

KD11-Z

11/44 UBI MAP
CKKUAA0

AH-F629A-MC

COPYRIGHT 1980

FICHE 1 OF 1

JAN 1980

digital

MADE IN USA

IDENTIFICATION

PRODUCT CODE: AC-F627A-MC
PRODUCT NAME: CKKUAAD 11/44 UBI MAP
DATE CREATED: OCT 1979
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHORS: JOHN W. CIUKAJ

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1979 BY DIGITAL EQUIPMENT CORPORATION

HISTORY SECTION

RR JAG WAS RELEASED OCT 1979

TABLE OF CONTENTS

- 1) ABSTRACT

- 2) REQUIREMENTS

 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3) LOADING PROCEDURE

 - 3.1 METHOD
- 4) STARTING PROCEDURE

 - 4.1 STARTING ADDRESS
 - 4.2 PROGRAM AND OPERATOR ACTION
 - 4.3 SPECIAL STARTING PROCEDURE
- 5) OPERATING PROCEDURE

 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUB-ROUTINE ABSTRACTS
 - 5.3 RUNNING UNDER APT
- 6) ERRORS

 - 6.1 ERROR HALTS AND DESCRIPTION
 - 6.2 ERROR RECOVERY
 - 6.3 SAMPLE ERROR MESSAGES
- 7) RESTRICTIONS

 - 7.1 STARTING RESTRICTIONS
 - 7.2 OPERATING RESTRICTIONS
- 8) MISCELLANEOUS

 - 8.1 EXECUTION TIME
 - 8.2 ADDRESS GENERATION IN THE PDP-11/44
- 9) PROGRAM DESCRIPTION

1. ABSTRACT

THIS PROGRAM IS DESIGNED TO BE RUN ON A PDP11/44 ON WHICH THE CPU, CACHE (IF APPLICABLE), AND MEMORY MANAGEMENT DIAGNOSTIC PROGRAMS HAVE BEEN RUN. THE PROGRAM WILL DETECT ALL ERRORS THAT ORIGINATE WITH THE MAP BOX AND PROVIDE LOOPING CAPABILITIES SO THAT THE FIELD SERVICE ENGINEER CAN VERIFY THE FAILURES. THERE MAY BE SOME CASES, SUCH AS THE CACHE REGISTER DATA PATH, AND CACHE MEMORY DATA PATH, WHERE INTERACTION BETWEEN MODULES PROHIBITS CLOSE ISOLATION, BUT THE FAILING FUNCTION WILL BE CALLED OUT SO THE FIELD SERVICE ENGINEER CAN COMPLETE THE ISOLATION PROCESS.

IF THE PROGRAM CATCHES AN ERROR IN AN EARLY TEST AND IS ALLOWED TO CONTINUE RUNNING THROUGH THE LATER TESTS THE ERROR INDICATIONS FROM THOSE LATER TESTS MAY BE INVALID. THIS IS DUE TO THE STRUCTURE OF THE PROGRAM, WHICH ASSUMES THAT ALL AREAS TESTED PRIOR TO THE CURRENT TEST ARE FUNCTIONING PROPERLY.

THE ERROR TYPE OUTS WILL BE IN TABLE FORMAT, WITH A MESSAGE INDICATING THE CLASS OF ERROR, A HEADER IDENTIFYING EACH COLUMN AND A REPORT OF ALL PERTINENT DATA. WHEN THE TEST CAN PRODUCE MORE THAN ONE ERROR CONDITION, A SUMMARY OF ERRORS WILL BE GIVEN AT THE END OF THAT TEST CONSISTING OF: THE LOGICAL 'AND' AND 'OR' OF THE DATA PREVIOUSLY REPORTED AND THE NUMBER OF ERRORS IN THIS TEST.

(SEE SECTION 6.3 FOR AN EXAMPLE OF THE ERROR TYPEOUTS.)

2. REQUIREMENTS

2.1 EQUIPMENT

THE BASIC PDP-11/44 COMPUTER, INCLUDING THE CPU, CACHE, MEMORY MANAGEMENT, AND AN LA-70 OR EQUIVALENT DEVICE FOR ERROR MESSAGES.

2.2 STORAGE

THIS PROGRAM WILL REQUIRE 8K TO LOAD BUT WILL UTILIZE ALL EXISTING CORE FOR A DUAL ADDRESSING TEST OF MEMORY FROM THE UNIBUS.

PRELIMINARY PROGRAMS

THE CPU, CACHE (IF APPLICABLE), AND MEMORY MANAGEMENT DIAGNOSTICS SHOULD BE RUN BEFORE THIS PROGRAM. THE MEMORY DIAGNOSTIC SHOULD AT LEAST, MAKE A QUICK VERIFY OF THE AREA OF MEMORY THIS PROGRAM WILL LOAD AND RUN IN.

3. LOADING PROCEDURE

3.1 METHOD.

THIS PROGRAM CAN BE LOADED FROM ANY DEVICE THAT IS SUPPORTED BY XXDP AND SHOULD BE LOADED USING THE XXDP PROCEDURE FOR THAT DEVICE.

4. STARTING PROCEDURE

4.1 STARTING ADDRESS

PROGRAM STARTS AT ADDRESS 200

4.2 PROGRAM AND/OR OPERATOR ACTION

PROGRAM WILL IDENTIFY ITSELF AND AT THE END OF EACH PASS WILL INDICATE THE TOTAL NUMBER OF ERRORS OCCURRING ON THAT PASS.

4.3 SPECIAL STARTING PROCEDURE

IF IT APPEARS THAT THE CACHE IS CAUSING SOME TROUBLE AND YOU STILL WANT TO RUN THIS PROGRAM, IT IS POSSIBLE TO RUN WITH THE CACHE DISABLED. SIMPLY LOAD THE CACHE CONTROL REGISTER (17777746) WITH THE DESIRED NUMBER. THEN LOAD THE PC (17777707) WITH THE STARTING ADDRESS (200) AND PRESS 'CONTINUE'. THE PROGRAM WILL NOW RUN NORMALLY EXCEPT THAT CERTAIN TESTS WILL BE SKIPPED SINCE THE CACHE IS DISABLED. THIS FACT IS INDICATED IN THE ABSTRACT OF EACH TEST THAT CHECKS THE CACHE CONTROL REGISTER.

DEFINITION OF THE BITS IN THE CACHE CONTROL REGISTER:
BIT00 -DISABLE TRAPS
BIT02 -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 0
BIT03 -FORCE MISS ON READ,WHERE ADDRESS BIT 12 IS 1
BIT09 -UNCONDITIONAL CACHE BYPASS

5. OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

| | | |
|------|----|---|
| SW15 | 1- | HALT ON ERROR |
| SW14 | 1- | LOOP ON TEST |
| SW13 | 1- | INHIBIT ERROR TYPEOUTS |
| SW12 | 1- | INHIBIT TRACE TRAP |
| SW11 | 1- | INHIBIT ITERATIONS |
| SW10 | 1- | BELL ON ERROR |
| SW09 | 1- | LOOP ON ERROR |
| SW08 | 1- | LOOP ON TEST IN SWR<05:00> |
| SW07 | 1- | INHIBIT MULTIPLE ERROR TYPE OUTS |
| SW06 | 1- | SELECT CACHE TESTS. THIS IS USED FOR MFG. QUICK VERIFY STATION AND CAN BE SELECTED BY APT SCRIPTING. THESE TESTS ASSUME THAT ALL MODULES EXCEPT UBI MODULE ARE KNOWN GOOD. |

5.2 SUB-ROUTINE ABSTRACTS

ALL SUBROUTINE ABSTRACTS APPEAR IN THE CODE BEFORE THEIR EXPANSION AND IN THE DOCUMENT THAT IMMEDIATELY FOLLOWS THIS. BELOW IS A LIST OF THE SUBROUTINE TITLES.

5.2.1 MACRO LIBRARY SUBROUTINES (FOUND IN MOST PROGRAMS)

SCOPE HANDLER ROUTINE
 ERROR HANDLER ROUTINE
 ERROR MESSAGE TYPE OUT ROUTINE
 CONVERT 16-BIT VIRTUAL ADDRESSES TO 22-BIT PHYSICAL ADDRESSES
 SAVE AND RESTORE R0-R5 ROUTINES
 TYPE ROUTINE
 BINARY TO OCTAL (ASCII) AND TYPE
 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
 TRAP DECODER
 POWER DOWN AND UP ROUTINES
 DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
 END OF PASS ROUTINE

5.2.2 SUBROUTINES UNIQUE TO THIS PROGRAM

TURN OFF AND SAVE T-BIT
 RESTORE T-BIT TO ITS PREVIOUS CONDITION
 SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
 SUBROUTINE TO REPORT MAP REGISTERS THAT WILL NOT HOLD ZERO
 SUBROUTINE TO REPORT DUAL ADDRESSING WHEN LOADING A MAP REGISTER
 SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN MAP REGISTERS
 SUBROUTINE TO REPORT COUNT PATTERN ERRORS ON UNIBUS DATA PATH
 SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN CACHE REGISTERS

5.2.3 TRAP AND ABORT HANDLER ROUTINES

CPU TRAP HANDLER ROUTINE
 CACHE TRAPS AND ABORTS HANDLER ROUTINE
 MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE

5.3 RUNNING UNDER APT

THE EXECUTION TIMES PROVIDED IN THE APT SCRIPT THAT FOLLOWS ARE FOR EXECUTION WITH A 11/44 PROCESSOR, CACHE, 16K CORE MEMORY, AND 300 BAUD.

THE FOLLOWING IS A PROGRAM LOAD FILE USED BY APT:

1. E TABLE 'A' IS USED FOR APT DUMP MODE.
 - A. IN ADDITION TO NORMAL CPU DIAGNOSTIC TESTS THIS TABLE WILL SELECT THE OPTIONAL CACHE TESTS. (\$SWREG=100) AND INHIBIT ITERATIONS (\$SWREG=4000)
2. E TABLE 'B' IS USED FOR APT QV MODE WHILE RUNNING ON A MANUFACTURING QV STATION. IT ACCOMPLISHES WHAT E TABLE 'A' DOES BUT ADDITIONALLY SUPPRESSES TYPEOUTS. (\$ENVM 240)
3. E TABLE 'C' IS USED FOR APT QV OR RUNTIME MODES WHILE RUNNING ON SYSTEMS OTHER THAN MFG. QV STATIONS. THIS TABLE DESELECTS THE OPTIONAL CACHE TESTS.

| | 1ST PASS RUN TIME 10 | LONGEST TEST TIME 5 | ADDITIONAL RUN TIME 0 | |
|---------------------------------------|----------------------------|---------------------------|-----------------------------|-------------------|
| | | E TABLES | | |
| E-MODE/S-MODE (\$ENVM/\$ENV) | | A 200/000 | B 240/001 | C 240/001 |
| SWITCH REGISTER 1 (\$SWREG) | | 004100 | 0004100 | 004000 |
| SWITCH REGISTER 2 CPU TYPE/OPTIONS | | 000000 00/0000 | 000000 00/0000 | 000000 00/0000 |

6. ERRORS

6.1 ERROR HALTS AND DESCRIPTION

WHEN AN ERROR IS DETECTED AN 'ERROR' (EMT) INSTRUCTION IS EXECUTED AND THE 'ERROR HANDLER ROUTINE' CHECKS THE SWITCH REGISTER FOR MODE SELECTED.

THE PROGRAM WILL:

| | |
|------------------------|-----------|
| HALT ON ERROR | IF SW15 1 |
| INHIBIT ERROR TYPE OUT | IF SW13 1 |
| RING BELL ON ERROR | IF SW10 1 |
| LOOP ON ERROR | IF SW9 1 |

6.2 ERROR RECOVERY

IF SW09=1, THE PROGRAM WILL LOOP BACK TO THE POINT WHERE THE INSTRUCTION THAT CAUSED THE ERROR WAS EXECUTED, WITHOUT ALLOWING ANY OF THE CONDITIONS TO CHANGE. THIS WILL PROVIDE THE TIGHTEST POSSIBLE SCOPE LOOP. IF SW09=0, EACH ERROR WILL BE REPORTED AND LOGGED AND, AT THE END OF EACH TEST, A SUMMARY OF ALL ERRORS OCCURRING IN THAT TEST WILL BE PROVIDED. THE SUMMARY CONSISTS OF THE LOGICAL AND AND OR OF THE ADDRESS AND/OR DATA THAT WAS WRONG.

6.3 SAMPLE ERROR TYPE OUTS
SEE 'ERRRTB:' FOR SAMPLE ERROR TYPEOUTS.

6.3.1 MULTIPLE TYPE ERRORS

THE FOLLOWING REGISTERS TIMED OUT WHEN REFERENCED

| REG.ADR | TESTNO | ERRORPC |
|---------|--------|---------|
| 170210 | 000001 | 015226 |
| 170212 | 000001 | 015232 |
| 170214 | 000001 | 015232 |
| 170216 | 000001 | 015232 |
| 170230 | 000001 | 015232 |
| 170232 | 000001 | 015232 |
| 170234 | 000001 | 015232 |
| 170236 | 000001 | 015232 |
| 170250 | 000001 | 015232 |
| 170252 | 000001 | 015232 |
| 170254 | 000001 | 015232 |
| 170256 | 000001 | 015232 |
| 170270 | 000001 | 015232 |
| 170272 | 000001 | 015232 |
| 170274 | 000001 | 015232 |
| 170276 | 000001 | 015232 |
| 170310 | 000001 | 015232 |
| 170312 | 000001 | 015232 |

| | | |
|--------|--------|--------|
| 170314 | 000001 | 015232 |
| 170316 | 000001 | 015232 |
| 170330 | 000001 | 015232 |
| 170332 | 000001 | 015232 |
| 170334 | 000001 | 015232 |
| 170336 | 000001 | 015232 |
| 170350 | 000001 | 015232 |
| 170352 | 000001 | 015232 |
| 170354 | 000001 | 015232 |
| 170356 | 000001 | 015232 |
| 170370 | 000001 | 015232 |
| 170372 | 000001 | 015232 |
| 170374 | 000001 | 015232 |
| 170376 | 000001 | 015232 |

SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ

| | | | | |
|---------|---------|---------|--------|---------|
| REGADRS | REGADRS | | | |
| 'OR' | 'AND' | #ERRORS | TESTNO | ERRORPC |
| 170376 | 170210 | 32 | 000001 | 010530 |

7. RESTRICTIONS

7.1 STARTING RESTRICTIONS

NONE

7.2 OPERATING RESTRICTIONS

NONE

8. MISCELLANEOUS

8.1 EXECUTION TIME

THE RUN TIME FOR A SINGLE PASS WITH NO ITERATIONS IS APPROXIMATELY 10 SECONDS.

8.2 ADDRESS GENERATION IN THE PDP-11/44

THE FOLLOWING IS AN EXAMPLE OF HOW A MEMORY ADDRESS IS GENERATED BY THE UNIBUS MAP. THIS ASSUMES THAT THE ADDRESS ORIGINATES IN THE CPU BUT THE PROCESS CAN APPLY TO ANY UNIBUS ADDRESS, STARTING AT LINE C2.

| | |
|---------------------------|---|
| A. VIRTUAL ADDRESS | 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 |
| A1. PAGE NUMBER (0-7) | 15 14 13 |
| A2. OFFSET | 12 11 10 09 08 07 06 05 04 03 02 01 00 |
| B. P.A.R.[PAGE NO.] + | 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 |
| <hr/> | |
| C. PHYSICAL ADDR. | 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 |
| C1. 17XXXXXX=> U.B.ADR. | 21 20 19 18 |
| C2. MAPPING REG.NO.(0-36) | 17 16 15 14 13 |
| C3. OFFSET | 12 11 10 09 08 07 06 05 04 03 02 01 00 |
| D. MAP REG.[NO.] + | 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 |
| <hr/> | |
| E. PHYSICAL ADDR. | 21 20 19 18 17 16 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00 |

DESCRIPTION OF LINES:

A: VIRTUAL ADDRESS (16 BITS)

A1: UPPER 3 BITS OF VIRTUAL ADDRESS, USED TO SELECT A PAGE ADDRESS REGISTER (PAR)

A2: LOWER 13 BITS OF VIRTUAL ADDRESS, ADDED TO SELECTED PAR

B: PAGE ADDRESS REGISTER (16 BITS), IN ADDITION PROCESS THIS GETS LEFT SHIFTED 6 BITS BEFORE ADDITION TO A2

C: PHYSICAL ADDRESS CREATED BY MEMORY MANAGEMENT, (22 BITS)

C1: IF UPPER 4 BITS ARE ALL ONES THEN BITS <17:00> GO OUT ON UNIBUS

C2: IF MAP RELOCATION IS ENABLED THEN BITS <17:13> SELECT ONE OF THE 36 (OCTAL) MAP REGISTERS.

C3: LOWER 13 BITS OF UNIBUS ADDRESS, ADDED TO SELECTED MAP REGISTER

D: MAP REGISTER (22 BITS), ADDED TO BITS <12:00> OF UNIBUS ADDRESS

E: PHYSICAL ADDRESS GENERATED BY UNIBUS MAP AND SENT TO THE CACHE.

9. PROGRAM DESCRIPTION

THE ASSEMBLED LISTING, CKKUA0.SEQ, HAS A PARAGRAPH DESCRIBING
EACH OF THE TESTS. THE PARAGRAPH WILL INDICATE IF THE TEST IS
RUN CONDITIONALLY ON THE STATUS ON THE CACHE CONTROL REGISTER.

•
.END

219

```
.TITLE CKKUAAO 11/44 UBI MAP
;*COPYRIGHT (C) OCT 1979
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY JOHN W. CIUKAJ
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-A5).
;*
```

220

```
.SBTTL OPERATIONAL SWITCH SETTINGS
;*.
;*.          SWITCH          USE
;*.          -----          -
;*.          15             HALT ON ERROR
;*.          14             LOOP ON TEST
;*.          13             INHIBIT ERROR TYPEOUTS
;*.          12             INHIBIT TRACE TRAP
;*.          11             INHIBIT ITERATIONS
;*.          10             BELL ON ERROR
;*.          9              LOOP ON ERROR
;*.          8              LOOP ON TEST IN SWR<5:0>
;*.          7              INHIBIT MULTIPLE ERROR TYPEOUTS
;*.          6              SELECT CACHE-CIS TESTS
```

221

222

```
.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100           ;;FIRST ADDRESS OF THE STACK
KERSTK= STACK        ;;KERNEL STACK
SUPSTK= STACK-200    ;;SUPERVISOR STACK
USESTK= STACK-300    ;;USER STACK
ERROR=EMT
SCOPE=IOT
PS      177776       ;;PROCESSOR STATUS WORD
PSW=PS
STKLMT= 177774       ;;STACK LIMIT REGISTER
PIRQ= 177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
;SWR= 177570         ;;** SWITCH REGISTER
;DISPLAY=SWR         ;;**
```

001100
001100
000700
000600
104000
000004
177776
177776
177774
177772

```
;*MISCELLANEOUS DEFINITIONS
HT= 11              ;;CODE FOR HORIZONTAL TAB
LF= 12              ;;CODE LINE FEED
CR= 15              ;;CODE CARRIAGE RETURN
CR.LF= 200         ;;CODE FOR CARRIAGE RETURN-LINE FEED
```

000011
000012
000015
000200

```
;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0             ;;GENERAL REGISTER
R1= %1             ;;GENERAL REGISTER
R2= %2             ;;GENERAL REGISTER
R3= %3             ;;GENERAL REGISTER
R4= %4             ;;GENERAL REGISTER
R5= %5             ;;GENERAL REGISTER
R6= %6             ;;GENERAL REGISTER
R7= %7             ;;GENERAL REGISTER
```

000000
000001
000002
000003
000004
000005
000006
000007

BASIC DEFINITIONS

| | |
|--------|--------|
| 000000 | R10=R0 |
| 000001 | R11=R1 |
| 000002 | R12=R2 |
| 000003 | R13=R3 |
| 000004 | R14=R4 |
| 000005 | R15=R5 |
| 000006 | SP=R6 |
| 000006 | KSP=SP |
| 000006 | SSP=SP |
| 000006 | USP=SP |
| 000007 | PC=R7 |

;*PRIORITY LEVEL DEFINITIONS

| | | | |
|--------|------|-----|--------------------|
| 000000 | PR0= | 0 | ::PRIORITY LEVEL 0 |
| 000040 | PR1= | 40 | ::PRIORITY LEVEL 1 |
| 000100 | PR2= | 100 | ::PRIORITY LEVEL 2 |
| 000140 | PR3= | 140 | ::PRIORITY LEVEL 3 |
| 000200 | PR4= | 200 | ::PRIORITY LEVEL 4 |
| 000240 | PR5= | 240 | ::PRIORITY LEVEL 5 |
| 000300 | PR6= | 300 | ::PRIORITY LEVEL 6 |
| 000340 | PR7= | 340 | ::PRIORITY LEVEL 7 |

;'SWITCH REGISTER' SWITCH DEFINITIONS

| | | |
|--------|-------|--------|
| 100000 | SW15= | 100000 |
| 040000 | SW14= | 40000 |
| 020000 | SW13= | 20000 |
| 010000 | SW12= | 10000 |
| 004000 | SW11= | 4000 |
| 002000 | SW10= | 2000 |
| 001000 | SW09= | 1000 |
| 000400 | SW08= | 400 |
| 000200 | SW07= | 200 |
| 000100 | SW06= | 100 |
| 000040 | SW05= | 40 |
| 000020 | SW04= | 20 |
| 000010 | SW03= | 10 |
| 000004 | SW02= | 4 |
| 000002 | SW01= | 2 |
| 000001 | SW00= | 1 |
| 001000 | SW9= | SW09 |
| 000400 | SW8= | SW08 |
| 000200 | SW7= | SW07 |
| 000100 | SW6= | SW06 |
| 000040 | SW5= | SW05 |
| 000020 | SW4= | SW04 |
| 000010 | SW3= | SW03 |
| 000004 | SW2= | SW02 |
| 000002 | SW1= | SW01 |
| 000001 | SW0= | SW00 |

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

| | | |
|--------|--------|--------|
| 100000 | BIT15= | 100000 |
| 040000 | BIT14= | 40000 |
| 020000 | BIT13= | 20000 |
| 010000 | BIT12= | 10000 |
| 004000 | BIT11= | 4000 |
| 002000 | BIT10= | 2000 |

BASIC DEFINITIONS

```

00*000 BIT09= 1000
000400 BIT08= 400
000200 BIT07= 200
000100 BIT06= 100
000040 BIT05= 40
000020 BIT04= 20
000010 BIT03= 10
000004 BIT02= 4
000002 BIT01= 2
000001 BIT00= 1
001000 BIT9=BIT09
000400 BIT8=BIT08
000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00
    
```

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
000010 RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC=14 ;;'T' BIT
000014 TRTVEC= 14 ;;TRACE TRAP
000014 BPTVEC= 14 ;;BREAKPOINT TRAP (BPT)
000020 IOTVEC= 20 ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC= 24 ;;POWER FAIL
000030 EMTVEC= 30 ;;EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC=34 ;;'TRAP' TRAP
000060 TKVEC= 60 ;;TTY KEYBOARD VECTOR
000064 TPVEC= 64 ;;TTY PRINTER VECTOR
000114 CACHVEC=114 ;;CACHE ERROR INTERRUPT VECTOR
000240 PIRQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR
000250 MMVEC= 250 ;;MEMORY MANAGEMENT VECTOR
    
```

.SBTTL CACHE REGISTER DEFINITIONS

```

177740 LOADRS - 177740 ;;LOWER 16 BITS OF ADDRESS THAT CAUSED ERROR
177742 HIADRS = 177742 ;;UPPER SIX BITS OF ADDRESS THAT CAUSED ERROR
177744 MEMERR = 177744 ;;CACHE ERROR REGISTER
177746 CONTRL = 177746 ;;MEMORY CONTROL REGISTER
177750 MAINT - 177750 ;;MEMORY MAINTENANCE REGISTER
177752 HITMIS - 177752 ;;HIT MISS REGISTER '1' IMPLIES HIT IN CACHE
    
```

.SBTTL CPU REGISTER DEFINITIONS

```

177760 SIZELO = 177760 ;;MEMORY SIZE REGISTER NUMBER TO PUT INTO A PAR
177762 SIZEHI = 177762 ;;TO GET TO THE LAST 32 WORDS OF MEMORY
177764 SYSTID = 177764 ;;HIGH SIZE REGISTER, RESERVED FOR FUTURE USE
177766 CPUERR = 177766 ;;CURRENTLY ALL ZERO
;;SYSTEM ID REGISTER
;;CPU ERROR REGISTER HOLDS CONDITION THAT CAUSED
    
