                        003A  7117 ;           6           0000FFFF
                        003A  7118 ;
                        003A  7119 ;
                        003A  7120 ; LOGIC DESCRIPTION:
                        003A  7121 ;
                        003A  7122 ;       THIS TEST INSURES THAT THE CPU CAN PERFORM A BUS/READ.LNG
                        003A  7123 ;       TO MAIN MEMORY.  BY USING VARIOUS TEST PATTERNS THE DATA INTEGRITY
                        003A  7124 ;       OF THE MDR GATE ARRAY AND CMI ARE ALSO TESTED.
                        003A  7125 ;
                        003A  7126 ;
                        003A  7127 ; ASSUMPTIONS:
                        003A  7128 ;
                        003A  7129 ;       THIS TEST ASSUMES THE CMI HAS BEEN TESTED FROM THE RDM
                        003A  7130 ;
                        003A  7131 ;
                        003A  7132 ; ERROR DESCRIPTION:
                        003A  7133 ;
                        003A  7134 ;       FIRST IFERROR:
                        003A  7135 ;               EXPECTED CONTENTS OF CSTRP REGISTER (NOT UNDR MICRO TRAP)
```

```
                       003A  7136 ;                RECEIVED CONTENTS OF CSTRP REGISTER
                       003A  7137 ;                LOOP COUNT I
                       003A  7138 ;
                       003A  7139 ;        SECOND IFERROR:
                       003A  7140 ;                EXPECTED CONTENTS OF CSTRP REGISTER (LAST DCS ADDRESS)
                       003A  7141 ;                RECEIVED CONTENTS OF CSTRP REGISTER
                       003A  7142 ;                LOOP COUNT I
                       003A  7143 ;
                       003A  7144 ;        THIRD IFERROR:
                       003A  7145 ;                EXPECTED CONTENTS OF WH REGISTER (TEST PATTERN)
                       003A  7146 ;                RECEIVED CONTENTS OF WH REGISTER
                       003A  7147 ;                LOOP COUNT I
                       003A  7148 ;
                       003A  7149 ;--
                       003A  7150 ;*******************************************************************************
                       003A  7151 ;///////////////////////////////////////////////////////////////////////////////
                       003A  7152
                       003A  7153          INITIALIZE                                      ;INIT CPU
                       003C  7154          EXPUTRAP                                        ;POSSIBLE FAILURE MODE
                       003E  7155          LOOP            I,1,6                           ;CREATE A TEST LOOP
                       0045  7156          SA01PAT         I,4$,,                          ;GENERATE A TEST PATTERN
                       004D  7157          LOADREG         WDREG,4$,,L                     ;TEST PATTERN TO WD REGISTER
                       0055  7158          ERRLOOP                                         ;ERROR LOOP
                       0057  7159          FETCH           0                               ;START DCS PROGRAM
                       005C  7160          BRSTCLK         <^X0B>                          ;END DCS PROGRAM
                       0060  7161          CMPREGMSK       CSTRP,1$,,2$,,W,NE              ;CHECK FOR UNDR U-TRAP
                       006C  7162          IFERROR         ,<<MDR><CML><CAK>>,
                       0074  7163          CMPREGMSK       CSTRP,3$,,2$,,W,                ;CHECK FOR UNEXPECTED U-TRAP
                       0080  7164          IFERROR         ,<<MDR><CML><CAK>>,<<MC0><MC1><MC2>>
                       008B  7165          CMPREG          WHREG,4$,,L,                    ;COMPARE RESULTS
                       0094  7166          IFERROR         ,<<MDR><CML><CAK>>,<<MC0><MC1><MC2>>
                       009F  7167          ENDLOOP         I                               ;END TEST LOOP
                       00A2  7168          SKIP
                       00A9  7169
                       00A9  7170 BEGIN_TEST_DATA
                       00A9
                       00A9                ;-------------------------------------------------------------------------
                       00A9  7171
                  0021 00A9  7172 1$:      .WORD   ^X0021                                  ;UNALIGNED DATA READ U-TRAP
                  3FFF 00AB  7173 2$:      .WORD   ^X3FFF                                  ;MASK FOR CMPREGMSK PSEUDO-OP
                  180B 00AD  7174 3$:      .WORD   ^X180B                                  ;ADD DCS SHOULD STOP AT
              00000000 00AF  7175 4$:      .LONG   ^X00000000                              ;TEST PATTERN STORAGE LOCATION
                       00B3  7176
                       00B3  7177 BEGIN_CPU_DATA
                       0002  7178
                       0002  7179 DCS_DATA
                       0001  7180
U 00, 7700,C800,0364,0300,0470,1801 0001  7181 ;% 00    NOP
U 01, 7700,C800,5BE4,0300,6870,1802 000C  7185 ;% 01    MEMSCAR_R[TEMP0]                ;PHY/VIR ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803 0017  7189 ;% 02    MEMSCR_R[TEMP1]                 ;SET MME OFF
U 03, 7700,8800,5BE4,0308,6870,1804 0022  7193 ;% 03    MEMSCAR_R[TEMP2]                ;REF CACHE ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805 002D  7197 ;% 04    MEMSCR_R[TEMP3]                 ;TURN CMI ON
U 05, 7700,0800,5BE4,0310,6870,1806 0038  7201 ;% 05    MEMSCAR_R[TEMP4]                ;CACHE CONT REG ADD TO MEMSCAR
U 06, 7700,4800,5BE4,0314,6070,1807 0043  7205 ;% 06    MEMSCR_R[TEMP5]                 ;TURN CACHE OFF
U 07, 7700,8800,5BE4,0304,4A70,1808 004E  7209 ;% 07    VA_R[TEMP1]                     ;3 TO VA REGISTER
```

```
U 08. 5700,4800,0364,0300,5480,1809  0059  7213 ;% 08   WB_RDM,WCTRL/WDR_WB.UR,WRITE.PHY          ;WRITE TEST PATTERN TO MEMORY
U 09. 7700,C800,0364,0300,0510,180A  0064  7217 ;% 09   READ.LONG                                 ;READ TEST PATTERN BACK TO CPU
U 0A. 6700,8812,5924,0300,0470,180B  006F  7221 ;% 0A   RDM_WB,WB_M[MDR]                          ;MOVE TEST PATTERN TO RDM
U 0B. 7300,4800,0364,0300,0470,180C  007A  7225 ;% 0B   NOP,STOP.CLOCK                           ;STOP CPU CLOCK
                                     0085  7229
                                     0085  7230 RTEMP_DATA
                                     0001  7231
                           00000000  0001  7232         .LONG   ^X00000000                     ;PHY/VIR REG ADDRESS
                           00000003  0005  7233         .LONG   ^X00000003                     ;DATA
                           0E000000  0009  7234         .LONG   ^X0E000000                     ;REF CACHE ONLY REG ADDRESS
                           00000000  000D  7235         .LONG   ^X00000000                     ;TURN CMI ON
                           06000000  0011  7236         .LONG   ^X06000000                     ;CACHE CONT REG ADDRESS
                           01000000  0015  7237         .LONG   ^X01000000                     ;TURN CACHE OFF
                                     0019  7238
                                     0019  7239 CACHE_DATA
                                     0001  7240
                           FFFFFFFF  0001  7241         .LONG   ^XFFFFFFFF                     ;CACHE ADD 0
                                     0005  7242
                                     0005  7243 END_CPU_DATA
                                     00B3  7244
                                     00B3  7245 END_TEST_DATA
                                     00B3
                                     00B3          ;-----------------------------------------------------------------------
                                     00B3  7246
                                     00B3  7247 ENDTEST
```

```
0038          .SBTTL  "TEST   091     Test READ.LNG.MOD Bus Micro Order using Main Memory
0038  7249 ;*******************************************************************************
0038  7250 ;++
0038  7251 ;
0038  7252 ; FUNCTIONAL DESCRIPTION:
0038  7253 ;
0038  7254 ;       THIS TEST WILL VERIFY THE CORRECT OPERATION OF THE "BUS/READ.LNG.MOD"
0038  7255 ;       MICRO ORDER USING MAIN MEMORY.
0038  7256 ;
0038  7257 ;
0038  7258 ; TEST ALGORITHM:
0038  7259 ;
0038  7260 ;       1.  LOAD PTE INTO DCS FOR THIS SUBTEST
0038  7261 ;       2.  EXECUTE DCS PROGRAM
0038  7262 ;               a.  TURN MME ON
0038  7263 ;               b.  TURN CMI ON
0038  7264 ;               c.  TURN CACHE OFF
0038  7265 ;               d.  SET CURRENT MODE TO KERNAL
0038  7266 ;               e.  PUT PTE IN LONGLIT REGISTER
0038  7267 ;               f.  LOAD 0'S INTO VA REGISTER
0038  7268 ;               g.  INVALIDATE TB LOCATION 0
0038  7269 ;               h.  MOVE PTE INTO TB
0038  7270 ;               i.  WRITE TEST PATTERN TO MAIN MEMORY ADDRESS 0
0038  7271 ;               j.  PERFORM BUS/READ.LNG.MOD
0038  7272 ;               k.  MOVE READ DATA TO RDM WH REGISTER
0038  7273 ;       3.  TEST CSTRP REGISTER FOR ACCESS VIOLATION, READ
0038  7274 ;           (SUBTEST #1 TESTS FOR NOT ACVR)
0038  7275 ;       4.  TEST CSTRP REGISTER FOR UNEXPECTED MICRO TRAPS (SUBTEST #1)
0038  7276 ;       5.  TEST WH REGISTER FOR CORRECT DATA (SUBTEST #1)
0038  7277 ;
0038  7278 ;
0038  7279 ; TEST PATTERNS:
0038  7280 ;
0038  7281 ;       SUBTEST         PTE             CSTRP           WHREG
0038  7282 ;          1          00000144          180F          FFFFFFFF
0038  7283 ;          2          00000134          002E            ??
0038  7284 ;
0038  7285 ;
0038  7286 ; LOGIC DESCRIPTION:
0038  7287 ;
0038  7288 ;       THIS TEST INSURES THAT THE ACV GATE ARRAY CAN DECODE AND EXECUTE
0038  7289 ;       A BUS/READ.LNG.MOD MICRO ORDER.  SUBTEST #1 HAS A PTE THAT ALLOWS
0038  7290 ;       WRITE ACCESS AND THEREFORE SHOULD NOT MICRO TRAP.  SUBTEST #2 HAS
0038  7291 ;       A PTE THAT DOES NOT ALLOW WRITE ACCESS AND SHOULD MICRO TRAP.
0038  7292 ;
0038  7293 ;
0038  7294 ; ASSUMPTIONS:
0038  7295 ;
0038  7296 ;       THIS TEST ASSUMES THAT THE CMI HAS BEEN TESTED FROM THE RDM
0038  7297 ;
0038  7298 ;
0038  7299 ; ERROR DESCRIPTION:
0038  7300 ;
0038  7301 ;       SUBTEST #1:
0038  7302 ;
```

```
                     0038  7303 ;                  FIRST IFERROR:
                     0038  7304 ;                          EXPECTED CSTRP REGISTER CONTENTS (NOT ACVR)
                     0038  7305 ;                          RECEIVED CSTRP REGISTER CONTENTS
                     0038  7306 ;
                     0038  7307 ;                  SECOND IFERROR:
                     0038  7308 ;                          EXPECTED
                     0038  7309 ;--
                     0038  7310 ;*******************************************************************************
                     0038  7311
                     0038  7312          SUBTEST                                         ;SUBTEST #1
                     003B
                     003B         ;////////////////////////////////////////////////////////////////////////////
                     003B  7313 ;+
                     003B  7314 ;SUBTEST #1
                     003B  7315 ;
                     003B  7316 ;       TEST "BUS/READ.LNG.MOD" USING MAIN MEMORY WITH NO MICRO TRAP
                     003B  7317 ;-
                     003B  7318          INITIALIZE                                      ;INIT CPU
                     003D  7319          LOADDCS         1$,,08,1                        ;MODIFY DCS ADDRESS
                     0044  7320          EXPUTRAP                                        ;POSSIBLE FAILURE MODE
                     0046  7321          ERRLOOP                                         ;ERROR LOOP
                     0048  7322          FETCH           0                               ;START DCS PROGRAM
                     004D  7323          BRSTCLK         <^X10>                          ;END DCS PROGRAM
                     0051  7324          CMPREGMSK       CSTRP,2$,,3$,,W,NE              ;CHECK FOR ACVR U-TRAP
                     005D  7325          IFERROR         ,<<ACV><CML>>,                  ;FAILING ARRAY AND MODULE
                     0064  7326          CMPREGMSK       CSTRP,4$,,3$,,W,                ;CHECK FOR UNEXPECTED U-TRAP
                     0070  7327          IFERROR         ,<<ACV><CML>>,<<MC0><MC1>>;FAILING ARRAY AND MODULE
                     0079  7328          CMPREG          WHREG,5$,,L,                    ;COMPARE RESULTS
                     0082  7329          IFERROR         ,<<CML><ACV>>,<<MC0><MC1><MC2>>;FAILING ARRAY AND MODULE
                     008C  7330
                     008C  7331          SUBTEST                                         ;SUBTEST #2
                     008F
                     008F         ;////////////////////////////////////////////////////////////////////////////
                     008F  7332 ;+
                     008F  7333 ;SUBTEST #2
                     008F  7334 ;
                     008F  7335 ;       TEST "BUS/READ.LNG.MOD" USING MAIN MEMORY WITH A MICRO TRAP
                     008F  7336 ;-
                     008F  7337          INITIALIZE                                      ;INIT CPU
                     0091  7338          LOADDCS         6$,,08,1                        ;MODIFY DCS ADDRESS
                     0098  7339          EXPUTRAP                                        ;EXPECTED UTRAP
                     009A  7340          ERRLOOP                                         ;ERROR LOOP
                     009C  7341          FETCH           0                               ;START DCS PROGRAM
                     00A1  7342          BRSTCLK         <^X10>                          ;END DCS PROGRAM
                     00A5  7343          CMPREGMSK       CSTRP,2$,,3$,,W,                ;COMPARE CS ADDRESS ACVR TRAP
                     00B1  7344          IFERROR         ,<<ACV><CML>>,                  ;FAILING ARRAY AND MODULE
                     00B8  7345          SKIP
                     00BF  7346
                     00BF  7347 BEGIN_TEST_DATA
                     00BF
                     00BF         ;-------------------------------------------------------------------------
                     00BF  7348
                     00BF  7349 1$:
U 08, 7700,7800,7FFF,FF5D,8470,1809  00BF  7350 ;* 08    LONLIT_[00000144]                ;DCS ADD 08 SUBTEST #1
                     00CA  7354
```