```


::THE TRAP TO ERRVEC (000004)

.SBTTL MEMORY MANAGEMENT DEFINITIONS

;*MEMORY MANAGEMENT STATUS REGISTER ADDRESSES

| | | |
|--------|-------|--------|
| 177572 | MMR0= | 177572 |
| 177574 | MMR1= | 177574 |
| 177576 | MMR2= | 177576 |
| 172516 | MMR3= | 172516 |
| 177572 | SR0= | MMR0 |
| 177574 | SR1= | MMR1 |
| 177576 | SR2= | MMR2 |
| 172516 | SR3= | MMR3 |

;*USER 'I' PAGE DESCRIPTOR REGISTERS

| | | |
|--------|---------|--------|
| 177600 | UIPDR0= | 177600 |
| 177602 | UIPDR1= | 177602 |
| 177604 | UIPDR2= | 177604 |
| 177606 | UIPDR3= | 177606 |
| 177610 | UIPDR4= | 177610 |
| 177612 | UIPDR5= | 177612 |
| 177614 | UIPDR6= | 177614 |
| 177616 | UIPDR7= | 177616 |

;*USER 'D' PAGE DESCRIPTOR REGISTERS

| | | |
|--------|---------|--------|
| 177620 | UDPDR0= | 177620 |
| 177622 | UDPDR1= | 177622 |
| 177624 | UDPDR2= | 177624 |
| 177626 | UDPDR3= | 177626 |
| 177630 | UDPDR4= | 177630 |
| 177632 | UDPDR5= | 177632 |
| 177634 | UDPDR6= | 177634 |
| 177636 | UDPDR7= | 177636 |

;*USER 'I' PAGE ADDRESS REGISTERS

| | | |
|--------|---------|--------|
| 177640 | UIPAR0= | 177640 |
| 177642 | UIPAR1= | 177642 |
| 177644 | UIPAR2= | 177644 |
| 177646 | UIPAR3= | 177646 |
| 177650 | UIPAR4= | 177650 |
| 177652 | UIPAR5= | 177652 |
| 177654 | UIPAR6= | 177654 |
| 177656 | UIPAR7= | 177656 |

;*USER 'D' PAGE ADDRESS REGISTERS

| | | |
|--------|---------|--------|
| 177660 | UDPAR0= | 177660 |
| 177662 | UDPAR1= | 177662 |
| 177664 | UDPAR2= | 177664 |

| | |
|--------|----------------|
| 177666 | UDPAR3= 177666 |
| 177670 | UDPAR4= 177670 |
| 177672 | UDPAR5= 177672 |
| 177674 | UDPAR6= 177674 |
| 177676 | UDPAR7= 177676 |

:*SUPERVISOR 'I' PAGE DESCRIPTOR REGISTERS

| | |
|--------|----------------|
| 172200 | SIPDR0= 172200 |
| 172202 | SIPDR1= 172202 |
| 172204 | SIPDR2= 172204 |
| 172206 | SIPDR3= 172206 |
| 172210 | SIPDR4= 172210 |
| 172212 | SIPDR5= 172212 |
| 172214 | SIPDR6= 172214 |
| 172216 | SIPDR7= 172216 |

:*SUPERVISOR 'D' PAGE DESCRIPTOR REGISTERS

| | |
|--------|----------------|
| 172220 | SDPDR0= 172220 |
| 172222 | SDPDR1= 172222 |
| 172224 | SDPDR2= 172224 |
| 172226 | SDPDR3= 172226 |
| 172230 | SDPDR4= 172230 |
| 172232 | SDPDR5= 172232 |
| 172234 | SDPDR6= 172234 |
| 172236 | SDPDR7= 172236 |

:*SUPERVISOR 'I' PAGE ADDRESS REGISTERS

| | |
|--------|----------------|
| 172240 | SIPAR0= 172240 |
| 172242 | SIPAR1= 172242 |
| 172244 | SIPAR2= 172244 |
| 172246 | SIPAR3= 172246 |
| 172250 | SIPAR4= 172250 |
| 172252 | SIPAR5= 172252 |
| 172254 | SIPAR6= 172254 |
| 172256 | SIPAR7= 172256 |

:*SUPERVISOR 'D' PAGE ADDRESS REGISTERS

| | |
|--------|----------------|
| 172260 | SDPAR0= 172260 |
| 172262 | SDPAR1= 172262 |
| 172264 | SDPAR2= 172264 |
| 172266 | SDPAR3= 172266 |
| 172270 | SDPAR4= 172270 |
| 172272 | SDPAR5= 172272 |
| 172274 | SDPAR6= 172274 |
| 172276 | SDPAR7= 172276 |

:*KERNEL 'I' PAGE DESCRIPTOR REGISTERS

| | |
|--------|----------------|
| 172300 | KIPDR0= 172300 |
| 172302 | KIPDR1= 172302 |
| 172304 | KIPDR2= 172304 |
| 172306 | KIPDR3= 172306 |
| 172310 | KIPDR4= 172310 |

172312 KIPDR5= 172312
172314 KIPDR6= 172314
172316 KIPDR7= 172316

:*KERNEL 'D' PAGE DESCRIPTOR REGISTERS

172320 KDPDR0= 172320
172322 KDPDR1= 172322
172324 KDPDR2= 172324
172326 KDPDR3= 172326
172330 KDPDR4= 172330
172332 KDPDR5= 172332
172334 KDPDR6= 172334
172336 KDPDR7= 172336

:*KERNEL 'I' PAGE ADDRESS REGISTERS

172340 KIPAR0= 172340
172342 KIPAR1= 172342
172344 KIPAR2= 172344
172346 KIPAR3= 172346
172350 KIPAR4= 172350
172352 KIPAR5= 172352
172354 KIPAR6= 172354
172356 KIPAR7= 172356

:*KERNEL 'D' PAGE ADDRESS REGISTERS

172360 KDPAR0= 172360
172362 KDPAR1= 172362
172364 KDPAR2= 172364
172366 KDPAR3= 172366
172370 KDPAR4= 172370
172372 KDPAR5= 172372
172374 KDPAR6= 172374
172376 KDPAR7= 172376

.SBTTL UNIBUS MAP REGISTER DEFINITIONS

:*THE LOWER 16 BITS OF THE MAP REGISTERS ARE LABELED 'MAPLYX'
:*THE UPPER 6 BITS OF THE MAP REGISTERS ARE LABELED 'MAPHXX'

170200 MAPL00 = 170200
170202 MAPH00 = 170202
170204 MAPL01 = 170204
170206 MAPH01 = 170206
170210 MAPL02 = 170210
170212 MAPH02 = 170212
170214 MAPL03 = 170214
170216 MAPH03 = 170216
170220 MAPL04 = 170220
170222 MAPH04 = 170222

| | | |
|--------|--------------|----------|
| 170224 | MAPL05 | 170224 |
| 170226 | MAPH05 | 170226 |
| 170230 | MAPL06 | 170230 |
| 170232 | MAPH06 | = 170232 |
| 170234 | MAPL07 | = 170234 |
| 170236 | MAPH07 | = 170236 |
| 170240 | MAPL10 | = 170240 |
| 170242 | MAPH10 | = 170242 |
| 170244 | MAPL11 | = 170244 |
| 170246 | MAPH11 | = 170246 |
| 170250 | MAPL12 | = 170250 |
| 170252 | MAPH12 | = 170252 |
| 170254 | MAPL13 | = 170254 |
| 170256 | MAPH13 | = 170256 |
| 170260 | MAPL14 | = 170260 |
| 170262 | MAPH14 | = 170262 |
| 170264 | MAPL15 | = 170264 |
| 170266 | MAPH15 | = 170266 |
| 170270 | MAPL16 | = 170270 |
| 170272 | MAPH16 | = 170272 |
| 170274 | MAPL17 | = 170274 |
| 170276 | MAPH17 | = 170276 |
| 170300 | MAPL20 | = 170300 |
| 170302 | MAPH20 | = 170302 |
| 170304 | MAPL21 | = 170304 |
| 170306 | MAPH21 | = 170306 |
| 170310 | MAPL22 | = 170310 |
| 170312 | MAPH22 | = 170312 |
| 170314 | MAPL23 | = 170314 |
| 170316 | MAPH23 | = 170316 |
| 170320 | MAPL24 | = 170320 |
| 170320 | MAPH24 | = 170320 |
| 170324 | MAPL25 | = 170324 |
| 170326 | MAPH25 | = 170326 |
| 170330 | MAPL26 | = 170330 |
| 170332 | MAPH26 | = 170332 |
| 170334 | MAPL27 | = 170334 |
| 170336 | MAPH27 | = 170336 |
| 170340 | MAPL30 | = 170340 |
| 170342 | MAPH30 | = 170342 |
| 170344 | MAPL31 | = 170344 |
| 170346 | MAPH31 | = 170346 |
| 170350 | MAPL32 | = 170350 |
| 170352 | MAPH32 | = 170352 |
| 170354 | MAPL33 | = 170354 |
| 170356 | MAPH33 | = 170356 |
| 170360 | MAPL34 | = 170360 |
| 170362 | MAPH34 | = 170362 |
| 170364 | MAPL35 | = 170364 |
| 170366 | MAPH35 | = 170366 |
| 170370 | MAPL36 | = 170370 |
| 170372 | MAPH36 | = 170372 |
| 170374 | MAPL37 | 170374 |
| 170376 | MAPH37 | = 170376 |
| 170200 | MAPL0=MAPL00 | |
| 170202 | MAPH0=MAPH00 | |
| 170204 | MAPL1=MAPL01 | |

| | |
|--------|--------------|
| 170206 | MAPH1=MAPH01 |
| 170210 | MAPL2=MAPL02 |
| 170212 | MAPH2=MAPH02 |
| 170214 | MAPL3=MAPL03 |
| 170216 | MAPH3=MAPH03 |
| 170220 | MAPL4=MAPL04 |
| 170222 | MAPH4=MAPH04 |
| 170224 | MAPL5=MAPL05 |
| 170226 | MAPH5=MAPH05 |
| 170230 | MAPL6=MAPL06 |
| 170232 | MAPH6=MAPH06 |
| 170234 | MAPL7=MAPL07 |
| 170236 | MAPH7=MAPH07 |

224

.SBTTL TRAP CATCHER

000000

. =0
;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A '+2,HALT'
;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

000200

.SBTTI STARTING ADDRESS(ES)
. -200

225

000200 000137 010000

JMP @START ;: JUMP TO STARTING ADDRESS OF PROGRAM
:*****

.SBTTL ACT11 HOOKS

:*THE FOLLOWING LOCATIONS ARE SETUP TO BE USED WITH ACT11:
:*
:*LOCATION 46 WILL CONTAIN THE ADDRESS OF THE LOGICAL
:*END OF THE PROGRAM.
:*LOCATION 52 IS USED TO SPECIFY PROGRAM OPERATING REQUIREMENTS
:*AND/OR RESTRICTIONS. THIS IS ACCOMPLISHED BY SETTING VARIOUS BITS
:*TO A ONE OR A ZERO. THE BITS USED AND THERE MEANING ARE:

:* BIT 15=1 PROGRAM SHOULD BE POWER FAILED WHILE RUNNING
:* -0 NO POWER FAIL DESIRED

:* BIT 14=1 PROGRAM RUN TIME IS MEMORY SIZE DEPENDENT
:* =0 RUN TIME IS NOT MEMORY SIZE DEPENDENT

:* BITS 13-0 MUST BE ZERO'S

000204
000046
000046 021472
000052
000052 000000
000204

\$.SVPC=. ;: SAVE LOCATION COUNTER
. -46 ;: SET LOCATION COUNTER
.WORD \$ENDAD ;: SET LOC.46 TO ADDRESS \$ENDAD
. =52 ;: SET LOCATION COUNTER
.WORD 0 ;: SET LOC.52 TO ZERO
. -\$SVPC ;: RESTORE LOCATION COUNTER

226

::*****

.SBTTL COMMON TAGS

:*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
:*USED IN THE PROGRAM.

001100 . =1100

| | | | | |
|--------|----------------------|-----------|-------------------------|--|
| 001100 | | \$CMTAG: | | :: START OF COMMON TAGS |
| 001100 | 000 | ;\$PASS: | .WORD 0 | :: ** CONTAINS PASS COUNT |
| 001101 | 000 | \$TSTNM: | .BYTE 0 | :: CONTAINS THE TEST NUMBER |
| 001102 | 000000 | \$ERFLG: | .BYTE 0 | :: CONTAINS ERROR FLAG |
| 001104 | 000000 | \$ICNT: | .WORD 0 | :: CONTAINS SUBTEST ITERATION COUNT |
| 001106 | 000000 | \$LPADR: | .WORD 0 | :: CONTAINS SCOPE LOOP |
| 001110 | 000000 | \$LPERR: | .WORD 0 | :: CONTAINS SCOPE RETURN FOR ERRORS |
| 001112 | 000 | \$ERTTL: | .WORD 0 | :: CONTAINS TOTAL ERRORS DETECTED |
| 001113 | 001 | \$ITEMB: | .BYTE 0 | :: CONTAINS ITEM CONTROL BYTE |
| 001114 | 000000 | \$ERMAX: | .BYTE 1 | :: CONTAINS MAX. ERRORS PER TEST |
| 001116 | 000000 | \$ERRPC: | .WORD 0 | :: CONTAINS PC OF LAST ERROR INSTRUCTION |
| 001120 | 000000 | \$GDADR: | .WORD 0 | :: CONTAINS OF 'GOOD' DATA |
| 001122 | 000000 | \$BDADR: | .WORD 0 | :: CONTAINS OF 'BAD' DATA |
| 001124 | 000000 | \$GDDAT: | .WORD 0 | :: CONTAINS 'GOOD' DATA |
| 001126 | 000000 000000 000000 | \$BDDAT: | .WORD 0 | :: CONTAINS 'BAD' DATA |
| 001134 | 177560 | | .WORD 0,0,0 | :: RESERVED--NOT TO BE USED |
| 001136 | 177562 | \$TKS: | 177560 | :: TTY KBD STATUS |
| 001140 | 177564 | \$TKB: | 177562 | :: TTY KBD BUFFER |
| 001142 | 177566 | \$TPS: | 177564 | :: TTY PRINTER STATUS REG. |
| 001144 | 000 | \$TPB: | 177566 | :: TTY PRINTER BUFFER REG. |
| 001145 | 002 | \$NULL: | .BYTE 0 | :: CONTAINS NULL CHARACTER FOR FILLS |
| 001146 | 012 | \$FILLS: | .BYTE 2 | :: CONTAINS # OF FILLER CHARACTERS REQUIRED |
| 001147 | 000 | \$FILLC: | .BYTE 12 | :: INSERT FILL CHARS. AFTER A 'LINE FEED' |
| | | \$TPFLG: | .BYTE 0 | :: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES) |
| 001150 | 177570 | \$WR: | .WORD 177570 | :: ** |
| 001152 | 177570 | DISPLAY: | .WORD 177570 | :: ** |
| 001154 | 000000 | \$REGAD: | .WORD 0 | :: CONTAINS THE FROM WHICH (\$REG0) WAS OBTAINED |
| 001156 | 000000 | | .REPT \$CM3 | |
| 001160 | 000000 | \$REG0: | .WORD 0 | :: CONTAINS ((\$REGAD)+0) |
| 001162 | 000000 | \$REG1: | .WORD 0 | :: CONTAINS ((\$REGAD)+2) |
| 001164 | 000000 | \$REG2: | .WORD 0 | :: CONTAINS ((\$REGAD)+4) |
| 001166 | 000000 | \$REG3: | .WORD 0 | :: CONTAINS ((\$REGAD)+6) |
| 001170 | 000000 | \$REG4: | .WORD 0 | :: CONTAINS ((\$REGAD)+10) |
| | 000006 | \$REG5: | .WORD 0 | :: CONTAINS ((\$REGAD)+12) |
| 001172 | 000000 | | .REPT 6 | |
| 001174 | 000000 | \$TMP0: | .WORD 0 | :: USER DEFINED |
| 001176 | 000000 | \$TMP1: | .WORD 0 | :: USER DEFINED |
| 001200 | 000000 | \$TMP2: | .WORD 0 | :: USER DEFINED |
| 001202 | 000000 | \$TMP3: | .WORD 0 | :: USER DEFINED |
| 001204 | 000000 | \$TMP4: | .WORD 0 | :: USER DEFINED |
| 001206 | 000000 | \$TMP5: | .WORD 0 | :: USER DEFINED |
| 001210 | 000000 | \$TIMES: | 0 | :: MAX. NUMBER OF ITERATIONS |
| 001212 | 207 | \$ESCAPE: | 0 | :: ESCAPE ON ERROR |
| 001215 | 00C | \$BELL: | .ASCII7 <207><377><377> | :: CODE FOR BELL |

COMMON TAGS

| | | | | | | |
|--------|--------|-----|---------|---------|------|---|
| 001216 | 077 | | \$QUES: | .ASCII | /?/ | ::QUESTION MARK |
| 001217 | 015 | | \$CRLF: | .ASCII | <15> | ::CARRIAGE RETURN |
| 001220 | 012 | 000 | \$LF: | .ASCIIZ | <12> | ::LINE FEED |
| 001222 | 000000 | | PADRSL: | .WORD | 0 | ::HOLDS THE LOWER 16 BITS OF A 22 BIT ADDRESS GENERATED FOR TYPE OUT. |
| 001224 | 000000 | | PADRSR: | .WORD | 0 | ::HOLDS THE UPPER 6 BITS OF A 22 BIT ADDRESS GENERATED FOR TYPE OUT |
| 001226 | 000000 | | ADRAND: | .WORD | 0 | ::LOGICAL AND OF FAILING ADDRESSES |
| 001230 | 000000 | | ADDROR: | .WORD | 0 | ::LOGICAL OR OF FAILING ADDRESSES |
| 001232 | 000000 | | DATAND: | .WORD | 0 | ::LOGICAL AND OF BAD DATA |
| 001234 | 000000 | | DATAOR: | .WORD | 0 | ::LOGICAL OR OF BAD DATA |
| 001236 | 000000 | | PATAND: | .WORD | 0 | ::LOGICAL AND OF PATTERN LOADED |
| 001240 | 000000 | | PATOR: | .WORD | 0 | ::LOGICAL OR OF PATTERN LOADED |
| 001242 | 000000 | | LOWEST: | .WORD | 0 | ::HOLDS NUMBER TO PUT IN PAR TO CAUSE THE LOWEST USEABLE MAP REGISTER TO RESPOND |
| 001244 | 000000 | | HIGEST: | .WORD | 0 | ::HOLDS NUMBER TO PUT IN PAR TO CAUSE THE HIGHEST USEABLE MAP REGISTER TO RESPOND |
| 001246 | 000000 | | LREGL: | .WORD | 0 | ::HOLDS I/O PAGE ADDR OF LOW 16 BITS OF THE LOWEST USEABLE MAP REGISTER |
| 001250 | 000000 | | LREGU: | .WORD | 0 | ::HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF OF THE LOWEST USEABLE MAP REGISTER |
| 001252 | 000000 | | HREGL: | .WORD | 0 | ::HOLDS I/O PAGE ADDR OF LOW 16 BITS OF THE HIGHEST USEABLE MAP REGISTER |
| 001254 | 000000 | | HREGU: | .WORD | 0 | ::HOLDS I/O PAGE ADDR OF HIGH 16 BITS OF THE HIGHEST USEABLE MAP REGISTER |
| 001256 | 000000 | | ERRCNT: | .WORD | 0 | ::MULTIPLE ERROR ERROR COUNTER |
| 001260 | 000000 | | CNTR: | .WORD | 0 | ::AUXILIARY COUNTER |
| 001262 | 000000 | | FLAG: | .WORD | 0 | ::FLAG TO INDICATE TO LAST PROGRAM PASS N |
| 001264 | 000000 | | TESTNO: | .WORD | 0 | ::HOLDS TEST NUMBER FOR ERROR TYPE OUTS |
| 001266 | 000000 | | CPUEXP: | .WORD | 0 | ::HOLDS THE EXPECTED CPU ERROR CODE |
| 001270 | 000000 | | PCPUER: | .WORD | 0 | ::HOLDS RECEIVED CPU ERROR CONDITION |
| 001272 | 000000 | | PPARER: | .WORD | 0 | ::HOLDS RECEIVED PARITY ERROR CONDITION |
| 001274 | 000000 | | PCONTR: | .WORD | 0 | ::HOLDS CONTENTS OF CONTROL REGISTER |
| 001276 | 000000 | | PMAINT: | .WORD | 0 | ::HOLDS CONTENTS OF MAINTENANCE REGISTER |
| 001300 | 000000 | | BADPC: | .WORD | 0 | ::HOLDS PC OF INST THAT CAUSED TRAP |
| 001302 | 000000 | | OLDPC: | .WORD | 0 | ::HOLDS THE RETURN ADDRESS AFTER A TRAP |
| 001304 | 000000 | | OLDPS: | .WORD | 0 | ::HOLDS THE OLD PROCESSOR STATUS |
| 001306 | 000000 | | OLDPSW: | .WORD | 0 | ::HOLDS OLD PSW FOR TBITRESTORE |
| 001310 | 000000 | | PMR0: | .WORD | 0 | ::HOLDS CONTENTS OF MMR0 AFTER TRAP |
| 001312 | 000000 | | PMR1: | .WORD | 0 | ::HOLDS CONTENTS OF MMR1 AFTER TRAP |
| 001314 | 000000 | | PMR2: | .WORD | 0 | ::HOLDS CONTENTS OF MMR2 AFTER TRAP |
| 001316 | 000000 | | RSIZE: | .WORD | 0 | ::WILL HOLD P.A.R. DATA FOR TOP OF MEMORY |
| 001320 | 000000 | | RETRY: | .WORD | 0 | ::RETRY FLAG IN CASE OF PARITY ABORTS |
| 001322 | 000000 | | NXTTST: | .WORD | 0 | ::LOCATION TO HOLD ESCAPE ADDRESS ON PARITY ERRORS. |
| 001324 | 000200 | | DATA: | .WORD | 200 | ::PATTERN TO BE USED TO LOAD INTO MEMORY |

.SBTTL ERROR POINTER TABLE

:*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
:*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
:*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
:*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
:*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

:* EM :POINTS TO THE ERROR MESSAGE
:* DH :POINTS TO THE DATA HEADER
:* DT :POINTS TO THE DATA
:* DF :POINTS TO THE DATA FORMAT

| | | | | |
|-----|--------|--------|----------|---|
| 227 | 001326 | | \$ERRTB: | |
| 228 | 001326 | 021554 | :ITEM1 | |
| 229 | 001330 | 030034 | EM1 | :NOT THE CORRECT CPU TRAP CONDITION THRU ERRVEC (#004) |
| 230 | 001332 | 032562 | DH1 | :RECEIVD EXPECTD TESTNO PC AT ABORT |
| 231 | 001334 | 033266 | DT1 | :PCPUER,CPUEXP,TESTNO,BADPC,0 |
| 232 | | | DF1 | : 0, 0, 0, 0 |
| 233 | | | :ITEM 2 | |
| 234 | 001336 | 021636 | EM2 | :UNEXPECTED CPU TRAP THRU ERRVEC (#004) |
| 235 | 001340 | 030100 | DH2 | :RECEIVD TESTNO PC AT ABORT |
| 236 | 001342 | 032574 | DT2 | :PCPUER,TESTNO,BADPC |
| 237 | 001344 | 033272 | DF2 | : 0, 0, 0 |
| 238 | | | :ITEM 3 | |
| 240 | 001346 | 022115 | EM5 | :MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS |
| 241 | 001350 | 030264 | DH5 | :STATUS AUTOI/D VIRTADR |
| 242 | | | | :REGISTR REGISTR REGISTR TESTNO PC AT ABORT |
| 243 | 001352 | 032616 | DT5 | :PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0 |
| 244 | 001354 | 033303 | DF5 | : 0, 0, 0, 0, 0 |
| 245 | | | :ITEM 4 | |
| 247 | 001356 | 022223 | EM6 | :SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ |
| 248 | 001360 | 030370 | DH6 | :REGADRS REGADRS |
| 249 | | | | : 'OR' 'AND' #ERRORS TESTNO ERRORPC |
| 250 | 001362 | 032632 | DT6 | :ADDROR,ADRAND,ERRCNT,TESTNO,\$ERRPC,0 |
| 251 | 001364 | 033310 | DF6 | : 0, 0, 1, 0, 0 |
| 252 | | | :ITEM 5 | |
| 254 | 001366 | 022365 | EM10 | :SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS |
| 255 | 001370 | 030460 | DH10 | :REGADRS REGADRS RECEIVD RECEIVD |
| 256 | | | | : 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC |
| 257 | 001372 | 032646 | DT10 | :ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,\$ERRPC,0 |
| 258 | 001374 | 033315 | DF10 | : 0, 0, 0, 0, 1, 0, 0 |
| 259 | | | :ITEM 6 | |
| 261 | 001376 | 022456 | EM11 | :SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS |
| 262 | 001400 | 030460 | DH10 | :REGADRS REGADRS RECEIVD RECEIVD |
| 263 | | | | : 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC |
| 264 | 001402 | 032646 | DT10 | :ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,\$ERRPC,0 |
| 265 | 001404 | 033315 | DF10 | : 0, 0, 0, 0, 1, 0, 0 |
| 266 | | | | |

| | | | | |
|-----|--------|--------|----------|--|
| 267 | | | :ITEM 7 | |
| 268 | 001406 | 022550 | EM12 | :POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00) |
| 269 | 001410 | 030610 | DH12 | :COUNT COUNT |
| 270 | | | | :EXPECTD RECEIVD TESTNO ERRORPC |
| 271 | 001412 | 032666 | DT12 | :\$REG2,\$REG0,TESTNO,\$ERRPC,0 |
| 272 | 001414 | 033324 | DF12 | : 0, 0, 0, 0 |
| 273 | | | | |
| 274 | | | :ITEM 10 | |
| 275 | 001416 | 022636 | EM13 | :NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20) |
| 276 | 001420 | 030610 | DH12 | :COUNT COUNT |
| 277 | | | | :EXPECTD RECEIVD TESTNO ERRORPC |
| 278 | 001422 | 032700 | DT13 | :\$REG3,\$REG1,TESTNO,\$ERRPC,0 |
| 279 | 001424 | 033324 | DF12 | : 0, 0, 0, 0 |
| 280 | | | | |
| 281 | | | :ITEM 11 | |
| 282 | 001426 | 022730 | EM14 | :SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS |
| 283 | 001430 | 030666 | DH14 | :REGLOAD REGLOAD REGDUAL REGDUAL |
| 284 | | | | : 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO |
| 285 | 001432 | 032712 | DT14 | :ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0 |
| 286 | 001434 | 033330 | DF14 | : 0, 0, 0, 0, 1, 0 |
| 287 | | | | |
| 288 | | | :ITEM 12 | |
| 289 | 001436 | 023023 | EM15 | :SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS |
| 290 | 001440 | 031005 | DH15 | :MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD |
| 291 | | | | : 'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO |
| 292 | 001442 | 032730 | DT15 | :ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0 |
| 293 | 001444 | 033336 | DF15 | : 0, 0, 0, 0, 0, 0, 1, 0 |
| 294 | | | | |
| 295 | | | :ITEM 13 | |
| 296 | 001446 | 023127 | EM16 | :SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS |
| 297 | 001450 | 031005 | DH15 | :MAPREG MAPREG EXPECTD EXPECTD RECEIVD RECEIVD |
| 298 | | | | : 'OR' 'AND' 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO |
| 299 | 001452 | 032730 | DT15 | :ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0 |
| 300 | 001454 | 033336 | DF15 | : 0, 0, 0, 0, 0, 0, 1, 0 |
| 301 | | | | |
| 302 | | | :ITEM 14 | |
| 303 | 001456 | 024206 | EM25 | :REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT |
| 304 | | | | :DIFFERENT THAN 770200 |
| 305 | 001460 | 031333 | DH25 | :ADDRUSED BITDIFF TESTNO ERRORPC |
| 306 | 001462 | 033010 | DT25 | :\$REG4,\$REG0,TESTNO,\$ERRPC,0 |
| 307 | 001464 | 033357 | DF25 | :3, 4, 0, 0 |
| 308 | | | | |
| 309 | | | :ITEM 15 | |
| 310 | 001466 | 024434 | EM30 | :CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF |
| 311 | | | | :SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION |
| 312 | 001470 | 031427 | DH30 | :TESTNO ERRORPC |
| 313 | 001472 | 033032 | DT30 | :TESTNO,\$ERRPC,0 |
| 314 | 001474 | 033366 | DF30 | :0, 0 |
| 315 | | | | |
| 316 | | | :ITEM 16 | |
| 317 | 001476 | 024610 | EM31 | :SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH |
| 318 | 001500 | 031214 | DH23 | :EXPECTD EXPECTD RECEIVD RECEIVD |
| 319 | | | | : 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO |
| 320 | 001502 | 032772 | DT23 | :PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0 |
| 321 | 001504 | 033351 | DF23 | : 0, 0, 0, 0, 1, 0 |
| 322 | | | | |
| 323 | | | :ITEM 17 | |

| | | | | |
|-----|--------|--------|----------|---|
| 324 | 001506 | 024702 | EM32 | :UNIBUS MAP IS RELOCATING WHEN NOT ENABLED |
| 325 | 001510 | 031427 | DH30 | :TESTNO ERRORPC |
| 326 | 001512 | 033032 | DT30 | :TESTNO,\$ERRPC,0 |
| 327 | 001514 | 033366 | DF30 | : 0, 0 |
| 328 | | | | |
| 329 | | | :ITEM 20 | |
| 330 | 001516 | 024754 | EM33 | :CANNOT USE ANY OF THE MAP REGISTERS OR PHYSICAL |
| 331 | | | | :ADDRESS BIT'4 IS STUCK LOW, MUST RESTART PROGRAM |
| 332 | | | | :IF YOU DON'T LOOP ON THIS PROBLEM. |
| 333 | 001520 | 031427 | DH30 | :TESTNO ERRORPC |
| 334 | 001522 | 033032 | DT30 | :TESTNO,\$ERRPC,0 |
| 335 | 001524 | 033366 | DF30 | : 0, 0 |
| 336 | | | | |
| 337 | 001526 | 000240 | NOP | |
| 338 | 001530 | 000240 | NOP | |
| 339 | 001532 | 000240 | NOP | |
| 340 | 001534 | 000240 | NOP | |
| 341 | | | | |
| 342 | | | :ITEM 22 | |
| 343 | 001536 | 025166 | EM35 | :THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET |
| 344 | | | | :IN THEIR DEFAULT POSITION. THIS WOULD ALLOW UNIBUS |
| 345 | | | | :ADDRESSES 000000 TO 757776 TO REFERECE MAIN MEMORY |
| 346 | | | | :THEIR CURRENT SETTING ALLOWS ONLY: |
| 347 | 001540 | 031507 | DH35 | :LOWEST HIGHEST TESTNO ERRORPC |
| 348 | 001542 | 033052 | DT35 | :LOWEST,HIGEST,TESTNO,\$ERRPC,0 |
| 349 | 001544 | 033374 | DF35 | : 4, 4, 0, 0 |
| 350 | | | | |
| 351 | | | :ITEM 23 | |
| 352 | 001546 | 025736 | EM36 | :MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST |
| 353 | 001550 | 031547 | DH36 | :TESTNO ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST |
| 354 | 001552 | 033064 | DT36 | :TESTNO,\$ERRPC,\$REG0,0 |
| 355 | 001554 | 033400 | DF36 | : 0, 0, 3 |
| 356 | | | | |
| 357 | | | :ITEM 24 | |
| 358 | 001556 | 026033 | EM37 | :SUMMARY OF UNIBUS ADDRESS ERRORS, WITH MAP RELOCATION DISABLED |
| 359 | 001560 | 031214 | DH33 | :EXPECTD EXPECTD RECEIVD RECEIVD |
| 360 | | | | : 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO |
| 361 | 001562 | 033074 | DT37 | :ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO |
| 362 | 001564 | 033403 | DF37 | : 0, 0, 0, 0, 1, 0 |
| 363 | | | | |
| 364 | | | :ITEM 25 | |
| 365 | 001566 | 026136 | EM40 | :MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY. |
| 366 | 001570 | 031641 | DH40 | :CONDITN CONDITN |
| 367 | | | | :EXPECTD RECEIVD TESTNO ERRORPC |
| 368 | 001572 | 033112 | DT40 | :CPUEXP,PCPUER,TESTNO,\$ERRPC,0 |
| 369 | 001574 | 033411 | DF40 | : 0, 0, 0, 0 |
| 370 | | | | |
| 371 | | | :ITEM 26 | |
| 372 | 001576 | 026231 | EM41 | :RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD. |
| 373 | 001600 | 031721 | DH41 | :CORRECT ADDRESS |
| 374 | | | | :ADDRESS FETCHED TESTNO ERRORPC |
| 375 | 001602 | 033124 | DT41 | :\$REG2,\$REG1,TESTNO,\$ERRPC |
| 376 | 001604 | 033415 | DF41 | : 0, 0, 0, 0 |
| 377 | | | | |
| 378 | | | :ITEM 27 | |
| 379 | 001606 | 026313 | EM42 | :RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION |
| 380 | 001610 | 032001 | DH42 | :CORRECT EXPECTD RECEIVD |

| | | | | |
|-----|--------|--------|-----------|---|
| 381 | | | | :ADDRESS DATA FROM UB TESTNO ERRORPC |
| 382 | 001612 | 033136 | DT42 | : \$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC |
| 383 | 001614 | 033421 | DF42 | : 3, 0, 0, 0, 0 |
| 384 | | | | |
| 385 | | | :ITEM 30 | |
| 386 | 001616 | 026473 | EM45 | :MAIN MEMORY TIME OUT OVER THE UNIBUS DID NOT OCCUR PROPERLY. |
| 387 | | | | :TEST BEING RUN OVER UNIBUS |
| 388 | 001620 | 031641 | DH40 | :CONDITN CONDITN |
| 389 | | | | :EXPECTD RECEIVD TESTNO ERRORPC |
| 390 | 001622 | 033112 | DT40 | :CPUEXP,PCPUER,TESTNO,\$ERRPC,0 |
| 391 | 001624 | 033411 | DF40 | : 0, 0, 0, 0 |
| 392 | | | | |
| 393 | | | :ITEM 31 | |
| 394 | 001626 | 026626 | EM46 | :RELOCATION THROUGH THE MAP WAS NOT CORRECT, FULL ADD. |
| 395 | | | | :TEST BEING RUN OVER UNIBUS |
| 396 | 001630 | 031721 | DH41 | :CORRECT ADDRESS |
| 397 | | | | :ADDRESS FETCHED TESTNO ERRORPC |
| 398 | 001632 | 033124 | DT41 | : \$REG2,\$REG1,TESTNO,\$ERRPC |
| 399 | 001634 | 033415 | DF41 | : 0, 0, 0, 0 |
| 400 | | | | |
| 401 | | | :ITEM 32 | |
| 402 | 001636 | 026750 | EM47 | :RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION |
| 403 | | | | :TEST BEING RUN OVER UNIBUS |
| 404 | 001640 | 032001 | DH42 | :CORRECT EXPECTD RECEIVD |
| 405 | | | | :ADDRESS DATA FROM UB TESTNO ERRORPC |
| 406 | 001642 | 033136 | DT42 | : \$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC |
| 407 | 001644 | 033421 | DF42 | : 3, 0, 0, 0, 0 |
| 408 | | | | |
| 409 | | | | |
| 410 | | | :ITEM 33 | |
| 411 | 001646 | 027270 | EM52 | :SUMMARY OF DUAL MAPPING ERRORS |
| 412 | 001650 | 031214 | DH23 | :EXPECTD EXPECTD RECEIVD RECEIVD |
| 413 | | | | : 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO |
| 414 | 001652 | 033074 | DT37 | :ADDROR,ADRAND,DATAOR,DATAND,\$ERRPC,TESTNO,0 |
| 415 | 001654 | 033403 | DF37 | : 0, 0, 0, 0, 1, 0 |
| 416 | | | | |
| 417 | 001656 | | ER200: | :THIS IS THE STARTING POINT FOR ERROR MESSAGES |
| 418 | | | | :201 THRU 377. THEY ARE USED FOR MULTIPLE |
| 419 | | | | :ERROR MESSAGES. |
| 420 | | | | |
| 421 | | | :ITEM 201 | |
| 422 | 001656 | 027327 | EM201 | :THE FOLLOWING REGISTERS TIMED OUT WHEN READ |
| 423 | 001660 | 032101 | DH201 | :REGADRS TESTNO ERRORPC |
| 424 | 001662 | 033152 | DT201 | : \$REG0,TESTNO,\$ERRPC,0 |
| 425 | 001664 | 033426 | DF201 | : 0, 0, 0 |
| 426 | | | | |
| 427 | | | :ITEM 202 | |
| 428 | 001666 | 027403 | EM202 | :THE FOLLOWING MAP REGISTERS WILL NOT CLEAR |
| 429 | 001670 | 032131 | DH202 | :REGADRS DATAREC TESTNO ERRORPC. |
| 430 | 001672 | 033162 | DT202 | : \$REG0,\$TMP0,TESTNO,\$ERRPC,0 |
| 431 | 001674 | 033431 | DF202 | : 0, 0, 0, 0 |
| 432 | | | | |
| 433 | | | :ITEM 203 | |
| 434 | 001676 | 027456 | EM203 | :THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP |
| 435 | 001700 | 032171 | DH203 | :MAPREG MAPREG |
| 436 | | | | :TESTING DUALED TESTNO ERRORPC |
| 437 | 001702 | 033174 | DT203 | : \$REG0,\$REG1,\$TESTNO,\$ERRPC,0 |

```

438 001704 033435      DF203      : 0, 0, 0, 0
439
440      :ITEM 204
441 001706 027551      EM204      :THE COUNT PATTERN THRU THE MAP REGISTERS FAILED
442 001710 032250      DH204      :REGADRS PATTERN EXPECTD RECEIVD TESTNO ERRORPC
443 001712 033206      DT204      :$REG0,$REG2,$REG4,$REG3,TESTNO,$ERRPC,0
444 001714 033441      DF204      : 0, 0, 0, 0, 0, 0
445
446      :ITEM 205
447 001716 027631      EM205      :UNIBUS DATA PATH COUNT PATTERN FAILURE
448 001720 032330      DH205      :EXPECTD RECEIVD ADDRSLD TESTNO ERRORPC
449 001722 033224      DT205      :$REG1,$REG0,$REG2,TESTNO,$ERRPC,0
450 001724 033447      DF205      :0, 0, 3, 0, 0
451
452      :ITEM 206
453 001726 027700      EM206      :UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED
454 001730 032402      DH206      :ADDRESS ADDRESS
455      :EXPECTD RECEIVD TESTNO ERRORPC
456 001732 033240      DT206      :$REG0,$REG3,TESTNO,$ERRPC,0
457 001734 033454      DF206      : 0, 0, 0, 0
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487

```

.SBTTL ERROR MESSAGE TYPE OUT ROUTINE

```

:*      THIS SUBROUTINE IS CALLED BY THE ERROR HANDLER TO TYPE
:*      THE ERROR MESSAGES. IT PICKS UP THE ITEM BYTE ($ITEMB) NUMBER
:*      AND USES THAT TO INDEX THROUGH THE ERROR TABLE. THE ERROR
:*      TABLE STARTS AT '$ERRTB' AND HAS FOUR (4) POINTERS FOR EACH
:*      ENTRY, 'EM', 'DH', 'DT', 'DF'. THE 'EM' POINTS TO THE ERROR
:*      MESSAGE WHICH IS AN ASCIZ STRING. THE 'DH' POINTS TO THE DATA
:*      HEADER WHICH IS ANOTHER ASCIZ STRING. THE 'DT' POINTS TO THE
:*      DATA TABLE WHICH IS A GROUP OF WORDS CONTAINING THE ADDRESSES
:*      OF THE DATA TO BY TYPED. THE FORMAT OF THIS DATA IS
:*      CONTROLLED BY THE 'DF' WHICH IS THE POINTER TO THE DATA FORMAT.
:*      THE DATA FORMAT IS A GROUP OF BYTES WHICH CONTAIN NUMBERS
:*      THAT CORRESPOND TO DIFFERENT TYPING FORMATS.
:*      0      -16 BIT OCTAL FORMAT
:*      1      -DECIMAL FORMAT
:*      2      -22 BIT OCTAL FORMAT. DATA IS LOWER 16 BITS OF THE
:*      PHYSICAL ADDRESS, UPPER 6 BITS ARE ADJACENT TO LOWER 16
:*      3      -22 BIT OCTAL FORMAT. DATA IS THE 16 BIT VIRTUAL
:*      ADDRESS IN KERNEL I-SPACE.
:*      4      -18 BIT OCTAL FORMAT. DATA IS A 16 BIT NUMBER THAT
:*      WILL BE CONVERTED INTO A UNIBUS ADDRESS BY LEFT
:*      SHIFTING IT 6 BITS.

```

```

488 001736 010046      ERTYPE: MOV    R0,-(KSP)      ;SAVE R0 ON STACK
489 001740 005000      CLR    R0                ;CLEAR R0
490 001742 113700 001112  MOVB   @($ITEMB),R0      ;PUT ITEM NUMBER IN R0
491 001746 001004      BNE   1$                ;BRANCH IF IT IS NON-ZERO
492 001750 016746 177140  MOV    $ERRPC,-(KSP)    ;PUT ERROR PC ON STACK FOR TYPING
493 001754 104402      TYPOC ;TYPE FAILING PC
494 001756 000526      BR    13$              ;GO TO RETURN

```

```

495 001760 005300          1$: DEC R0 ;ADJUST ITEM NUMBER TO BE A POINTER
496 001762 072027 000003  ASH #3,R0 ;LEFT SHIFT ITEM NO. 3 PLACES
497 001766 100024          BPL 22$ ;BRANCH IF ITEM #LESS THAN 200
498 001770 032700 001000  BIT #BIT9,R0 ;SEE IF ITEM # WAS 3XX
499 001774 001415          BEQ 21$ ;BRANCH IF ITEM # WAS 2XX
500                                     ;AND TYPE ERROR MESSAGE
501 001776 032777 000200 177144 BIT #SW7,@SWR ;SEE IF SWITCH 7 IS UP
502 002004 001404          BEQ 20$ ;BRANCH IF SWITCH IS NOT UP
503                                     ;AND TYPE DATA, ON MULTIPLE ERRORS
504 002006 062766 000004 000002 ADD #4,2(KSP) ;SKIP 'TYPE $CRLF' IF SW 7 IS UP
505                                     ;INHIBIT MULTIPLE ERROR TYPEOUTS
506 002014 000507          BR 13$ ;BRANCH TO EXIT
507 002016 042700 177000 20$: BIC #177000,R0 ;CLEAR UPPER BYTE OF R0
508 002022 062700 001662  ADD #ER200+4,R0 ;POINT TO DATA TABLE ENTRY
509 002026 000424          BR 5$ ;GO TYPE DATA TABLE
510 002030 042700 177000 21$: BIC #177000,R0 ;CLEAR UPPER BYTE OF R0
511 002034 062700 000330  ADD #<ER200-$ERRTB>,R0 ;ADD DIFFERENCE BETWEEN
512                                     ;ITEM 1 AND ITEM 201
513                                     GET POINTER TO ERROR MESSAGE AND TYPE IT
514                                     IF THE POINTER IS NOT ZERO
515 002040 062700 001326 22$: ADD #ERRTB,R0 ;ADD BASE OF ERROR TABLE
516 002044 012067 000004  MOV (R0)+,2$ ;PUT MESSAGE POINTER IN TYPE STATEMENT
517 002050 001404          BEQ 3$ ;BRANCH IF NO ERROR MESSAGE
518 002052 104400          TYPE ;TYPE ERROR MESSAGE
519 002054 000000 2$: .WORD 0 ;POINTER TO ERROR MESSAGE
520 002056 104400 001217  TYPE , $CRLF ;TYPE CRLF
521                                     GET THE POINTER TO THE DATA HEADER AND
522                                     TYPE IT IF THE POINTER IS NOT ZERO
523 002062 012067 000004 3$: MOV (R0)+,4$ ;PUT HEADER POINTER IN TYPE STATEMENT
524 002066 001404          BEQ 5$ ;BRANCH IF NO DATA HEADER
525 002070 104400          TYPE ;TYPE THE DATA HEADER
526 002072 000000 4$: .WORD 0 ;POINTER TO DATA HEADER
527 002074 104400 001217  TYPE , $CRLF ;TYPE CRLF
528                                     THIS IS THE START OF THE DATA OUTPUT IF THE
529                                     DATA POINTER IS NOT ZERO. R0 POINTS TO THE
530                                     DATA FORMAT, R1 POINTS TO THE ADDRESS OF
531                                     THE DATA WORDS.
532 002100 010146 5$: MOV R1,-(KSP) ;SAVE R1 ON THE STACK
533 002102 012001  MOV (R0)+,R1 ;PUT DATA TABLE POINTER IN R1
534 002104 001452          BEQ 12$ ;BRANCH IF NO DATA TABLE
535 002106 012000  MOV (R0)+,R0 ;PICK UP DATA FORMAT POINTER
536 002110 105710 6$: TSTB (R0) ;IS THIS WORD OCTAL
537 002112 001003  BNE 7$ ;BRANCH IF NOT 16-BIT OCTAL
538                                     THIS IS 16 BIT OCTAL FORMAT (DF = 0)
539 002114 013146  MOV @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
540 002116 104402  TYPOC ;TYPE THE WORD ON STACK AS 16 BIT OCTAL
541 002120 000436  BR 11$ ;GET READY FOR NEXT WORD
542 002122 122710 000001 7$: (MPB #1,(R0) ;IS THE WORD DECIMAL
543 002126 001003  BNE 8$ ;BRANCH IF NOT DECIMAL
544                                     THIS IS DECIMAL FORMAT (DF = 1)
545 002130 013146  MOV @ (R1)+,-(KSP) ;PUT WORD ON STACK FOR TYPING
546 002132 104410  TYPDS ;TYPE THE WORD ON STACK AS DECIMAL
547 002134 000430  BR 11$ ;GET READY FOR NEXT WORD
548 002136 122710 000002 8$: (MPB #2,(R0) ;IS WORD 22-BIT PHYSICAL ADDRESS
549 002142 001012  BNE 9$ ;BRANCH IF NOT 22-BIT PHYSICAL ADDR
550                                     THIS IS 22-BIT PHYSICAL FORMAT (DF = 2)
551 002144 012146  MOV (R1)+,-(KSP) ;PUT ADDR OF LOW WORD ON STACK
    
```

```

552 002146 004767 001562 JSR PC,$DB20 ;CONVERT NUMBER TO OCTAL ASCIZ
553 002152 062716 000003 ADD #3,(KSP) ;ONLY WANT 8 D'GITS
554 002156 012667 000002 MOV (KSP)+,30$ ;PUT POINTER AFTER 'TYPE' CALL
555 002162 104400 TYPE ;TYPE ASCIZ STRING
556 002164 000000 30$: .WORD 0 ;WORD HOLDS POINTER TO ASCIZ STRING
557 002166 000413 BR 11$ ;GET READY FOR NEXT WORD
558 002170 122710 000003 9$: (MPB #3,(R0) ;IS THIS A 16-BIT VIRTUAL ADDRESS
559 002174 001004 BNE 10$ ;BRANCH IF NOT 16-BIT VIRT. ADDR.
560 ;: THIS IS 22-BIT VIRTUAL ADDRESS FORMAT
561 ;: KERNEL I-SPACE ASSUMED. (DF = 3)
562 002176 013146 MOV @ (R1)+,-(KSP) ;PUT 16-BIT VIRTUAL ADDR ON STACK
563 002200 004767 000040 JSR PC,TYPVAD ;GO TYPE 22-BIT ADDRESS FROM 16-BIT V.A.
564 002204 000404 BR 11$ ;GET READY FOR NEXT WORD
565 ;: THIS IS FORMAT 4. DATA WORD IS A UNIBUS ADDRESS
566 ;: OUTPUT WILL BE 18-BITS WORD LEFT SHIFTED 6.
567 002206 013146 10$: MOV @ (R1)+,-(KSP) ;PUSH 16-BIT UNIBUS ADDRESS ON STACK
568 002210 004767 000*36 JSR PC,UBADDR ;CONVERT TO 18-BIT UNIBUS ADDR AND TYPE
569 002214 000400 BR 11$ ;GET READY FOR NEXT WORD
570 002216 005200 11$: INC R0 ;POINT TO NEXT FORMAT BYTE
571 002220 104400 002240 TYPE ,32$ ;TYPE TWO SPACES
572 002224 005711 TST (R1) ;IS THERE ANOTHER WORD?
573 002226 001401 BEQ 12$ ;BRANCH IF ALL DONE
574 002230 000727 BR 6$ ;GO BACK FOR NEXT NUMBER
575 002232 012601 12$: MOV (KSP)+,R1 ;RESTORE R1
576 002234 012600 13$: MOV (KSP)+,R0 ;RESTORE R0
577 002236 000207 RTS PC ;RETURN TO ERROR ROUTINE
578 002240 040 000 32$: .ASCIZ ? ? ;TWO SPACES
579 .EVEN

```

```

582 .SBTTL CONVERT 16-BIT VIRTUAL ADDRESS TO 22-BIT PHYSICAL ADDRESS
583 ;*
584 ;* THIS ROUTINE IS CALLED BY A 'JSR PC' AFTER THE VIRTUAL ADDRESS
585 ;* IS PUSHED ON THE KERNEL STACK. THE V.A. IS THEN LOADED INTO
586 ;* R1 AND THE UPPER 3 BITS ARE SHIFTED INTO R0 TO SELECT THE
587 ;* CORRECT KERNEL I-SPACE PAR. THE LOWER 12 BITS OF THE VIRTUAL
588 ;* ADDRESS ARE ADDED TO THE PAR AS THEY ARE BY MEMORY MANAGEMENT
589 ;* AND THE PHYSICAL ADDRESS IS SAVED IN MEMORY TO BE CONVERTED
590 ;* TO ASCIZ AND TYPED.
591 ;*

```

```

592 002244 104412 TYPVAD: SAVREG ;SAVE ALL REGISTERS
593 002246 016601 000002 MOV 2(KSP),R1 ;PUT VIRTUAL ADDR IN R1
594 002252 005000 CLR R0 ;CLEAR R0 FOR CALCULATIONS
595 002254 073027 000003 ASHC #3,R0 ;LEFT SHIFT R0,R1 3 PLACES
596 002260 006300 ASL R0 ;LEFT SHIFT R0 ONE MORE PLACE
597 002262 006001 ROR R1 ;RIGHT SHIFT R1 SO OFFSET IS CORRECT
598 002264 006001 ROR R1 ;RIGHT SHIFT R1
599 002266 006001 ROR R1 ;RIGHT SHIFT R1
600 002270 062700 172340 ADD #KIPAR0,R0 ;FORM DESIRED PAR ADDR IN R0
601 002274 011003 MOV (R0),R3 ;PUT CONTENTS OF PAR IN R3
602 002276 005002 CLR R2 ;CLEAR R2 FOR PHYSICAL ADDR CALCULATION
603 002300 073227 000006 ASHC #6,R2 ;LEFT SHIFT <R2,R3> 6 PLACES
604 002304 060103 ADD R1,R3 ;ADD OFFSET IN R1 TO BASE IN R3
605 002306 005502 ADC R2 ;ADD ANY POSSIBLE CARRY TO UPPER 6 BITS
606 002310 010267 176710 MOV R2,PADRSR ;PUT UPPER 6 BITS OF ADDR IN CORE
607 002314 010367 176702 MOV R3,PADRSL ;PUT LOWER 16 BITS OF ADDR IN CORE
608 002320 012746 001222 MOV #PADRSL,-(KSP) ;PUT POINTER TO LOWER 16 BITS ON STACK

```

```

609 002324 004767 001404      JSR    PC,$DB20      ;CONVERT NUMBER TO OCTAL ASCIZ
610 002330 062716 000003      ADD    #3,(KSP)      ;ONLY TYPE 8 DIGITS
611 002334 012667 000002      MOV    (KSP)+,3$     ;PUT POINTER AFTER TYPE INST
612 002340 104400                TYPE                                ;TYPE THE 22-BIT VIRTUAL ADDRESS
613 002342 000000      3$:  .WORD    0      ;THIS WORD HOLDS THE POINTER TO
614                                ;THE ASCIZ STRING
615 002344 104414      RESREG                                ;RESTORE ALL THE REGISTERS
616 002346 012616      MOV    (KSP)+,(KSP)  ;LEAVE ONLY RETURN ADDR ON STACK
617 002350 000207      RTS    PC           ;RETURN TO ERROR HANDLER

```

```

618
619
620      ;*THIS SUBROUTINE IS USED TO CONVERT THE A WORD PUSHED
621      ;*ON THE STACK INTO A UNIBUS ADDRESS AND TYPE IT AS A
622      ;*6 DIGIT NUMBER. IT USES R1 & R0 AND LEAVES
623      ;*ALL OTHER REGISTERS UNCHANGED.

```

```

624 002352 104412      UBADDR: SAVREG
625 002354 016601 000002      MOV    2(KSP),R1    ;LOAD 16 BIT ADDRESS INTO R1
626 002360 005000      CLR    R0          ;CLEAR R0 FOR CALCULATIONS
627 002362 073027 000006      ASHC  #6,R0        ;LEFT SHIFT <R0:R1> 6 PLACES
628 002366 010167 176630      MOV    R1,PADRSL   ;PUT LOWER 16 BITS IN PADRSL
629 002372 010067 176626      MOV    R0,PADRSH   ;PUT UPPER 6 BITS IN PADRSH
630 002376 012746 001222      MOV    #PADRSL,-(KSP) ;PUSH POINTER TO WORDS ON STACK
631 002402 004767 001326      JSR    PC,$DB20    ;JUMP TO CONVERT ROUTINE
632 002406 062716 000005      ADD    #5,(KSP)    ;ONLY USE LOWER 6 CHARS.
633 002412 012667 000002      MOV    (KSP)+,3$   ;PUT POINTER AFTER TYPE CALL.
634 002416 104400      TYPE
635 002420 000000      3$:  .WORD    0      ;HOLDS POINTER TO FIRST CHAR.
636 002422 104414      RESREG
637 002424 012616      MOV    (KSP)+,(KSP) ;LEAVE ONLY RETURN ADDRESS ON STACK
638 002426 000207      RTS    PC           ;RETURN TO ERROR TYPE ROUTINE.

```

```

639
640
641      ;:*****

```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

```

;*SAVE R0-R5
;*CALL:
;* SAVREG
;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
;*
;*TOP---(+16)
;* +2---(+18)
;* +4---R5
;* +6---R4
;* +8---R3
;*+10---R2
;*+12---R1
;*+14---R0

```

```

002430      $SAVREG:
002430      MOV    R0,-(SP)  ;: PUSH R0 ON STACK
002432      MOV    R1,-(SP)  ;: PUSH R1 ON STACK
002434      MOV    R2,-(SP)  ;: PUSH R2 ON STACK
002436      MOV    R3,-(SP)  ;: PUSH R3 ON STACK
002440      MOV    R4,-(SP)  ;: PUSH R4 ON STACK
002442      MOV    R5,-(SP)  ;: PUSH R5 ON STACK

```

```

002444 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PS OF MAIN FLOW
002450 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PC OF MAIN FLOW
002454 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PS OF CALL
002460 016646 000022      MOV      22(SP),-(SP)    ;;SAVE PC OF CALL
002464 000002
    
```

```

;*RESTORE R0-R5
;*CALL:
;*
;* RESREG
$RESREG:
    
```

```

002466 012666 000022      MOV      (SP)+,22(SP)    ;;RESTORE PC OF CALL
002472 012666 000022      MOV      (SP)+,22(SP)    ;;RESTORE PS OF CALL
002476 012666 000022      MOV      (SP)+,22(SP)    ;;RESTORE PC OF MAIN FLOW
002502 012666 000022      MOV      (SP)+,22(SP)    ;;RESTORE PS OF MAIN FLOW
002506 012605              MOV      (SP)+,R5        ;;POP STACK INTO R5
002510 012604              MOV      (SP)+,R4        ;;POP STACK INTO R4
002512 012603              MOV      (SP)+,R3        ;;POP STACK INTO R3
002514 012602              MOV      (SP)+,R2        ;;POP STACK INTO R2
002516 012601              MOV      (SP)+,R1        ;;POP STACK INTO R1
002520 012600              MOV      (SP)+,R0        ;;POP STACK INTO R0
002522 000002      RTI
    
```

642

.SBTTL TYPE ROUTINE

```

;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
;*NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
;*NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
;*NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
    
```

```

;*CALL:
;*1) USING A TRAP INSTRUCTION
;*   TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;*OR
;*   TYPE
;*   MESADR
;*2) USING A JSR INSTRUCTION
;*   MOV      PS,-(SP)      ;;PUSH PROCESSOR STATUS WORD ON THE STACK
;*   JSR      PC,$TYPE      ;;CALL TYPE ROUTINE
;*   MESADDR      ;;FIRST ADDRESS OF MESSAGE
    
```

```

002524 105767 176417      $TYPE: TSTB      $TPFLG      ;;IS THERE A TERMINAL?
002530 100002              BPL      1$              ;;BR IF YES
002532 000000              HALT                    ;;HALT HERE IF NO TERMINAL
002534 000413              BR      3$              ;;LEAVE
002536 010046              1$:  MOV      R0,-(SP)      ;;SAVE R0
002540 017600 000002      MOV      @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
    
```

```

002544 132767 000040 0152E3      BITB      #40,$ENVM      ;;** SUPPRESS TYPEOUTS?
002552 001003              BNE      60$            ;;** YES
    
```

```

002554 112046              2$:  MOVB      (R0)+,-(SP)  ;;PUSH CHARACTER TO BE TYPED ONTO STACK
002556 001005              BNE      4$              ;;BR IF IT ISN'T THE TERMINATOR
002560 005726              TST      (SP)+          ;;IF TERMINATOR POP IT OFF THE STACK
002562 01260C              4$:  MOV      (SP)+,R0      ;;** RESTORE R0
    
```



```

002564 062716 000002 3$: ADD #2,(SP) ::ADJUST RETURN PC
002570 000002 RTI ::RETURN
002572 122716 000011 4$: CMPB #HT,(SP) ::BRANCH IF <HT>
002576 001426 BEQ 8$
002600 122716 000200 CMPB #CRLF,(SP) ::BRANCH IF NOT
002604 001004 BNE 5$
002606 005726 TST (SP)+ ::POP <CR><LF> EQUIV
002610 104400 001217 TYPE $CRLF
002614 000757 BR 2$ ::GET NEXT CHARACTER
002616 004767 000056 5$: JSR PC,$TYPEC ::GO TYPE THIS CHARACTER
002622 126726 176320 6$: CMPB $FILLC,(SP)+ ::IS IT TIME FOR FILLER CHARS.?
002626 001352 BNE 2$ ::IF NO GO GET NEXT CHAR.
002630 016746 176310 MOV $NULL,-(SP) ::GET # OF FILLER CHARS. NEEDED
::AND THE NULL CHAR.
002634 105366 000001 7$: DECB 1(SP) ::DOES A NULL NEED TO BE TYPED?
002640 002770 BLT 6$ ::BR IF NO--GO POP THE NULL OFF OF STACK
002642 004767 000032 JSR PC,$TYPEC ::GO TYPE A NULL
002646 105367 000142 DECB $CHARCNT ::DON'T COUNT THE NULL AS A CHARACTER
002652 000770 BR 7$ ::LOOP

```

:::HORIZONTAL TAB PROCESSOR

```

002654 112716 000040 8$: MOVB #' ,(SP) ::REPLACE TAB WITH SPACE
002660 004767 000014 9$: JSR PC,$TYPEC ::TYPE A SPACE
002664 132767 000007 000122 BITB #7,$CHARCNT ::BRANCH IF NOT AT
002672 001372 BNE 9$ ::TAB STOP
002674 005726 TST (SP)+ ::POP SPACE OFF STACK
002676 000726 BR 2$ ::GET NEXT CHARACTER
002700 105777 176234 $TYPEC: TSTB @STPS ::WAIT UNTIL PRINTER IS READY
002704 100375 BPL $TYPEC
002706 116677 000002 176226 MOVB 2(SP),@STPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
002714 105777 176214 TSTB @STKS ::** SEE IF KEYBOARD IS TALKING
002720 100021 BPL 2$ ::**
002722 017746 176210 MOV @STKB,-(SP) ::**
002726 042716 177600 BIC #177600,(SP) ::** BIT CLEAR TOP BYTE AND PARITY BIT
002732 022726 000023 CMP #23,(SP)+ ::** SEE IF IT IS A ^S
002736 001012 BNE 2$ ::** BR IF IT ISN'T
002740 105777 176170 3$: TSTB @STKS ::**
002744 100375 BPL 3$ ::** BR BACK IF NOT READY
002746 017746 176164 MOV @STKB,-(SP) ::** PUSH NEXT CHARACTER ON STACK
002752 042716 177600 BIC #177600,(SP) ::**
002756 022726 000021 CMP #21,(SP)+ ::** IS IT A ^Q
002762 001366 BNE 3$ ::** BR IF NOT
002764 122766 000015 000002 2$: CMPB #CR,2(SP) ::BRANCH IF
002772 001003 BNE 1$ ::NOT <CR>
002774 105067 000014 CLRB $CHARCNT
003000 000406 BR $TYPEX ::EXIT
003002 122766 000012 000002 1$: CMPB #LF,2(SP) ::BRANCH IF
003010 001402 BEQ $TYPEX ::<LF>
003012 105227 INCB (PC)+ ::INC SPACE
003014 000000 $CHARCNT:=.WORD 0 ::COUNT
003016 000207 $TYPEX: RTS PC

```

:::*****

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

.*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
 *.OCTAL (ASCII) NUMBER AND TYPE IT.
 *\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE

*CALL:
 * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
 * TYPOS ;;CALL FOR TYPEOUT
 * .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
 * .BYTE M ;;M=1 OR 0
 * ;;1=TYPE LEADING ZEROS
 * ;;0=SUPPRESS LEADING ZEROS

*\$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
 *\$TYPOS OR \$TYPOC

*CALL:
 * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
 * TYPON ;;CALL FOR TYPEOUT

*\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

*CALL:
 * MOV NUM,-(SP) ;;NUMBER TO BE TYPED
 * TYPOC ;;CALL FOR TYPEOUT

| | | | | | | | |
|--------|--------|--------|--------|----------|------|------------------|------------------------------------|
| 003020 | 017646 | 000000 | | \$TYPOS: | MOV | @(SP),-(SP) | ;;PICKUP THE MODE |
| 003024 | 116667 | 000001 | 000211 | | MOVB | 1(SP),\$OFILL | ;;LOAD ZERO FILL SWITCH |
| 003032 | 112667 | 000207 | | | MOVB | (SP)+,\$SOMODE+1 | ;;NUMBER OF DIGITS TO TYPE |
| 003036 | 062716 | 000002 | | | ADD | #2,(SP) | ;;ADJUST RETURN ADDRESS |
| 003042 | 000406 | | | | BR | \$TYPON | |
| 003044 | 112767 | 000001 | 000171 | \$TYPOC: | MOVB | #1,\$OFILL | ;;SET THE ZERO FILL SWITCH |
| 003052 | 112767 | 000006 | 000165 | | MOVB | #6,\$SOMODE+1 | ;;SET FOR SIX(6) DIGITS |
| 003060 | 112767 | 000005 | 000154 | \$TYPON: | MOVB | #5,\$SOCNT | ;;SET THE ITERATION COUNT |
| 003066 | 010346 | | | | MOV | R3,-(SP) | ;;SAVE R3 |
| 003070 | 010446 | | | | MOV | R4,-(SP) | ;;SAVE R4 |
| 003072 | 010546 | | | | MOV | R5,-(SP) | ;;SAVE R5 |
| 003074 | 116704 | 000145 | | | MOVB | \$SOMODE+1,R4 | ;;GET THE NUMBER OF DIGITS TO TYPE |
| 003100 | 005404 | | | | NEG | R4 | |
| 003102 | 062704 | 000006 | | | ADD | #6,R4 | ;;SUBTRACT IT FOR MAX. ALLOWED |
| 003106 | 110467 | 000132 | | | MOVB | R4,\$SOMODE | ;;SAVE IT FOR USE |
| 003112 | 116704 | 000125 | | | MOVB | \$OFILL,R4 | ;;GET THE ZERO FILL SWITCH |
| 003116 | 016605 | 000012 | | | MOV | 12(SP),R5 | ;;PICKUP THE INPUT NUMBER |
| 003122 | 005003 | | | | CLR | R3 | ;;CLEAR THE OUTPUT WORD |
| 003124 | 006105 | | | 1\$: | ROL | R5 | ;;ROTATE MSB INTO 'C' |
| 003126 | 000404 | | | | BR | 3\$ | ;;GO DO MSB |
| 003130 | 006105 | | | 2\$: | ROL | R5 | ;;FORM THIS DIGIT |
| 003132 | 006105 | | | | ROL | R5 | |
| 003134 | 006105 | | | | ROL | R5 | |
| 003136 | 010503 | | | | MOV | R5,R3 | |
| 003140 | 006103 | | | 3\$: | ROL | R3 | ;;GET LSB OF THIS DIGIT |
| 003142 | 105367 | 000076 | | | DECB | \$SOMODE | ;;TYPE THIS DIGIT? |
| 003146 | 100016 | | | | BPL | 7\$ | ;;BR IF NO |
| 003150 | 042703 | 177770 | | | BIC | #177770,R3 | ;;GET RID OF JUNK |
| 003154 | 001002 | | | | BNE | 4\$ | ;;TEST FOR 0 |
| 003156 | 005704 | | | | TST | R4 | ;;SUPPRESS THIS 0? |
| 003160 | 001403 | | | | BEQ | 5\$ | ;;BR IF YES |
| 003162 | 005204 | | | 4\$: | INC | R4 | ;;DON'T SUPPRESS ANYMORE 0'S |
| 003164 | 052703 | 000060 | | | BIS | #'0,R3 | ;;MAKE THIS DIGIT ASCII |

```

003170 052703 000040      5$:   BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
003174 110367 000040      :   MOVB     R3 ,8$      ;;SAVE FOR TYPING
003200 104400 003240      :   TYPE     ,8$      ;;GO TYPE THIS DIGIT
003204 105367 000032      7$:   DECB     $OCNT      ;;COUNT BY 1
003210 003347      :   BGT      2$      ;;BR IF MORE TO DO
003212 002402      :   BLT      6$      ;;BR IF DONE
003214 005204      :   INC      R4      ;;INSURE LAST DIGIT ISN'T A BLANK
003216 000744      :   BR       2$      ;;GO DO THE LAST DIGIT
003220 012605      6$:   MOV      (SP)+,R5      ;;RESTORE R5
003222 012604      :   MOV      (SP)+,R4      ;;RESTORE R4
003224 012603      :   MOV      (SP)+,R3      ;;RESTORE R3
003226 016666 000002 000004 :   MOV      2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
003234 012616      :   MOV      (SP)+,(SP)
003236 000002      :   RTI      ;;RETURN
003240      000      8$:   .BYTE     0      ;;STORAGE FOR ASCII DIGIT
003241      000      :   .BYTE     0      ;;TERMINATOR FOR TYPE ROUTINE
003242      000      $OCNT: .BYTE     0      ;;OCTAL DIGIT COUNTER
003243      000      $OFILL: .BYTE     0      ;;ZERO FILL SWITCH
003244 000000      $OMODE: .WORD     0      ;;NUMBER OF DIGITS TO TYPE
    
```

644

 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

;;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
 ;;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
 ;;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
 ;;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
 ;;REPLACED WITH SPACES.

```

*CALL:
*   MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
*   TYPDS      ;;GO TO THE ROUTINE
    
```

```

003246      010046      $TYPDS: MOV      R0,-(SP)      ;;PUSH R0 ON STACK
003246 010146      :   MOV      R1,-(SP)      ;;PUSH R1 ON STACK
003252 010246      :   MOV      R2,-(SP)      ;;PUSH R2 ON STACK
003254 010346      :   MOV      R3,-(SP)      ;;PUSH R3 ON STACK
003256 010546      :   MOV      R5,-(SP)      ;;PUSH R5 ON STACK
003260 012746 020200 :   MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
003264 016605 000020 :   MOV      20(SP),R5     ;;GET THE INPUT NUMBER
003270 100004      :   BPL      1$      ;;BR IF INPUT IS POS.
003272 005405      :   NEG      R5      ;;MAKE THE BINARY NUMBER POS.
003274 112766 000055 000001 :   MOVB     #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
003302 005000      1$:   CLR      R0      ;;ZERO THE CONSTANTS INDEX
003304 012703 003462 :   MOV      #$DBLK,R3     ;;SETUP THE OUTPUT POINTER
003310 112723 000040 :   MOVB     #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
003314 005002      2$:   CLR      R2      ;;CLEAR THE BCD NUMBER
003316 016001 003452 :   MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
003322 160105      3$:   SUB      R1,R5      ;;FORM THIS BCD DIGIT
003324 002402      :   BLT      4$      ;;BR IF DONE
003326 005202      :   INC      R2      ;;INCREASE THE BCD DIGIT BY 1
003330 000774      :   BR       3$
003332 060105      4$:   ADD      R1,R5      ;;ADD BACK THE CONSTANT
003334 005702      :   TST      R2      ;;CHECK IF BCD DIGIT 0
003336 001002      :   BNE      5$      ;;FALL THROUGH IF 0
003340 105716      :   TSTB     (SP)      ;;STILL DOING LEADING 0'S?
003342 100407      :   BMI      7$      ;;BR IF YES
    
```

```

003344 106316          5$: ASLB (SP)          ;;MSD?
003346 103003          BCC 6$          ;;BR IF NO
003350 116663 000001 177777  MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
003356 052702 000060          6$: BIS #'0,R2      ;;MAKE THE BCD DIGIT ASCII
003362 052702 000040          7$: BIS #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
003366 110223          MOVB R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
003370 005720          TST (R0)+        ;;JUST INCREMENTING
003372 020027 000010          CMP R0,#10      ;;CHECK THE TABLE INDEX
003376 002746          BLT 2$          ;;GO DO THE NEXT DIGIT
003400 003002          BGT 8$          ;;GO TO EXIT
003402 010502          MOV R5,R2       ;;GET THE LSD
003404 000764          BR 5$          ;;GO CHANGE TO ASCII
003406 105726          8$: TSTB (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
003410 100003          BPL 9$          ;;BR IF NO
003412 116663 177777 177776  9$: MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
003420 105013          CLRB (R3)      ;;SET THE TERMINATOR
003422 012605          MOV (SP)+,R5   ;;POP STACK INTO R5
003424 012603          MOV (SP)+,R3   ;;POP STACK INTO R3
003426 012602          MOV (SP)+,R2   ;;POP STACK INTO R2
003430 012601          MOV (SP)+,R1   ;;POP STACK INTO R1
003432 012600          MOV (SP)+,R0   ;;POP STACK INTO R0
003434 104400 003462          TYPE $DBLK     ;;NOW TYPE THE NUMBER
003440 016666 000002 000004  MOV 2(SP),4(SP) ;;ADJUST THE STACK
003446 012616          MOV (SP)+,(SP)
003450 000002          RTI          ;;RETURN TO USER
003452 023420          $DIBL: 10000.
003454 001750          1000.
003456 000144          100.
003460 000012          10.
003462          $DBLK: .BLKW 4
    
```

645

.SBTTL TRAP DECODER

;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
 ;*GO TO THAT ROUTINE.

```

003472 010046          $TRAP: MOV R0,-(SP)      ;;SAVE R0
003474 016600 000002      MOV 2(SP),R0      ;;GET TRAP ADDRESS
003500 005740          TST -(R0)        ;;BACKUP BY 2
003502 111000          MOVB (R0),R0     ;;GET RIGHT BYTE OF TRAP
003504 016000 003512      MOV $TRPAD(R0),R0 ;;INDEX TO TABLE
003510 000200          RTS R0          ;;GO TO ROUTINE
    
```

.SBTTL TRAP TABLE

;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
 ;*BY THE 'TRAP' INSTRUCTION.

```

: ROUTINE
: -----
003512 $TRPAD: $TYPE ;;CALL=TYPE TRAP+0(104400) TTY TYPEOUT ROUTINE
003512 002524 $TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
003514 003044
    
```

003516 003020
003520 003060
003522 003246
003524 002430
003526 002466
646 003530 004054
647 003532 004102
648
649

\$TYPOS ::CALL=TYPOS TRAP+4(104404) TYPE OCTAL NUMBER (NO LEADING ZEROS)
\$TYPON ::CALL=TYPON TRAP+6(104406) TYPE OCTAL NUMBER (AS PER LAST CALL)
\$TYPDS ::CALL=TYPDS TRAP+10(104410) TYPE DECIMAL NUMBER (WITH SIGN)
\$SAVREG ::CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE
\$RESREG ::CALL=RESREG TRAP+14(104414) RESTORE R0-R5 ROUTINE
TBITOFF ::CALL-TBITO TRAP+16(104416) THIS WILL TURN OFF T BIT TRAPPING
TBITRESTORE ::CALL=TBITR TRAP+20(104420) THIS WILL RETURN THE T BIT TO PREVIOUS

::*****

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

003534 012737 003662 000024 \$PWRDN: MOV \$SILLUP,@PWRVEC ::SET FOR FAST UP
003542 012737 000340 000026 MOV #340,@PWRVEC+2 ::PRIO:7
003550 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
003552 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
003554 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
003556 010346 MOV R3,-(SP) ::PUSH R3 ON STACK
003560 010446 MOV R4,-(SP) ::PUSH R4 ON STACK
003562 010546 MOV R5,-(SP) ::PUSH R5 ON STACK
003564 010667 000076 MOV SP,\$SAVR6 ::SAVE SP
003570 012737 003602 000024 MOV \$PWRUP,@PWRVEC ::SET UP VECTOR
003576 000000 HALT
003600 000776 BR .-2 ::HANG UP

:POWER UP ROUTINE

003602 016706 000060 \$PWRUP: MOV \$SAVR6,SP ::GET SP
003606 005067 000054 CLR \$SAVR6 ::WAIT LOOP FOR THE TTY
003612 005267 000050 1\$: INC \$SAVR6 ::WAIT FOR THE INC
003616 001375 BNE 1\$::OF WORD
003620 012605 MOV (SP)+,R5 ::POP STACK INTO R5
003622 012604 MOV (SP)+,R4 ::POP STACK INTO R4
003624 012603 MOV (SP)+,R3 ::POP STACK INTO R3
003626 012602 MOV (SP)+,R2 ::POP STACK INTO R2
003630 012601 MOV (SP)+,R1 ::POP STACK INTO R1
003632 012600 MOV (SP)+,R0 ::POP STACK INTO R0
003634 012737 003534 000024 MOV \$PWRDN,@PWRVEC ::SET UP THE POWER DOWN VECTOR
003642 012737 000340 000026 MOV #340,@PWRVEC+2 ::PRIO:7
003650 104400 TYPE ::REPORT THE POWER FAILURE
003652 003670 \$PWRMG: .WORD PWRMSG ::POWER FAIL MESSAGE POINTER
003654 012716 MOV (PC)+,(SP) ::RESTART AT START
003656 010000 \$PWRAD: .WORD START ::RESTART ADDRESS
003660 000002 RTI
003662 000000 \$SILLUP: HALT ::THE POWER UP SEQUENCE WAS STARTED
003664 000776 BR .-2 ::BEFORE THE POWER DOWN WAS COMPLETTE
003666 000000 \$SAVR6: 0 ::PUT THE SP HERE
650 003670 012 015 120 PWRMSG: .ASCII <12><15>?POWER FAILURE, RESTARTING PROGRAM?
003673 117 127 105
003676 122 040 106
003701 101 111 114
003704 125 122 105
003707 054 040 122
003712 105 123 124
003715 101 122 124
003720 111 116 107
003723 040 120 122

003726 117 107 122
 003731 101 115 000
 651
 652
 653

.EVEN

::*****

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

::*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
 ::*UNSIGNED OCTAL ASCII NUMBER.

::*CALL
 ::* MOV #PNTR,-(SP) ::: POINTER TO LOW WORD OF BINARY NUMBER
 ::* JSR PC,@#SDB20 ::: CALL THE ROUTINE
 ::* RETURN ::: THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

| | | | | | | |
|--------|--------|--------|----------|--------|----------------|--|
| 003734 | 104412 | | \$DB20: | SAVREG | | ::: SAVE ALL REGISTERS |
| 003736 | 016601 | 000002 | | MOV | 2(SP),R1 | ::: PICKUP THE POINTER TO LOW WORD |
| 003742 | 012705 | 004053 | | MOV | #SOCTVL+13.,R5 | ::: POINTER TO DATA TABLE |
| 003746 | 012704 | 000014 | | MOV | #12.,R4 | ::: DO ELEVEN CHARACTERS |
| 003752 | 012703 | 177770 | | MOV | #^C7,R3 | ::: MASK |
| 003756 | 012100 | | | MOV | (R1)+,R0 | ::: LOWER WORD |
| 003760 | 012101 | | | MOV | (R1)+,R1 | ::: HIGH WORD |
| 003762 | 005002 | | | CLR | R2 | ::: TERMINATOR |
| 003764 | 110245 | | 1\$: | MOVB | R2,-(R5) | ::: PUT CHARACTER IN DATA TABLE |
| 003766 | 010002 | | | MOV | R0,R2 | ::: GET THIS DIGIT |
| 003770 | 005304 | | | DEC | R4 | ::: COUNT THIS CHARACTER |
| 003772 | 003007 | | | BGT | 3\$ | ::: BR IF NOT THE LAST DIGIT |
| 003774 | 001405 | | | BEQ | 2\$ | ::: BR IF IT IS THE LAST DIGIT |
| 003776 | 005205 | | | INC | R5 | ::: ALL DIGITS DONE-ADJUST POINTER FOR FIRST |
| 004000 | 010566 | 000002 | | MOV | R5,2(SP) | ::: ASCII CHAR. & PUT IT ON THE STACK |
| 004004 | 104414 | | | RESREG | | ::: RESTORE ALL REGISTERS |
| 004006 | 000207 | | | RTS | PC | ::: RETURN TO USER |
| 004010 | 006203 | | 2\$: | ASR | R3 | ::: POSITION THE MASK FOR THE LAST DIGIT |
| 004012 | 006001 | | 3\$: | ROR | R1 | ::: POSITION THE BINARY NUMBER FOR |
| 004014 | 006000 | | | ROR | R0 | ::: THE NEXT OCTAL DIGIT |
| 004016 | 006001 | | | ROR | R1 | |
| 004020 | 006000 | | | ROR | R0 | |
| 004022 | 006001 | | | ROR | R1 | |
| 004024 | 006000 | | | ROR | R0 | |
| 004026 | 040302 | | | BIC | R3,R2 | ::: MASK OUT ALL JUNK |
| 004030 | 062702 | 000060 | | ADD | #'0,R2 | ::: MAKE THIS CHAR. ASCII |
| 004034 | 000753 | | | BR | 1\$ | ::: GO PUT IT IN THE DATA TABLE |
| 004036 | | | \$OCTVL: | .BLKB | 14. | ::: RESERVE DATA TABLE |

655

657
658
659
660

.SBTTL ***** SUBROUTINES UNIQUE TO THIS PROGRAM *****

***** SUBROUTINES UNIQUE TO THIS PROGRAM *****

```

662
663
664
665
666
667
668
669
670
671
672      000020
673 004054
674 004054 032766 000020 000002
675 004062 001406
676 004064 016667 000002 175214
677 004072 042766 000020 000002
678 004100 000006
679
680
681
682
683
684
685
686
687
688
689
690
691
692 004102
693 004102 016766 175200 000002

```

```

.SBTTL  TURN OFF AND SAVE T-BIT
*****
THIS SUBROUTINE IS REACHED BY THE TRAP CALL 'TBITO', IT IS
USED TO TURN OFF THE T-BIT IF IT IS ON.  THE PROCESSOR STATUS
IS SAVED IN 'OLDPSW' SO THAT THE T-BIT CAN BE RESTORED TO ITS
PREVIOUS STATUS WHEN CONDITIONS WARRANT.
*****
      TBIT=BIT4
TBITOFF:
      BIT      #TBIT,2(KSP)      :IS THE T-BIT ON?
      BEQ      1$                :BRANCH TO EXIT IF IT IS NOT ON
      MOV      2(KSP),OLDPSW     :SAVE OLD PSW FOR RESTORING T BIT
      BIC      #TBIT,2(KSP)     :CLEAR T BIT IF IT IS ON
1$:      RTT                    :RETURN TO PROGRAM

```

```

.SBTTL  RESTORE T-BIT TO ITS PREVIOUS CONDITION
*****
THIS SUBROUTINE CAN BE REACHED BY THE TRAP CALL 'TBITR', IT IS
USED TO RESTORE THE T-BIT AFTER A PARTICULAR TES THAT CANNOT
BE RUN WITH THE T-BIT ON.  IT USES THE PROCESSOR STATUS STORED
IN 'OLDPSW' BY 'TBITO', REPLACES THE PS ON THE STACK WITH IT
AND DOES AN 'RTT'.
*****
TBITRESTORE:
      MOV      OLDPSW,2(KSP)     :PUT OLD PSW ON STACK

```

```

695 004110 042767 000020 175170      BIC      #TBIT,OLDPSW      ;CLEAR T-BIT IN 'OLDPSW' SO THAT
696                                     ;IT WON'T BE TURNED ON BY ACCIDENT
697 004116 000006                      RTT                          ;RETURN TO PROGRAM AND INHIBIT
698                                     ;T BIT TRAP AFTER THIS INSTRUCTION.
699
700
701

```

.SBTTL SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS

 * THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS BY LOADING
 * THE ADDRESS OF MAPLOO INTO R3 AND THEN CLEARING THE
 * REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE MAPH37.

```

702                                     .SBTTL SUBROUTINE TO CLEAR ALL OF THE MAP REGISTERS
703                                     *****
704                                     * THIS SUBROUTINE CLEARS ALL OF THE MAP REGISTERS BY LOADING
705                                     * THE ADDRESS OF MAPLOO INTO R3 AND THEN CLEARING THE
706                                     * REGISTER POINTED TO BY R3 UNTIL R3 POINTS ABOVE MAPH37.
707                                     *****
709 CLRMAP: MOV      #MAPLO,R3          ;PUT FIRST MAP ADDR IN R3
710 1$: CLR      (R3)+                ;CLEAR MAP REGS
711 111 004124 005023                 ;SEE IF LAST ADDR IS IN R3
712 112 004126 022703 170376        CMP      #MAPH37,R3
713 113 004132 103374                 ;BRANCH IF R3 LE THAN LAST ADDR
714 114 004134 000207                 ;RETURN TO MAIN PROGRAM
715                                     *****

```

.SBTTL SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS

 * THIS SUBROUTINE IS USED TO LOG AND REPORT THE FACT THAT A
 * REFERENCE TO A MAPPING REGISTER TIMED OUT ON THE UNIBUS. IT
 * KEEPS A 'LOGICAL AND' AND A 'LOGICAL OR' OF EACH ADDRESS THAT
 * TIMES OUT.