```
                          002E  00CA  7355 2$:     .WORD   ^X002E                          ;ADD FOR ACCESS VIOLATION READ
                          3FFF  00CC  7356 3$:     .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO-OP
                          1810  00CE  7357 4$:     .WORD   ^X1810                          ;ADD DCS PROGRAM SHOULD END AT
                      FFFFFFFF  00D0  7358 5$:     .LONG   ^XFFFFFFFF                      ;EXPECTED RESULTS
                                00D4  7359
                                00D4  7360 6$:
U 08, 7700,3800,7FFF,FF65,8470,1809   00D4  7361 ;* 08   LONLIT_[00000134]                 ;DCS ADD 08 SUBTEST #2
                                00DF  7365
                                00DF  7366 BEGIN_CPU_DATA
                                0002  7367
                                0002  7368 DCS_DATA
                                0001  7369
U 00, 7700,C800,0364,0300,0470,1801   0001  7370 ;* 00   NOP
U 01, 7700,C800,5BE4,0300,6870,1802   000C  7374 ;* 01   MEMSCAR_R[TEMP0]                  ;PHY/VIR REG ADD TO MEMSCAR
U 02, 7700,8800,5BE4,0304,6070,1803   0017  7378 ;* 02   MEMSCR_R[TEMP1]                   ;TURN MME ON
U 03, 7700,8800,5BE4,0308,6870,1804   0022  7382 ;* 03   MEMSCAR_R[TEMP2]                  ;REF CACHE REG ADD TO MEMSCAR
U 04, 7700,C800,5BE4,030C,6070,1805   002D  7386 ;* 04   MEMSCR_R[TEMP3]                   ;TURN CMI ON
U 05, 7700,0800,5BE4,0310,6870,1806   0038  7390 ;* 05   MEMSCAR_R[TEMP4]                  ;CACHE CONT ADD TO MEMSCAR
U 06, 7700,4800,5BE4,0314,6070,1807   0043  7394 ;* 06   MEMSCR_R[TEMP5]                   ;TURN CACHE OFF
U 07, 7700,0800,5BF4,0300,0070,1808   004E  7398 ;* 07   PSL_R[TEMP0]                      ;SET MODE TO KERNEL
U 08, 7700,F800,7FFF,FF5F,8470,1809   0059  7402 ;* 08   LONLIT_[00000140]               ;PTE TO LONGLIT REGISTER
U 09, 7700,0800,5BE6,0300,0470,180A   0064  7406 ;* 09   D_R[TEMP0]                       ;0'S TO D REGISTER
U 0A, 7700,5800,DB13,0300,4A70,180B   006F  7410 ;* 0A   VA_Q_D_D+ZLIT8[0]               ;VA GETS ADDRESS
U 0B, 7700,9800,DB12,0300,5360,180C   007A  7414 ;* 0B   VA_D_D+ZLIT8[0] INVALIDATE TB   ;INVALIDATE TB AT VA
U 0C, 7700,481B,5BE4,03D4,5070,180D   0085  7418 ;* 0C   TB_R[LONLIT]                     ;PTE TO TB
U 0D, 7700,4800,5BE4,0318,5C80,180E   0090  7422 ;* 0D   WRITE.PHY R[TEMP6]              ;WRITE TEST PATTERN TO MEMORY
U 0E, 7700,4800,0364,0300,0550,180F   009B  7426 ;* 0E   READ.LONG.MOD                   ;PERFORM READ LONG WITH MODIFY
U 0F, 6700,8812,5924,0300,0470,1810   00A6  7430 ;* 0F   RDM_WB,WB_M[MDR]               ;MOVE TEST PATTERN TO RDM
U 10, 7300,4800,0364,0300,0470,1811   00B1  7434 ;* 10   NOP,STOP.CLOCK                  ;STOP CPU CLOCK
                                00BC  7438
                                00BC  7439 RTEMP_DATA
                                0001  7440
                      00000000  0001  7441          .LONG   ^X00000000                  ;PHY/VIR REGISTER ADDRESS
                      01000000  0005  7442          .LONG   ^X01000000                  ;TURNS MME ON
                      0E000000  0009  7443          .LONG   ^X0E000000                  ;REF CACHE ONLY REG ADDRESS
                      00000000  000D  7444          .LONG   ^X00000000                  ;TURNS CMI ON
                      06000000  0011  7445          .LONG   ^X06000000                  ;CACHE CONT REG ADDRESS
                      01000000  0015  7446          .LONG   ^X01000000                  ;TURNS CACHE OFF
                      FFFFFFFF  0019  7447          .LONG   ^XFFFFFFFF                  ;TEST PATTERN
                                001D  7448
                                001D  7449 CACHE_DATA
                                0001  7450
                      AAAAAAAA  0001  7451          .LONG   ^XAAAAAAAA                  ;DATA FOR CACHE ADD 0
                                0005  7452
                                0005  7453 END_CPU_DATA
                                00DF  7454
                                00DF  7455 END_TEST_DATA
                                00DF
                                00DF      ;-------------------------------------------------------------------------
                                00DF  7456
                                00DF  7457 ENDTEST
```