```

724                                     .SBTTL SUBROUTINE TO LOG AND REPORT TIMEOUTS OF MAP REGISTERS
725                                     *****
726 004136 0052.                     TIMEOUT:                      ;STARTING ADDRESS OF SUBROUTINE
727 004140 177777                    TOFLAG: .WORD      -1          ;INCREMENT ONE TIME GATE
728 004142 001401                    BEQ      10$             ;ONE TIME ENTRANCE FLAG
729 004144 000000                    HALT                    ;BRANCH IF FLAG IS NOW ZERO
730                                     ;I HAVE ENTERED THIS ROUTINE BEFORE
731                                     ;I FINISHED REPORTING THE FIRST ERROR
732                                     ;THE SECOND ENTRY ADDRESS IS ON THE
733                                     ;STACK AND THE FIRST ERROR CONDITION
734 004146 012667 175130              10$: MOV      (KSP)+,OLDPC      ;SAVE RETURN ADDRESS
735 004152 012667 175126              MOV      (KSP)+,OLDPS      ;SAVE OLD PSW
736 004156 016767 173604 175104      MOV      CPUERR,PCPUER     ;SAVE CPU ERROR REGISTER
737 004164 016767 175100 173574      MOV      PCPUER,CPUERR     ;CLEAR CPU ERROR REGISTER
738 004172 026767 175072 175066      CMP      PCPUER,CPUEXP     ;SEE IF EXPECTED CONDITION CAME UP.
739 004200 001402                    BEQ      1$              ;BRANCH IF IT WAS A TIMEOUT
740 004202 104001                    EMT+    1
741 004204 000414                    BR      3$              ;BRANCH TO EXIT
742 004206 050067 175016              1$: BIS      R0,ADDROR     ;PERFORM LOGIAL OR OF FAILING ADDRESS
743 004212 005100                    COM      R0              ;GET R0 READY FOR AND
744 004214 040067 175006              BIC      R0,ADRAND        ;PERFORM LOGICAL AND
745 004222 005767 175030              COM      R0              ;PUT R0 BACK AS IT WAS
746 004226 001002                    TST      ERRCNT          ;IS THIS THE FIRST ERROR
747 004232 104201                    BNE      2$              ;BRANCH IF NOT FIRST ERROR
748 004234 000401                    EMT+    201
749 004234 104301                    BR      3$              ;BRANCH TO EXIT
750 004234 104301                    EMT+    301

```


797 004410
 004410 104303
 798 004412 000177 174664
 799

1\$:
 EMT+ 303
 2\$: JMP @OLDPC ;RETURN TO TEST

800
 801
 802
 803
 804
 805
 806
 807
 808
 809

.SBTTL SUBROUTINE TO REPORT COUNT PATTERN ERRORS IN MAP REGISTERS

 :* THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN
 :* THE COUNT PATTERN IN THE MAP REGISTERS. IT KEEPS THE LOGICAL
 :* 'AND' AND 'OR' OF THE FAILING REGISTER'S ADDRESS, DATA RECEIVED,
 :* AND PATTERN LOADED. EVERY ERROR IS REPORTED AND AT THE END OF
 :* A TEST AN ERROR SUMMARY IS GIVEN.
 :*

810 004416 012667 174660
 811 004422 050067 174602
 812 004426 005100
 004430 040067 174572
 004434 005100
 813 004436 050367 174572
 814 004442 005103
 004444 040367 174562
 004450 005103
 815 004452 050467 174562
 816 004456 005104
 004460 040467 174552
 004464 005104
 817 004466 005767 174564
 818 004472 001002
 819 004474 104204
 820 004476 000401
 821 004500
 004500 104304
 822 004502 000177 174574

COUNT: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
 BIS R0,ADDROR ;LOGICAL OR OF BAD ADDRESSES
 COM R0 ;GET R0 READY FOR AND
 BIC R0,ADRAND ;PERFORM LOGICAL AND
 COM R0 ;PUT R0 BACK AS IT WAS
 BIS R3,DATAOR ;LOGICAL OR OF BAD DATA
 COM R3 ;GET R3 READY FOR AND
 BIC R3,DATAND ;PERFORM LOGICAL AND
 COM R3 ;PUT R3 BACK AS IT WAS
 BIS R4,PATTOR ;LOGICAL OR OF PATTERN LOADED
 COM R4 ;GET R4 READY FOR AND
 BIC R4,PATAND ;PERFORM LOGICAL AND
 COM R4 ;PUT R4 BACK AS IT WAS
 TST ERRCNT ;IS THIS FIRST ERROR
 BNE 1\$;BRANCH IF NOT FIRST ERROR
 EMT+ 204
 BR 2\$;SKIP NEXT STATEMENT
 1\$:
 EMT+ 304
 2\$: JMP @OLDPC ;RETURN TO THE TEST

823
 824
 825
 826
 827
 828
 829
 830
 831
 832
 833
 834

.SBTTL SUBROUTINE TO REPORT COUNT PATTERN ERRORS ON UNIBUS DATA PATH

 :* THIS SUBROUTINE IS CALLED WHENEVER AN ERROR IS DISCOVERED IN THE
 :* COUNT PATTERN THAT IS RUN OVER THE UNIBUS INTO MAIN MEMORY. IT
 :* KEEPS THE LOGICAL 'AND' AND 'OR' OF THE PATTERN LOADED AND THE
 :* DATA RECEIVED, AND REPORTS ALL ERRORS. AT THE END OF THE TEST
 :* IT GIVES A SUMMARY MESSAGE OF ALL THE ERRORS.
 :*

835 004506 012667 174570
 836 004512 050167 174522
 837 004516 005101
 004520 040167 174512
 004524 005101
 838 004526 050067 174502
 839 004532 005100
 004534 040067 174472
 004540 005100
 840 004542 005767 174510
 841 004546 001002

UNBCOUNT: MOV (KSP)+,OLDPC ;SAVE RETURN ADDRESS IN CASE OF ERROR LOOP
 BIS R1,PAT*OR ;LOGICAL OR OF COUNT LOADED
 COM R1 ;GET R1 READY FOR AND
 BIC R1,PATAND ;PERFORM LOGICAL AND
 COM R1 ;PUT R1 BACK AS IT WAS
 BIS R0,DATAOR ;LOGICAL OR OF DATA READ
 COM R0 ;GET R0 READY FOR AND
 BIC R0,DATAND ;PERFORM LOGICAL AND
 COM R0 ;PUT R0 BACK AS IT WAS
 TST ERRCNT ;IS THIS THE FIRST ERROR?
 BNE 1\$;BRANCH IF NOT FIRST

```

842 004550 104205          FMT+ 205
843 004552 000401          BR 2$          ;SKIP NEXT INSTRUCTION
844 004554          1$:
      004554 104305          FMT+ 305
845 004556 000177 174520 2$: JMP @OLDPC      ;RETURN TO THE TEST
846
847
848
849

```

.SBTTL ***** TRAP HANDLING ROUTINES *****

.SBTTL CPU TRAP HANDLER ROUTINE

```

*****
*
* THIS SUBROUTINE WILL HANDLE ALL CPU TRAPS AND ABORTS, THRU
* 'ERRVEC' (000004). IF THIS SUBROUTINE IS ENTERED BY A SECOND
* TRAP BEFORE THE FIRST HAS BEEN PROCESSED A HALT IS EXECUTED.
* IF THE WORD 'CPUEXP' IS ZERO, NO TRAP WAS EXPECTED AND AN
* UNEXPECTED ERROR MESSAGE IS GIVEN. IF THE WORD 'CPUEXP' IS
* NOT ZERO THEN THE CPU ERROR REGISTER 'CPUERR' IS COMPARED WITH
* 'CPUEXP' TO SEE IF THE PROPER CONDITION OCCURRED. 'PCPUER' CAN
* BE USED AS A FLAG TO INDICATE THAT A TRAP HAS OCCURRED SINCE IT
* IS LOADED WITH THE ERROR REGISTER IF A TRAP VECTORS HERE
*
*****

```

```

864 004562 005227          (CPUER: INC (PC)+          ;MAKE FLAG ZERO IF FIRST TIME
865 004564 177777          (CPFLAG: .WORD -1          ;NEGATIVE ONE FOR A FLAG
866 004566 001401          BEQ 10$          ;BRANCH IF FIRST TIME IN
867 004570 000000          HALT          ;I HAVE ENTERED THIS ROUTINE BEFORE
868                                     ;I FINISHED REPORTING THE FIRST ERROR
869                                     ;THE SECOND ENTRY ADDRESS IS ON THE
870                                     ;STACK AND THE FIRST ERROR CONDITION
871                                     ;IS PROBABLY STILL LOCKED UP
872 004572 012667 174504 10$: MOV (KSP)+,OLDPC      ;SAVE RETURN ADDRESS IN CASE OF LOOP
873 004576 012667 174502   MOV (KSP)+,OLDPS      ;SAVE OLD PSW IN CASE OF LOOP
874 004602 012706 001100   MOV #KERSTK,KSP      ;MAKE SURE THAT THE KERNEL STACK POINTER
875                                     ;IS AT 1100.
876 004606 016767 173154 174454 MOV CPUERR,PCPUER    ;SAVE CPU ERROR REGISTER
877 004614 016767 174462 174456 MOV OLDPC,BADPC      ;SAVE PC+2 AT TIME OF ABORT
878 004622 005767 174440   TST CPUEXP          ;SEE IF ANY CONDITION WAS EXPECTED
879 004626 001406          BEQ 2$          ;BRANCH IF NO TRAP WAS EXPECTED
880 004630 026767 174434 174430 CMP PCPUER,CPUEXP    ;SEE IF EXPECTED ERROR OCCURED
881 004636 001403          BEQ 1$          ;BRANCH IF ERROR CODES MATCH
882 004640 104001          EMT+ 1
883 004642 000401          BR 1$          ;SKIP NEXT INSTRUCTION
884 004644          2$:
      004644 104002          EMT+ 2
885 004646 016767 174416 173112 1$: MOV PCPUER,CPUERR    ;CLEAR CPU ERROR REGISTER
886 004654 012767 177777 177702 MOV #-1,CPFLAG      ;MAKE FLAG NEGATIVE ONE FOR NEXT TIME
887 004662 016746 174416   MOV OLDPS,-(KSP)     ;PUSH OLD PSW BACK ON STACK
888 004666 016746 174410   MOV OLDPC,-(KSP)     ;PUSH RETURN ADDRESS BACK ON STACK
889 004672 000006          RTI          ;RETURN FROM INTERRUPT OR ABOR
890
891
892
893
894
895
896

```

.SBTTL MEMORY MANAGEMENT TRAPS AND ABORTS HANDLER ROUTINE

```

*****
*
* THIS ROUTINE WILL HANDLE ALL SPURIOUS MEMORY MANAGEMENT TRAP
*
*****

```

```

897      ;*      AND ABORTS. IT WILL REPORT THE CONDITION OF ALL THE MEMORY
898      ;*      MANAGEMENT STATUS REGISTERS, AND THEN RETURN TO THE TEST AND
899      ;*      TRY TO CONTINUE RUNNING.
900      ;*
901      ;*
902      ;*
903      ;*
904      ;*
905      ;*
906      ;*
907      ;*
908      ;*
909      ;*
910      ;*
911      ;*
912      ;*
913      ;*
914      ;*
915      ;*
916      ;*
917      ;*
918      ;*
919      ;*
920      ;*
921      ;*
922      ;*
923      ;*
924      ;*
925      ;*
926      ;*
927      ;*
928      ;*
929      ;*
930      ;*
931      ;*
932      ;*
933      ;*
934      ;*
935      ;*

```

| | | | | | | | | | |
|-----|--------|--------|--------|--------|---------|---------|--------------------------------|---|--|
| 902 | 004674 | 005227 | | | MMTRAP: | INC | (PC)+ | | : MAKE FLAG ZERO IF FIRST TIME |
| 903 | 004676 | 77777 | | | MMFLAG: | .WORD | -1 | | : FLAG SHOULD BE NEG ONE |
| 904 | 004700 | 001401 | | | | BEG | 10\$ | | : BRANCH IF FIRST TIME INTO ROUTINE |
| 905 | 004702 | 000000 | | | | HALT | | | : I HAVE ENTERED THIS ROUTINE BEFORE |
| 906 | | | | | | | | | : I FINISHED REPORTING THE FIRST ERROR |
| 907 | | | | | | | | | : THE SECOND ENTRY ADDRESS IS ON THE |
| 908 | | | | | | | | | : STACK AND THE FIRST ERROR CONDITION |
| 909 | | | | | | | | | : IS PROBABLY STILL LOCKED UP |
| 910 | 004704 | 011667 | 174370 | | 10\$: | MOV | (KSP),BADPC | | : SAVE PC AT TIME OF ABORT OR TRAP |
| 911 | 004710 | 012667 | 174366 | | | MOV | (KSP)+,OLDPC | | : SAVE RETURN ADDRESS IN CASE OF LOOP |
| 912 | 004714 | 012667 | 174364 | | | MOV | (KSP)+,OLDPS | | : SAVE OLD PSW IN CASE OF LOOP |
| 913 | 004720 | 016767 | 172646 | 174362 | | MOV | MMR0,PMR0 | | : SAVE STATUS REGISTER |
| 914 | 004726 | 016767 | 172642 | 174356 | | MOV | MMR1,PMR1 | | : SAVE AUTO INC/DEC REGISTER |
| 915 | 004734 | 016767 | 172636 | 174352 | | MOV | MMR2,PMR2 | | : SAVE VIRTUAL ADDRESS REGISTER |
| 916 | 004742 | 104003 | | | | EMT+ | 3 | | |
| 917 | 004744 | 042737 | 177776 | 177572 | 1\$: | BIC | #177776,@MMR0 | | : CLEAR ALL BITS EXCEPT 0 |
| 918 | 004752 | 012767 | 177777 | 177716 | | MOV | #-1,MMFLAG | | : RESTORE A NEGATIVE ONE TO FLAG |
| 919 | 004760 | 016746 | 174320 | | | MOV | OLDPS,-(KSP) | | : PUSH OLD PSW ONTO STACK |
| 920 | 004764 | 016746 | 174312 | | | MOV | OLDPC,-(KSP) | | : PUSH RETURN ADDRESS ON STACK |
| 921 | 004770 | 000006 | | | | RTT | | | : RETURN TO MAIN PROGRAM |
| 922 | | | | | | | | | |
| 923 | | | | | | | | | |
| 924 | | | | | | | | | |
| 925 | | | | | | | | | |
| 926 | | | | | | .SBTTL | ***** | | |
| 927 | | | | | | .SBTTL | ***** START OF TEST CODE ***** | | |
| 928 | | 010000 | | | | .=10000 | | : START TEST CODE AT ADDRESS 10000 (2K) | |
| 929 | | | | | | | | | |
| 930 | 010000 | 132767 | 000200 | 010027 | START: | BITB | #200,\$ENVM | | : IS APT SIZING |
| 931 | 010006 | 001403 | | | | BEG | 1\$ | | : NO |
| 932 | 010010 | 012767 | 020036 | 171132 | | MOV | \$\$SWREG,\$WR | | : USE APT SWITCH REGISTER |
| 933 | 010016 | | | | 1\$: | | | | |
| 934 | | | | | | | | | |
| 935 | 010016 | 012737 | 000340 | 177776 | | MOV | #340,@MPS | | : LOCK OUT ALL INTERRUPTS |
| | 010024 | 012706 | 001100 | | | MOV | \$\$SCMTAG,R6 | | : FIRST LOCATION TO BE CLEARED |
| | 010030 | 005026 | | | | CLR | (R6)+ | | : CLEAR MEMORY LOCATION |
| | 010032 | 022706 | 001134 | | | CMF | \$\$TKS,R6 | | : DONE? |
| | 010036 | 001374 | | | | BNE | .-6 | | : LOOP BACK IF NO |
| | 010040 | 005067 | 007756 | | | CLR | \$PASS | | : ** INITIALIZE PASS COUNT |
| | 010044 | 012706 | 001100 | | | MOV | \$\$STACK,SP | | : SETUP THE STACK POINTER |
| | 010050 | 012737 | 020140 | 000020 | | MOV | \$\$SCOPE,@IOTVEC | | : IOT VECTOR FOR SCOPE ROUTINE |
| | 010056 | 012737 | 000340 | 000022 | | MOV | #340,@IOTVEC+2 | | : LEVEL 7 |
| | 010064 | 012737 | 020470 | 000030 | | MOV | \$\$ERROR,@EMTVEC | | : EMT VECTOR FOR ERROR ROUTINE |
| | 010072 | 012737 | 000340 | 000032 | | MOV | #340,@EMTVEC+2 | | : LEVEL 7 |
| | 010100 | 012737 | 003472 | 000034 | | MOV | \$\$TRAP,@TRAPVEC | | : TRAP VECTOR FOR TRAP CALLS |
| | 010106 | 012737 | 000340 | 000036 | | MOV | #340,@TRAPVEC+2 | | : LEVEL 7 |
| | 010114 | 012737 | 003534 | 000024 | | MOV | \$\$SPWRDN,@PWRVEC | | : POWER FAILURE VECTOR |
| | 010122 | 012737 | 000340 | 000026 | | MOV | #340,@PWRVEC+2 | | : LEVEL 7 |
| | 010130 | 016767 | 011164 | 011154 | | MOV | \$\$ENDCT,\$EOPCT | | : SETUP END-OF-PROGRAM COUNTER |
| | 010136 | 005067 | 171044 | | | CLR | \$\$TIMES | | : INITIALIZE NUMBER OF ITERATION |

***** START OF TEST CODE *****

```

010142 005067 171042 CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
010146 112767 000001 170737 MOV# #1,$ERRMAX ;:ALLOW ONE ERROR PER TEST
010154 012737 021540 000014 MOV $SRTN,@TBITVEC ;:SET 'T' BIT VECTOR TO $SRTN
010162 012737 000340 000016 MOV #340,@TBITVEC+2 ;:LEVEL 7
010170 012767 000002 011342 MOV #RTI,$SRTN ;:SET $SRTN TO A RTI
010176 012737 010224 000010 MOV #65,$PRESVEC ;:TRY TO DO A RTT
010204 005046 CLR -(SP) ;:DUMMY PS
010206 012746 010214 MOV #64,-(SP) ;:AND PC
010212 000006 RTT ;:TRY THE RTT
010214 012767 000006 011316 64$: MOV #RTT,$SRTN ;:RTT IS LEGAL--SET $SRTN TO A RTT
010222 000402 BR 66$ ;:
010224 062706 000010 65$: ADD #10,SP ;:RTT ILLEGAL--CLEAN OFF THE STACK
010230 012737 000012 000010 66$: MOV #PRESVEC+2,@PRESVEC ;:RESTORE TRAP CATCHER
010236 005067 011304 CLR $TBIT ;:CLEAR 'T' BIT SWITCH
010242 012767 010242 170634 MOV #,$LPPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
010250 012767 010250 170630 MOV #,$LPPERR ;:SETUP THE ERROR LOOP ADDRESS
+36 010256 005227 177777 INC #-1 ;:FIRST TIME?
010262 001023 BNE 67$ ;:BRANCH IF NO
010264 022737 021472 000042 CMP #SENDAD,@#42 ;:ACT-11?
010272 001417 BEQ 67$ ;:BRANCH IF YES
010274 104400 010302 TYPE ,68$ ;:TYPE ASCIZ STRING
010300 000414 BR 67$ ;:GET OVER THE ASCIZ
;:68$: .ASCIZ <CRLF>?CKKUA0 11/44 UBI MAP?<CRLF>
67$:
937 010332 LOOP:
938 010332 012767 004674 167710 MOV #MMTRAP,MMVEC ;:LOAD MEMORY MANAGEMENT TRAP
939 ;:SERVICE ROUTINE ADDRESS
940 010340 012767 000340 167704 MOV #340,MMVEC+2 ;:SET PRIORITY SEVEN
941 010346 012767 004562 167430 MOV #CPUER,ERRVEC ;:LOAD CPU TRAP SERVICE ROUTINE ADDR
942 010354 012767 000340 167424 MOV #340,ERRVEC+2 ;:SET PRIORITY SEVEN
943 010362 005067 170700 CLR CPUEXP ;:NOT EXPECTING ANY CPU ERRORS
944 010366 005037 177572 CLR @MMR0 ;:START IN 16 BIT MAPPING
945 010372 005037 172516 CLR @MMR3 ;:DISABLE MAP AND 22-BIT MAPPING
946 010376 012700 177777 MOV #-1,R0 ;:NEGATIVE ONE USED TO INITIALIZE FLAGS
947 010402 010067 174156 MOV R0,CPFLAG ;:INITIALIZE FLAGS
948 010406 010067 173526 MOV R0,TOFLAG ;:INITIALIZE FLAGS
949 010412 010067 174260 MOV R0,MMFLAG ;:INITIALIZE FLAGS
950 010416 010037 177766 MOV R0,@CPUERR ;:CLEAR CPU ERROR REGISTER
951
952
962

```

```

*****
*TEST 1 MAP REGISTER RESPONSE TEST
*
* THIS TEST IS USED TO ENSURE THAT ALL THE UNIBUS MAP REGISTERS
* CAN BE REFERENCED UNDER PROGRAM CONTROL, WITHOUT TIMING OUT.
* THE ADDRESSES OF ANY MAP REGISTERS THAT TIME OUT WILL BE REPORTED
* AND, AT THE END OF THE TEST, A SUMMARY OF THOSE REGISTERS WILL
* BE GIVEN.
*****

```

```

010422 TST1:
010422 000004 SCOPE
010424 012767 010506 170670 MOV #TST2,NXTTST ;:SAVE STARTING ADDRESS OF NEXT TEST
;:FOR ESCAPE ON PARITY ERRORS
963 010432 012767 004136 167344 20$: MOV #TIMEOUT,ERRVEC ;:LOAD ERRVEC WITH ROUTINE ADDR.
964 010440 012700 170200 MOV #MAPLO,R0 ;:PUT FIRST MAP REG ADDR IN R0
965 010444 012767 010452 170434 MOV #1,$LPPERR ;:SET LOOP ON ERROR POINTER TO 1$

```

```

966 010452 000240      1$:  NOP
967 010454 011002      MOV      (R0),R2      ;READ MAP REGISTERS TO R2
968 010456 062700 000002  ADD      #2,R0        ;POINT TONEXT MAP REGISTER
969 010462 022700 170376  CMP      #MAPH37,R0   ;COMPARE LAST MAP REG ADDR WITH R0
970 010466 103371      BHIS     1$           ;CONTINUE READING IF NOT AT LAST REG.
971 010470 012767 010432 170410  MOV      #20$,$LPERR  ;INITIALIZE LOOPING POINTER IN CASE
972                                ;OF INTERMITTENT FAULTS.
973 010476 005767 170554      TST      ERRCNT      ;SEE IF THERE WERE ANY ERRORS
974 010502 001401      BEQ      TST2        ;:NO ERRORS GO TO NEXT TEST
975 010504 104004      EMT+     4
976
988

```

TEST 2 CLEAR AND READ LOW 20 REGISTERS LOWER 16 BITS

THIS TEST WILL ENSURE THAT MAPL00 - MAPL17 WILL ALL HOLD ZERO.
IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL00 - MAPL17 WILL
GIVE MORE HELP.

```

010506
010506 000004
010510 012767 010604 170604
989 010516 012767 004562 167260
990 010524 012700 170200
991 010530 012767 010536 170350
992 010536 000240
993 010540 005010
994 010542 011067 170424
995 010546 001402
996 010550 004767 173502
997 010554 062700 000004
998 010560 022700 170274
999 010564 103364
1000 010566 012767 010524 170312
1001
1002 010574 005767 170456
1003 010600 001401
1004 010602 104005
1005
1006
1007

TST2:
SCOPE
MOV      #TST3,NXTTST  ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
MOV      #CPUER,ERRVEC ;LOAD CPU TRAP SERVICE ROUTINE ADDR
20$:    MOV      #MAPL0,R0 ;LOAD STARTING MAP REGISTER ADDR
MOV      #1$,$LPERR   ;SET LOOP ON ERROR POINTER HERE
1$:     NOP
CLR      (R0)         ;CLEAR MAP REGISTER
MOV      (R0),$TMP0   ;SEE IF MAP REGISTER IS ZERO
BEQ      2$          ;BRANCH IF REGISTER IS ZERO
JSR      PC,WRITEERROR ;REPORT AND LOG ERROR
2$:     ADD      #4,R0 ;POINT TO NEXT MAP REG LOWER 16 BITS
CMP      #MAPL17,R0  ;SEE IF THIS SECTION IS TESTED
BHIS     1$          ;BRANCH IF NOT
MOV      #20$,$LPERR ;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
TST      ERRCNT      ;WERE THERE ANY ERRORS ON THIS TEST
BEQ      TST3        ;:BRANCH IF NO ERRORS
EMT+     5
;HOLD ZERO

```


1020

 *TEST 3 CLEAR AND READ LOW 20 REGISTERS UPPER 6 BITS

THIS TEST WILL ENSURE THAT MAPH00 - MAPH17 WILL ALL HOLD ZERO.
 IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
 CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
 PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
 ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
 IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH00 - MAPH17 WILL
 GIVE MORE HELP.

TST3:

| | | | | | | | |
|--------|--------|--------|--------|--------|-------|---------------|-------------------------------------|
| 010604 | 000004 | | | | SCOPE | | |
| 010604 | 012767 | 010674 | 170506 | | MOV | #TST4,NXTTST | ;SAVE STARTING ADDRESS OF NEXT TEST |
| | | | | | | | ;FOR ESCAPE ON PARITY ERRORS |
| 1021 | 010614 | 012700 | 170202 | 20\$: | MOV | #MAPH0,R0 | ;LOAD STARTING MAP REGISTER ADDR |
| 1022 | 010620 | 012767 | 010626 | 170260 | MOV | #1\$,\$LPERR | ;SET LOOP ON ERROR POINTER HERE |
| 1023 | 010626 | 000240 | | 1\$: | NOP | | |
| 1024 | 010630 | 005010 | | | CLR | (R0) | ;CLEAR MAP REGISTER |
| 1025 | 010632 | 011067 | 170334 | | MOV | (R0),\$STMP0 | ;SEE IF MAP REGISTER IS ZERO |
| 1026 | 010636 | 001402 | | | BEQ | 2\$ | ;BRANCH IF REGISTER IS ZERO |
| 1027 | 010640 | 004767 | 173412 | | JSR | PC,WRITEERROR | ;REPORT AND LOG ERROR |
| 1028 | 010644 | 062700 | 000004 | 2\$: | ADD | #4,R0 | ;POINT TO NEXT MAP REG UPPER 6 BITS |
| 1029 | 010650 | 022700 | 170276 | | CMP | #MAPH17,R0 | ;SEE IF TESTING IS OVER |
| 1030 | 010654 | 103364 | | | BHIS | 1\$ | ;BRANCH IF MORE REGS TO TEST |
| 1031 | 010656 | 012767 | 010614 | 170222 | MOV | #20\$,\$LPERR | ;INITIALIZE LOOPING POINTER IN CASE |
| 1032 | | | | | | | ;OF INTERMITTENT FAULTS. |
| 1033 | 010664 | 005767 | 170366 | | TST | ERRCNT | ;WHERE THERE ANY ERRORS ON THIS |
| 1034 | 010670 | 001401 | | | BEQ | TST4 | ;BRANCH IF NO ERRORS |
| 1035 | 010672 | 104006 | | | EMT+ | 6 | |
| 1036 | | | | | | | ;HOLD ZERO |
| 1037 | | | | | | | |
| 1038 | | | | | | | |
| 1050 | | | | | | | |

 *TEST 4 CLEAR AND READ HIGH 20 REGISTERS LOWER 16 BITS

THIS TEST WILL ENSURE THAT MAPL20 - MAPL37 WILL ALL HOLD ZERO.
 IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
 CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
 PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
 ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
 IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPL20 - MAPL37 WILL
 GIVE MORE HELP.

TST4:

| | | | | | | | |
|--------|--------|--------|--------|--------|-------|---------------|-------------------------------------|
| 010674 | 000004 | | | | SCOPE | | |
| 010674 | 012767 | 010764 | 170416 | | MOV | #TST5,NXTTST | ;SAVE STARTING ADDRESS OF NEXT TEST |
| | | | | | | | ;FOR ESCAPE ON PARITY ERRORS |
| 1051 | 010704 | 012700 | 170300 | 20\$: | MOV | #MAPL20,R0 | ;LOAD STARTING MAP REGISTER ADDR |
| 1052 | 010710 | 012767 | 010716 | 170170 | MOV | #1\$,\$LPERR | ;SET LOOP ON ERROR POINTER HERE |
| 1053 | 010716 | 000240 | | 1\$: | NOP | | |
| 1054 | 010720 | 005010 | | | CLR | (R0) | ;CLEAR MAP REGISTER |
| 1055 | 010722 | 011067 | 170244 | | MOV | (R0),\$STMP0 | ;SEE IF MAP REGISTER IS ZERO |
| 1056 | 010726 | 001402 | | | BEQ | 2\$ | ;BRANCH IF REGISTER IS ZERO |
| 1057 | 010730 | 004767 | 173322 | | JSR | PC,WRITEERROR | ;REPORT AND LOG ERROR |

```

1058 010734 062700 000004      2$:  ADD    #4,R0          ;POINT TO NEXT REGISTER
1059 010740 022700 170374      CMP    #MAPL37,R0       ;SEE IF THIS SECTION IS TESTED
1060 010744 103364              BHIS   1$              ;BRANCH IF NOT
1061 010746 012767 010704 170132 MOV    #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
1062                                ;OF INTERMITTENT FAULTS.
1063 010754 005767 170276      TST    ERRCNT          ;WERE THERE ANY ERRORS ON THIS
1064 010760 001401              BEQ    TST5            ;:BRANCH IF NO ERRORS
1065 010762 104005              EMT+   5
1066                                ;HOLD ZERO
1067
1068
1080
  
```

 *TEST 5 CLEAR AND READ HIGH 20 REGISTERS UPPER 6 BITS

THIS TEST WILL ENSURE THAT MAPH20 - MAPH37 WILL ALL HOLD ZERO.
 IF ANY REGISTERS ARE NOT ZERO AFTER BEING CLEARED, THEIR ADDRESS AND
 CONTENTS ARE REPORTED. IF ALL THE REGISTERS HAVE RANDOM DATA THEN
 PROBABLY NO WRITE PULSE IS BEING GENERATED, BUT IF THE ERROR IS IN
 ONLY ONE REGISTER OR IN ONE BIT IT MIGHT BE MORE DIFFICULT TO FIND.
 IT IS POSSIBLE THAT THE COUNT PATTERN TEST FOR MAPH20 - MAPH37 WILL
 GIVE MORE HELP.

```

010764
010764 000004
010766 012767 011054 170326      SCOPE
MOV    #TST6,NXTTST          ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
1081 010774 012700 170302      20$:  MOV    #MAPH20,R0     ;LOAD STARTING MAP REGISTER ADDR
1082 011000 012767 011006 170100 MOV    #1$,$LPERR        ;SET LOOP ON ERROR POINTER HERE
1083 011006 000240              1$:   NOP
1084 011010 005010              CLR    (R0)            ;CLEAR MAP REGISTER
1085 011012 011067 170154      MOV    (R0),$TMP0       ;SEE IF MAP REGISTER IS ZERO
1086 011016 001402              BEQ    2$              ;BRANCH IF REGISTER IS ZERO
1087 011020 004767 173232      JSR    PC,WRITEERROR    ;REPORT AND LOG ERROR
1088 011024 062700 000004      2$:   ADD    #4,R0          ;POINT TO NEXT HIGH REGISTER
1089 011030 022700 170376      CMP    #MAPH37,R0       ;SEE IF THERE ARE MORE REGS LEFT
1090 011034 103364              BHIS   1$              ;BRANCH IF MORE REGS TO TEST
1091 011036 012767 010774 170042 MOV    #20$,$LPERR     ;INITIALIZE LOOPING POINTER IN CASE
1092                                ;OF INTERMITTENT FAULTS.
1093 011044 005767 170206      TST    ERRCNT          ;WERE THERE ANY ERRORS ON THIS TEST
1094 011050 001401              BEQ    TST6            ;:BRANCH IF NO ERRORS
1095 011052 104006              EMT+   6
1096                                ;HOLD ZERO
1097
1108
  
```

 *TEST 6 DATA PATH TEST TO MAP REGISTER LOWER 16 BITS

THIS TEST WILL ENSURE THAT THE DATA PATH TO AND FROM THE UNIBUS
 MAP REGISTERS IS FUNCTIONING PROPERLY. IT RUNS A COUNT PATTERN
 THROUGH MAPL00, AND IF IT DETECTS AN ERROR THE EXPECTED COUNT
 AND THE RECEIVED COUNT ARE REPORTED. THE TEST THEN TRIES TO RUN
 THE COUNT THROUGH MAPL20, IN CASE THE ERROR IS IN THE LOWER
 GROUP OF REGISTERS AND NOT IN THE DATA PATH.

011054
 011054 000004

 *TST6: SCOPE

```

011056 012767 011200 170236      MOV      #TST7,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
011064 012767 000012 170114      MOV      #12,$TIMES      ;;DO 12 ITERATIONS
1109 011072 005003 20$:      CLR      R3              ;START COUNT PATTERN AT 0
1110 011074 005002          CLR      R2              ;START COUNT PATTERN AT 0
1111 011076 012767 011104 170002  MOV      #1$, $LPERR      ;SET LOOP ON ERROR POINTER TO 1$
1112 011104 000240 1$:      NOP
1113 011106 010267 157066      MOV      R2,MAPL00        ;LOAD MAP REG 0 LOW 16 BITS
1114 011112 016700 157062      MOV      MAPL00,R0        ;READ CONTENTS OF MAP REGISTER 0
1115 011116 020200          CMP      R2,R0           ;COMPARE DATA & PATTERN
1116 011120 001402          BEQ      2$              ;BRANCH IF DATA DOES NOT MATCH
1117 011122 104007          EMT+    7
1118 011124 000404          BR      3$              ;BRANCH TO TEST DATA PATH WITH
1119                                ;A REGISTER OF THE NEXT GROUP
1120 011126 062702 000002 2$:      ADD      #2,R2           ;ADD 2 TO COUNT PATTERN
1121 011132 001364          BNE     1$              ;BRANCH IF COUNT NOT COMPLETE.
1122 011134 000416          BR      10$            ;DATA PATH OKAY FOR LOW 16 BITS
1123 011136 012767 011144 167742 3$:      MOV      #12$, $LPERR    ;SET LOOP ON ERROR POINTER TO 12$
1124 011144 000240 12$:      NOP
1125 011146 010367 157126      MOV      R3,MAPL20        ;LOAD MAP REG 20 LOW 16 BITS
1126 011152 016701 157122      MOV      MAPL20,R1        ;READ DATA STORED IN MAP REGISTER 20
1127 011156 020301          CMP      R3,R1           ;COMPARE DATA & PATTERN
1128 011160 001003          BNE     4$              ;CAUGHT ERROR,NOW SEE IF SAME.
1129 011162 062703 000002      ADD      #2,R3           ;ADD 2 TO COUNT PATTERN
1130 011166 001357          BNE     2$              ;BRANCH IF COUNT NOT 0
1131 011170 4$:      EMT+    10
011170 104010          MOV      #20$, $LPERR    ;SET LOOP POINTER TO START OF TEST
1132 011172 012767 011072 167706 10$:
1133
1134
1145
    
```

 *TEST 7 DATA PATH TEST TO MAP REGISTER UPPER 6 BITS

* THIS TEST WILL ENSURE THAT THE DATA PATH TO AND FROM THE UNIBUS
 * MAP REGISTERS IS FUNCTIONING PROPERLY. IT RUNS A COUNT PATTERN
 * THROUGH MAPH00, AND IF IT DETECTS AN ERROR THE EXPECTED COUNT
 * AND THE RECEIVED COUNT ARE REPORTED. THE TEST THEN TRIES TO RUN
 * THE COUNT THROUGH MAPH20, IN CASE THE ERROR IS IN THE LOWER
 * GROUP OF REGISTERS AND NOT IN THE DATA PATH.

```

011200
011200 000004
011202 012767 011322 170112      SCOPE
                                MOV      #TST10,NXTTST      ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
1146 011210 005002 20$:      CLR      R2              ;START COUNT PATTERN AT 0
1147 011212 005003          CLR      R3              ;START COUNT PATTERN AT 0
1148 011214 012767 011222 167664  MOV      #1$, $LPERR      ;SET LOOP ON ERROR POINTER TO 1$
1149 011222 000240 1$:      NOP
1150 011224 010237 170202      MOV      R2,MAPH00        ;LOAD MAP REG 00 HIGH 6 BITS
1151 011230 016700 156746      MOV      MAPH00,R0        ;READ BACK MAP REG 00 UPPER SIX BITS
1152 011234 020200          CMP      R2,R0           ;COMPARE DATA AND PATTERN LOADED
1153 011236 001402          BEQ      2$              ;BRANCH IF DATA IS CORRECT
1154 011240 104007          EMT+    7
1155 011242 000405          BR      3$              ;BRANCH TO TRY SECOND BANK OF MAP REGS
1156 011244 005202 2$:      INC      R2              ;CHANGE DATA PATTERN
1157 011246 022702 000100      CMP      #100,R2         ;HAS COUNT REACH MAXIMUM?
    
```

```

1158 011252 001363      BNE      1$      ;BRANCH IF TEST NOT OVER
1159 011254 000417      BR       10$     ;UPPER SIX BIT DATA PATH IS OKAY
1160 011256 012767 011264 167622 3$:  MOV     #12$, $LPERR ;SET LOOP ON ERROR POINTER TO 12$
1161 011264 000240      NOP
1162 011266 010337 170302  MOV     R3, @MAPH20 ;LOAD MAP REG 20 HIGH 6 BITS
1163 011272 016701 157004  MOV     MAPH20, R1 ;READ BACK MAP REG 20 UPPER SIX BITS
1164 011276 020301      CMP     R3, R1   ;COMPARE DATA AND PATTERN
1165 011300 001004      BNE     4$      ;BRANCH IF DATA DOES NOT MATCH
1166 011302 005203      INC     R3      ;CHANGE DATA PATTERN
1167 011304 022703 000100  CMP     #100, R3 ;HAS COUNT REACHED MAXIMUM?
1168 011310 001355      BNE     2$      ;BRANCH IF TEST NOT OVER
1169 011312      4$:
1170 011314 012767 011210 167564 10$: EMT+    10
1171 011314 012767 011210 167564 10$: MOV     #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
1184

```

 *TEST 10 DUAL ADDRESS LOADS & READS MAP REG'S

THIS TEST ENSURES THAT ONLY ONE UNIBUS MAP REGISTER IS LOADED DURING A 'MOV #DATA, @MAPREG' INSTRUCTION. ALL MAP REGISTERS ARE CLEARED AND ONE REGISTER AT A TIME, STARTING WITH MAPLO, IS LOADED A NEGATIVE ONE. THEN, ALL MAP REGISTERS ARE READ, STARTING WITH MAPH37, AND VERIFIED TO BE ZERO. ANY REGISTER THAT IS NOT ZERO AND WHOSE UNIBUS ADDRESS DOES NOT MATCH THAT OF THE REGISTER UNDER TEST IS REPORTED TO BE IN ERROR. AT THE END OF THE TEST A SUMMARY OF ALL DUALED REGISTERS IS GIVEN.

 TST10:

```

011322
011322 000004
011324 012767 011446 167770  SCOPE
MOV     #TST11, NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
1185 011332 012767 000144 167646 20$: MOV     #144, $TIMES ;DO 144 ITERATIONS
1186 011340 012700 170200  MOV     #MAPLO, R0 ;LOAD FIRST MAP REG ADDR INTO R0
1187 011344 012767 011352 167534  MOV     #1$, $LPERR ;SET LOOP ON ERROR POINTER HERE
1188 011352 012701 170376 1$:  MOV     #MAPH37, R1 ;PUT ADDRESS OF HIGHEST MAP REG IN R1
1189 011356 004767 172536  JSR     PC, CLRMAP ;CLEAR ALL MAP REGISTERS IN CASE OF
;A 'FLAKEY' PROBLEM
1190 011362 000240      NOP
1191 011364 012710 177777  MOV     #177777, (R0) ;LOAD MAP REGISTER UNDER TEST
1192 011370 011102 2$:  MOV     (R1), R2 ;READ MAP REGS STARTING FROM TOP
1193 011372 001404      BEQ     3$      ;GO TO 3$ IF ZERO
1194 011374 020100      CMP     R1, R0 ;SEE IF NON-ZERO ADDRS MATCH
1195 011376 001402      BEQ     3$      ;GO TO 3$ IF MATCH
1196 011400 004767 172736  JSR     PC, DUALADR ;JUMP TO SUBROUTINE TO LOG ALL ERRORS
1197 011404 162701 000002 3$:  SUB     #2, R1 ;POINT TO NEXT MAP REGISTER
1198 011410 022701 170200  CMP     #MAPLO, R1 ;HAVE ALL REGISTERS BEEN READ
1199 011414 101765      BLOS   2$      ;IF NOT CONTINUE TEST
1200 011416 062700 000002  ADD     #2, R0 ;POINT TO NEXT MAP REGISTER
1201 011422 022700 170376  CMP     #MAPH37, R0 ;SEE IF TEST IS OVER
1202 011426 103351      BHS    1$      ;CONTINUE TESTING
1203 011430 012767 011340 167450  MOV     #20$, $LPERR ;INITIALIZE LOOPING POINTER IN CASE
1204 011436 005767 167614  TST     ERRCNT ;SEE IF THERE WERE ANY ERRORS
1205 011442 001401      BEQ    TST11 ;GOTO NEXT TEST IF NO ERRORS
1206 011444 104014      EMT+    14
1207 011444 104014      EMT+    14
1208

```

;ON LOADING MAP REGISTERS

1209
1210
1211
1212
1213
1223

.*THE NEXT FOUR(4) TESTS RUN A COUNT PATTERN THROUGH EACH
.*MAP REGISTER, CHECKING FOR STUCK BITS, AND SHORTED PINS.

.....
.*TEST 11 CNT PATRN LOW 20 MAP REG'S LOW 16 BITS

.* THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
.* MAPL00 - MAPL17. IF THE COUNT RECEIVED DOES NOT MATCH THE
.* EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
.* PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED. AT THE END
.* OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
.* DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
.*

.....
TST11:

| | | | | | | |
|-------------|--------|--------|--------|-------|-------|---|
| 011446 | 011446 | 000004 | | | SCOPE | |
| 011450 | 012767 | 011572 | 167644 | | MOV | #TST12,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST |
| | | | | | | ;FOR ESCAPE ON PARITY ERRORS |
| 1224 011456 | 012767 | 000012 | 167522 | | MOV | #12,\$TIMES ;DO 12 ITERATIONS |
| 011464 | 005002 | | | 20\$: | CLR | R2 ;START WITH AN ALL ZERO PATTERN |
| 011466 | 012767 | 011506 | 167412 | | MOV | #2\$,\$LPERR ;SET LOOP ON ERROR POINTER TO 2\$ |
| 011474 | 010204 | | | 1\$: | MOV | R2,R4 ;PUT COUNT IN R4 FOR MASK |
| 011476 | 042704 | 000001 | | | BIC | #000001,R4 ;CLEAR BITS IN MASK FOR PROPER COMPARE |
| 011502 | 012700 | 170200 | | | MOV | #MAPL00,R0 ;LOAD STARTING MAP REGISTER ADDRESS |
| 011506 | 000240 | | | 2\$: | NOP | |
| 011510 | 010210 | | | | MOV | R2,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN |
| 011512 | 011003 | | | | MOV | (R0),R3 ;READ MAP REGISTER INTO R3 |
| 011514 | 020403 | | | | CMR | R4,R3 ;COMPARE MASK WITH DATA |
| 011516 | 001402 | | | | BEQ | 3\$;BRANCH IF DATA MATCHES |
| 011520 | 004767 | 172672 | | | JSR | PC,COUNT ;JUMP TO COUNT TO LOG ALL ERRORS |
| 011524 | 005003 | | | 3\$: | CLR | R3 ;CLEAR DATA HOLDER |
| 011526 | 062700 | 000004 | | | ADD | #4,R0 ;POINT TO NEXT MAP REGISTER UNDER TEST |
| 011532 | 022700 | 170274 | | | CMR | #MAPL17,R0 ;SEE IF ALL REGISTERS HAVE BEEN LOADED |
| 011536 | 103363 | | | | BHIS | 2\$;BRANCH IF STILL MORE TO GO |
| 011540 | 022702 | 177777 | | | CMR | #177777,R2 ;SEE IF COUNT HAS CYCLED |
| 011544 | 001403 | | | | BEQ | 4\$;BRANCH IF TEST IS DONE |
| 011546 | 062702 | 000401 | | | ADD | #401,R2 ;CHANGE COUNT PATTERN |
| 011552 | 000750 | | | | BR | 1\$;CONTINUE TESTING |
| 011554 | 012767 | 011464 | 167324 | 4\$: | MOV | #20\$,\$LPERR ;INITIALIZE LOOPING POINTER IN CASE |
| | | | | | | ;OF INTERMITTENT FAULTS. |
| 011562 | 005767 | 167470 | | | TST | ERRCNT ;SEE IF THERE WERE ANY ERRORS |
| 011566 | 001401 | | | | BEQ | TST12 ;BRANCH IF NO ERRORS |
| 011570 | 104015 | | | | EMT+ | 15 ;IN MAPPING REGISTERS |

1225
1226
1236

.....
.*TEST 12 COUNT PATTERN LOW 20 MAP REGISTERS UPPER 6 BITS

.* THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
.* MAPH00 - MAPH17. IF THE COUNT RECEIVED DOES NOT MATCH THE
.* EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
.* PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED. AT THE END
.* OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
.* DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
.*

.....