```
0019          .SBTTL  "TEST   092     PC Increment on IRD1
0019  7459 ;*******************************************************************************
0019  7460 ;++
0019  7461 ;
0019  7462 ; FUNCTIONAL DESCRIPTION
0019  7463 ;
0019  7464 ;       This test will exaine the PC+2 increment on a BUT/IRD1TST.
0019  7465 ;
0019  7466 ; TEST ALGORITHM
0019  7467 ;
0019  7468 ;       1. Initialize the CPU.
0019  7469 ;       2. Expect addresses outside of DCS range.
0019  7470 ;       3. Load the WDREG with all 0.
0019  7471 ;       4. Execute micro-code to:
0019  7472 ;               1. Write the VA with the WDREG.
0019  7473 ;               2. Load a model instruction into the cache.
0019  7474 ;               3. Load the PC with the WDREG.
0019  7475 ;               4. Execute a BUT/IRD1TST.
0019  7476 ;               5. Halt.
0019  7477 ;       5. Execute more micro-code to:
0019  7478 ;               6. Read the value of the PC to the RDM WHREG.
0019  7479 ;               7. Halt.
0019  7480 ;       6. Compare the WHREG with the expected result of 2.
0019  7481 ;       7. If the two are not equal, signal an error.
0019  7482 ;       8. End of the test.
0019  7483 ;
0019  7484 ; TEST DATA
0019  7485 ;
0019  7486 ;       The only data needed for this test is the initial value of 0 for the
0019  7487 ;       VA and the PC.
0019  7488 ;
0019  7489 ; LOGICAL DESCRIPTION
0019  7490 ;
0019  7491 ;       On a BUT/IRD1, the PC will always be incremented by two.
0019  7492 ;
0019  7493 ; ASSUMPTIONS
0019  7494 ;
0019  7495 ;       This test assumes that the CACHE, XB0, XB1 the IR and the OSR have
0019  7496 ;       all been tested.
0019  7497 ;
0019  7498 ; ERROR DESCRIPTION
0019  7499 ;
0019  7500 ;       On error this test will type out the following information:
0019  7501 ;
0019  7502 ;       EXPECTED PC VALUE
0019  7503 ;       RECEIVED PC VALUE
0019  7504 ;
0019  7505 ;--
0019  7506 ;*******************************************************************************
0019  7507 ;///////////////////////////////////////////////////////////////////////////////
0019  7508
0019  7509          INITIALIZE                      ; INITIALIZE THE CPU
001B  7510          EXPUTRAP                        ; EXPECT ADDRESSES OUTSIDE OF DCS RANGE
001D  7511          LOADREG         WDREG,1$,,L     ; LOAD A ZERO INTO THE WDREG
0025  7512          ERRLOOP                         ; LOOP ON ERROR FROM HERE
```

```
                                0027  7513          FETCH           0                       ; EXECUTE THE MICRO-CODE FOR AN IRD1
                                002C  7514          BRSTCLK         <^X0B>,INH              ; ...
                                0030  7515          FETCH           <^X0C>                  ; EXECUTE THE MICRO-CODE TO READ THE PC
                                0035  7516          BRSTCLK         <^X0E>,INH              ; ...
                                0039  7517          CMPREG          WHREG,2$,,L             ; DOES THE WHREG = 2?
                                0042  7518          IFERROR         ,,<DPM>                 ; IF NOT SIGNAL AN ERROR
                                0048  7519          SKIP                                    ; END OF TEST
                                004F  7520
                                004F  7521 BEGIN_TEST_DATA
                                004F
                                004F       ;-------------------------------------------------------------------------
                                004F  7522
                    00000000    004F  7523 1$:      .LONG   0
                    00000002    0053  7524 2$:      .LONG   ^X 00000002
                                0057  7525
                                0057  7526 BEGIN_CPU_DATA
                                0002  7527
                                0002  7528 DCS_DATA
                                0001  7529
U 00, 7700,C800,0364,0300,0470,1801   0001  7530 ;% 00    NOP                             ; LET THE CPU GET UP TO SPEED
U 01, 5700,C800,0364,0300,4A70,1802   000C  7534 ;% 01    VA_RDM                          ; LOAD THE VA WITH ZERO
U 02, 7700,F800,57D7,D7EF,0470,1803   0017  7538 ;% 02    LONLIT_[50505021]               ; LOAD A MODEL INSTRUCTION INTO LONLIT
U 03, 7701,8800,5BE4,02D4,5590,1804   0022  7542 ;% 03    WRITE.LONG R[LONLIT]            ; WRITE LONLIT TO CACHE
U 04, 7700,4800,5BE4,03D4,5470,1805   002D  7546 ;% 04    WDR_UNROT(R[LONLIT])            ; ...
U 05, 7700,C800,0364,0300,4470,1806   0038  7550 ;% 05    VA_VA+4                         ; INCREMENT THE VA
U 06, 7700,7800,57D7,D7D7,8470,1807   0043  7554 ;% 06    LONLIT_[50505050]               ; WRITE THE REST OF THE INSTRUCTION TO
U 07, 7701,8800,5BE4,02D4,5590,1808   004E  7558 ;% 07    WRITE.LONG R[LONLIT]            ; CACHE
U 08, 7700,4800,5BE4,03D4,5470,1809   0059  7562 ;% 08    WDR_UNROT(R[LONLIT])            ; ...
U 09, 5700,C800,0364,0300,4870,180A   0064  7566 ;% 09    PC_RDM                          ; START INSTRUCTION PREFETCH
U 0A, 7700,C800,0364,1700,0470,180B   006F  7570 ;% 0A    BUT/IRD1TST                     ; EXECUTE THE IRD1TST
U 0B, 7300,4800,0364,0300,0470,180C   007A  7574 ;% 0B    STOP.CLOCK                      ; HALT
U 0C, 7700,C800,0364,0300,0470,180D   0085  7578 ;% 0C    NOP                             ; LET THE CPU GET UP TO SPEED
U 0D, 6700,C81A,0024,03D8,0470,180E   0090  7582 ;% 0D    RDM_PC                          ; READ BACK THE PC
U 0E, 7300,4800,0364,0300,0470,180F   009B  7586 ;% 0E    STOP.CLOCK                      ; HALT
                                00A6  7590
                                00A6  7591 END_CPU_DATA
                                0057  7592
                                0057  7593 END_TEST_DATA
                                0057
                                0057       ;-------------------------------------------------------------------------
                                0057  7594 ENDTEST
```

```
0014          .SBTTL  "TEST   093     Test BUT/WCSENA
0014  7596  ;*******************************************************************
0014  7597  ;++
0014  7598  ;
0014  7599  ; FUNCTIONAL DESCRIPTION:
0014  7600  ;
0014  7601  ;       THIS TEST CHECKS FOR WCS PRESENT VIA THE DPM BRANCHING LOGIC.
0014  7602  ;
0014  7603  ;
0014  7604  ; TEST ALGORITHM:
0014  7605  ;
0014  7606  ;       1.  EXECUTE DCS PROGRAM
0014  7607  ;               a.  ENABLE THE CMI
0014  7608  ;               b.  PUT ADDRESS OF WCS INTO VA REGISTER
0014  7609  ;               c.  WRITE BIT 20 TO THE WCS
0014  7610  ;               d.  ALLOW TIME FOR WRITE TO COMPLETE
0014  7611  ;               e.  BRANCH ON WCS PRESENT
0014  7612  ;       2.  TEST CS ADDRESS FOR WCS PRESENT AND PRINT MESSAGE ACCORDINGLY
0014  7613  ;
0014  7614  ;
0014  7615  ; TEST PATTERNS:
0014  7616  ;
0014  7617  ;       NONE
0014  7618  ;
0014  7619  ;
0014  7620  ; LOGIC DESCRIPTION:
0014  7621  ;
0014  7622  ;       THIS TEST USES THE BUT LOGIC ON DPM TO DETERMINE IF THE WCS IS
0014  7623  ;       PRESENT.  IT IS UP TO THE USER TO DETERMINE IF THE PRINTED MESSAGE
0014  7624  ;       IS CORRECT.  IF THE MESSAGE IS INCORRECT THEN EITHER THE WCS ENABLE
0014  7625  ;       FLIP FLOP OR THE DPM BRANCHING LOGIC ARE AT FALUT.
0014  7626  ;
0014  7627  ;
0014  7628  ; ASSUMPTIONS:
0014  7629  ;
0014  7630  ;       THIS TEST ASSUMES THE CMI IS OPERATIONAL
0014  7631  ;
0014  7632  ;
0014  7633  ; ERROR DESCRIPTION:
0014  7634  ;
0014  7635  ;       WCS INSTALLED
0014  7636  ; or
0014  7637  ;       WCS NOT INSTALLED
0014  7638  ; or
0014  7639  ;       EXPECTED CS ADDRESS
0014  7640  ;       RECEIVED CS ADDRESS
0014  7641  ;
0014  7642  ;--
0014  7643  ;*******************************************************************
0014  7644
0014  7645
0014  7646          INITIALIZE                              ;INIT THE CPU
0016  7647          ERRLOOP                                 ;ERROR LOOP POINT
0018  7648          FETCH           0                       ;START DCS PROGRAM
001D  7649          BRSTCLK         <^X09>                  ;END DCS PROGRAM
```

```
                                0021  7650          CMPREGMSK       CSTRP,1$,,3$,,W,        ;TEST CS ADDRESS
                                002D  7651          SKIP            10$,ONERROR             ;NO WCS
                                0034  7652          PRINT_S         4$                      ;WCS PRESENT
                                0038  7653          SKIP                                    ;GO TO NEXT TEST
                                003F  7654 10$:     CMPREGMSK       CSTRP,2$,,3$,,W,        ;TEST CS ADDRESS
                                004B  7655          SKIP            20$,ONERROR             ;INVALID ADDRESS
                                0052  7656          PRINT_S         5$                      ;NO WCS
                                0056  7657          SKIP                                    ;GO TO NEXT TEST
                                005D  7658 20$:     IFERROR                                 ;FAILURE IN BRANCH LOGIC
                                0062  7659          SKIP                                    ;GO TO NEXT TEST
                                0069  7660
                                0069  7661
                                0069  7662 BEGIN_TEST_DATA
                                0069  7663          ;-------------------------------------------------------------------------
                          1800  0069  7664 1$:       .WORD   ^X1800                          ;WCS PRESENT
                          1801  006B  7665 2$:       .WORD   ^X1801                          ;NO WCS
                          3FFF  006D  7666 3$:       .WORD   ^X3FFF                          ;MASK FOR CMPREGMSK PSEUDO OP
41 54 53 4E 49 20 53 43 57 0A 0D 00'  006F  7667 4$:       .ASCIC  <13><10>/WCS INSTALLED/<13><10>
                0A 0D 44 45 4C 4C     007B
                                11    006F
49 20 54 4F 4E 20 53 43 57 0A 0D 00'  0081  7668 5$:       .ASCIC  <13><10>/WCS NOT INSTALLED/<13><10>
      0A 0D 44 45 4C 4C 41 54 53 4E   008D
                                15    0081
                                0097  7669
                                0097  7670
                                0097  7671 BEGIN_CPU_DATA
                                0002  7672
                                0002  7673 DCS_DATA
                                0001  7674
U 00, 7700,C800,0364,0300,0470,1801   0001  7675 ;% 00    NOP
U 01, 7700,8800,5BE4,0308,6870,1802   000C  7679 ;% 01    MEMSCAR_R[TEMP2]                ;ENABLE THE CMI
U 02, 7700,C800,5BE4,03D8,6070,1803   0017  7683 ;% 02    MEMSCR_R[ZERO]                  ;...
U 03, 7700,C800,5BE4,0300,4A70,1804   0022  7687 ;% 03    VA_R[TEMP0]                     ;ADDRESS OF WCS
U 04, 7700,8800,5BE4,0304,5C80,1805   002D  7691 ;% 04    WRITE.PHY R[TEMP1]              ;ENABLE BIT TO WCS
U 05, 7700,4800,0364,0300,0470,1806   0038  7695 ;% 05    NOP                             ;ALLOW WRITE TO COMPLETE
U 06, 7700,C800,0364,0300,0470,1807   0043  7699 ;% 06    NOP                             ;...
U 07, 7700,C800,0364,0300,0470,1808   004E  7703 ;% 07    NOP                             ;...
U 08, 7700,4800,0364,9F00,0470,1800   0059  7707 ;% 08    BUT/WCSENA,NEXT/1800            ;BRANCH ON WCS
U 09, 7300,4800,0364,0300,0470,180A   0064  7711 ;% 09    NOP,STOP.CLOCK                  ;STOP THE CPU CLOCK
                                006F  7715
                                006F  7716 RTEMP_DATA
                                0001  7717
                    00F00000    0001  7718          .LONG   ^X00F00000                      ;ADDRESS OF WCS
                    00100000    0005  7719          .LONG   ^X00100000                      ;WCS ENABLE BIT
                    0E000000    0009  7720          .LONG   ^X0E000000                      ;ENABLE CMI STATUS REGISTER
                                000D  7721
                                000D  7722 END_CPU_DATA
                                0097  7723
                                0097  7724
                                0097  7725 END_TEST_DATA
                                0097          ;-------------------------------------------------------------------------
                                0097  7726 ENDTEST
```