```

011572                                TST12:
011572 000004                          SCOPE
011574 012767 011716 167520            MOV      #TST13,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
1237 011602 012767 000144 167376      MOV      #144,$TIMES ;DO 144 ITERATIONS
011610 005002                          CLR      R2 ;START WITH AN ALL ZERO PATTERN
011612 012767 011632 167266          20$:    MOV      #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
011620 010204                          1$:    MOV      R2,R4 ;PUT COUNT IN R4 FOR MASK
011622 042704 177700                    BIC      #177700,R4 ;CLEAR BITS IN MASK FOR PROPER COMPARE
011626 012700 170202                    MOV      #MAPH00,R0 ;LOAD STARTING MAP REGISTER ADDRESS
011632 000240                          2$:    NOP
011634 010210                          MOV      R2,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN
011636 011003                          MOV      (R0),R3 ;READ MAP REGISTER INTO R3
011640 020403                          CMP      R4,R3 ;COMPARE MASK WITH DATA
011642 001402                          BEQ      3$ ;BRANCH IF DATA MATCHES
011644 004767 172546                    JSR      PC,COUNT ;JUMP TO COUNT TO LOG ALL ERRORS
011650 005003                          3$:    CLR      R3 ;CLEAR DATA HOLDER
011652 062700 000004                    ADD      #4,R0 ;POINT TO NEXT MAP REGISTER UNDER TEST
011656 022700 170276                    CMP      #MAPH17,R0 ;SEE IF ALL REGISTERS HAVE BEEN LOADED
011662 103363                          BHIS    2$ ;BRANCH IF STILL MORE TO GO
011664 022702 000177                    CMP      #177,R2 ;SEE IF COUNT HAS CYCLED
011670 001403                          BEQ      4$ ;BRANCH IF TEST IS DONE
011672 062702 000001                    ADD      #1,R2 ;CHANGE COUNT PATTERN
011676 000750                          BR       1$ ;CONTINUE TESTING
011700 012767 011610 167200          4$:    MOV      #20$, $LPERR ;INITIALIZE LOOPING POINTER IN CASE
;OF INTERMITTENT FAULTS.
011706 005767 167344                    TST      ERRCNT ;SEE IF THERE WERE ANY ERRORS
011712 001401                          BEQ     TST13 ;BRANCH IF NO ERRORS
011714 104013                          EMT+    13
;IN MAPPING REGISTERS
  
```

1238
 1239
 1240
 1250

```

*****
*TEST 13      CNT PATRN HI 20 MAP REG'S LOW 16 BITS
*
* THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
* MAPL20 - MAPL37. IF THE COUNT RECIEVED DOES NOT MATCH THE
* EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
* PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED. AT THE END
* OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
* DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
*****
  
```

```

011716                                TST13:
011716 000004                          SCOPE
011720 012767 012042 167374            MOV      #TST14,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
1251 011726 012767 000012 167252      MOV      #12,$TIMES ;DO 12 ITERATIONS
011734 005002                          CLR      R2 ;START WITH AN ALL ZERO PATTERN
011736 012767 011756 167142          20$:    MOV      #2$, $LPERR ;SET LOOP ON ERROR POINTER TO 2$
011744 010204                          1$:    MOV      R2,R4 ;PUT COUNT IN R4 FOR MASK
011746 042704 000001                    BIC      #000001,R4 ;CLEAR BITS IN MASK FOR PROPER COMPARE
011752 012700 170300                    MOV      #MAPL20,R0 ;LOAD STARTING MAP REGISTER ADDRESS
011756 000240                          2$:    NOP
011760 010210                          MOV      R2,(R0) ;LOAD MAP REGISTER WITH COUNT PATTERN
011762 011003                          MOV      (R0),R3 ;READ MAP REGISTER INTO R3
011764 020403                          CMP      R4,R3 ;COMPARE MASK WITH DATA
  
```

```

011766 001402          BEQ      3$          ;BRANCH IF DATA MATCHES
011770 004767 172422   JSR      PC,COUNT     ;JUMP TO COUNT TO LOG ALL ERRORS
011774 005003          CLR      R3           ;CLEAR DATA HOLDER
011776 062700 000004   ADD      #4,R0        ;POINT TO NEXT MAP REGISTER UNDER TEST
012002 022700 170374   CMP      #MAPL37,R0   ;SEE IF ALL REGISTERS HAVE BEEN LOADED
012006 103363          BHS     2$           ;BRANCH IF STILL MORE TO GO
012010 022702 177777   CMP      #177777,R2   ;SEE IF COUNT HAS CYCLED
012014 001403          BEQ     4$           ;BRANCH IF TEST IS DONE
012016 062702 000401   ADD      #401,R2     ;CHANGE COUNT PATTERN
012022 000750          BR      1$           ;CONTINUE TESTING
012024 012767 011734 167054 4$:  MOV     #20$,SLPERR  ;INITIALIZE LOOPING POINTER IN CASE
                                ;OF INTERMITTENT FAULTS.
012032 005767 167220   TST     ERRCNT       ;SEE IF THERE WERE ANY ERRORS
012036 001401          BEQ     TST14        ;BRANCH IF NO ERRORS
012040 104012          EMT+    12          ;IN MAPPING REGISTERS

```

1252
1253
1263

```

*****
*TEST 14      CNT PATRN HI 20 MAP REG'S UPPER 6 BITS
*
* THIS TEST WILL RUN A COUNT PATTERN THROUGH UNIBUS MAP REGISTERS
* MAPH20 - MAPH37. IF THE COUNT RECEIVED DOES NOT MATCH THE
* EXPECTED PATTERN THEN THE MAP REGISTER ADDRESS, DATA RECEIVED,
* PATTERN LOADED, AND PATTERN EXPECTED ARE REPORTED. AT THE END
* OF THE TEST A SUMMARY OF ALL ERRORS IS GIVEN SO THAT YOU CAN
* DETERMINE IF THE ERROR IS BIT SENSITIVE OR REGISTER SENSITIVE.
*****

```

```

012042          TST14:
012042 000004          SCOPE
012044 012767 012330 167250  MOV     #TST15,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
1264 012052 012767 000144 167126  MOV     #144,$TIMES   ;DO 144 ITERATIONS
012060 005002 20$:  CLR      R2           ;START WITH AN ALL ZERO PATTERN
012062 012767 012102 167016  MOV     #2$,SLPERR    ;SET LOOP ON ERROR POINTER TO 2$
012070 010204 1$:  MOV     R2,R4        ;PUT COUNT IN R4 FOR MASK
012072 042704 177700  BIC     #177700,R4    ;CLEAR BITS IN MASK FOR PROPER COMPARE
012076 012700 170302  MOV     #MAPH20,R0    ;LOAD STARTING MAP REGISTER ADDRESS
012102 000240 2$:  NOP
012104 010210  MOV     R2,(R0)     ;LOAD MAP REGISTER WITH COUNT PATTERN
012106 011003  MOV     (R0),R3      ;READ MAP REGISTER INTO R3
012110 020403  CMP     R4,R3        ;COMPARE MASK WITH DATA
012112 001402  BEQ     3$           ;BRANCH IF DATA MATCHES
012114 004767 172276   JSR     PC,COUNT     ;JUMP TO COUNT TO LOG ALL ERRORS
012120 005003 3$:  CLR      R3           ;CLEAR DATA HOLDER
012122 062700 000004   ADD     #4,R0        ;POINT TO NEXT MAP REGISTER UNDER TEST
012126 022700 170376   CMP     #MAPH37,R0   ;SEE IF ALL REGISTERS HAVE BEEN LOADED
012132 103363  BHS     2$           ;BRANCH IF STILL MORE TO GO
012134 022702 000177   CMP     #177,R2     ;SEE IF COUNT HAS CYCLED
012140 001403  BEQ     4$           ;BRANCH IF TEST IS DONE
012142 062702 000001   ADD     #1,R2       ;CHANGE COUNT PATTERN
012146 000750  BR      1$           ;CONTINUE TESTING
012150 012767 012060 166730 4$:  MOV     #20$,SLPERR  ;INITIALIZE LOOPING POINTER IN CASE
                                ;OF INTERMITTENT FAULTS.
012156 005767 167074   TST     ERRCNT       ;SEE IF THERE WERE ANY ERRORS
012162 001401  BEQ     RELC22       ;BRANCH IF NO ERRORS
012164 104013  EMT+    13

```

:IN MAPPING REGISTERS

```

1265
1266
1267
1268
1269
1270
1271
1272
1273 012166 104420          RELC22: TBITR          :RESTORE THE T BIT TO ITS CONDITION
1274                                     :BEFORE THE LAST TEST
1275 012170 012700 077406    MOV      #77406,R0      :MAKE THE KERNEL I-SPACE PAGES ALL
1276                                     :4K, UPWARD EXPANDABLE, READ/WRITE
1277 012174 010067 160100    MOV      R0,KIPDR0     :KERNEL I-SPACE PAGE 0
1278 012200 010067 160076    MOV      R0,KIPDR1     :KERNEL I-SPACE PAGE 1
1279 012204 010067 160074    MOV      R0,KIPDR2     :KERNEL I-SPACE PAGE 2
1280 012210 010067 160072    MOV      R0,KIPDR3     :KERNEL I-SPACE PAGE 3
1281 012214 010067 160070    MOV      R0,KIPDR4     :KERNEL I-SPACE PAGE 4
1282 012220 010067 160066    MOV      R0,KIPDR5     :KERNEL I-SPACE PAGE 5
1283 012224 010067 160064    MOV      R0,KIPDR6     :KERNEL I-SPACE PAGE 6
1284 012230 010067 160062    MOV      R0,KIPDR7     :KERNEL I-SPACE PAGE 7
1285 012234 012767 000000 160076    MOV      #000,KIPAR0   :MAP TO PHYSICAL 0
1286 012242 012767 000200 160072    MOV      #200,KIPAR1   :MAP TO PHYSICAL 4K - 8K
1287 012250 012767 000400 160066    MOV      #400,KIPAR2   :MAP TO PHYSICAL 8K - 12K
1288 012256 012767 000600 160062    MOV      #600,KIPAR3   :MAP TO PHYSICAL 12K - 16K
1289 012264 012767 001000 160056    MOV      #1000,KIPAR4  :MAP TO PHYSICAL 16K - 20K
1290 012272 012767 001200 160052    MOV      #1200,KIPAR5  :MAP TO PHYSICAL 20K - 24K
1291 012300 012767 170000 160046    MOV      #170000,KIPAR6 :MAP TO UNIBUS
1292 012306 012767 177600 160042    MOV      #177600,KIPAR7 :MAP TO I/O PAGE
1293 012314 012767 000001 165250    MOV      #BIT0,MMR0    :ENABLE FULL 18-BIT MAPPING
1294 012322 012767 000020 160156    MOV      #BIT4,MMR3    :ENABLE 22-BIT MAPPING
1295
1306

```

:TEST 15 MAP REGISTER ADDRESS DECODE TEST

: THIS TEST TRIES TO VERIFY THAT NONE OF THE INPUTS TO THE ADDRESS
: DECODER FOR THE UNIBUS MAP REGISTERS IS STUCK TRUE.
: KIPAR6 IS SETUP TO HOLD 177702 AND R4 HAS THE VIRTUAL ADDRESS
: TO SELECT MAPL00, THRU KIPAR6. THE TEST THEN CHANGES ONE BIT
: AT A TIME IN PAR6 SO THAT IT SHOULD NEVER REFERENCE MAPL00. IF
: IT DOES AN ERROR IS REPORTED.

TST15:

```

012330
012330 012767 012360 166546    MOV      #20$,SLPADR   :SET LOOP ON TEST POINTER TO 20$
012336 012767 012360 166542    MOV      #20$,SLPERR   :SET LOOP ON ERROR POINTER TO 20$
012344 012767 000015 166526    MOV      #15,$STSTM    :SETUP TEST NUMBER AND CLR ERROR FLAG
012352 016777 166522 166572    MOV      $STSTM,@DISPLAY :DISPLAY TEST NUMBER FOR ALL TO SEE
1307                                     TIMEOUT=BIT4
1308 012360 012767 000020 166700 20$: MOV      #TIMOUT,CPUEXP  :EXPECTING CPU TIME OUT ON UNIBUS
1309 012366 012767 012426 166512    MOV      #10$,SLPERR   :SET LOOP ON ERROR POINTER TO 10$
1310 012374 012767 177702 157752    MOV      #177702,KIPAR6 :PUT MAP REG 0 ADDR IN PAR6
1311 012402 012702 175254          MOV      #175254,R2     :PATTERN FOR TESTING.
1312 012406 010237 170200          MOV      R2,@MAPL0     :LOAD MAP REGISTER 0
1313 012412 012700 004000          MOV      #BIT11,R0     :SET BIT 11 TO FLOAT THRU PAR6
1314 012416 012704 140000          MOV      #140000,R4    :VIRT.ADDR. TO SELECT PAR6
1315 012422 074067 157726          $: XOR      R0,KIPAR6   :CHANGE A BIT OF MAP REG 0'S ADDR

```



```

1316 012426 000240      10$:  NOP
1317 012430 011401      MOV      (R4),R1      ;READ LOCATION POINTED TO BY PAR6
1318 012432 020201      LMP      R2,R1      ;SEE IF DATA FETCHED MATCHES PATTERN
1319 012434 001007      BNE      3$          ;BRANCH IF NOT SAME
1320 012436 012714 000000      MOV      #0,(R4)     ;TRY TO CLEAR THIS LOCATION
1321 012442 005737 170200      TST      @MAPLC     ;SEE IF MAP REG 0 GOT CLEARED
1322 012446 001001      BNE      2$          ;BRANCH IF MAP REG NOT ZERO
1323 012450 104014      EMT+     14
1324
1325 012452 010114      2$:  MOV      R1,(R4)     ;DIFFERENT IN ADDRESS FROM 770200
1326 012454 074067 157674      3$:  XOR      R0,KIPAR6  ;RELOAD THE LOCATION
1327 012460 006200      ASR      R0          ;RESTORE BIT TO ORIGINAL STATUS
1328 012462 020027 000001      CMP      R0,#1      ;RIGHT SHIFT ONE PLACE
1329 012466 001355      BNF      1$          ;SEE IF TEST IS OVER
1330 012470 012767 012360 166410      MOV      #20$,$LPERR ;BRANCH TO CONTINUE TEST
1331 012476 005067 166564      CLR      CPUEXP     ;SET LOOP POINTER TO START OF TEST
1332
1345
  
```

 *TEST 16 DATA PATH, UNIBUS TO MAIN MEMORY

THIS TEST RUNS A COUNT PATTERN THROUGH A MEMORY LOCATION VIA THE UNIBUS. THE UNIBUS MAP IS LEFT OFF DURING THIS TEST SO THAT THE ADDRESS IS NOT RELOCATED. THE TEST TRIES TO LOAD THE PATTERN INTO ADDRESS 040000 (8K) BUT IF THE MAP JUMPERS ARE SET NOT TO RESPOND TO THAT ADDRESS THE NEXT 4K IS TRIED UNTIL THE TEST GETS TO MAIN MEMORY FROM THE UNIBUS. IF THIS TEST DETERMINES THAT IT CANNOT GET TO MAIN MEMORY FROM THE UNIBUS IT REPORTS THE FACT AND SKIPS THE NEXT TEST FOR VERIFICATION.

 TST16:

```

012502
012502 000004
012504 012767 012734 166610      SCOPE
MOV      #TST17,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
012512 012767 000012 166466      MOV      #12,$TIMES  ;DO 12 ITERATIONS
1346 012520 004767 171374      20$:  JSR      PC,CLRMAP  ;CLEAR ALL MAP REGISTERS
1347 012524 012767 000020 166534      MOV      #TIMOUT,CPUEXP ;TIMEOUTS MIGHT OCCUR IN THIS TEST.
1348 012532 012767 012540 166346      MOV      #10$,$LPERR  ;SET LOOP ON ERROR POINTER TO 10$
1349 012540 012767 170400 157606      10$:  MOV      #170400,KIPAR6 ;START WITH ADDRESS 8K FROM UNIBUS
1350 012546 005067 166516      1$:  CLR      PCPUER    ;CLEAR ERROR CONDITION LOCATION
1351 012552 000240      NOP
1352 012554 013700 140000      MOV      @140000,R0  ;TRY TO READ ADDRESS POINTED TO BY PAR6
1353 012560 005767 166504      TST      PCPUER    ;SEE IF READ OF ADDRESS TIMED OUT
1354 012564 001412      BEQ      2$          ;BRANCH IF REFERENCE WAS GOOD
1355 012566 062767 000200 157560 4$:  ADD      #200,KIPAR6 ;TRY NEXT 4K BLOCK OF MEMORY
1356 012574 022767 177600 157552      CMP      #177600,KIPAR6 ;SEE IF YOU'VE POINTED TO I/O PAGE
1357 012602 001361      BNE      1$          ;BRANCH IF NOT
1358 012604 104015      EMT+     15
1359 012606 000167 000252      JMP      SIZEJ      ;JUMP TO SIZE JUMPER TEST
1360 012612 016767 157536 157532 2$:  MOV      KIPAR6,KIPAR5 ;PUT PAR6 INTO PAR5
1361 012620 042767 170300 157524      BIC      #170000,KIPAR5 ;MAKE PAR5 A NON UNIBUS ADDRESS
1362 012626 012737 173214 120000      MOV      #173214,@120000 ;PUT RANDOM NUMBER INTO TEST LOCATION BY FAST H
1363 012634 013701 140000      MOV      @140000,R1 ;READ TEST LOCATION VIA UNIBUS
1364 012640 022701 173214      CMP      #173214,R1 ;SEE IF DATA WAS READ PROPERLY
1365 012644 001403      BEQ      3$          ;DATA OKAY NOW VERIFY DATA PATH
1366 012646 020001      CMP      R0,R1      ;SEE IF DATA CHANGED FROM FIRST READ
1367 012650 001001      BNE      3$          ;BRANCH IF DATA CHANGED
  
```

```

1368 012652 000745          BR      4$          ;TRY NEXT 4K BLOCK
1369 012654 005001          CLR      R1          ;CLEAR REGISTER TO HOLD COUNT
1370 012656 012767 012670 166222 3$:  MOV     #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
1371 012664 012702 140000    MOV     #140000, R2   ;LOAD VIRTUAL ADDRESS INTO R2
1372 012670 000240          NOP
1373 012672 010112          5$:  MOV     R1, (R2)     ;LOAD COUNT INTO TEST LOCATION VIA U.B.
1374 012674 011200          MOV     (R2), R0     ;READ TEST LOCATION BACK VIA UNIBUS
1375 012676 020100          CMP     R1, R0       ;COMPARE COUNT WITH DATA READ
1376 012700 001402          BEQ     6$           ;BRANCH IF DATA MATCHES
1377 012702 004767 171600    JSR     PC, UBCOUNT  ;COUNT FAILED REPORT ERROR
1378 012706 005201          6$:  INC     R1           ;INCREASE COUNT
1379 012710 001370          BNE     5$           ;BRANCH IF COUNT HASN'T CYCLED
1380 012712 005767 166340    TST     ERRCNT       ;WERE THERE ANY ERRORS ON THIS TEST
1381 012716 001401          BEQ     19$          ;BRANCH IF NO ERRORS ON THIS TEST
1382 012720 104016          EMT+    16
1383                                ;DATA PATH
1384 012722 012767 012520 166156 19$: MOV     #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
1385 012730 005067 166332    CLR     CPUEXP       ;ZERO EXPECTED CPU TRAP CONDITION
1386
1387
1401
  
```

 TEST 17 MAP DOESN'T RELOCATE IF NOT ENABLED

THIS TEST VERIFIES THAT THE UNIBUS MAP DOES NOT RELOCATE IF BIT 5 OF MMR3 IS NOT SET. THE TEST ASSUMES THAT THE PREVIOUS TEST HAS RUN SUCCESSFULLY AND LEFT KIPAR6 POINTING TO THE FIRST UNIBUS MAPPING REGISTER THAT THE UNIBUS MAP WILL RESPOND TO GREATER THAN OR EQUAL TO MAPREG #2. KIPAR5 IS ALSO POINTING TO THE SAME MEMORY BASE ADDRESS EXCEPT IT POINTS OVER THE FASTBUS. THE TEST THEN SETS ONE BIT IN EACH A.L.U. OF THE UNIBUS MAP AND TRIES TO REFERENCE MAIN MEMORY OVER THE UNIBUS. SINCE THE MAP IS NOT ENABLED THE LOAD WILL GO TO MAIN MEMORY UNRELOCATED.

```

012734
012734 000004
012736 012767 013046 166356  TST17: SCOPE
                                MOV     #TST20, NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
1402 012744 012767 000020 166314 20$: MOV     #TIMOUT, CPUEXP ;MIGHT TIME OUT OVER UNIBUS
1403 012752 012767 013010 166126    MOV     #10$, $LPERR ;SET LOOP ON ERROR POINTER TO 0$
1404 012760 016700 157370    MOV     KIPAR6, R0   ;PUT UNIBUS ADDRESS OF MAP REG IN R0
1405 012764 072027 177773    ASH     #-5, R0      ;RIGHT SHIFT R0 5 PLACES
1406 012770 042700 007400    BIC     #007400, R0  ;STRIP OFF EXTRANEIOUS BITS.
1407 012774 012720 021042    MOV     #021042, (R0)+ ;SET BOTTOM BIT IN EACH ALU
1408 013000 012710 000042    MOV     #42, (R0)    ;SET BOTTOM BIT IN EACH ALU
1409 013004 005037 120000    CLR     @#120000     ;CLEAR TEST LOCATION VIA FAST BUS
1410 013010 000240
1411 013012 012737 043207 140000 10$: MOV     #43207, @#140000 ;LOAD TEST LOCATION VIA UNIBUS
1412                                ;THIS LOAD SHOULD NOT BE RELOCATED
1413                                ;BY THE UNIBUS MAP, SINCE BIT05 OF
1414                                ;MMR3 IS CLEAR.
1415 013020 013703 120000    MOV     @#120000, R3 ;READ TEST LOCATION VIA FAST BUS
1416 013024 022703 043207    CMP     #43207, R3   ;SEE IF DATA MATCHES
1417 013030 001401          BEQ     1$           ;BRANCH IF DATA GOOD
1418 013032 104017          EMT+    17
1419 013034 012767 012744 166044 1$:  MOV     #20$, $LPERR ;SET LOOP POINTER TO START OF TEST
1420 013042 005067 166220    CLR     CPUEXP       ;ZERO EXPECTED CPU TRAP CONDITION
  
```

1421
 1422
 1432

 *TEST 20 SIZE JUMPER LOCATION
 *
 * THIS TEST DETERMINES THE SETTING OF THE JUMPERS ON THE UNIBUS
 * MAP WHICH ALLOW THE MAP TO RESPOND TO THOSE ADDRESSES BETWEEN
 * THE JUMPER RANGE. THE DEFAULT SETTING ALLOWS THE MAP TO RESPOND
 * TO ADDRESSES 000000 - 757776 ON THE UNIBUS. IF THE JUMPERS ARE
 * NOT SET IN THEIR DEFAULT POSITION AN ERROR MESSAGE IS GIVEN.
 *

| | | | | | | | | |
|--------|--------|--------|--------|--------|--------|-------|------------------|---|
| 013046 | 013046 | C00004 | | | TST20: | SCOPE | | |
| 013050 | 012767 | 013420 | 166244 | | | MOV | #TST21,NXTTST | :SAVE STARTING ADDRESS OF NEXT TEST |
| | | | | | | | | :FOR FESCAPE ON PARITY ERRORS |
| 013056 | 012767 | 000001 | 166122 | | | MOV | #1,\$TIMES | ::DO 1 ITERATION |
| 1433 | 013064 | | | | SIZEJ: | MOV | #20\$,SLPADR | :SET LOOP ON TEST POINTER TO 20\$ |
| | 013064 | 012767 | 013114 | 166012 | | MOV | #20\$,SLPERR | :SET LOOP ON ERROR POINTER TO 20\$ |
| | 013072 | 012767 | 013114 | 166006 | | MOV | #20,\$TSTNM | :SETUP TEST NUMBER AND CLR ERROR FLAG |
| | 013100 | 012767 | 000020 | 165772 | | MOV | \$TSTNM,@DISPLAY | :DISPLAY TEST NUMBER FOR ALL TO SEE |
| | 013106 | 016777 | 165766 | 166036 | | MOV | #TIMOUT,CPUEXP | :EXPECTING CPU TIME OUT ON UNIBUS |
| 1434 | 013114 | 012767 | 000020 | 166144 | 20\$: | MOV | #MAPLO,R0 | :LOAD ADDRESS OF FIRST MAP REG |
| 1435 | 013122 | 012700 | 170200 | | | MOV | #20000,(R0)+ | :LOAD 4K INTO ALL MAP REGISTERS |
| 1436 | 013126 | 012720 | 020000 | | 1\$: | CLR | (R0)+ | :INSURE THAT ALL REGS HAVE UPPER BITS CLR |
| 1437 | 013132 | 005020 | | | | CMP | #MAPL37,R0 | :SEE IF LAST REG IS LOADED |
| 1438 | 013134 | 022700 | 170374 | | | BHI | 1\$ | :BRANCH IF THERE ARE MORE TO LOAD |
| 1439 | 013140 | 101372 | | | | BIS | #BITS,@MMR3 | :TURN ON MAP RELOCATION |
| 1440 | 013142 | 052737 | 000040 | 172516 | | MOV | #117776,R0 | :THIS WILL BE USED TO SELECT PAR 4 |
| 1441 | 013150 | 012700 | 117776 | | | MOV | #170000,KIPAR4 | :WE START TESTING WITH MAP 0 |
| 1442 | 013154 | 012767 | 170000 | 157166 | | MOV | #200,R1 | :CONSTANT USED TO ADD TO PAR 4 |
| 1443 | 013162 | 012701 | 000200 | | | MOV | #125252,R2 | :CONSTANT TO LOAD INTO LOCATION 37776 |
| 1444 | 013166 | 012702 | 125252 | | 2\$: | CLR | @37776 | :CLEAR TEST LOCATION |
| 1445 | 013172 | 005037 | 037776 | | | MOV | R2,(R0) | :TRY TO LOAD TEST CELL THROUGH MAP |
| 1446 | 013176 | 010210 | | | | CMP | @37776,R2 | :SEE IF TEST LOCATION WAS LOADED |
| 1447 | 013200 | 023702 | 037776 | | | BEQ | 3\$ | :BRANCH IF IT WAS LOADED |
| 1448 | 013204 | 001411 | | | | ADD | R1,KIPAR4 | :CELL NOT LOADED, TEST NEXT MAP REG |
| 1449 | 013206 | 060167 | 157136 | | | CMP | #177600,KIPAR4 | :SEE IF YOU'RE POINTING TO I/O PAGE |
| 1450 | 013212 | 022767 | 177600 | 157130 | | BNE | 2\$ | :GO TYPE NEXT MAP REGISTER |
| 1451 | 013220 | 001364 | | | | EMT+ | 20 | |
| 1452 | 013222 | 104020 | | | | JMP | START | :RESTART PROGRAM |
| 1453 | 013224 | 000167 | 174550 | | | MOV | KIPAR4,LOWEST | :FOUND THE LOWEST USABLE MAP REG |
| 1454 | 013230 | 016767 | 157114 | 166004 | 3\$: | MOV | KIPAR4,HIGEST | :THIS WILL END UP BEING LAST USABLE REG |
| 1455 | 013236 | 016767 | 157106 | 166000 | 4\$: | ADD | R1,KIPAR4 | :TRY NEXT MAP REG TO SEE IF IT RESPONDS |
| 1456 | 013244 | 060167 | 157100 | | | CMP | #177600,KIPAR4 | :SEE IF ALL MAP REGS HAVE BEEN TRIED |
| 1457 | 013250 | 022767 | 177600 | 157072 | | BEQ | 7\$ | :BRANCH IF ALL ARE DONE |
| 1458 | 013256 | 001406 | | | | CLR | @37776 | :CLEAR TEST LOCATION |
| 1459 | 013260 | 005037 | 037776 | | | MOV | R2,(R0) | :TRY TO LOAD TEST CELL THROUGH THE MAP |
| 1460 | 013264 | 010210 | | | | CMP | @37776,R2 | :SEE IF TEST LOCATION WAS LOADED |
| 1461 | 013266 | 023702 | 037776 | | | BEQ | 4\$ | :BRANCH IF IT WAS LOADED |
| 1462 | 013272 | 001761 | | | 7\$: | CLR | CPUEXP | :NO CPU TRAPS EXPECTED IN NEXT TEST |
| 1463 | 013274 | 005067 | 165766 | | | TST | \$PASS | :SEE IF THIS IS FIRST PASS |
| 1464 | 013300 | 005767 | 004516 | | | BNE | TST21 | :GO TO NEXT TEST IF NOT THE FIRST PASS |
| 1465 | 013304 | 001045 | | | | CMP | LOWEST,#170000 | :SEE IF LOWER JUMPER IS DEFAULT |
| 1466 | 013306 | 026727 | 165730 | 170000 | | BNE | 8\$ | :BRANCH IF NOT |
| 1467 | 013314 | 001004 | | | | CMP | HIGEST,#177400 | :SEE IF UPPER JUMPER IS DEFAULT |
| 1468 | 013316 | 026727 | 165722 | 177400 | | | | |

1469 013324 001401
 1470 013326
 013326 104022
 1471
 1472
 1473
 1474
 1475 013330 016700 165710
 1476 013334 016701 165702
 1477 013340 042700 170000
 1478 013344 042701 170000
 1479 013350 072027 177773
 1480 013354 072127 177773
 1481 013360 062701 170200
 1482 013364 062700 170200
 1483 013370 010167 165652
 1484 013374 062701 000002
 1485 013400 010167 165644
 1486 013404 010067 165642
 1487 013410 062700 000002
 1488 013414 010067 165634
 1489
 1490
 1491
 1504

```

    BEQ    9$          ;BRANCH IF JUMPERS DEFAULT.
    EMT+   22
    ::
    ::
    ::
    9$:   MOV    HIGEST,R0
    MOV    LOWEST,R1
    BIC    #170000,R0
    BIC    #170000,R1
    ASH    #-5,R0      ;RIGHT SHIFT R0 5 PLACES
    ASH    #-5,R1      ;RIGHT SHIFT R1 5 PLACES
    ADD    #170200,R1
    ADD    #170200,R0
    MOV    R1,LREGL
    ADD    #2,R1        ;POINT TO UPPER BITS OF MAP REG
    MOV    R1,LREGU
    MOV    R0,HREGL
    ADD    #2,R0        ;POINT TO UPPER BITS OF MAP REG
    MOV    R0,HREGU
    
```

 TEST 21 ENSURE THAT THERE IS NO DUAL MAPPING

THIS TEST VERIFIES THAT THERE IS NO DUAL MAPPING. IT CLEARS ALL THE MAP REGISTERS EXCEPT THE ONE UNDER TEST, AND LOADS THAT ONE WITH 00020000. THE TEST THEN USES A VIRTUAL ADDRESS TO SELECT THAT MAP REGISTER AND ADD 17776, SO THAT IT SHOULD REFERENCE ADDRESS 00037776. A REFERENCE IS MADE THROUGH EACH OF THE REGISTERS AND ANY THAT FETCH THE CORRECT DATA ARE CHECKED TO SEE THAT IT WAS THE MAP REGISTER UNDER TEST. IF NOT BOTH THE MAP REGISTER UNDER TEST AND THE DUALED REGISTER ARE REPORTED.

 TST21:

```

    013420
    013420 000004
    013422 012767 013614 165672
    SCOPE
    MOV    #TST22,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
    ;FOR ESCAPE ON PARITY ERRORS
    1505 013430 012767 000144 165550
    20$:   MOV    #144,$TIMES ;DO 144 ITERATIONS
    JSR    PC,CLRMAP      ;CLEAR ALL MAP REGISTERS
    CLR    $TMP0          ;USED AS FLAG IN TEST
    MOV    #100000,R3     ;SELECT P.A.R. 4 OFFSET OF ZERO
    1508 013452 012767 013504 165426
    MOV    #2$,$LPERR     ;SET LOOP ON ERROR POINTER TO 2$
    MOV    LREGL,R2       ;PUT ADDRESS OF LOWEST USABLE MAP REG IN R2
    1510 013464 016700 165552
    MOV    LOWEST,R0      ;MAP REGISTER UNDER TEST IN R0
    1511 013470 016701 165546
    1$:   MOV    LOWEST,R1  ;MAP REGISTER USED IN CURRENT REFERENCE
    MOV    #40000,(R2)    ;LOAD MAP REG UNDER TEST WITH 8K BASE
    1512 013474 012712 040000
    MOV    R2,#40000      ;LOAD TEST LOCATION WITH THE ADDRESS
    1513 013500 010237 040000
    ;OF THE MAP REGISTER UNDER TEST
    1514
    2$:   MOV    R1,KIPAR4  ;LOAD PAR 4 WITH NEXT MAP REG UNIBUS ADDR
    MOV    (R3),R4        ;READ THROUGH THE MAP
    1516 013510 011304
    CMP    R4,R2          ;SEE IF CORRECT DATA WAS FETCHED
    1517 013512 020402
    BNE   4$             ;BRANCH IF NO MATCH
    1518 013514 001010
    CMP    R0,R1         ;SEE IF MAP REGS ARE THE SAME
    1519 013516 020001
    
```

```

1520 013520 001403 BEQ 3$ :BRANCH IF CORRECT MAP REG WAS USED
1521 013522 004767 170614 JSR PC,DUALADR :LOG AND REPORT ALL ERRORS
1522 013526 000403 BR 4$ :SKIP NEXT INSTRUCTION
1523 013530 012767 000001 165434 3$: MOV #1,$TMP0 :SET FLAG WHEN ADDR MATCH
1524 013536 062701 000200 4$: ADD #200,R1 :TRY NEXT MAP REG
1525 013542 026701 165476 CMP HIGEST,R1 :SEE IF ALL HAVE BEEN TRIED
1526 013546 103356 BHIS 2$ :BRANCH IF STILL MORE TO TRY
1527 013550 005767 165416 TST $TMP0 :SEE THAT THERE WAS A SUCCESSFUL MATCH
1528 013554 001001 BNE 5$ :BRANCH IF THERE WAS
1529 013556 104023 EMT+ 23
1530 013560 005067 165406 5$: CLR $TMP0 :CLEAR FLAG FOR NEXT REG
1531 013564 005012 CLR (R2) :CLEAR MAP REG JUST TESTED
1532 013566 062702 000004 ADD #4,R2 :POINT TO NEXT MAP REG TO LOAD
1533 013572 062700 000200 ADD #200,R0 :POINT TO NEXT MAP REG UNDER TEST
1534 013576 026700 165442 CMP HIGEST,R0 :SEE IF ALL MAP REGS HAVE BEEN TESTED
1535 013602 103332 BHIS 1$ :BRANCH IF STILL MORE TO TEST
1536 013604 005767 165446 TST ERRCNT :SEE IF THERE WERE ANY ERRORS
1537 013610 001401 BEQ TST22 :BRANCH TO NEXT TEST IF NO ERRORS
1538 013612 104010 EMT+ 10
1539
1540
1541
1542
1551
    