```
                0002  7729 .END
```

```
$CONTROL_C_CODE= 0000000A          CONT_FLAG       = 00000020          END_TEST_146     00000057 R     63
$FILE_NOT_FOUND= 00000009          CONT_KEY        = 0000001C          END_TEST_147     00000097 R     69
$STN             = 00000000        CPU_RUN         = 00000010          EN_TRACE_REG    = 00000008
$TEST_NUMBER     = 00000094        CSAMR           = 00007404          ERRLOG_FLAG     = 00000000
ACV              = 0000000F        CSBUF           = 00007424          ERROR_FLAG      = 00000010
AC_LOW           = 00000080        CSTRP           = 00007422          ERROR_LOG_ADR   = 000032FC
ADD              = 00000019        CS_ADD_BUFF_HGH= 00007425           ERR_LOG_INIT    = 00000001
ADDR_MATCH_HI    = 00007405        CS_ADD_BUFF_LOW= 00007424           EXP_STALL_FLAG  = 00000010
ADDR_MATCH_LO    = 00007404        CS_ADD_TRAP_HGH= 00007423           EXP_UTRAP_FLAG  = 00000040
ADK              = 0000000B        CS_ADD_TRAP_LOW= 00007422           FAC             = 00000020
ALK              = 00000000        CYCLE_FLAG      = 00000008          FAULT           = 00000002
ALP              = 00000016        CYCLE_KEY       = 00000020          FCC             = 00000015
ARG_CNT          = 0000005E R      69 D0           = 00000003          FCS             = 00000021
A_REG_BYTE_0     = 00007410        D1              = 00000009          FETCH_FLAG      = 00000008
A_REG_BYTE_1     = 00007411        D2              = 00000000          FEX             = 0000001E
A_REG_BYTE_2     = 00007412        DATA_FLAG       = 00000000          FFH             = 00000024
A_REG_BYTE_3     = 00007413        DATA_LENGTH     = 00000001          FFL             = 00000023
B                = 00000001        DCSAD           = 00007401          FIC             = 0000001F
BADD1            = 00007426        DCSCF           = 00007403          FIO             = 0000001D
BELL_FLAG        = 00000008        DCSCR           = 00007402          FLAG_KEY        = 00000004
BELL_KEY         = 0000000E        DCSDA           = 00007400          FMR             = 00000022
BUFF_ADDR_LOW    = 00007426        DCS_ADDR_CLEAR  = 00000002          FPA             = 00000002
BURST_STOP_FLAG= 00000080          DCS_ADDR_REG_RO= 00007427           FP_BOOT_0       = 00000001
BYTE             = 00000001        DCS_ADDR_REG_WO= 00007401           FP_BOOT_1       = 00000002
CACHE_FLAG       = 00000000        DCS_CTRL_FILE   = 00007403          FP_START_0      = 00000004
CACHE_FLAG1      = 00000000        DCS_CTRL_REG    = 00007402          FP_START_1      = 00000008
CACHE_HEAD       = 00000000 R      60 DCS_DATA_REG = 00007400          FQA             = 00000014
CACHE_LENGTH     = 00000001        DCS_FLAG        = 00000000          FRONT1          = 00007420
CAK              = 0000000A        DCS_FLAG1       = 00000000          FRONT2          = 00007421
CCC              = 00000004        DCS_HEAD        = 00000000 R      6A FRONT_PNL_1     = 00007420
CCS              = 00000005        DCS_LENGTH      = 00000001          FRONT_PNL_2     = 00007421
CF_KEY           = 00000024        DCS_SCRATCH_ADR= 00000036           FRONT_PNL_LOCK  = 00000080
CLA              = 00000003        DCS_START       = 00001800          HALT_FLAG       = 00000001
CLEAR_CTRL_FILE= 00000002          DC_LOW          = 00000040          HALT_KEY        = 00000008
CLKDCS           = 00007427        DIAGNOSE_FLAG   = 00000004          HALT_ON_MATCH   = 00000001
CLK_CTRL_0       = 00000020        DPM             = 00000000          HEAD            = FFFF7C00
CLK_CTRL_0_RO    = 00000040        DUM1            = FFFFFFF3          HIGH            = 00000001
CLK_CTRL_1       = 00000040        ENDTEST_LOC     = 0000009C R      69 H_FROM_WBUS     = 00000010
CLK_CTRL_1_RO    = 00000080        END_SLICE       = 00000025          IB_FLAG         = 00000020
CLOCK_STOPED     = 00000080        END_TEST_130      00000673 R      02 IB_KEY          = 00000012
CMI              = 00000007        END_TEST_131      0000065B R      09 INSTR_FLAG      = 00000004
CMICTL           = 0000741C        END_TEST_132      00000637 R      0F INSTR_KEY       = 00000018
CMI_COMPLETE     = 00000020        END_TEST_133      00000652 R      15 INT             = 00000010
CMI_CTRL_CLEAR   = 00000001        END_TEST_134      0000068A R      1B IRD             = 00000005
CMI_CTRL_REG     = 0000741C        END_TEST_135      00000672 R      21 I_INDEX_FLAG    = 00000000
CMI_GO           = 00000008        END_TEST_136      00000682 R      27 I_INIT          = 00000000
CMI_READ         = 00000040        END_TEST_137      0000069D R      2D J_INDEX_FLAG    = 00000000
CMI_STATUS_0     = 00000008        END_TEST_138      000000F8 R      33 J_INIT          = 00000000
CMI_STATUS_1     = 00000010        END_TEST_139      000000D4 R      39 K_INDEX_FLAG    = 00000000
CMI_WRITE        = 00000020        END_TEST_140      00000129 R      3F K_INIT          = 00000000
CML              = 0000000E        END_TEST_141      0000030E R      45 L               = 00000004
COMPOSIT_LENGTH= 00000003          END_TEST_142      000001A6 R      4B LIST_END        = 00000062 R     69
CON              = 00000011        END_TEST_143      000000AD R      51 LIST_HEAD       = 0000005F R     69
CONTINUE_FLAG    = 00000002        END_TEST_144      000000B3 R      57 LIST_LENGTH     = 00000003
CONTROL_C_FLAG   = 00000001        END_TEST_145      000000DF R      5D 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$TEST_SIZE     = 000007A2
M$TEST_SIZE_D   = 000003E2
MA0             = 00000008
MA1             = 0000000A
MA2             = 0000000C
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
MC2             = 0000000B
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
MEC             = 0000001A
MHREG           = 00007418
MH_REG_BYTE_0   = 00007418
MH_REG_BYTE_1   = 00007419
MH_REG_BYTE_2   = 0000741A
MH_REG_BYTE_3   = 0000741B
MIC             = 00000001
MICROWORD       = 0000000A
MICRO_ADDR_INH  = 00000080
MONITOR_SIZE    = 00002C5E
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      6E
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_HEAD      = 00000000 R      6B
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            = 00000063
TEMP1           = 0000009D
TEMP2           = 0000007D
TEMP3           = 00000002
TEMP4           = 00000002 R      68
TEST            = 00000004
TEST_FLAG       = 00000000
TEST_HEAD       = 00000011 R      69
TEST_KEY        = 00000000
TEST_LENGTH     = 00000002
TEST_SECT_HEAD  = 00000000 R      69
THIS_TEST       = 00000093
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 synopsis