```

```

*****
*TEST 22 LOAD LOC'S 4000-7776 WITH THEIR ADRES'S
*
* THIS TEST IS USED TO LOAD MAIN MEMORY FROM ADDRESS 00040000 TO
* ADDRESS 00077776 WITH ITS OWN ADDRESS. IT THEN CHECKS THAT
* MEMORY OVER THE UNIBUS AND LOGS AND REPORTS ANY ERRORS THAT
* IT FINDS.
*****
    
```

```

013614
013614 000004
013616 012767 014074 165476
1552 013624 042767 000040 156664
1553 013632 012767 000400 156510
1554 013640 012700 040000
1555 013644 012701 100000 1$: MOV #100000,R1 :VIRTUAL ADDRESS
1556 013650 012702 010000 1$: MOV #D4096,R2 :LOAD 4096 LOCATIONS AT A TIME
1557 013654 010021 1$: MOV R0,(R1)+ :LOAD PHY. ADDR. INTO EACH MEMORY LOC.
1558 013656 062700 000002 ADD #2,R0 :POINT TO NEXT PHYSICAL ADDRESS
1559 013662 077204 SOB R2,2$ :BRANCH IF 4K OF MEMORY NOT LOADED
1560 013664 062767 000200 156456 ADD #200,KIPAR4 :POINT TO NEXT 4K BANK OF MEMORY
1561 013672 022767 001000 156450 CMP #1000,KIPAR4 :SEE IF 16K IS LOADED
1562 013700 101361 BHI 1$ :BRANCH IF MORE MEMORY TO LOAD
1563
1564
1565
1566 013702 012767 013710 165174
1567 013710 012767 013746 165170 20$: MOV #20$,$LPADR :SET LOOP ADDRESS TO 20$
1568 013716 022767 171000 156430 MOV #4$,$LPERR :SET LOOP ON ERROR POINTER TO 4$
1569 CMP #171000,KIPAR6 :DID I USE ANY MAP REGISTER
1570 013724 101463 BLOS TST23 :BELOW REGISTER 6 (UB. ADDR 100000)
1571 013726 016700 156422 MOV KIPAR6,R0 :BRANCH TO NEXT TEST IF NOT
1572 :LOAD PAR6 INTO R0 TO GET
:THE STARTING DATA PATTERN
    
```

| | | | | | | | |
|------|--------|--------|--------|-----|-----|------------|---|
| 1573 | 013732 | 072027 | 000006 | | ASH | #6,R0 | :R0 NOW HOLDS THE STARTING DATA PATTERN |
| 1574 | 013736 | 012701 | 140000 | 38: | MOV | #140000,R1 | :STARTING VIRTUAL ADDRESS |
| 1575 | 013742 | 012702 | 010000 | | MOV | #D4096,R2 | :PREPARE TO READ 4K AT A TIME |
| 1576 | 013746 | 000240 | | 48: | NOP | | |
| 1577 | 013750 | 011103 | | | MOV | (R1),R3 | :READ MAIN MEMORY THRU UNIBUS |
| 1578 | 013752 | 020003 | | | CMF | R0,R3 | :SEE IF THE ADDRESSES MATCH |
| 1579 | 013754 | 001015 | | | BNE | ES | :BRANCH IF ERROR |
| 1580 | 013756 | 062701 | 000002 | 58: | ADD | #2,R1 | :CHANGE VIRTUAL ADDRESS |

```

1582 013762 062700 000002      ADD      #2,R0      ;CHANGE PHYSICAL ADDRESS
1583 013766 077211      SOB      R2,4$     ;BRANCH IF 4K OF MEMORY NOT READ
1584 013770 062767 000200 156356  ADD      #200,KIPAR6 ;POINT TO NEXT BANK OF 4K THRU UNIBUS
1585 013776 022767 171000 156350  CMP      #171000,KIPAR6 ;SEE IF THIS POINTS TO 16K PLUS 2
1586 014004 101354      BHI      3$       ;BRANCH IF 16K OF MEMORY NOT CHECKED
1587 014006 000423      BR       10$     ;TEST FINISHED, BRANCH TO EXIT
1588
1589 014010      6$:
014010 005100      COM      R0       ;GET R0 READY FOR AND
014012 040067 165210      BIC      R0,ADRAND ;PERFORM LOGICAL AND
014016 005100      COM      R0       ;PUT R0 BACK AS IT WAS
1590 014020 005103      COM      R3       ;GET R3 READY FOR AND
014022 040367 165204      BIC      R3,DATAND ;PERFORM LOGICAL AND
014026 005103      COM      R3       ;PUT R3 BACK AS IT WAS
1591 014030 050067 165174      BIS      R0,ADDROR ;LOGICAL OR OF PHYSICAL ADDRESS
1592 014034 050367 165174      BIS      R3,DATAOR ;LOGICAL OR OF ADDRESS FETCHED
1593 014040 005767 165212      TST      ERRCNT   ;IS THIS THE FIRST ERROR HERE?
1594 014044 001002      BNE      7$       ;BRANCH IF NOT FIRST ERROR
1595 014046 104206      EMT+     206
1596 014050 000742      BR       5$       ;CONTINUE TESTING
1597 014052      7$:
014052 104306      EMT+     306
1598 014054 000740      BR       5$       ;CONTINUE WITH TEST
1599 014056 005767 165174      10$: TST      ERRCNT ;WERE THERE ANY ERRORS ON THIS TEST?
1600 014062 001401      BEQ      19$     ;BRANCH IF NO ERRORS ON THIS TEST
1601 014064 104024      EMT+     24
1602 014066 012767 013710 165012 19$: MOV      #20$,$LPERR ;SET LOOP POINTER TO 20$
1603
1604
1613

```

```

*****
*TEST 23      MAIN MEMORY TIMEOUT THROUGH MAP
*
*      THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING
*      TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY. IT USES THE LOWEST
*      USEABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER
*      ZERO.
*****

```

```

014074
014074 000004
014076 012767 014206 165216      SCOPE
MOV      #TST24,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
1614 014104 052767 000040 156404 20$: BIS      #BITS5,MMR3 ;TURN MAP RELOCATION BACK ON
1615 014112 012767 000020 165146      MOV      #TIMOUT,CPUEXP ;EXPECTING TIMEOUT IN THIS TEST
1616 014120 016767 165116 156222      MOV      LOWEST,KIPAR4 ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
1617 014126 012777 000074 165114      MOV      #74,@LREGU ;LOAD UPPER 6 BITS OF LOWEST MAP REG
1618 014134 012777 000000 165104      MOV      #000000,@LREGL ;LOAD LOWER 16 BITS OF LOWEST MAP REG
1619 014142 012767 014150 164736      MOV      #1$,$LPERR ;SET LOOP ON ERROR POINTER TO 1$
1620 014150 005067 165114      1$: CLR      PCPUER ;CPU ERROR REG LOCATION
1621 014154 000240      NOP
1622 014156 013703 100000      MOV      @#100000,R3 ;TRY TO READ THRU PAGE 4
1623 ;THIS REFERENCE WILL GO OUT ON THE
1624 ;UNIBUS TO SELECT THE LOWEST USEABLE
1625 ;MAP REGISTER (DEFAULT MAP REG. 0).
1626 ;PHYSICAL ADDRESS 17700000 IS THEN
1627 ;GENERATED, WHICH SHOULD TIME OUT SINCE
1628 ;IT IS THE FIRST NON-EXISTANT LOCATION.

```

```

1629 014162 022767 000020 165100    CMP    #TIMOUT,PCPUER ;THE UNIBUS SHOULD HAVE TIMED OUT
1630 014170 001401                BEQ    10$             ;BRANCH IF CONDITION WAS CORRECT
1631 014172 104025                EMT+   25
1632 014174 012767 014104 164704 10$:  MOV    #20$,$LPERR   ;SET LOOP POINTER TO START OF TEST
1633 014202 005067 165060                CLR    CPUEXP        ;NO CPU TRAPS EXPECTED FOR AWHILE
1634
1635
1646
  
```

```

*****
*TEST 24      RELOCATION TEST USING LOWEST USABLE MAPPING REG
*
* THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS
* MAP A.L.U.. IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT
* IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS
* THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER.
* IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS
* THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.
*****
  
```

```

014206
014206 000004
014210 012767 015114 165104    SCOPE
                                MOV    #TST25,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
                                                ;FOR ESCAPE ON PARITY ERRORS
  
```

```

1647
1648
1649
1650
1651 014216 012767 014250 164662 100$: MOV    #1$,$LPERR   ;SET LOOP ON ERROR POINTER TO 1$
                                CLR    @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
                                MOV    #060000,@LREGL ;LOAD LOWER BITS OF MAPPING REG
                                MOV    LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
                                MOV    #140000,R0    ;SELECT PAR6, OFFSET IS 0000
                                1$:  NOP
                                MOV    (R0),R1      ;READ LOCATION 060000 THRU THE UNIBUS
                                MOV    #060000,R2    ;THE EXPECTED PHYSICAL ADDRESS IS 060000
                                CMP    R1,R2        ;SEE IF THE MAP'S FETCH WAS CORRECT
                                BEQ    2$           ;BRANCH IF FETCHED DATA MATCHES ADDRESS
                                EMT+   26
  
```

```

1652
1653
1654
1655
1656 014266 012767 014320 164612 2$:  MOV    #3$,$LPERR   ;SET LOOP ON ERROR POINTER TO 3$
                                CLR    @LREGU      ;CLEAR UPPER BITS OF MAPPING REG
                                MOV    #052524,@LREGL ;LOAD LOWER BITS OF MAPPING REG
                                MOV    LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
                                MOV    #145252,R0    ;SELECT PAR6, OFFSET IS 05252
                                3$:  NOP
                                MOV    (R0),R1      ;READ LOCATION 057776 THRU THE UNIBUS
                                MOV    #057776,R2    ;THE EXPECTED PHYSICAL ADDRESS IS 057776
                                CMP    R1,R2        ;SEE IF THE MAP'S FETCH WAS CORRECT
                                BEQ    4$           ;BRANCH IF FETCHED DATA MATCHES ADDRESS
                                EMT+   26
  
```

```

1657
1658
1659
1660
1661 014336 012767 014370 164542 4$:  MOV    #5$,$LPERR   ;SET LOOP ON ERROR POINTER TO 5$
  
```



```

014344 005077 164700 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
014350 012777 045252 164670 MOV #045252,@LREGL ;LOAD LOWER BITS OF MAPPING REG
014356 016737 164660 172354 MOV LOWEST,@KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
014364 012700 152524 MOV #152524,R0 ;SELECT PAR6, OFFSET IS 12524
014370 000240 5$: NOP
014372 011001 MOV (R0),R1 ;READ LOCATION 057776 THRU THE UNIBUS
014374 012702 057776 MOV #057776,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 057776
014400 020102 CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
014402 001401 BEQ 6$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
014404 104026 EMT+ 26
  
```

```

1662 ::
1663 ;*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
1664 ;*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
1665 ::
  
```

```

1666 014406 012767 014440 164472 6$: MOV #7$, $LPERR ;SET LOOP ON ERROR POINTER TO 7$
014414 005077 164630 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
014420 012777 050420 164620 MOV #050420,@LREGL ;LOAD LOWER BITS OF MAPPING REG
014426 016737 164610 172354 MOV LOWEST,@KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
014434 012700 150420 MOV #150420,R0 ;SELECT PAR6, OFFSET IS 10420
014440 000240 7$: NOP
014442 011001 MOV (R0),R1 ;READ LOCATION 061040 THRU THE UNIBUS
014444 012702 061040 MOV #061040,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
014450 020102 CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
014452 001401 BEQ 8$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
014454 104026 EMT+ 26
  
```

```

1667 ::
1668 ;*THE RELOCATION HERE USES A BASE OF 00054630 AND AN
1669 ;*OFFSET OF 04210 TO PRODUCE AN ADDRESS OF 00061040
1670 ::
  
```

```

1671 014456 012767 014510 164422 8$: MOV #9$, $LPERR ;SET LOOP ON ERROR POINTER TO 9$
014464 005077 164560 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
014470 012777 054630 164550 MOV #054630,@LREGL ;LOAD LOWER BITS OF MAPPING REG
014476 016737 164540 172354 MOV LOWEST,@KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
014504 012700 144210 MOV #144210,R0 ;SELECT PAR6, OFFSET IS 04210
014510 000240 9$: NOP
014512 011001 MOV (R0),R1 ;READ LOCATION 061040 THRU THE UNIBUS
014514 012702 061040 MOV #061040,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
014520 020102 CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
014522 001401 BEQ 10$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
014524 104026 EMT+ 26
  
```

```

1672 ::
1673 ;*THE RELOCATION HERE USES A BASE OF 00044210 AND AN
1674 ;*OFFSET OF 14630 TO PRODUCE AN ADDRESS OF 00061040
1675 ::
  
```

```

1676 014526 012767 014560 164352 10$: MOV #11$, $LPERR ;SET LOOP ON ERROR POINTER TO 11$
014534 005077 164510 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
014540 012777 044210 164500 MOV #044210,@LREGL ;LOAD LOWER BITS OF MAPPING REG
014546 016737 164470 172354 MOV LOWEST,@KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
014554 012700 154630 MOV #154630,R0 ;SELECT PAR6, OFFSET IS 14630
014560 000240 11$: NOP
014562 011001 MOV (R0),R1 ;READ LOCATION 061040 THRU THE UNIBUS
014564 012702 061040 MOV #061040,R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
014570 020102 CMP R1,R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
014572 001401 BEQ 12$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
014574 104026 EMT+ 26
  
```

```

1677 ::
1678 ;*THE RELOCATION HERE USES A BASE OF 00056734 AND AN
  
```

```

1679      ;*OFFSET OF 02104 TO PRODUCE AN ADDRESS OF 00061040
1680
1681 014576 012767 014630 164302 12$:  MOV    #13$, $LPERR      ;SET LOOP ON ERROR POINTER TO 13$
      014604 005077 164440          CLR    @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
      014610 012777 056734 164430  MOV    #056734, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      014616 016737 164420 172354  MOV    LOWEST, @*KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      014624 012700 142104          MOV    #142104, R0     ;SELECT PAR6, OFFSET IS 02104
      014630 000240          13$:  NOP
      014632 011001          MOV    (R0), R1       ;READ LOCATION 061040 THRU THE UNIBUS
      014634 012702 061040          MOV    #061040, R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
      014640 020102          CMP    R1, R2        ;SEE IF THE MAP'S FETCH WAS CORRECT
      014642 001401          BEQ   14$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
      014644 104026          EMT+ 26
  
```

```

1682      ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
1683      ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
1684
1685
1686 014646 012767 014700 164232 14$:  MOV    #15$, $LPERR      ;SET LOOP ON ERROR POINTER TO 15$
      014654 005077 164370          CLR    @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
      014660 012777 042104 164360  MOV    #042104, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      014666 016737 164350 172354  MOV    LOWEST, @*KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      014674 012700 156734          MOV    #156734, R0     ;SELECT PAR6, OFFSET IS 16734
      014700 000240          15$:  NOP
      014702 011001          MOV    (R0), R1       ;READ LOCATION 061040 THRU THE UNIBUS
      014704 012702 061040          MOV    #061040, R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
      014710 020102          CMP    R1, R2        ;SEE IF THE MAP'S FETCH WAS CORRECT
      014712 001401          BEQ   16$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
      014714 104026          EMT+ 26
  
```

```

1687      ;*THE RELOCATION HERE USES A BASE OF 00057776 AND AN
1688      ;*OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
1689
1690
1691 014716 012767 014750 164162 16$:  MOV    #17$, $LPERR      ;SET LOOP ON ERROR POINTER TO 17$
      014724 005077 164320          CLR    @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
      014730 012777 057776 164310  MOV    #057776, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      014736 016737 164300 172354  MOV    LOWEST, @*KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      014744 012700 141042          MOV    #141042, R0     ;SELECT PAR6, OFFSET IS 01042
      014750 000240          17$:  NOP
      014752 011001          MOV    (R0), R1       ;READ LOCATION 061040 THRU THE UNIBUS
      014754 012702 061040          MOV    #061040, R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
      014760 020102          CMP    R1, R2        ;SEE IF THE MAP'S FETCH WAS CORRECT
      014762 001401          BEQ   18$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
      014764 104026          EMT+ 26
  
```

```

1692      ;*THE RELOCATION HERE USES A BASE OF 00041042 AND AN
1693      ;*OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
1694
1695
1696 014766 012767 015020 164112 18$:  MOV    #19$, $LPERR      ;SET LOOP ON ERROR POINTER TO 19$
      014774 005077 164250          CLR    @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
      015000 012777 041042 164240  MOV    #041042, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      015006 016737 164230 172354  MOV    LOWEST, @*KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      015014 012700 157776          MOV    #157776, R0     ;SELECT PAR6, OFFSET IS 17776
      015020 000240          19$:  NOP
      015022 011001          MOV    (R0), R1       ;READ LOCATION 061040 THRU THE UNIBUS
      015024 012702 061040          MOV    #061040, R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
      015030 020102          CMP    R1, R2        ;SEE IF THE MAP'S FETCH WAS CORRECT
      015032 001401          BEQ   20$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
  
```

```
015034 104026 EMT+ 26
1697
1698
1699
1700
1701 015036 012767 015070 164042 20$: MOV #21$, $LPERR ;SET LOOP ON ERROR POINTER TO 21$
015044 005077 164200 CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
015050 012777 057776 164170 MOV #057776, @LREGL ;LOAD LOWER BITS OF MAPPING REG
015056 016737 164160 172354 MOV LOWEST, @KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
015064 012700 140002 MOV #140002, R0 ;SELECT PAR6, OFFSET IS 00002
015070 000240 21$: NOP
015072 011001 MOV (R0), R1 ;READ LOCATION 060000 THRU THE UNIBUS
015074 012702 060000 MOV #060000, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 060000
015100 020102 (MP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
015102 001401 BEQ 22$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
015104 104026 EMT+ 26
1702 015106 012767 014216 163772 22$: MOV #100$, $LPERR ;SET LOOP POINTER TO START OF TEST
1703
1704
1713
```

```
*****
*TEST 25 TEST CARRY PROP OF MAP'S RELOC ADDER
*
* EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
* WITH 00030000 UP TO 17000000. THAT IS, THE FIRST OF EVERY 2K
* WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
* WORKING PROPERLY .
*****
```

```
TST25:
015114
015114 000004 SCOPE
015116 012767 015450 164176 MOV #TST26, NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
;FOR ESCAPE ON PARITY ERRORS
015124 012767 000012 164054 MOV #12, $TIMES ;DO 12 ITERATIONS
000040 NEXMEM=BITS
1714
1715
1716 015132 012767 177777 000126 20$: MOV #-1, 4$ ;THE CPU ERROR REGISTER
1717 015140 005077 164104 CLR @LREGU ;INITIALIZE FLAG AS NEGATIVE ONE
1718 015144 012777 020000 164074 MOV #20000, @LREGL ;CLEAR UPPER 6 BITS OF MAP REG
1719 015152 012701 100100 MOV #100100, R1 ;LOAD 4K BASE INTO MAP REGISTER
1720 015156 012700 150000 MOV #150000, R0 ;LOAD BITS TO SELECT PAR 4, OFFSET 100
1721 015162 012767 000277 155160 MOV #277, KIPAR4 ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
1722 015170 016767 164046 155156 MOV LOWEST, KIPAR6 ;START WITH PHYSICAL 6K
1723 015176 012767 015236 163702 MOV #10$, $LPERR ;LOAD PAR 6 WITH MAP REG'S ADDR
1724 015204 012767 000020 164054 3$: MOV #20, CPUEXP ;SET LOOP ON ERROR POINTER TO 10$
1725 015212 005067 164052 CLR PCPUER ;EXPECTING A UNIBUS TIME OUT DURING TEST
1726 015216 016710 164102 MOV DATA, (R0) ;CLEAR TIME OUT FLAG
1727 ;THIS LOAD WILL TIME OUT WHEN YOU
1728 ;HAVE REACHED THE TOP OF MEMORY
1729 ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
1730 ;<XXX XXI>0000 ON THE UNIBUS.
1731 ;THE X'S WILL SELECT THE LOWEST USEABLE
1732 ;MAPPING REGISTER. THE DEFAULT CASE IS
015222 005767 164042 TST PCPUER ;010000, SELECTING MAPPING REGISTER 0.
015226 001016 BNE 1$ ;SEE IF THERE WAS MAIN MEMORY
1735 015230 012767 000040 164030 MOV #NEXMEM, CPUEXP ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
1736 015236 011103 10$: MOV (R1), R3 ;POSSIBLE CACHE NON-EXISTANT MEMORY
1737 015240 022767 000040 164022 (MP #NEXMEM, PCPUER ;READ TEST LOCATION VIA FASTBUS
;WAS THIS CACHE NON-EXISTANT MEMORY
```

```

1738 015246 001414      BEQ      2$      ;BRANCH IF NON-EXISTANT MEMORY
1739 015250 000240      NOP
1740 015252 011002      MOV      (R0),R2 ;READ TEST LOCATION VIA UNIBUS MAP
1741 015254 020203      CMP      R2,R3   ;COMPARE TEST DATA R2-MAP DATA
1742                                ;R3=FASTBUS DATA
1743 015256 001410      BEQ      2$      ;BRANCH IF IT WAS THE SAME
1744 015260 104027      EMT+     27
1745 015262 000406      BR       2$      ;BRANCH TO UPDATE ROUTINE
1746 015264 005227      1$: INC    (PC)+  ;INCREMENT ONE TIME ENTRANCE FLAG
1747 015266 177777      4$: .WORD  -1     ;USE NEGATIVE ONE FOR FLAG
1748 015270 001003      BNE      2$      ;BRANCH IF YOU'VE BEEN HERE BEFORE
1749 015272 016767 155052 164016  MOV      KIPAR4,RSIZE ;SAVE UPPER LIMIT OF MEMORY
1750 015300 062767 000100 164016  2$: ADD    #100,DATA ;CHANGE PATTERN FOR NEXT LOAD
1751 015306 062767 000100 155034  ADD      #100,KIPAR4 ;ADD 2K TO PAR4
1752 015314 062777 010000 163724  ADD      #10000,@LREGI ;ADD 2K TO MAP REGISTER
1753 015322 001330      BNE      3$      ;BRANCH IF MAP REGISTER NOT ZERO
1754 015324 005277 163720  INC      @LREGJ    ;ADD ONE TO UPPER 6 BITS OF MAP REG
1755 015330 022777 000073 163712  CMP      #73,@LREGU ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
1756 015336 103322      BHIS     3$      ;BRANCH IF NOT PAST IT
1757 015340 005267 163760  INC      DATA    ;CHANGE DATA PATTERN FOR NEXT PASS
1758 015344 042767 177700 163752  BIC      #177700,DATA ;CLEAR UPPER 10 BITS OF DATA PATTERN
1759 015352 052767 000300 163744  BIS      #300,DATA ;START WITH 3XX IN DATA PATTERN
1760 015360 012767 015132 163520  MOV      #20$,$LPERR ;SET LOOP POINTER TO START OF TEST
1761 015366 005067 163674  19$: CLR    (CPUEXP) ;NO CPU TRAPS EXPECTED FOR AWHILE
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772 015372 000004      UBMAP: SCOPE    ;LOOP ON PREVIOUS TEST
1773 015374 104420      TBTR      ;RESTORE T-BIT IF IT WAS ON
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785 015376 016701 163646      MOV      LREGU,R1 ;PUT POINTER TO LOWEST MAP REG IN R1
1786 015402 005061 000004      CLR      4(R1)    ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
1787 015406 005061 000010      CLR      10(R1)   ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
1788 015412 012761 020000 00000?  MOV      #20000,2(R1) ;LOAD LOWER 16 BITS OF (LOWEST + 1)
1789                                ;SO THAT IT POINTS TO 4K - 8K
1790 015420 005061 000006      CLR      6(R1)    ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
1791 015424 016701 163612      MOV      LOWEST,R1 ;PREPARE TO LOAD PAR1
1792 015430 062701 000200      ADD      #200,R1  ;POINTER TO (LOWEST + 1)
1793 015434 010167 154702      MOV      R1,KIPAR1 ;LOAD PAR1 TO POINT TO LOWEST + 1
1794 015440 062701 000200      ADD      #200,R1  ;ADJUST R1 FOR KIPARO
  
```

```

.SBTTL *****
.SBTTL THE FOLLOWING TESTS ARE RUN THROUGH THE UNIBUS MAP
.SBTTL *****
  
```

```

:
:
: THIS CODE SETS UP THE TWO MAP REGISTERS ABOVE THE LOWEST
: USEABLE ONE TO POINT TO PHYSICAL MEMORY FROM 0 - 8K. IN THE
: DEFAULT CASE, IN MANUFACTURING AND IF THERE IS NO UNIBUS MEMORY,
: THIS WILL BE MAP REGISTERS 1 AND 2. MAP REGISTER 1 WILL POINT
: TO PHYSICAL 4K - 8K SO 'ACT-11' WILL WORK PROPERLY AND MAP
: REGISTER 2 WILL POINT TO PHYSICAL 0 - 4K. THIS MEANS THAT
: KIPAR0 SHOULD GET 170400 SO IT PUTS ADDRESSES 040000 TO 057776
: ON THE UNIBUS AND KIPAR1 SHOULD GET 170200 SO IT PUTS ADDRESSES
: 020000 TO 037776 ON THE UNIBUS.
:
:
  
```

```

MOV      LREGU,R1 ;PUT POINTER TO LOWEST MAP REG IN R1
CLR      4(R1)    ;CLEAR UPPER 6 BITS OF (LOWEST + 1)
CLR      10(R1)   ;CLEAR UPPER 6 BITS OF (LOWEST + 2)
MOV      #20000,2(R1) ;LOAD LOWER 16 BITS OF (LOWEST + 1)
;SO THAT IT POINTS TO 4K - 8K
CLR      6(R1)    ;CLEAR LOWER 16 BITS OF (LOWEST + 2)
MOV      LOWEST,R1 ;PREPARE TO LOAD PAR1
ADD      #200,R1  ;POINTER TO (LOWEST + 1)
MOV      R1,KIPAR1 ;LOAD PAR1 TO POINT TO LOWEST + 1
ADD      #200,R1  ;ADJUST R1 FOR KIPARO
  
```

1795 015444 010167 154670
 1796
 1797
 1809

MOV R1,KIPAR0 ;LOAD PAR0 TO POINT TO (LOWEST + 2)

 *TEST 26 MAIN MEM. T.O. THRU MAP, CODE RUN OVER U.B.

THIS TEST GENERATES A TIME OUT THROUGH THE UNIBUS MAP BY TRYING TO REFERENCE ADDRESS 17000000 IN MAIN MEMORY. IT USES THE LOWEST USEABLE MAP REGISTER, WHICH IN THE DEFAULT CASE IS MAP REGISTER ZERO.

THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU THE UNIBUS MAP.

 *TST26:

015450
 015450 012767 015506 163426
 015456 012767 015506 163422
 015464 012767 000026 163406
 015472 016777 163402 163452
 1810 015500 012767 015602 163614
 1811 015506 012767 000020 163552
 1812 015514 016767 163522 154626
 1813 015522 012777 000074 163520
 1814 015530 012777 000000 163510
 1815 015536 012767 015544 163342
 1816 015544 005067 163520
 1817 015550 000240
 1818 015552 013703 100000
 1819
 1820
 1821
 1822
 1823
 1824
 1825 015556 022767 000020 163504
 1826 015564 001401
 1827 015566 104030
 1828 015570 012767 015506 163310
 1829 015576 005067 163464
 1830
 1831
 1845

```

MOV #20$,SLPADR ;SET LOOP ON TEST POINTER TO 20$
MOV #20$,SLPERR ;SET LOOP ON ERROR POINTER TO 20$
MOV #26,$STSTM ;SETUP TEST NUMBER AND CLR ERROR FLAG
MOV $STSTM,@DISPLAY ;DISPLAY TEST NUMBER FOR ALL TO SEE
MOV #TST27,NXTTST ;SET UP ESCAPE VECTOR IN CASE OF PARITY ERRORS
20$: MOV #20,CPUEXP ;EXPECTING CPU TIMEOUT IN THIS TEST
MOV LOWEST,KIPAR4 ;LOAD PAR 4 WITH LOWEST USABLE MAP REG
MOV #74,@LREGU ;LOAD UPPER 6 BITS OF LOWEST MAP REG
MOV #000000,@LREGL ;LOAD LOWER 16 BITS OF LOWEST MAP REG
1$: MOV #1$,SLPERR ;SET LOOP ON ERROR POINTER TO 1$
CLR PCPUER ;CPU ERROR REG LOCATION
NOP
MOV @#100000,R3 ;TRY TO READ THRU PAGE 4
;THIS REFERENCE WILL GO OUT ON THE
;UNIBUS TO SELECT THE LOWEST USEABLE
;MAP REGISTER (DEFAULT MAP REG. 0).
;PHYSICAL ADDRESS 17700000 IS THEN
;GENERATED, WHICH SHOULD TIME OUT SINCE
;IT IS THE FIRST NON-FIXISTANT LOCATION.
;THE UNIBUS SHOULD HAVE TIMED OUT
CMP #20,PCPUER ;BRANCH IF UNIBUS TIMED OUT
BEQ 10$
EMT+ 30
10$: MOV #20$,SLPERR ;SET LOOP POINTER TO START OF TEST
CLR CPUEXP ;NO CPU TRAPS EXPECTED IN NEXT TEST
    
```

 *TEST 27 RELOCATION TEST USING LOWEST USABLE MAPPING REG

THIS TEST CHECKS OUT THE FULL ADDITION PROPERTIES OF THE UNIBUS MAP A.L.U.. IN THE DEFAULT CASE IT USES MAP REGISTER ZERO BUT IF THE MAP JUMPERS HAVE BEEN ALTERED TO DE-SELECT SOME MAP REGISTERS THIS TEST WILL USE THE LOWEST USEABLE MAP REGISTER. IF AN ERROR OCCURS THE TEST WILL REPORT THE PHYSICAL ADDRESS THAT WAS DESIRED, AND THE DATA AT THE ADDRESS THAT WAS REFERENCED.

THIS TEST IS BEING RUN WITH ALL MEMORY REFERENCES GOING THRU THE UNIBUS MAP.

 *TST27:

015602

```

015602 000004          SCOPE
015604 012767 016516 163510  MOV      #TST30,NXTTST  ;SAVE STARTING ADDRESS OF NEXT TEST
                                ;FOR ESCAPE ON PARITY ERRORS
1846 015612 012737 060000 060000  MOV      #060000,@#060000 ;MAKE SURE THAT ADDRESS 060000
1847                                     ;CONTAINS ITS OWN ADDRESS AS DATA
1848
1849                                     ;*THE RELOCATION HERE USES A BASE OF 00060000 AND AN
1850                                     ;*OFFSET OF 00000 TO PRODUCE AN ADDRESS OF 00060000
1851
1852 015620 012767 015652 163260 100$:  MOV      #1$,SLPERR      ;SET LOOP ON ERROR POINTER TO 1$
015626 005077 163416          CLR      @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
015632 012777 060000 163406  MOV      #060000,@LREGL ;LOAD LOWER BITS OF MAPPING REG
015640 016737 163376 172354  MOV      LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
015646 012700 140000          MOV      #140000,R0      ;SELECT PAR6, OFFSET IS 00000
015652 000240          1$:  NOP
015654 011001          MOV      (R0),R1        ;READ LOCATION 060000 THRU THE UNIBUS
015656 012702 060000          MOV      #060000,R2        ;THE EXPECTED PHYSICAL ADDRESS IS 060000
015662 020102          CMP      R1,R2          ;SEE IF THE MAP'S FETCH WAS CORRECT
015664 001401          BEQ      2$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
015666 104031          EMT+    31

1853                                     ;*THE RELOCATION HERE USES A BASE OF 00052524 AND AN
1854                                     ;*OFFSET OF 05252 TO PRODUCE AN ADDRESS OF 00057776
1855
1856
1857 015670 012767 015722 163210 2$:  MOV      #3$,SLPERR      ;SET LOOP ON ERROR POINTER TO 3$
015676 005077 163346          CLR      @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
015702 012777 052524 163336  MOV      #052524,@LREGL ;LOAD LOWER BITS OF MAPPING REG
015710 016737 163326 172354  MOV      LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
015716 012700 145252          MOV      #145252,R0      ;SELECT PAR6, OFFSET IS 05252
015722 000240          3$:  NOP
015724 011001          MOV      (R0),R1        ;READ LOCATION 057776 THRU THE UNIBUS
015726 012702 057776          MOV      #057776,R2        ;THE EXPECTED PHYSICAL ADDRESS IS 057776
015732 020102          CMP      R1,R2          ;SEE IF THE MAP'S FETCH WAS CORRECT
015734 001401          BEQ      4$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
015736 104031          EMT+    31

1858                                     ;*THE RELOCATION HERE USES A BASE OF 00045252 AND AN
1859                                     ;*OFFSET OF 12524 TO PRODUCE AN ADDRESS OF 00057776
1860
1861
1862 015740 012767 015772 163140 4$:  MOV      #5$,SLPERR      ;SET LOOP ON ERROR POINTER TO 5$
015746 005077 163276          CLR      @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
015752 012777 045252 163266  MOV      #045252,@LREGL ;LOAD LOWER BITS OF MAPPING REG
015760 016737 163256 172354  MOV      LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
015766 012700 152524          MOV      #152524,R0      ;SELECT PAR6, OFFSET IS 12524
015772 000240          5$:  NOP
015774 011001          MOV      (R0),R1        ;READ LOCATION 057776 THRU THE UNIBUS
015776 012702 057776          MOV      #057776,R2        ;THE EXPECTED PHYSICAL ADDRESS IS 057776
016002 020102          CMP      R1,R2          ;SEE IF THE MAP'S FETCH WAS CORRECT
016004 001401          BEQ      6$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016006 104031          EMT+    31

1863                                     ;*THE RELOCATION HERE USES A BASE OF 00050420 AND AN
1864                                     ;*OFFSET OF 10420 TO PRODUCE AN ADDRESS OF 00061040
1865
1866
1867 016010 012767 016042 163070 6$:  MOV      #7$,SLPERR      ;SET LOOP ON ERROR POINTER TO 7$
016016 005077 163226          CLR      @LREGU          ;CLEAR UPPER BITS OF MAPPING REG
016022 012777 050420 163216  MOV      #050420,@LREGL ;LOAD LOWER BITS OF MAPPING REG
  
```

```

016030 016737 163206 172354      MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016036 012700 150420                MOV     #150420,R0      ;SELECT PAR6, OFFSET IS 10420
016042 000240                7$:    NOP
016044 011001                MOV     (R0),R1        ;READ LOCATION 061040 THRU THE UNIBUS
016046 012702 061040                MOV     #061040,R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
016052 020102                CMP     R1,R2          ;SEE IF THE MAP'S FETCH WAS CORRECT
016054 001401                BEQ     8$             ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016056 104031                EMT+   31
  
```

```

1868
1869      ;*THE RELOCATION HERE USES A BASE OF 00054630 AND AN
1870      ;*OFFSET OF 04210 TO PRODUCE AN ADDRESS OF 00061040
1871
1872
  
```

```

016060 012767 016112 163020      8$:    MOV     #9$,SLPERR    ;SET LOOP ON ERROR POINTER TO 9$
016066 005077 163156                CLR     @LREGU         ;CLEAR UPPER BITS OF MAPPING REG
016072 012777 054630 163146      MOV     #054630,@LREGL ;LOAD LOWER BITS OF MAPPING REG
016100 016737 163136 172354      MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016106 012700 144210                MOV     #144210,R0    ;SELECT PAR6, OFFSET IS 04210
016112 000240                9$:    NOP
016114 011001                MOV     (R0),R1        ;READ LOCATION 061040 THRU THE UNIBUS
016116 012702 061040                MOV     #061040,R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
016122 020102                CMP     R1,R2          ;SEE IF THE MAP'S FETCH WAS CORRECT
016124 001401                BEQ     10$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016126 104031                EMT+   31
  
```

```

1873
1874      ;*THE RELOCATION HERE USES A BASE OF 00044210 AND AN
1875      ;*OFFSET OF 14630 TO PRODUCE AN ADDRESS OF 00061040
1876
1877
  
```

```

016130 012767 016162 162750      10$:   MOV     #11$,SLPERR   ;SET LOOP ON ERROR POINTER TO 11$
016136 005077 163106                CLR     @LREGU         ;CLEAR UPPER BITS OF MAPPING REG
016142 012777 044210 163076      MOV     #044210,@LREGL ;LOAD LOWER BITS OF MAPPING REG
016150 016737 163066 172354      MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016156 012700 154630                MOV     #154630,R0    ;SELECT PAR6, OFFSET IS 14630
016162 000240                11$:   NOP
016164 011001                MOV     (R0),R1        ;READ LOCATION 061040 THRU THE UNIBUS
016166 012702 061040                MOV     #061040,R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
016172 020102                CMP     R1,R2          ;SEE IF THE MAP'S FETCH WAS CORRECT
016174 001401                BEQ     12$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016176 104031                EMT+   31
  
```

```

1878
1879      ;*THE RELOCATION HERE USES A BASE OF 00056734 AND AN
1880      ;*OFFSET OF 02104 TO PRODUCE AN ADDRESS OF 00061040
1881
1882
  
```

```

016200 012767 016232 162700      12$:   MOV     #13$,SLPERR   ;SET LOOP ON ERROR POINTER TO 13$
016206 005077 163036                CLR     @LREGU         ;CLEAR UPPER BITS OF MAPPING REG
016212 012777 056734 163026      MOV     #056734,@LREGL ;LOAD LOWER BITS OF MAPPING REG
016220 016737 163016 172354      MOV     LOWEST,@#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
016226 012700 142104                MOV     #142104,R0    ;SELECT PAR6, OFFSET IS 02104
016232 000240                13$:   NOP
016234 011001                MOV     (R0),R1        ;READ LOCATION 061040 THRU THE UNIBUS
016236 012702 061040                MOV     #061040,R2    ;THE EXPECTED PHYSICAL ADDRESS IS 061040
016242 020102                CMP     R1,R2          ;SEE IF THE MAP'S FETCH WAS CORRECT
016244 001401                BEQ     14$            ;BRANCH IF FETCHED DATA MATCHES ADDRESS
016246 104031                EMT+   31
  
```

```

1883
1884      ;*THE RELOCATION HERE USES A BASE OF 00042104 AND AN
1885      ;*OFFSET OF 16734 TO PRODUCE AN ADDRESS OF 00061040
1886
  
```

```

1887 016250 012767 016302 162630 14$: MOV #15$, $LPERR ;SET LOOP ON ERROR POINTER TO 15$
      016256 005077 162766          CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
      016262 012777 042104 162756 MOV #042104, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      016270 016737 162746 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      016276 012700 156734          MOV #156734, R0 ;SELECT PAR6, OFFSET IS 16734
      016302 000240          15$: NOP
      016304 011001          MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
      016306 012702 061040          MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
      016312 020102          CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
      016314 001401          BEQ 16$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
      016316 104031          EMT+ 31
  
```

```

1888
1889      ::
1890      *THE RELOCATION HERE USES A BASE OF 00057776 AND AN
1891      *OFFSET OF 01042 TO PRODUCE AN ADDRESS OF 00061040
  
```

```

1892 016320 012767 016352 162560 16$: MOV #17$, $LPERR ;SET LOOP ON ERROR POINTER TO 17$
      016326 005077 162716          CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
      016332 012777 057776 162706 MOV #057776, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      016340 016737 162676 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      016346 012700 141042          MOV #141042, R0 ;SELECT PAR6, OFFSET IS 01042
      016352 000240          17$: NOP
      016354 011001          MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
      016356 012702 061040          MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
      016362 020102          CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
      016364 001401          BEQ 18$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
      016366 104031          EMT+ 31
  
```

```

1893
1894      ::
1895      *THE RELOCATION HERE USES A BASE OF 00041042 AND AN
1896      *OFFSET OF 17776 TO PRODUCE AN ADDRESS OF 00061040
  
```

```

1897 016370 012767 016422 162510 18$: MOV #19$, $LPERR ;SET LOOP ON ERROR POINTER TO 19$
      016376 005077 162646          CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
      016402 012777 041042 162636 MOV #041042, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      016410 016737 162626 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      016416 012700 157776          MOV #157776, R0 ;SELECT PAR6, OFFSET IS 17776
      016422 000240          19$: NOP
      016424 011001          MOV (R0), R1 ;READ LOCATION 061040 THRU THE UNIBUS
      016426 012702 061040          MOV #061040, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 061040
      016432 020102          CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
      016434 001401          BEQ 20$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
      016436 104031          EMT+ 31
  
```

```

1898
1899      ::
1900      *THE RELOCATION HERE USES A BASE OF 00057776 AND AN
1901      *OFFSET OF 00002 TO PRODUCE AN ADDRESS OF 00060000
  
```

```

1902 016440 012767 016472 162440 20$: MOV #21$, $LPERR ;SET LOOP ON ERROR POINTER TO 21$
      016446 005077 162576          CLR @LREGU ;CLEAR UPPER BITS OF MAPPING REG
      016452 012777 057776 162566 MOV #057776, @LREGL ;LOAD LOWER BITS OF MAPPING REG
      016460 016737 162556 172354 MOV LOWEST, @#KIPAR6 ;LOAD PAR6 WITH ADDR OF LOWEST MAP REG
      016466 012700 140002          MOV #140002, R0 ;SELECT PAR6, OFFSET IS 00002
      016472 000240          21$: NOP
      016474 011001          MOV (R0), R1 ;READ LOCATION 060000 THRU THE UNIBUS
      016476 012702 060000          MOV #060000, R2 ;THE EXPECTED PHYSICAL ADDRESS IS 060000
      016502 020102          CMP R1, R2 ;SEE IF THE MAP'S FETCH WAS CORRECT
      016504 001401          BEQ 22$ ;BRANCH IF FETCHED DATA MATCHES ADDRESS
      016506 104031          EMT+ 31
  
```

```

1903 016510 012767 015620 162370 22$: MOV #100$, $LPERR ;SET LOOP POINTER TO START OF TEST
  
```


1904
 1905
 1906
 1915
 1916

.....
 TEST 30 TEST CARRY PROP OF MAP'S RELOC ADDER

 EVERY ADDRESS OF THE FORM XXXX0000 IS GENERATED HERE STARTING
 WITH 00030000 UP TO 17000000. THAT IS THE FIRST OF EVERY 2K
 WORDS IS ADDRESSED, TO INSURE THAT THE ADDER IN THE MAP IS
 WORKING PROPERLY .


```

016516
016516 000004
016520 012767 016774 162574
          SCOPE
          MOV #TST31,NXTTST ;SAVE STARTING ADDRESS OF NEXT TEST
          ;FOR ESCAPE ON PARITY ERRORS
          MOV #12,$TIMES ;DO 12 ITERATIONS
1917 016534 012767 177777 000126 20$: MOV #-1,$S ;INITIALIZE FLAG AS NEGATIVE ONE
1918 016542 005077 162502 CLR @LREGU ;CLEAR UPPER 6 BITS OF MAP REG
1919 016546 012777 020000 162472 MOV #20000,@LREGL ;LOAD 4K BASE INTO MAP REGISTER
1920 016554 012701 100100 MOV #100100,R1 ;LOAD BITS TO SELECT PAR 4, OFFSET 100
1921 016560 012700 150000 MOV #150000,R0 ;LOAD BITS TO SELECT PAR 6, OFFSET 2K
1922 016564 012767 000277 153556 MOV #277,KIPAR4 ;START WITH PHYSICAL 6K
1923 016572 016767 162444 153554 MOV LOWEST,KIPAR6 ;LOAD PAR 6 WITH MAP REG'S ADDR
1924 016600 012767 016640 162300 MOV #10$,SLPERR ;SET LOOP ON ERROR POINTER TO 10$
1925 016606 012767 000020 162452 3$: MOV #TIMOUT,CPUEXP ;EXPECTING A UNIBUS TIME OUT DURING TEST
1926 016614 005067 162450 CLR PCPUER ;CLEAR TIME OUT FLAG
1927 016620 016710 162500 MOV DATA,(R0) ;THIS LOAD WILL TIME OUT WHEN YOU
1928 ;HAVE REACHED THE TOP OF MEMORY
1929 ;IT SELECTS PAR 6 WHICH WILL PUT ADDR
1930 ;<XXX XX1>0000 ON THE UNIBUS.
1931 ;THE X'S WILL SELECT THE LOWEST USEABLE
1932 ;MAPPING REGISTER. THE DEFAULT CASE IS
1933 ;010000, SELECTING MAPPING REGISTER C.
1934 016624 005767 162440 TST PCPUER ;SEE IF THERE WAS MAIN MEMORY
1935 016630 001016 BNE 1$ ;BRANCH IF NO MAIN MEMORY FROM UNIBUS
1936 016632 012767 000040 162426 10$: MOV #NEXMEM,CPUEXP ;POSSIBLE CACHE NON-EXISTANT MEMORY
1937 016640 011103 MOV (R1),R3 ;READ TEST LOCATION VIA FASTBUS
1938 016642 022767 000040 162420 CMP #NEXMEM,PCPUER ;WAS THERE CACHE NON-EXISTANT MEMORY
1939 016650 001414 BEQ 2$ ;BRANCH IF NON-EXISTANT MEMORY
1940 016652 000240 NOP
1941 016654 011002 MOV (R0),R2 ;READ TEST LOCATION VIA UNIBUS MAP
1942 016656 020203 CMP R2,R3 ;COMPARE TEST DATA R2=MAP DATA
1943 ;R3 FASTBUS DATA
1944 016660 001410 BEQ 2$ ;BRANCH IF IT WAS THE SAME
1945 016662 104032 EMT+ 32
1946 016664 000406 BR 2$ ;BRANCH TO UPDATE ROUTINE
1947 016666 005227 1$: INC (PC)+ ;INCREMENT ONE TIME ENTRANCE FLAG
1948 016670 177777 4$: .WORD -1 ;USE NEGATIVE ONE FOR FLAG
1949 016672 001003 BNE 2$ ;BRANCH IF YOU'VE BEEN HERE BEFORE
1950 016674 016767 153450 162414 MOV KIPAR4,RSIZE ;SAVE UPPER LIMIT OF MEMORY
1951 016702 062767 000100 162414 2$: ADD #100,DATA ;CHANGE PATTERN FOR NEXT LOAD
1952 016710 062767 000100 153432 ADD #100,KIPAR4 ;ADD 2K TO PAR4
1953 016716 062777 010000 162322 ADD #10000,@LREGL ;ADD 2K TO MAP REGISTER
1954 016724 001330 BNE 3$ ;BRANCH IF MAP REGISTER NOT ZERO
1955 016726 005277 162316 INC @LREGU ;ADD ONE TO UPPER 6 BITS OF MAP REG
  
```

```

1956 016732 022777 000073 162310      CMP      #73,@LREGU      ;SEE IF TOP 128K BLOCK HAS BEEN PASSED
1957 016740 103322                    BHS      3$              ;BRANCH IF NOT PAST IT
1958 016742 005267 162356                    INC      DATA           ;CHANGE DATA PATTERN FOR NEXT PASS
1959 016746 042767 177700 162350      BIC      #177700,DATA    ;CLEAR UPPER 10 BITS OF DATA PATTERN
1960 016754 052767 000300 162342      BIS      #300,DATA       ;START WITH 3XX IN DATA PATTERN
1961 016762 012767 016534 162116      MOV      #20$,SLPERR     ;SET LOOP POINTER TO START OF TEST
1962 016770 005067 162272      19$:    CLR      CPUEXP     ;NO CPU TRAPS EXPECTED IN NEXT TEST
1963
1964
1965
1990
  
```

```

*****
:TEST 31      VERIFY TRAP DUE TO CACHE PARITY INTERRUPT
:THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
:SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
:OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
:THEN BIT 08 OF $SWREG IS SET TO 1 THRU APT SCRIPTING.
:
:THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE
:TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE INTERRUPT
:IS BEING CALLED FOR(CACHE PE INTR L).
:
:THIS TEST ASSUMES THAT ALL MODULES EXCEPT THE UBI MODULE ARE KNOWN
:GOOD MODULES.
:
:THIS TEST TOGETHER WITH OTHER CACHE TESTS,ALLOW MFG. TO ELIMINATE
:HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
:VERIFY TESTING OF THE UBI MODULE.
:
:TEST DESCRIPTION:
:VERIFY INTERRUPT LOGIC BY ASSURING THAT A TRAP OCCURS TO LOCATION
:114 WHEN A LOCATION PREVIOUSLY WRITTEN
:WITH WRONG HI/LO BYTE PARITY IS ACCESSED.
:CONDITIONS:      PEA=0
:                  DCPI=0
*****
  
```

```

016774
016774 000004
016776 012767 017324 162316      SCOPE
MOV      #TST32,NXTTST           ;SAVE STARTING ADDRESS OF NEXT TEST
:FOR ESCAPE ON PARITY ERRORS
1991 017004 012767 017020 162072      MOV #20$,SLPADR             ;SETUP LOOP ON TEST POINTER
1992 017012 012767 017020 162066      MOV #20$,SLPERR            ;SETUP LOOP ON ERROR POINTER
1993 017020
1994 017020 132767 000200 001007      20$:  BITB     #200,$ENVM       ;IS APT SIZING?
1995 017026 001405                    BEQ      12$              ;NO;TRY HARDWARE SWITCH REGISTER
1996 017030 032767 000100 001000      BIT      #100,$SWREG      ;YES APT IS SIZING;DOES APT SAY TO DO
1997
1998 017036 001530                    BEQ      10$              ;THIS TEST
1999 017040 000404                    BR       1$               ;NO;SKIP TEST
2000 017042 032737 000100 177570      12$:  BIT      #100,@#177570 ;YES,DO TEST
2001
2002 017050 001523                    BEQ      10$              ;DOES HARDWARE SWITCH REGISTER SAY TO
2003
2004 017052 042737 000001 177572      1$:    BIC #1,@#177572     ;TO DO TEST?
2005 017060 052737 000400 177746      BIS #400,@#177746        ;NO,SKIP TEST
:TURN OFF RELOCATION
:FLUSH CACHE TO INVALIDATE ALL CACHE LOCATIONS
  
```

```
2007 017066 032737 010000 177746 BIT #10000,2#177746 ;WAIT TILL DONE
2008 017074 001374 BNE .-6
2009 017076 013702 000000 MOV 2#0,R2 ;SAVE ADDR. 0 CONTENTS
2010 017102 005037 000000 CLR 2#0 ;0'S TO MAIN MEMORY LOCATION 0.
2011 017106 013700 000114 MOV 2#114,R0 ;SAVE VECTORS
2012 017112 013701 000116 MOV 2#116,R1
2013 017116 012737 017232 000114 MOV #45,2#114 ;SETUP FOR CACHE TRAP
2014 017124 012737 000340 000116 MOV #340,2#116
2015
2016 017132 112737 000002 177750 MOVB #2,2#177750 ;MODE ALLOWS CACHE UPDATES
2017 ;AND CLOCKING OF PARITY INFO TO INTERRUPT LOGIC
2018 ; ONLY DURING THE DESTINATION ACCESS OF
2019 ;AN INSTRUCTION.
2020 017140 005003 CLR R3 ;CLEAR ERROR FLAG
2021 017142 012737 000015 177746 MOV #15,2#177746 ;NO UCB SO AS TO WRITE CACHE STORES
2022 017150 005737 000000 TST 2#0
2023 017154 005737 040000 TST 2#40000 ;UPDATE CACHE LOCATION 0000 WITH CORRECT PARITY STORE
2024 017160 052737 000100 177746 BIS #100,2#177746 ;ALLOW WRITE WRONG PARITY DATA TO LO
2025 ; & HI BYTE PARITY STORE.
2026 017166 005737 000000 TST 2#0 ;READ UPDATE TO CACHE LOCATION 0000;
2027 ;WRITE WRONG PARITY TO HI/LO BYTE PARITY STORES
2028 017172 042737 000100 177746 BIC #100,2#177746 ;DISABLE WMPD
2029 017200 005037 177744 CLR 2#177744 ;CLEAR CME AND PARITY DETECT LOGIC
2030 017204 042737 000005 177746 BIC #5,2#177746 ;ALLOW FOR INTERRUPT TO OCCUR
2031 ;AND ENABLE LOW CACHE
2032 017212 005737 000000 TST 2#0 ;READ HIT;
2033 ;LO & HI BYTE PARITY CHECK GENERATORS WILL
2034 ; DETECT WRONG PARITY AND THE PARITY
2035 ;ERROR WILL BE CLOCKED TO INTERRUPT
2036 ;LOGIC
2037 017216 000240 NOP
2038 017220 005203 INC R3 ;INDICATE THAT TRAP DID NOT OCCUR
2039 017222 012737 001015 177746 MOV #1015,2#177746 ;DISABLE CACHE
2040 017230 000404 BR 25$
2041 017232 012737 001015 177746 4$: MOV #1015,2#177746 ;DISABLE CACHE
2042 017240 022626 CMP (R6)+,(R6)+ ;READJUST STACK DUE TO INTERRUPT
2043 017242 000240 25$: NOP
2044 017244 000240 NOP
2045 017246 105037 177750 CLR R3 ;DISABLE MAINT. MODE
2046 017252 010237 000000 MOV R2,2#0 ;RESTORE LOCATION 0
2047 017256 010037 000114 MOV R0,2#114 ;RESTORE CACHE INTERRUPT VECTORS
2048 017262 010137 000116 MOV R1,2#116
2049 017266 052737 000400 177746 BIS #400,2#177746 ;BEFORE LEAVING TEST FLUSH CACHE TO
2050 ;ELIMINATE ANY EFFECTS OF WMPD
2051 017274 032737 010000 177746 BIT #10000,2#177746 ;WAIT TILL DONE
2052 017302 001374 BNE .-6
2053 017304 012737 000000 177746 MOV #0,2#177746 ;TURN CACHE ON
2054 017312 005703 TST R3 ;DID TRAP OCCUR?
2055 017314 001401 BEQ 10$ ;YES
2056 017316 000000 HALT ;INTERRUPT/ABORT LOGIC TESTS
2057 ;TRAP TO LOCATION 114 DID NOT OCCUR
2058 017320 000240 10$: NOP ;END OF TEST
2059 017322 000240 NOP
2060
2061
2062
2096
```

```

: *TEST 32 VERIFY TRAP DUE TO CACHE PARITY ABORT
: * THIS TEST IS OPTIONAL AND IS SELECTED BY SETTING MFM HARDWARE
: * SWITCH REGISTER BIT 08 TO A 1 IN THE CASE OF STANDALONE OPERATION
: * OF THE DIAGNOSTIC. IN THE CASE OF MANUFACTURING APT RUNTIME MODE
: * THEN BIT 08 OF $SWREG IS SET TO 1 THRU APT SCRIPTING.

: * THE TEST VERIFIES THE SIGNAL GENERATED FROM THE CACHE (BUS PBL)
: * TO THE UBI MODULE WHICH INDICATES TO THE UBI THAT A CACHE ABORT
: * IS BEING CALLED FOR.

: * THIS TEST ASSUMES THAT ALL MODULES EXCEPT UBI ARE KNOWN GOOD MODULES

: * THIS TEST TOGETHER WITH OTHER CACHE TESTS, ALLOW MFG. TO ELIMINATE
: * HAVING TO RUN THE CACHE DIAGNOSTIC DURING QUICK
: * VERIFY TESTING OF THE UBI MODULE.
    
```

TEST DESCRIPTION:

VERIFY ABORT LOGIC BY THE FOLLOWING RESULTS WHEN A LOCATION PREVIOUSLY WRITTEN WITH WRONG HI/LO BYTE PARITY IS ACCESSED.

1. INSTRUCTION CYCLE WILL BE ABORTED
2. THE ABORT CAUSES TRAP TO 114

PROCEDURE: INHIBIT CLOCKING OF PARITY ERROR SIGNAL TO INTERRUPT LOGIC. ALLOW CME<15> TO BE SET BY ABORT SIGNAL WHICH IS ASSERTED BY PARITY ERROR SIGNAL TO ABORT LOGIC.

CONDITIONS: PEA=1
DCPI=1

```

*****
TST32: SCOPE
2097 017324 000004
2097 017326 016767 001720 161766 MOV $EOP,NXTTST ;POINT TO ESCAPE VECTOR
2098 017334 012767 017350 161542 MOV #20$, $LPADR ;SETUP LOOP ON TEST VECTOR
2099 017342 012767 017350 161536 MOV #20$, $LPERR ;SETUP LOOP ON ERROR VECTOR
2100 017350
2101 017350 132767 000200 000457 20$: BITB #200, $ENVM ;IS APT SIZING?
2102 017356 001405 BEQ 12$ ;NO; TRY HARDWARE SWITCH REGISTER
2103 017360 032767 000100 000450 BIT #100, $SWREG ;YES APT IS SIZING; DOES APT SAY TO DO
2104 ; THIS TEST
2105 017366 001534 BEQ 10$ ;NO; SKIP TEST
2106 017370 000404 BR 1$ ;YES; DO TEST
2107 017372 032737 000100 177570 12$: BIT #100, @177570 ;DOES HARDWARE SWITCH REGISTER SAY TO
2108 ; TO DO TEST?
2109 017400 001527 BEQ 10$ ;NO; SKIP TEST
2110
2111 017402 042737 000001 177572 1$: BIC #1, @177572 ;TURN OFF RELOCATION
2112 017410 052737 000400 177746 BIS #400, @177746 ;FLUSH CACHE TO INVALIDATE ALL CACHE LOCATIONS
2113 017416 032737 010000 177746 BIT #10000, @177746 ;WAIT TILL DONE
2114 017424 001374 BNE -6
2115 017426 013702 000000 MOV @0, R2 ;SAVE ADDRESS 0 CONTENTS
2116 017432 005037 000000 CLR @0 ;ALL 0'S TO LOCATION 0
2117 017436 005003 CLR R3 ;ADDRESS 0 TO R3
2118 017440 013700 000114 MOV @114, R0 ;SAVE VECTORS
2119 017444 013701 000116 MOV @116, R1
    
```

| | | | | | | | |
|------|--------|--------|--------|--------|------|---------------------|--|
| 2120 | 017450 | 012737 | 017570 | 000114 | | MOV #45,@#114 | :SETUP FOR TRAP |
| 2121 | 017456 | 012737 | 000340 | 000116 | | MOV #340,@#116 | |
| 2122 | | | | | | | |
| 2123 | 017464 | 112737 | 000002 | 177750 | | MOVB #2,@#177750 | :HODO ALLOWS CACHE UPDATES |
| 2124 | | | | | | | :AND CLOCKING OF PARITY INFO TO INTERRUPT LOGIC |
| 2125 | | | | | | | : ONLY DURING THE DESTINATION ACCESS OF |
| 2126 | | | | | | | :AN INSTRUCTION. |
| 2127 | 017472 | 005005 | | | | CLR R5 | :CLEAR ERROR FLAG |
| 2128 | 017474 | 012704 | 177777 | | | MOV #-1,R4 | :ALL 1'S TO R4 |
| 2129 | 017500 | 012737 | 000015 | 177746 | | MOV #15,@#177746 | :NO UCB SO AS TO WRITE CACHE STORES |
| 2130 | 017506 | 005713 | | | | TST (R3) | |
| 2131 | 017510 | 005737 | 040000 | | | TST @#40000 | :UPDATE CACHE LOCATION 0000 WITH CORRECT PARITY STOR |
| 2132 | 017514 | 052737 | 000100 | 177746 | | BIS #100,@#177746 | :ALLOW WRITE WRONG PARITY DATA TO LO |
| 2133 | | | | | | | : & HI BYTE PARITY STORE. |
| 2134 | 017522 | 005713 | | | | TST (R3) | :READ UPDATE TO CACHE LOCATION 0000 |
| 2135 | | | | | | | :WRITE WRONG PARITY TO HI/LO BYTE PARITY STORES |
| 2136 | 017524 | 042737 | 000100 | 177746 | | BIC #100,@#177746 | :DISABLE WWP |
| 2137 | 017532 | 005037 | 177744 | | | CLR @#177744 | :CLEAR CME AND PARITY DETECT LOGIC |
| 2138 | 017536 | 042737 | 000004 | 177746 | | BIC #4,@#177746 | : ENABLE LOW CACHE |
| 2139 | 017544 | 052737 | 000200 | 177746 | | BIS #200,@#177746 | :ALLOW FOR ABORT |
| 2140 | 017552 | 011304 | | | | MOV (R3),R4 | :READ HIT; |
| 2141 | | | | | | | :LO & HI BYTE PARITY CHECK GENERATORS WILL |
| 2142 | | | | | | | : DETECT WRONG PARITY |
| 2143 | | | | | | | :USING HODO AND SOURCE MODE FOR READING |
| 2144 | | | | | | | :LOCATION 0 WILL INHIBIT PARITY ERROR |
| 2145 | | | | | | | :FROM BEING CLOCKED TO INTERRUPT LOGIC |
| 2146 | | | | | | | :HOWEVER, THE PARITY ERROR SIGNAL |
| 2147 | | | | | | | :WILL CAUSE THE ABORT SIGNAL TO BE |
| 2148 | | | | | | | :ASSERTED.THE ABORT SIGNAL WILL BE |
| 2149 | | | | | | | :CAUSE CME<15> TO BE SET. |
| 2150 | | | | | | | :THIS INSTRUCTION SHOULD BE ABORTED |
| 2151 | 017554 | 000240 | | | | NOP | |
| 2152 | 017556 | 005205 | | | | INC R5 | :INDICATE NO TRAP OCCURED |
| 2153 | 017560 | 012737 | 001015 | 177746 | | MOV #1015,@#177746 | :DISABLE CACHE |
| 2154 | 017566 | 000404 | | | | BR 5\$ | |
| 2155 | 017570 | 012737 | 001015 | 177746 | 4\$: | MOV #1015,@#177746 | :DISABLE CACHE |
| 2156 | 017576 | 022626 | | | | CMPI (R6)+,(R6)+ | :READJUST STACK |
| 2157 | | | | | | | |
| 2158 | 017600 | 000240 | | | 5\$: | NOP | |
| 2159 | 017602 | 000240 | | | | NOP | |
| 2160 | 017604 | 105037 | 177750 | | | CLRB @#177750 | :DISABLE MAINT. MODE |
| 2161 | 017610 | 010237 | 000000 | | | MOV R2,@#0 | :RESTORE VECTORS |
| 2162 | 017614 | 010037 | 000114 | | | MOV R0,@#114 | |
| 2163 | 017620 | 010137 | 000116 | | | MOV R1,@#116 | |
| 2164 | 017624 | 052737 | 000400 | 177746 | | BIS #400,@#177746 | :BEFORE LEAVING TEST FLUSH CACHE TO |
| 2165 | | | | | | | :ELIMINATE ANY EFFECTS OF WWP |
| 2166 | 017632 | 032737 | 010000 | 177746 | | BIT #10000,@#177746 | :WAIT TILL DONE |
| 2167 | 017640 | 001374 | | | | BNE #-6 | |
| 2168 | 017642 | 022704 | 177777 | | 8\$: | CMPI #-1,R4 | :WAS INSTRUCTION ABORTED LEAVING R4 INTACT? |
| 2169 | 017646 | 001401 | | | | BEQ 9\$ | :YES |
| 2170 | 017650 | 000000 | | | | HALT | : INTERRUPT/ABORT TESTS |
| 2171 | | | | | | | :R4 WAS OVERWRITTEN WITH DATA INDICATING |
| 2172 | | | | | | | :THAT INSTRUCTION WAS NOT ABORTED |
| 2173 | 017652 | 005705 | | | 9\$: | TST R5 | :DID TRAP OCCUR |
| 2174 | 017654 | 001401 | | | | BEQ 10\$ | :YES;PASS |
| 2175 | 017656 | 000000 | | | | HALT | : INTERRUPT/ABORT TESTS |
| 2176 | | | | | | . | :TRAP DID NOT OCCUR DUE TO ABORT |

2177 017660 000240 108: NOP ;END OF TEST
2178 017662 000240 NOP
2179
2180
2181
2182
2183
2184

| | | | | |
|------|--------|--------|--------|-----------|
| 2186 | 017664 | 000167 | 001362 | JMP SEOP |
| 2187 | | | | |
| 2188 | | | | |
| 2189 | 020000 | | | . = 20000 |
| 2190 | | | | |
| 2191 | | | | |
| 2192 | | | | |

2194

.SBTTL APT PARAMETER BLOCK

```

*****
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
*****
.SX=      ;;SAVE CURRENT LOCATION
.=24     ;;SET POWER FAIL TO POINT TO START OF PROGRAM
200      ;;FOR APT START UP
.=44     ;;POINT TO APT INDIRECT ADDRESS PNTR.
$APTHDR  ;;POINT TO APT HEADER BLOCK
--.$X    ;;RESET LOCATION COUNTER
*****
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
:INTERFACE SPEC.

```

```

020000
000024 000024
000024 000200
000044 000044
000044 020000
020000

```

```

020000
020000 000000
020002 020014
020004 000005
020006 000010
020010 000000
020012 000052

```

```

$APTHD:
$HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL ;;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT: .WORD 5 ;;RUN TIM OF LONGEST TEST
$PASTM: .WORD 10 ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD 0 ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)

```

2195

.SBTTL APT MAILBOX-ETABLE

```

*****
.EVEN
$MAIL:      ;;APT MAILBOX
$MSGTY: .WORD AMSGTY ;;MESSAGE TYPE CODE
$FATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
$TESTN: .WORD ATESTN ;;TEST NUMBER
$PASS: .WORD APASS ;;PASS COUNT
$DEVCT: .WORD ADEVCT ;;DEVICE COUNT
$UNIT: .WORD AUNIT ;;I/O UNIT NUMBER
$MSGAD: .WORD AMSGAD ;;MESSAGE ADDRESS
$MSGLG: .WORD AMSGLG ;;MESSAGE LENGTH
$ETABLE:   ;;APT ENVIRONMENT TABLE
$ENV: .BYTE AENV ;;ENVIRONMENT BYTE
$ENVM: .BYTE AENVM ;;ENVIRONMENT MODE BITS
$SWREG: .WORD ASWREG ;;APT SWITCH REGISTER
$USWR: .WORD AUSWR ;;USER SWITCHES
$CPUOP: .WORD ACPUOP ;;CPU TYPE,OPTIONS
        BITS 15-11=CPU TYPE
        11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
        11/70=06,PDQ=07,Q=10
        BIT 10=REAL TIME CLOCK
        BIT 9=FLOATING POINT PROCESSOR
        BIT 8=MEMORY MANAGEMENT
$MAMS1: .BYTE AMAMS1 ;;HIGH ADDRESS,M.S. BYTE
$MTYP1: .BYTE AMTYP1 ;;MEM. TYPE,BLK#1
        MEM. TYPE BYTE -- (HIGH BYTE)
        900 NSEC CORE=001
        300 NSEC BIPOLAR=002
        500 NSEC MOS=003
$MADR1: .WORD AMADR1 ;;HIGH ADDRESS,BLK#1
        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
$MAMS2: .BYTE AMAMS2 ;;HIGH ADDRESS,M.S. BYTE
$MTYP2: .BYTE AMTYP2 ;;MEM. TYPE,BLK#2

```

```

020014
020014 000000
020016 000000
020020 000000
020022 000000
020024 000000
020026 000000
020030 000000
020032 000000
020034
020034 000
020035 000
020036 000000
020040 000000
020042 000000

```

```

020044 000
020045 000

```

```

020046 000000
020050 000
020051 C

```


| | | | | | |
|--------|--------|----------|-------|--------|--|
| 020052 | 000000 | \$MADR2: | .WORD | AMADR2 | ::MEM.LAST ADDRESS,BLK#2 |
| 020054 | 000 | \$MAMS3: | .BYTE | AMAMS3 | ::HIGH ADDRESS,M.S.BYTE |
| 020055 | 000 | \$MTYP3: | .BYTE | AMTYP3 | ::MEM.TYPE,BLK#3 |
| 020056 | 000000 | \$MADR3: | .WORD | AMADR3 | ::MEM.LAST ADDRESS,BLK#3 |
| 020060 | 000 | \$MAMS4: | .BYTE | AMAMS4 | ::HIGH ADDRESS,M.S.BYTE |
| 020061 | 000 | \$MTYP4: | .BYTE | AMTYP4 | ::MEM.TYPE,BLK#4 |
| 020062 | 000000 | \$MADR4: | .WORD | AMADR4 | ::MEM.LAST ADDRESS,BLK#4 |
| 020064 | 000000 | \$VECT1: | .WORD | AVECT1 | ::INTERRUPT VECTOR#1,BUS PRIORITY#1 |
| 020066 | 000000 | \$VECT2: | .WORD | AVECT2 | ::INTERRUPT VECTOR#2BUS PRIORITY#2 |
| 020070 | 000000 | \$BASE: | .WORD | ABASE | ::BASE ADDRESS OF EQUIPMENT UNDER TEST |
| 020072 | 000000 | \$DEVN: | .WORD | ADEVN | ::DEVICE MAP |
| 020074 | 000000 | \$CDW1: | .WORD | ACDW1 | ::CONTROLLER DESCRIPTION WORD#1 |
| 020076 | 000000 | \$CDW2: | .WORD | ACDW2 | ::CONTROLLER DESCRIPTION WORD#2 |
| 020100 | 000000 | \$DDW0: | .WORD | ADDW0 | ::DEVICE DESCRIPTOR WORD#0 |
| 020102 | 000000 | \$DDW1: | .WORD | ADDW1 | ::DEVICE DESCRIPTOR WORD#1 |
| 020104 | 000000 | \$DDW2: | .WORD | ADDW2 | ::DEVICE DESCRIPTOR WORD#2 |
| 020106 | 000000 | \$DDW3: | .WORD | ADDW3 | ::DEVICE DESCRIPTOR WORD#3 |
| 020110 | 000000 | \$DDW4: | .WORD | ADDW4 | ::DEVICE DESCRIPTOR WORD#4 |
| 020112 | 000000 | \$DDW5: | .WORD | ADDW5 | ::DEVICE DESCRIPTOR WORD#5 |
| 020114 | 000000 | \$DDW6: | .WORD | ADDW6 | ::DEVICE DESCRIPTOR WORD#6 |
| 020116 | 000000 | \$DDW7: | .WORD | ADDW7 | ::DEVICE DESCRIPTOR WORD#7 |
| 020120 | 000000 | \$DDW8: | .WORD | ADDW8 | ::DEVICE DESCRIPTOR WORD#8 |
| 020122 | 000000 | \$DDW9: | .WORD | ADDW9 | ::DEVICE DESCRIPTOR WORD#9 |
| 020124 | 000000 | \$DDW10: | .WORD | ADDW10 | ::DEVICE DESCRIPTOR WORD#10 |
| 020126 | 000000 | \$DDW11: | .WORD | ADDW11 | ::DEVICE DESCRIPTOR WORD#11 |
| 020130 | 000000 | \$DDW12: | .WORD | ADDW12 | ::DEVICE DESCRIPTOR WORD#12 |
| 020132 | 000000 | \$DDW13: | .WORD | ADDW13 | ::DEVICE DESCRIPTOR WORD#13 |
| 020134 | 000000 | \$DDW14: | .WORD | ADDW14 | ::DEVICE DESCRIPTOR WORD#14 |
| 020136 | 000000 | \$DDW15: | .WORD | ADDW15 | ::DEVICE DESCRIPTOR WORD#15 |

020140

2196

\$ETEND:

::*****

.SBTTL SCOPE HANDLER ROUTINE

```

;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW14-1      LOOP ON TEST
;*SW11-1      INHIBIT ITERATIONS
;*SW09=1      LOOP ON ERROR
;*SW08-1      LOOP ON TEST IN SWR<5:0>
;*CALL
;*          SCOPE          ;;SCOPE-IOT
    
```

020140

020140 005067 161154

\$SCOPE:

| | | | | | |
|--------|--------|--------|-----|------------|--|
| 020144 | 005067 | 161106 | CLR | RETRY | ::CLEAR RETRY FLAG AN THE START OF |
| 020150 | 005067 | 161060 | CLR | ERRCNT | ::EACH TEST |
| 020154 | 005067 | 161050 | CLR | DATAOR | ::CLEAR THE MULTIPLE ERROR COUNTER |
| 020160 | 005067 | 161054 | CLR | ADDROR | ::LOCATION FOR LOGICAL OR OF BAD DATA |
| 020164 | 012700 | 177777 | CLR | PATTOR | ::LOCATION FOR LOGICAL OR OF ADDRESS |
| 020170 | 010067 | 161036 | MOV | #-1,R0 | ::LOCATION FOR LOGICAL OR OF PATTERN LOADED |
| | | | MOV | R0,DATAAND | ::LOAD -1 INTO R0 TO INITIALIZE LOGICAL AND OF |
| | | | | | ::LOCATION FOR LOGICAL AND OF BAD DATA |

```

020174 010067 161026      MOV      R0,ADRAND      ;:LOCATION FOR LOGICAL AND OF ADDRESS
020200 010067 161032      MOV      R0,PATAND     ;:LOCATION FOR LOGICAL AND OF PATTERN LOADED
020204 032777 040000 160736  BIT      #40000,@SWR   ;:** LOOP ON PRESENT TEST?
020212 001117      BNE     $OVER          ;:** YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
020214 000416      $XSTR: BR      6$     ;:IF RUNNING ON THE 'XOR' TESTER CHANGE
;:THIS INSTRUCTION TO A 'NOP' (NOP-240)
020216 013746 000004      MOV      @ERRVEC,-(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
020222 012737 020242 000004      MOV      #5$,@ERRVEC  ;:SET FOR TIMEOUT
020230 005737 177060      TST     @177060       ;:TIME OUT ON XOR?
020234 012637 000004      MOV      (SP)+,@ERRVEC ;:RESTORE THE ERROR VECTOR
020240 000466      BR      $SVLAD        ;:GO TO THE NEXT TEST
020242 022626      5$:     CMP      (SP)+,(SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
020244 012637 000004      MOV      (SP)+,@ERRVEC ;:RESTORE THE ERROR VECTOR
020250 000426      BR      7$           ;:LOOP ON THE PRESENT TEST
020252      6$::*****END OF CODE FOR THE XOR TESTER*****
020252 032777 000400 160670      BIT      #BIT08,@SWR   ;:** LOOP ON SPEC. TEST?
020260 001407      BEQ     2$           ;:BR IF NO
020262 017746 160662      MOV      @SWR,-(SP)   ;:** SET DESIRED TEST NUM. FROM SWR
020266 042716 000300      BIC     #5$SWRMK,(SP) ;:STRIP AWAY UNDESIRED BITS
020272 122667 160602      CMPB   (SP)+,$STSTNM ;:ON THE RIGHT TEST?
020276 001465      BEQ     $OVER        ;:BR IF YES
020300 105767 160575      2$:     TSTB   $ERFLG     ;:HAS AN ERROR OCCURRED?
020304 001421      BEQ     3$           ;:BR IF NO
020306 126767 160601 160565      CMPB   $ERMAX,$ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
020314 101015      BEQ     3$           ;:BR IF NO
020316 032777 001000 160624      BIT      #BIT09,@SWR   ;:** LOOP ON ERROR?
020324 001404      BEQ     4$           ;:BR IF NO
020326 016767 160554 160550  7$:     MOV      $LPERR,$LPADR ;:SET LOOP ADDRESS TO LAST SCOPE
020334 000446      BR      $OVER        ;:
020336 105067 160537      4$:     CLRB   $ERFLG     ;:ZERO THE ERROR FLAG
020342 005067 160640      CLR     $TIMES       ;:CLEAR THE NUMBER OF ITERATIONS TO MAKE
020346 000415      BR      1$           ;:ESCAPE TO THE NEXT TEST
020350 032777 004000 160572  3$:     BIT      #BIT11,@SWR   ;:** INHIBIT ITERATIONS?
020356 001011      BNE     1$           ;:BR IF YES
020360 005767 177436      TST     $PASS        ;:IF FIRST PASS OF PROGRAM
020364 001406      BEQ     1$           ;:
020366 005267 160510      INC     $ICNT        ;:INCREMENT ITERATION COUNT
020372 026767 160610 160502      CMP     $TIMES,$ICNT ;:CHECK THE NUMBER OF ITERATIONS MADE
020400 002024      BGE     $OVER        ;:BR IF MORE ITERATION REQUIRED
020402 012767 000001 160472  1$:     MOV      #1,$ICNT    ;:REINITIALIZE THE ITERATION COUNTER
020410 016767 000052 160570      MOV     $MXCNT,$TIMES ;:SET NUMBER OF ITERATIONS TO DO
020416 105267 160456      $SVLAD: INCB   $STSTNM ;:COUNT TEST NUMBERS
020422 116767 160452 177370      MOVB   $STSTNM,$TESTN ;:** SET TEST # IN APT MAILBOX
020430 011667 160450      MOV     (SP),$LPADR  ;:SAVE SCOPE LOOP ADDRESS
020434 011667 160446      MOV     (SP),$LPERR  ;:SAVE ERROR LOOP ADDRESS
020440 005067 160544      CLR     $ESCAPE      ;:CLEAR THE ESCAPE FROM ERROR ADDRESS
020444 112767 000001 160441      MOVB   #1,$ERMAX     ;:ONLY ALLOW ONE(1) ERROR ON NEXT TEST
020452 016777 160422 160472  $OVER:  MOV     $STSTNM,@DISPLAY ;:** DISPLAY TEST NUMBER
020460 016716 160420      MOV     $LPADR,(SP)  ;:FUDGE RETURN ADDRESS
020464 000002      RTI                    ;:FIXES PS
020466 000002      $MXCNT: 2.            ;:MAX. NUMBER OF ITERATIONS
;*****

```

2197

.SBTTL ERROR HANDLER ROUTINE

;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT.