| PSECT name | Allocation | PSECT No. | Attributes |
|---|---|---|---|
| . ABS . | 00000000 ( 0.) | 00 ( 0.) | NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE |
| X_0130_D | 00000002 ( 2.) | 01 ( 1.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0130_T | 00000680 ( 1664.) | 02 ( 2.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0130_ERTEMP | 00000039 ( 57.) | 03 ( 3.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0130_FCACHE | 00000001 ( 1.) | 04 ( 4.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0130_GDCS | 00000001 ( 1.) | 05 ( 5.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0130_HFILL | 00000043 ( 67.) | 06 ( 6.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| DIRECTORY | 00000075 ( 117.) | 07 ( 7.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0131_D | 00000002 ( 2.) | 08 ( 8.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0131_T | 00000680 ( 1664.) | 09 ( 9.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0131_ERTEMP | 00000039 ( 57.) | 0A ( 10.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0131_FCACHE | 00000001 ( 1.) | 0B ( 11.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0131_GDCS | 00000001 ( 1.) | 0C ( 12.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0131_HFILL | 00000043 ( 67.) | 0D ( 13.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0132_D | 00000002 ( 2.) | 0E ( 14.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0132_T | 00000680 ( 1664.) | 0F ( 15.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0132_ERTEMP | 00000039 ( 57.) | 10 ( 16.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0132_FCACHE | 00000001 ( 1.) | 11 ( 17.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0132_GDCS | 00000001 ( 1.) | 12 ( 18.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0132_HFILL | 00000043 ( 67.) | 13 ( 19.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0133_D | 00000002 ( 2.) | 14 ( 20.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0133_T | 00000680 ( 1664.) | 15 ( 21.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0133_ERTEMP | 00000039 ( 57.) | 16 ( 22.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0133_FCACHE | 00000001 ( 1.) | 17 ( 23.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0133_GDCS | 00000001 ( 1.) | 18 ( 24.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0133_HFILL | 00000043 ( 67.) | 19 ( 25.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0134_D | 00000002 ( 2.) | 1A ( 26.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0134_T | 00000700 ( 1792.) | 1B ( 27.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0134_ERTEMP | 00000035 ( 53.) | 1C ( 28.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0134_FCACHE | 00000001 ( 1.) | 1D ( 29.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0134_GDCS | 00000001 ( 1.) | 1E ( 30.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0134_HFILL | 00000047 ( 71.) | 1F ( 31.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0135_D | 00000002 ( 2.) | 20 ( 32.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0135_T | 00000680 ( 1664.) | 21 ( 33.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0135_ERTEMP | 00000035 ( 53.) | 22 ( 34.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0135_FCACHE | 00000001 ( 1.) | 23 ( 35.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0135_GDCS | 00000001 ( 1.) | 24 ( 36.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0135_HFILL | 00000047 ( 71.) | 25 ( 37.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0136_D | 00000002 ( 2.) | 26 ( 38.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0136_T | 00000700 ( 1792.) | 27 ( 39.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0136_ERTEMP | 00000035 ( 53.) | 28 ( 40.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0136_FCACHE | 00000001 ( 1.) | 29 ( 41.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0136_GDCS | 00000001 ( 1.) | 2A ( 42.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0136_HFILL | 00000047 ( 71.) | 2B ( 43.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0137_D | 00000002 ( 2.) | 2C ( 44.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0137_T | 00000700 ( 1792.) | 2D ( 45.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0137_ERTEMP | 00000035 ( 53.) | 2E ( 46.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |
| X_0137_FCACHE | 00000001 ( 1.) | 2F ( 47.) | NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X_0137_GDCS | 00000001 | ( 1.) | 30 | ( 48.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0137_HFILL | 00000047 | ( 71.) | 31 | ( 49.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0138_D | 00000002 | ( 2.) | 32 | ( 50.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0138_T | 00000100 | ( 256.) | 33 | ( 51.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0138_GDCS | 00000017 | ( 23.) | 34 | ( 52.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0138_ERTEMP | 00000001 | ( 1.) | 35 | ( 53.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0138_FCACHE | 00000001 | ( 1.) | 36 | ( 54.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0138_HFILL | 00000065 | ( 101.) | 37 | ( 55.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0139_D | 00000002 | ( 2.) | 38 | ( 56.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0139_T | 00000100 | ( 256.) | 39 | ( 57.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0139_GDCS | 0000007A | ( 122.) | 3A | ( 58.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0139_ERTEMP | 00000015 | ( 21.) | 3B | ( 59.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0139_FCACHE | 00000001 | ( 1.) | 3C | ( 60.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0139_HFILL | 0000006E | ( 110.) | 3D | ( 61.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0140_D | 00000002 | ( 2.) | 3E | ( 62.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0140_T | 00000180 | ( 384.) | 3F | ( 63.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0140_GDCS | 00000090 | ( 144.) | 40 | ( 64.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0140_ERTEMP | 00000015 | ( 21.) | 41 | ( 65.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0140_FCACHE | 00000001 | ( 1.) | 42 | ( 66.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0140_HFILL | 00000058 | ( 88.) | 43 | ( 67.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0141_D | 00000002 | ( 2.) | 44 | ( 68.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0141_T | 00000380 | ( 896.) | 45 | ( 69.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0141_ERTEMP | 0000001D | ( 29.) | 46 | ( 70.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0141_FCACHE | 00000001 | ( 1.) | 47 | ( 71.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0141_GDCS | 00000001 | ( 1.) | 48 | ( 72.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0141_HFILL | 0000005F | ( 95.) | 49 | ( 73.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0142_D | 00000002 | ( 2.) | 4A | ( 74.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0142_T | 00000200 | ( 512.) | 4B | ( 75.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0142_ERTEMP | 0000000D | ( 13.) | 4C | ( 76.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0142_FCACHE | 00000001 | ( 1.) | 4D | ( 77.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0142_GDCS | 00000001 | ( 1.) | 4E | ( 78.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0142_HFILL | 0000006F | ( 111.) | 4F | ( 79.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0143_D | 00000002 | ( 2.) | 50 | ( 80.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0143_T | 00000100 | ( 256.) | 51 | ( 81.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0143_GDCS | 00000090 | ( 144.) | 52 | ( 82.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0143_ERTEMP | 0000001D | ( 29.) | 53 | ( 83.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0143_FCACHE | 00000005 | ( 5.) | 54 | ( 84.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0143_HFILL | 0000004C | ( 76.) | 55 | ( 85.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0144_D | 00000002 | ( 2.) | 56 | ( 86.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0144_T | 00000100 | ( 256.) | 57 | ( 87.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0144_GDCS | 00000085 | ( 133.) | 58 | ( 88.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0144_ERTEMP | 00000019 | ( 25.) | 59 | ( 89.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0144_FCACHE | 00000005 | ( 5.) | 5A | ( 90.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0144_HFILL | 0000005B | ( 91.) | 5B | ( 91.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0145_D | 00000002 | ( 2.) | 5C | ( 92.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0145_T | 00000100 | ( 256.) | 5D | ( 93.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0145_GDCS | 000000BC | ( 188.) | 5E | ( 94.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0145_ERTEMP | 0000001D | ( 29.) | 5F | ( 95.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0145_FCACHE | 00000005 | ( 5.) | 60 | ( 96.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0145_HFILL | 00000020 | ( 32.) | 61 | ( 97.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0146_D | 00000002 | ( 2.) | 62 | ( 98.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0146_T | 00000080 | ( 128.) | 63 | ( 99.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0146_GDCS | 000000A6 | ( 166.) | 64 | (100.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0146_ERTEMP | 00000001 | ( 1.) | 65 | (101.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |
| X_0146_FCACHE | 00000001 | ( 1.) | 66 | (102.) | NOPIC | USR | CON | REL | LCL | NOSHR | EXE | RD | WRT NOVEC BYTE |

```
X_0146_HFILL                    00000056  (    86.)  67 (103.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0147_D                        00000002  (     2.)  68 (104.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0147_T                        00000100  (   256.)  69 (105.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0147_GDCS                     0000006F  (   111.)  6A (106.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0147_ERTEMP                   0000000D  (    13.)  6B (107.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0147_FCACHE                   00000001  (     1.)  6C (108.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0147_HFILL                    00000001  (     1.)  6D (109.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
X_0148_D                        00000000  (     0.)  6E (110.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
A_DIRECTORY                     00000002  (     2.)  6F (111.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
```

## Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 108 | 00:00:00.36 | 00:00:02.66 |
| Command processing | 874 | 00:00:00.92 | 00:00:04.65 |
| Pass 1 | 2864 | 00:01:52.02 | 00:04:16.07 |
| Symbol table sort | 0 | 00:00:01.05 | 00:00:01.52 |
| Pass 2 | 37 | 00:00:29.49 | 00:00:59.85 |
| Symbol table output | 2 | 00:00:00.19 | 00:00:00.40 |
| Psect synopsis output | 2 | 00:00:00.38 | 00:00:00.80 |
| Cross-reference output | 0 | 00:00:00.00 | 00:00:00.00 |
| Assembler run totals | 3888 | 00:02:24.42 | 00:05:25.99 |

```
The working set limit was 6150 pages.
1255025 bytes (2452 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 348 non-local and 462 local symbols.
8447 source lines were read in Pass 1, producing 274 object records in Pass 2.
105 pages of virtual memory were used to define 49 macros.
```

## Macro library statistics

| Macro library name | Macros defined |
|---|---|
| DISK$UCODPACK00:[VAX750]COMETMAC.MLB;1 | 49 |
| SYS$COMMON:[SYSLIB]STARLET.MLB;1 | 0 |
| TOTALS (all libraries) | 49 |

```
2105 GETS were required to define 49 macros.

There were 0 errors, 8 warnings and 0 information messages, on lines:
  719 (1)      1432 (1)      2154 (1)      2881 (1)      3610 (1)
 4332 (1)      5057 (1)      5790 (1)

MACRO UCOD$0:[VAX750.MIC]TITLE+DISK$UCODPACK00:[VAX750]COMETASM+UCOD$0:[VAX750.MIC]ECKAC4/LIS=ECKAC4/OBJ=ECKAC4
```

| Fiche | Frame | Sequence | |
|---|---|---|---|
| 2 | M8 | 309 | V08.00 "TEST 03F TB GRO17-FEB-1986 |
| 2 | C9 | 312 | V08.00 "TEST 040 TB GRO17-FEB-1986 |
| 2 | G9 | 316 | V08.00 "TEST 041 TB GRO17-FEB-1986 |
| 2 | K9 | 320 | V08.00 "TEST 042 TB GRO17-FEB-1986 |
| 2 | C10 | 325 | V08.00 "TEST 043 TB GRO17-FEB-1986 |
| 2 | H10 | 330 | V08.00 "TEST 044 TB GRO17-FEB-1986 |
| 2 | L10 | 334 | V08.00 "TEST 045 TB GRO17-FEB-1986 |
| 2 | C11 | 338 | V08.00 "TEST 046 TB INV17-FEB-1986 |
| 2 | F11 | 341 | V08.00 "TEST 047 TB INV17-FEB-1986 |
| 2 | H11 | 343 | V08.00 "TEST 048 TB GRO17-FEB-1986 |
| 2 | L11 | 347 | V08.00 "TEST 049 TB GRO17-FEB-1986 |
| 2 | C12 | 351 | V08.00 "TEST 04A TEST T17-FEB-1986 |
| 2 | F12 | 354 | V08.00 "TEST 04B TEST T17-FEB-1986 |
| 2 | I12 | 357 | V08.00 "TEST 04C TEST T17-FEB-1986 |
| 2 | L12 | 360 | V08.00 "TEST 04D TEST T17-FEB-1986 |
| 2 | B13 | 363 | V08.00 "TEST 04E TEST F17-FEB-1986 |
| 2 | E13 | 366 | V08.00 "TEST 04F VA_PC+17-FEB-1986 |
| 2 | J13 | 371 | V08.00 "TEST 050 TEST X17-FEB-1986 |
| 2 | N13 | 375 | V08.00 "TEST 051 PREFET17-FEB-1986 |
| 2 | G14 | 381 | V08.00 "TEST 052 Test P17-FEB-1986 |
| 2 | M14 | 387 | V08.00 "TEST 053 TEST E17-FEB-1986 |
| 2 | E15 | 392 | V08.00 "TEST 054 TEST F17-FEB-1986 |
| 2 | H15 | 395 | V08.00 "TEST 055 TEST F17-FEB-1986 |
| 2 | K15 | 398 | V08.00 "TEST 056 TEST X17-FEB-1986 |
| 2 | B16 | 402 | V08.00 "TEST 057 TEST X17-FEB-1986 |
| 2 | F16 | 406 | V08.00 "TEST 058 TEST X17-FEB-1986 |
| 2 | I16 | 409 | V08.00 "TEST 059 TEST X17-FEB-1986 |
| 2 | L16 | 412 | V08.00 "TEST 05A TEST F17-FEB-1986 |
| 3 | D1 | 415 | V08.00 "TEST 05B TEST F17-FEB-1986 |
| 3 | H1 | 419 | V08.00 "TEST 05C TEST X17-FEB-1986 |
| 3 | K1 | 422 | V08.00 "TEST 05D TEST X17-FEB-1986 |
| 3 | N1 | 425 | V08.00 "TEST 05E TEST D17-FEB-1986 |
| 3 | E2 | 429 | Symbol table 17-FEB-1986 |
| 3 | H2 | 432 | Psect synopsis 17-FEB-1986 |
| 3 | M2 | 437 | VAX-11 Macro Run Statistics 17-FEB-1986 |
| 3 | N2 | 438 | Table of contents 3-MAR-1986 |
| 3 | B3 | 439 | V08.00 3-MAR-1986 |
| 3 | C3 | 440 | V08.00 "MIC Revision Hi 3-MAR-1986 |
| 3 | G3 | 444 | V08.00 "DIAGNOSTIC MICR 3-MAR-1986 |
| 3 | I3 | 446 | V08.00 " Diagnostic Mac 3-MAR-1986 |
| 3 | J3 | 447 | 0000 166 3-MAR-1986 |
| 3 | K3 | 448 | V08.00 " Diagnostic Mac 3-MAR-1986 |
| 3 | L3 | 449 | V08.00 " L0001 MULTIPLI 3-MAR-1986 |
| 3 | N3 | 451 | V08.00 " L0002 GATE ARR 3-MAR-1986 |
| 3 | C4 | 453 | V08.00 " L0003 GATE ARR 3-MAR-1986 |
| 3 | E4 | 455 | V08.00 " L0004 GATE ARR 3-MAR-1986 |
| 3 | G4 | 457 | V08.00 " L0011 & L0016 3-MAR-1986 |
| 3 | I4 | 459 | V08.00 "" 3-MAR-1986 |
| 3 | J4 | 460 | V08.00 "EQUATED SYMBOLS 3-MAR-1986 |
| 3 | M4 | 463 | V08.00 "TEST 05F TEST X 3-MAR-1986 |
| 3 | C5 | 466 | V08.00 "TEST 060 TEST X 3-MAR-1986 |
| 3 | F5 | 469 | V08.00 "TEST 061 TEST I 3-MAR-1986 |
| 3 | K5 | 474 | V08.00 "TEST 062 COMPAT 3-MAR-1986 |
| 3 | N5 | 477 | V08.00 "TEST 063 MEMORY 3-MAR-1986 |
| 3 | D6 | 480 | V08.00 "TEST 064 PAGE B 3-MAR-1986 |
| 3 | J6 | 486 | V08.00 "TEST 065 UNALIG 3-MAR-1986 |
| 3 | N6 | 490 | V08.00 "TEST 066 UNALIG 3-MAR-1986 |