```

: *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
: *AND GO TO ERTYPE ON ERROR
: *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
: *SW15=1      HALT ON ERROR
: *           HALT CAN OCCUR BEFORE AND AFTER THE ERROR TYPEOUT
: *SW13=1      INHIBIT ERROR TYPEOUTS
: *SW10=1      BELL ON ERROR
: *SW09=1      LOOP ON ERROR
: *CALL
: *          ERROR N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
  
```

```

020470          $ERROR:
020470 116767 160404 '60566      MOVB  $STNM,TESTNO      ;SAVE TEST NUMBER FOR ERROR TYPE OUT
020476 005267 160554          INC  ERRCNT      ;COUNT ALL MULTIPLE ERRORS
020502 010067 160450          MOV  R0,$REG0      ;SAVE R0 FOR POSSIBLE TYPE OUT
020506 010167 160446          MOV  R1,$REG1      ;SAVE R1 FOR POSSIBLE TYPE OUT
020512 010267 160444          MOV  R2,$REG2      ;SAVE R2 FOR POSSIBLE TYPE OUT
020516 010367 160442          MOV  R3,$REG3      ;SAVE R3 FOR POSSIBLE TYPE OUT
020522 010467 160440          MOV  R4,$REG4      ;SAVE R4 FOR POSSIBLE TYPE OUT
020526 010567 160436          MOV  R5,$REG5      ;SAVE R5 FOR POSSIBLE TYPE OUT
020532 105267 160343          7$: INCB $ERFLG      ;;SET THE ERROR FLAG
020536 001775          BEQ  7$      ;;DON'T LET THE FLAG GO TO ZERO
020540 016777 160334 160404      MOV  $STNM,@DISPLAY      ;** DISPLAY TEST NUMBER AND ERROR FLAG
020546 005777 160376          TST  @SWR      ;** HALT ON ERROR = 1?
020552 100001          BPL  8$      ;;BRANCH IF NO
020554 000000          HALT          ;;YES--HALT
020556 032777 002000 160364 8$: BIT  #BIT10,@SWR      ;** BELL ON ERROR?
020564 001402          BEQ  1$      ;;NO - SKIP
020566 104400 001212          TYPE $BELL      ;;RING BELL
020572 005267 160312          1$: INC  $ERTL      ;;COUNT THE NUMBER OF ERRORS
020576 011667 160312          MOV  (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
020602 162767 000002 160304      SUB  #2,$ERRPC
020610 117767 160300 160274      MOVB @SERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
020616 032777 020000 160324      BIT  #BIT13,@SWR      ;;** SKIP TYPEOUT IF SET
020624 001017          BNE  2$      ;;SKIP TYPEOUTS
020626 004767 161104          JSR  PC,ERTYPE      ;;GO TO USER ERROR ROUTINE
020632 104400 001217          TYPE ,SCLF

020636 122767 000001 177170      CMPB #1,$ENV      ;;** IS THIS APT?
020644 001007          BNE  2$      ;;** NO
020646 116767 160240 000004      MOVB $ITEMB,21$      ;;**
020654 004767 000142          JSR  PC,$ATY4      ;;**
020660          000          21$: .BYTE 0      ;;**
020661          000          .BYTE 0      ;;**
020662 000777          22$: BR  22$      ;;**

020664 005777 160260          2$: TST  @SWR      ;;** HALT ON ERROR
020670 100001          BPL  9$      ;;SKIP IF CONTINUE
020672 000000          HALT          ;;HALT ON ERROR.
020674 022767 021472 157140 9$: CMP  #SENDAD,42      ;;ACT-11?
020702 001001          BNE  3$      ;;BRANCH IF NO
020704 000000          HALT          ;;YES
020706 032777 001000 160234 3$: BIT  #BIT09,@SWR      ;** LOOP ON ERROR SWITCH SET?
020714 001402          BEQ  4$      ;;BR IF NO
020716 016716 160164          MOV  $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
020722 005767 160262          4$: TST  $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
020726 001402          BEQ  5$      ;;BR IF NONE
  
```

2198

```

020730 016716 160254          MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
020734 032777 001000 160206 5$:      BIT      #SW9,@SWR          ;ARE YOU LOOPING ON THIS ERROR?
020742 001417          BEQ      EREXIT           ;BRANCH IF NOT LOOPING ON ERROR
020744 012737 177777 177766      MOV      #-1,@CPUERR      ;CLEAR CPU ERROR REGISTER
020752 042737 177776 177572      BIC      #177776,@MMRO    ;CLEAR MEMORY MANAGEMENT STATUS REGISTER
020760 012767 177777 163152      MOV      #-1,@OFLAG      ;INITIALIZE TRAP FLAG
020766 012767 177777 163570      MOV      #-1,@CPFLAG     ;INITIALIZE CP TRAP FLAG
020774 012767 177777 163674      MOV      #-1,@MMFLAG     ;INITIALIZE MEMORY MANAGEMENT TRAP FLAG
021002 000002          EREXIT: RTI             ;RETURN TO TEST
          .SBTTL  APT COMMUNICATIONS ROUTINE

```

```

*****
021004 112767 000001 000236 $ATY1:  MOVB   #1,$FFLC      ;;TO REPORT FATAL ERROR
021012 112767 000001 000226 $ATY3:  MOVB   #1,$MFLG     ;;TO TYPE A MESSAGE
021020 000403          BR       $ATYC
021022 112767 000001 000220 $ATY4:  MOVB   #1,$FFLG     ;;TO ONLY REPORT FATAL ERROR
021030 $ATYC:
021030 010046          MOV      R0,-(SP)        ;;PUSH R0 ON STACK
021032 010146          MOV      R1,-(SP)        ;;PUSH R1 ON STACK
021034 105767 000206          TSTB   $MFLG           ;;SHOULD TYPE A MESSAGE?
021040 001450          BEQ     5$              ;;IF NOT: BR
021042 122767 000001 176764      CMPS   #APTENV,$ENV     ;;OPERATING UNDER APT?
021050 001031          BNE     3$              ;;IF NOT: BR
021052 132767 000100 176755      BITB   #APT$POOL,$ENVM  ;;SHOULD SPOOL MESSAGES?
021060 001425          BEQ     3$              ;;IF NOT: BR
021062 017600 000004          MOV     @4(JP),R0        ;;GET MESSAGE ADDR.
021066 062766 000002 000004      ADD     #2,4(SP)         ;;BUMP RETURN ADDR.
021074 005767 176714          1$:     TST     $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
021100 001375          BNE     1$              ;;IF NOT: WAIT
021102 010067 176722          MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
021106 105720          2$:     TSTB   (R0)+        ;;FIND END OF MESSAGE
021110 001376          BNE     2$
021112 166700 176712          SUB     $MSGAD,R0        ;;SUB START OF MESSAGE
021116 006200          ASR    R0               ;;GET MESSAGE LNTH IN WORDS
021120 010067 176706          MOV     R0,$MSGGLT      ;;PUT LENGTH IN MAILBOX
021124 012767 000004 176662      MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
021132 000413          BR      5$
021134 017667 000004 000016 3$:     MOV     @4(SP),4$        ;;PUT MSG ADDR IN JSR LINKAGE
021142 062766 000002 000004      ADD     #2,4(SP)         ;;BUMP RETURN ADDRESS
021150 016746 156622          MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
021154 004767 161344          JSR    PC,$TYPE        ;;CALL TYPE MACRO
021160 000000          4$:     .WORD    0
021162          5$:
021162 105767 000062          10$:    TSTB   $FFLG         ;;SHOULD REPORT FATAL ERROR?
021166 001416          BEQ    12$              ;;IF NOT: BR
021170 005767 176640          TST    $ENV             ;;RUNNING UNDER APT?
021174 001413          BEQ    12$              ;;IF NOT: BR
021176 005767 176612          11$:   TST     $MSGTYPE     ;;FINISHED LAST MESSAGE?
021202 001375          BNE    11$              ;;IF NOT: WAIT
021204 017667 000004 176604      MOV     @4(SP),$FATAL    ;;GET ERROR #
021212 062766 000002 000004      ADD     #2,4(SP)         ;;BUMP RETURN ADDR.
021220 005267 176570          INC    $MSGTYPE        ;;TELL APT TO TAKE ERROR
021224 105067 000020          12$:   CLRB   $FFLG         ;;CLEAR FATAL FLAG
021230 105067 000013          CLRB   $LFLG          ;;CLEAR LOG FLAG
021234 105067 000006          CLRB   $MFLG          ;;CLEAR MESSAGE FLAG
021240 012601          MOV     (SP)+,R1        ;;POP STACK INTO W

```

APT COMMUNICATIONS ROUTINE

| | | | | |
|--------|--------|---------|----------|---------------------|
| 021242 | 012600 | MOV | (SP)+,R0 | ::POP STACK INTO R0 |
| 021244 | 000207 | RTS | PC | ::RETURN |
| 021246 | 000 | \$MFLG: | .BYTE 0 | ::MESSG. FLAG |
| 021247 | 000 | \$LFLG: | .BYTE 0 | ::LOG FLAG |
| 021250 | 000 | \$FFLG: | .BYTE 0 | ::FATAL FLAG |

000200
000001
000100
000040

APTSIZE=200
APTENV=001
APTSPOOL=100
APTCSUP=040
:*****

2196

.SBT*L END OF PASS ROUTINE

::INCREMENT THE PASS NUMBER (\$PASS)
 ::INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
 ::TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYYY'
 ::WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
 ::IF SW12=1 INHIBIT TRACE TRAP
 ::IF THERES A MONITOR GO TO IT
 ::IF THERE ISN'T JUMP TO LOOP

| | | | |
|--------|--------|--------|--------|
| 021252 | | | |
| 021252 | 000004 | | |
| 021254 | 005067 | 156312 | |
| 021260 | 005067 | 151232 | |
| 021264 | 104420 | | |
| 021266 | 005067 | 157606 | |
| 021272 | 005067 | 157710 | |
| 021276 | 005267 | 176520 | |
| 021302 | 042767 | 100000 | 176512 |
| 021310 | 005327 | | |
| 021312 | 000001 | | |
| 021314 | 003072 | | |
| 021316 | 012737 | | |
| 021320 | 000001 | | |
| 021322 | 021312 | | |
| 021324 | 104400 | 021332 | |
| 021330 | 000407 | | |
| 021350 | | | |
| 021350 | 016746 | 176446 | |
| 021354 | 104410 | | |
| 021356 | 104400 | 021364 | |
| 021362 | 000421 | | |
| 021426 | | | |
| 021426 | 016746 | 157456 | |
| 021432 | 104410 | | |
| 021434 | 104400 | 001217 | |
| 021440 | 005067 | 157444 | |
| 021444 | 013700 | 000042 | |
| 021450 | 001414 | | |
| 021452 | 005046 | | |
| 021454 | 012746 | 021462 | |

\$EOP:
 SCOPE ::LOOP ON LAST TEST
 CLR MMRO ::TURN OFF FULL RELOCATION
 CLR MMR3 ::DISABLE THE UNIBUS MAP
 TBCTR ::RESTORE THE T BIT IF IT WAS ON
 CLR \$STNM ::ZERO THE TEST NUMBER
 CLR \$TIMES ::ZERO THE NUMBER OF ITERATIONS
 INC \$PASS ::INCREMENT THE PASS NUMBER
 BIC #100000,\$PASS ::DON'T ALLOW A NEG. NUMBER
 DEC (PC)+ ::LOOP?
 \$EOPCT: .WORD 1
 BGT \$DOAGN ::YES
 MOV (PC)+,@(PC)+ ::RESTORE COUNTER
 \$ENDCT: .WORD 1
 \$EOPCT
 TYPE ,65\$::TYPE ASCIZ STRING
 BR 64\$::GET OVER THE ASCIZ
 ::65\$: .ASCIZ <12><15>/END PASS #/
 64\$:
 MOV \$PASS,-(SP) ::SAVE \$PASS FOR TYPEOUT
 ::TYPE PASS NUMBER
 TYPDS ::GO TYPE--DECIMAL ASCII WITH SIGN
 TYPE ,67\$::TYPE ASCIZ STRING
 BR 66\$::GET OVER THE ASCIZ
 ::67\$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
 66\$:
 MOV \$ERTTL,-(SP) ::SAVE \$ERTTL FOR TYPEOUT
 ::TOTAL NUMBER OF ERRORS
 TYPDS ::GO TYPE--DECIMAL ASCII WITH SIGN
 TYPE ,SCLRF ::TYPE CARRIAGE RETURN, LINE FEED
 CLR \$ERTTL ::CLEAR ERROR TOTAL
 \$GET42: MOV @42,R0 ::GET MONITOR ADDRESS
 BEQ \$DOAGN ::BRANCH IF NO MONITOR
 CLR -(SP) ::INSURE THE 'T' BIT IS CLEAR
 MOV #SCLRF,-(SP) ::SETUP FOR AN RTI OR RTT

```
021460 000427          BR      $RTRN          ::GO DO AN RTI OR RTT TO LOAD THE PSW
                                           ::WITH A CLEARED 'T' BIT
021462          $CLR.T:
021462 013700 000042    MOV      @#42,R0          ::INSURE R0 CONTAINS THE MONITORS
021466 001405          BEQ      $DOAGN          ::RETURN ADDRESS
021470 000005          RESET          ::CLEAR THE WORLD
021472 004710          $ENDAD: JSR      PC,(R0)        ::GO TO MONITOR
021474 000240          NOP
021476 000240          NOP
021500 000240          NOP
021502          $DOAGN:
021502 013746 177776    MOV      @#PS,-(SP)        ::PUT THE PS ON THE STACK AND
021506 042716 000020    BIC      #20,(SP)        ::CLEAR THE 'T' BIT
021512 032777 010000 157470    BIT      #B112,@SWR      ::** RUN WITH TRACE TRAP?
021520 001005          BNE      1$              ::BR IF NO
021522 005167 000020    COM      $TBIT          ::IS IT TIME FOR TRACE TRAP
021526 100402          BMI      1$              ::BR IF NO
021530 052716 000020    BIS      #20,(SP)        ::SET TRACE TRAP
021534 012746 021542 1$:      MOV      #SLOOP,-(SP)    ::JUMP TO START OF TEST
021540 000002          $RTRN: RTI              ::RETURN--THIS IS CHANGED TO
                                           ::AN 'RTT' IF 'RTT' IS A LEGAL
                                           ::INSTRUCTION
021542          $LOOP:
021542 000137 010332    JMP      @#LOOP          ::RETURN
021546 000000          $TBIT: .WORD          0          ::'T' BIT STATE INDICATOR
021550 377 377 000 $ENULL: .BYTE          -1,-1,0    ::NULL CHARACTER STRING
                                           .EVEN
```

| 2201 | | | | | SBTTL | ERROR MESSAGES AND DATA TABLES |
|------|--------|-----|-----|-----|-------|---|
| 2202 | 021554 | 116 | 117 | 124 | EM1: | .ASCIZ ?NOT THE CORRECT TRAP CONDITION THRU ERRVEC (#004)? |
| | 021557 | 040 | 124 | 110 | | |
| | 021562 | 105 | 040 | 103 | | |
| | 021565 | 117 | 122 | 122 | | |
| | 021570 | 105 | 103 | 124 | | |
| | 021573 | 040 | 124 | 122 | | |
| | 021576 | 101 | 120 | 040 | | |
| | 021601 | 103 | 117 | 116 | | |
| | 021604 | 104 | 111 | 124 | | |
| | 021607 | 111 | 117 | 116 | | |
| | 021612 | 040 | 124 | 110 | | |
| | 021615 | 122 | 125 | 040 | | |
| | 021620 | 105 | 122 | 122 | | |
| | 021623 | 126 | 105 | 103 | | |
| | 021626 | 040 | 050 | 043 | | |
| | 021631 | 060 | 060 | 064 | | |
| | 021634 | 051 | 000 | | | |
| 2203 | 021636 | 125 | 116 | 105 | EM2: | .ASCIZ ?UNEXPECTED CPU TRAP THRU ERRVEC (#004)? |
| | 021641 | 130 | 120 | 105 | | |
| | 021644 | 103 | 124 | 105 | | |
| | 021647 | 104 | 040 | 103 | | |
| | 021652 | 120 | 125 | 040 | | |
| | 021655 | 124 | 122 | 101 | | |
| | 021660 | 120 | 040 | 124 | | |
| | 021663 | 110 | 122 | 125 | | |
| | 021666 | 040 | 105 | 122 | | |
| | 021671 | 122 | 126 | 105 | | |
| | 021674 | 103 | 040 | 050 | | |
| | 021677 | 043 | 060 | 060 | | |
| | 021702 | 064 | 051 | 000 | | |
| 2204 | 021705 | 125 | 116 | 105 | EM3: | .ASCIZ ?UNEXPECTED CACHE PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE? |
| | 021710 | 130 | 120 | 105 | | |
| | 021713 | 103 | 124 | 105 | | |
| | 021716 | 104 | 040 | 103 | | |
| | 021721 | 101 | 103 | 110 | | |
| | 021724 | 105 | 040 | 120 | | |
| | 021727 | 101 | 122 | 111 | | |
| | 021732 | 124 | 131 | 040 | | |
| | 021735 | 105 | 122 | 122 | | |
| | 021740 | 117 | 122 | 040 | | |
| | 021743 | 124 | 110 | 122 | | |
| | 021746 | 125 | 040 | 103 | | |
| | 021751 | 101 | 103 | 110 | | |
| | 021754 | 126 | 105 | 103 | | |
| | 021757 | 054 | 040 | 127 | | |
| | 021762 | 111 | 114 | 114 | | |
| | 021765 | 040 | 122 | 105 | | |
| | 021770 | 124 | 122 | 131 | | |
| | 021773 | 040 | 124 | 105 | | |
| | 021776 | 123 | 124 | 040 | | |
| | 022001 | 117 | 116 | 103 | | |
| | 022004 | 105 | 000 | | | |
| 2205 | 022006 | 125 | 116 | 105 | EM4: | .ASCIZ ?UNEXPECTED MAIN MEMORY PARITY ERROR THRU CACHVEC, WILL RETRY TEST ONCE? |
| | 022011 | 130 | 120 | 105 | | |
| | 022014 | 103 | 124 | 105 | | |
| | 022017 | 104 | 040 | 115 | | |

ERROR MESSAGES AND DATA TABLES

| | | | | |
|------|--------|-----|-----|-----|
| | 022022 | 101 | 111 | 116 |
| | 022025 | 040 | 115 | 105 |
| | 022030 | 115 | 117 | 122 |
| | 022033 | 131 | 040 | 120 |
| | 022036 | 101 | 122 | 111 |
| | 022041 | 124 | 131 | 040 |
| | 022044 | 105 | 122 | 122 |
| | 022047 | 117 | 122 | 040 |
| | 022052 | 124 | 110 | 122 |
| | 022055 | 125 | 040 | 103 |
| | 022060 | 101 | 103 | 110 |
| | 022063 | 126 | 105 | 103 |
| | 022066 | 054 | 040 | 127 |
| | 022071 | 111 | 114 | 114 |
| | 022074 | 040 | 122 | 105 |
| | 022077 | 124 | 122 | 131 |
| | 022102 | 040 | 124 | 105 |
| | 022105 | 123 | 124 | 040 |
| | 022110 | 117 | 116 | 103 |
| | 022113 | 105 | 000 | |
| 2206 | 022115 | 125 | 116 | 105 |
| | 022120 | 130 | 120 | 105 |
| | 022123 | 103 | 124 | 105 |
| | 022126 | 104 | 040 | 115 |
| | 022131 | 105 | 115 | 117 |
| | 022134 | 122 | 131 | 040 |
| | 022137 | 115 | 101 | 116 |
| | 022142 | 101 | 107 | 105 |
| | 022145 | 115 | 105 | 116 |
| | 022150 | 124 | 040 | 124 |
| | 022153 | 122 | 101 | 120 |
| | 022156 | 054 | 040 | 115 |
| | 022161 | 105 | 115 | 117 |
| | 022164 | 122 | 131 | 040 |
| | 022167 | 115 | 101 | 116 |
| | 022172 | 101 | 107 | 105 |
| | 022175 | 115 | 105 | 116 |
| | 022200 | 124 | 040 | 123 |
| | 022203 | 124 | 101 | 124 |
| | 022206 | 125 | 123 | 040 |
| | 022211 | 122 | 105 | 107 |
| | 022214 | 111 | 123 | 124 |
| | 022217 | 105 | 122 | 123 |
| | 022222 | 000 | | |
| 2207 | 022223 | 123 | 125 | 115 |
| | 022226 | 115 | 101 | 122 |
| | 022231 | 131 | 040 | 117 |
| | 022234 | 106 | 040 | 115 |
| | 022237 | 101 | 120 | 040 |
| | 022242 | 122 | 105 | 107 |
| | 022245 | 111 | 123 | 124 |
| | 022250 | 105 | 122 | 123 |
| | 022253 | 040 | 124 | 110 |
| | 022256 | 101 | 124 | 040 |
| | 022261 | 124 | 111 | 115 |
| | 022264 | 105 | 104 | 040 |
| | 022267 | 117 | 125 | 124 |

EM5: .ASCIZ ?UNEXPECTED MEMORY MANAGEMENT TRAP, MEMORY MANAGEMENT STATUS REGISTERS?

EM6: .ASCIZ ?SUMMARY OF MAP REGISTERS THAT TIMED OUT ON READ?

ERROR MESSAGES AND DATA TABLES

| | | | | | |
|------|--------|-----|-----|-----|--|
| | 022272 | 040 | 117 | 116 | |
| | 022275 | 040 | 122 | 105 | |
| | 022300 | 101 | 104 | 000 | |
| 2208 | 022303 | 123 | 125 | 115 | EM7: .ASCIZ ?SUMMARY OF CACHE REGISTERS THAT TIMED OUT ON READ? |
| | 022306 | 115 | 101 | 122 | |
| | 022311 | 131 | 040 | 117 | |
| | 022314 | 106 | 040 | 103 | |
| | 022317 | 101 | 103 | 110 | |
| | 022322 | 105 | 040 | 122 | |
| | 022325 | 105 | 107 | 111 | |
| | 022330 | 123 | 124 | 105 | |
| | 022333 | 122 | 123 | 040 | |
| | 022336 | 124 | 110 | 101 | |
| | 022341 | 124 | 040 | 124 | |
| | 022344 | 111 | 115 | 105 | |
| | 022347 | 104 | 040 | 117 | |
| | 022352 | 125 | 124 | 040 | |
| | 022355 | 117 | 116 | 040 | |
| | 022360 | 122 | 105 | 101 | |
| 2209 | 022363 | 104 | 000 | | |
| | 022365 | 123 | 125 | 115 | EM10: .ASCIZ ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN LOW 16 BITS? |
| | 022370 | 115 | 101 | 122 | |
| | 022373 | 131 | 040 | 117 | |
| | 022376 | 106 | 040 | 115 | |
| | 022401 | 101 | 120 | 040 | |
| | 022404 | 122 | 105 | 107 | |
| | 022407 | 111 | 123 | 124 | |
| | 022412 | 105 | 122 | 123 | |
| | 022415 | 040 | 116 | 117 | |
| | 022420 | 124 | 040 | 110 | |
| | 022423 | 117 | 114 | 104 | |
| | 022426 | 111 | 116 | 107 | |
| | 022431 | 040 | 132 | 105 | |
| | 022434 | 122 | 117 | 040 | |
| | 022437 | 111 | 116 | 040 | |
| | 022442 | 114 | 117 | 127 | |
| | 022445 | 040 | 061 | 066 | |
| | 022450 | 040 | 102 | 111 | |
| | 022453 | 124 | 123 | 000 | |
| 2210 | 022456 | 123 | 125 | 115 | EM11: .ASCIZ ?SUMMARY OF MAP REGISTERS NOT HOLDING ZERO IN UPPER 6 BITS? |
| | 022461 | 115 | 101 | 122 | |
| | 022464 | 131 | 040 | 117 | |
| | 022467 | 106 | 040 | 115 | |
| | 022472 | 101 | 120 | 040 | |
| | 022475 | 122 | 105 | 107 | |
| | 022500 | 111 | 123 | 124 | |
| | 022503 | 105 | 122 | 123 | |
| | 022506 | 040 | 116 | 117 | |
| | 022511 | 124 | 040 | 110 | |
| | 022514 | 117 | 114 | 104 | |
| | 022517 | 111 | 116 | 107 | |
| | 022522 | 040 | 132 | 105 | |
| | 022525 | 122 | 117 | 040 | |
| | 022530 | 111 | 116 | 040 | |
| | 022533 | 125 | 120 | 120 | |
| | 022536 | 105 | 122 | 040 | |
| | 022541 | 066 | 040 | 102 | |

ERROR MESSAGES AND DATA TABLES

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 022544 | 111 | 124 | 123 | |
| | 022547 | 000 | | | |
| 2211 | 022550 | 120 | 117 | 123 | EM12: .ASCIZ ?POSSIBLE ERROR IN MAP REGISTER DATA PATH (MAP REG 00)? |
| | 022553 | 123 | 111 | 102 | |
| | 022556 | 114 | 105 | 040 | |
| | 022561 | 105 | 122 | 122 | |
| | 022564 | 117 | 122 | 040 | |
| | 022567 | 111 | 116 | 040 | |
| | 022572 | 115 | 101 | 120 | |
| | 022575 | 040 | 122 | 105 | |
| | 022600 | 107 | 111 | 123 | |
| | 022603 | 124 | 105 | 122 | |
| | 022606 | 040 | 104 | 101 | |
| | 022611 | 124 | 101 | 040 | |
| | 022614 | 120 | 101 | 124 | |
| | 022617 | 110 | 040 | 050 | |
| | 022622 | 115 | 101 | 120 | |
| | 022625 | 040 | 122 | 105 | |
| | 022630 | 107 | 040 | 060 | |
| | 022633 | 060 | 051 | 000 | |
| 2212 | 022636 | 116 | 117 | 127 | EM13: .ASCIZ ?NOW PROBABLE ERROR IN MAP REGISTER DATA PATH (MAP REG 20)? |
| | 022641 | 040 | 120 | 122 | |
| | 022644 | 117 | 102 | 101 | |
| | 022647 | 102 | 114 | 105 | |
| | 022652 | 040 | 105 | 122 | |
| | 022655 | 122 | 117 | 122 | |
| | 022660 | 040 | 111 | 116 | |
| | 022663 | 040 | 115 | 101 | |
| | 022666 | 120 | 040 | 122 | |
| | 022671 | 105 | 107 | 111 | |
| | 022674 | 123 | 124 | 105 | |
| | 022677 | 122 | 040 | 104 | |
| | 022702 | 101 | 124 | 101 | |
| | 022705 | 040 | 120 | 101 | |
| | 022710 | 124 | 110 | 040 | |
| | 022713 | 050 | 115 | 101 | |
| | 022716 | 120 | 040 | 122 | |
| | 022721 | 105 | 107 | 040 | |
| | 022724 | 062 | 060 | 051 | |
| | 022727 | 000 | | | |
| 2213 | 022730 | 123 | 125 | 115 | EM14: .ASCIZ ?SUMMARY OF DUAL ADDRESSING ERRORS ON LOADING MAP REGISTERS? |
| | 022733 | 115 | 101 | 122 | |
| | 022736 | 131 | 040 | 117 | |
| | 022741 | 106 | 040 | 104 | |
| | 022744 | 125 | 101 | 114 | |
| | 022747 | 040 | 101 | 104 | |
| | 022752 | 104 | 122 | 105 | |
| | 022755 | 123 | 123 | 111 | |
| | 022760 | 116 | 107 | 040 | |
| | 022763 | 105 | 122 | 122 | |
| | 022766 | 117 | 122 | 123 | |
| | 022771 | 040 | 117 | 116 | |
| | 022774 | 040 | 114 | 117 | |
| | 022777 | 101 | 104 | 111 | |
| | 023002 | 116 | 107 | 040 | |
| | 023005 | 115 | 101 | 120 | |
| | 023010 | 040 | 122 | 105 | |

| | | | | | |
|------|--------|-----|-----|-----|--|
| | 023013 | 107 | 111 | 123 | |
| | 023016 | 124 | 105 | 122 | |
| | 023021 | 123 | 000 | | |
| 2214 | 023023 | 123 | 125 | 115 | EM15: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES IN LOWER 16 BITS OF MAP REGISTERS? |
| | 023026 | 115 | 101 | 122 | |
| | 023031 | 131 | 040 | 117 | |
| | 023034 | 106 | 040 | 103 | |
| | 023037 | 117 | 125 | 116 | |
| | 023042 | 124 | 040 | 120 | |
| | 023045 | 101 | 124 | 124 | |
| | 023050 | 105 | 122 | 116 | |
| | 023053 | 040 | 106 | 101 | |
| | 023056 | 111 | 114 | 125 | |
| | 023061 | 122 | 105 | 123 | |
| | 023064 | 040 | 111 | 116 | |
| | 023067 | 040 | 114 | 117 | |
| | 023072 | 127 | 105 | 122 | |
| | 023075 | 040 | 061 | 066 | |
| | 023100 | 040 | 102 | 111 | |
| | 023103 | 124 | 123 | 040 | |
| | 023106 | 117 | 106 | 040 | |
| | 023111 | 115 | 101 | 120 | |
| | 023114 | 040 | 122 | 105 | |
| | 023117 | 107 | 111 | 123 | |
| | 023122 | 124 | 105 | 122 | |
| | 023125 | 123 | 000 | | |
| 2215 | 023127 | 123 | 125 | 115 | EM16: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES IN UPPER 6 BITS OF MAP REGISTERS? |
| | 023132 | 115 | 101 | 122 | |
| | 023135 | 131 | 040 | 117 | |
| | 023140 | 106 | 040 | 103 | |
| | 023143 | 117 | 125 | 116 | |
| | 023146 | 124 | 040 | 120 | |
| | 023151 | 101 | 124 | 124 | |
| | 023154 | 105 | 122 | 116 | |
| | 023157 | 040 | 106 | 101 | |
| | 023162 | 111 | 114 | 125 | |
| | 023165 | 122 | 105 | 123 | |
| | 023170 | 040 | 111 | 116 | |
| | 023173 | 040 | 125 | 120 | |
| | 023176 | 120 | 105 | 122 | |
| | 023201 | 040 | 066 | 040 | |
| | 023204 | 102 | 111 | 124 | |
| | 023207 | 123 | 040 | 117 | |
| | 023212 | 106 | 040 | 115 | |
| | 023215 | 101 | 120 | 040 | |
| | 023220 | 122 | 105 | 107 | |
| | 023223 | 111 | 123 | 124 | |
| | 023226 | 105 | 122 | 123 | |
| | 023231 | 000 | | | |
| 2216 | 023232 | 103 | 117 | 125 | EM17: .ASCII ?COULD NOT CLEAR CACHE CONTROL REGISTER?<CRLF> |
| | 023235 | 114 | 104 | 040 | |
| | 023240 | 116 | 117 | 124 | |
| | 023243 | 040 | 103 | 114 | |
| | 023246 | 105 | 101 | 122 | |
| | 023251 | 040 | 103 | 101 | |
| | 023254 | 103 | 110 | 105 | |
| | 023257 | 040 | 103 | 117 | |

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 023262 | 116 | 124 | 122 | |
| | 023265 | 117 | 114 | 040 | |
| | 023270 | 122 | 105 | 10 | |
| | 023273 | 111 | 123 | 124 | |
| | 023276 | 105 | 122 | 200 | |
| 2217 | 023301 | 120 | 117 | 123 | .ASCII ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH? |
| | 023304 | 123 | 111 | 102 | |
| | 023307 | 114 | 105 | 040 | |
| | 023312 | 105 | 122 | 122 | |
| | 023315 | 117 | 122 | 040 | |
| | 023320 | 111 | 116 | 040 | |
| | 023323 | 103 | 101 | 103 | |
| | 023326 | 110 | 105 | 040 | |
| | 023331 | 122 | 105 | 107 | |
| | 023334 | 111 | 123 | 124 | |
| | 023337 | 105 | 122 | 040 | |
| | 023342 | 104 | 101 | 124 | |
| | 023345 | 101 | 040 | 120 | |
| | 023350 | 101 | 124 | 110 | |
| | 023353 | 000 | | | |
| 2218 | 023354 | 103 | 117 | 125 | EM20: .ASCII ?COULD NOT CLEAR CACHE MAINTENANCE REGISTER?<CRLF> |
| | 023357 | 114 | 104 | 040 | |
| | 023362 | 116 | 117 | 124 | |
| | 023365 | 040 | 103 | 114 | |
| | 023370 | 105 | 101 | 122 | |
| | 023373 | 040 | 103 | 101 | |
| | 023376 | 103 | 110 | 105 | |
| | 023401 | 040 | 115 | 101 | |
| | 023404 | 111 | 116 | 124 | |
| | 023407 | 105 | 116 | 105 | |
| | 023412 | 116 | 103 | 105 | |
| | 023415 | 040 | 122 | 105 | |
| | 023420 | 107 | 111 | 123 | |
| | 023423 | 124 | 105 | 122 | |
| | 023426 | 200 | | | |
| 2219 | 023427 | 120 | 117 | 123 | .ASCII ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH? |
| | 023432 | 123 | 111 | 102 | |
| | 023435 | 114 | 105 | 040 | |
| | 023440 | 105 | 122 | 122 | |
| | 023443 | 117 | 122 | 040 | |
| | 023446 | 111 | 116 | 040 | |
| | 023451 | 103 | 101 | 103 | |
| | 023454 | 110 | 105 | 040 | |
| | 023457 | 122 | 105 | 107 | |
| | 023462 | 111 | 123 | 124 | |
| | 023465 | 105 | 122 | 040 | |
| | 023470 | 104 | 101 | 124 | |
| | 023473 | 101 | 040 | 120 | |
| | 023476 | 101 | 124 | 110 | |
| | 023501 | 000 | | | |
| 2220 | 023502 | 103 | 117 | 125 | EM21: .ASCII ?COULD NOT READ 177740 FROM CACHE LO ADDRESS REG (LOADRS)?<CRLF> |
| | 023505 | 114 | 104 | 040 | |
| | 023510 | 116 | 117 | 124 | |
| | 023513 | 040 | 122 | 105 | |
| | 023516 | 101 | 104 | 040 | |
| | 023521 | 061 | 067 | 067 | |
| | 023524 | 067 | 064 | 060 | |

| | | | | |
|------|--------|-----|-----|-----|
| | 023527 | 040 | 106 | 122 |
| | 023532 | 117 | 115 | 040 |
| | 023535 | 103 | 101 | 103 |
| | 023540 | 110 | 105 | 040 |
| | 023543 | 114 | 117 | 040 |
| | 023546 | 101 | 104 | 104 |
| | 023551 | 122 | 105 | 123 |
| | 023554 | 123 | 040 | 122 |
| | 023557 | 105 | 107 | 040 |
| | 023562 | 050 | 114 | 117 |
| | 023565 | 101 | 104 | 122 |
| 2221 | 023570 | 123 | 051 | 200 |
| | 023573 | 120 | 117 | 123 |
| | 023576 | 123 | 111 | 102 |
| | 023601 | 114 | 105 | 040 |
| | 023604 | 105 | 122 | 122 |
| | 023607 | 117 | 122 | 040 |
| | 023612 | 111 | 116 | 040 |
| | 023615 | 103 | 101 | 103 |
| | 023620 | 110 | 105 | 040 |
| | 023623 | 122 | 105 | 107 |
| | 023626 | 111 | 123 | 124 |
| | 023631 | 105 | 122 | 040 |
| | 023634 | 104 | 101 | 124 |
| | 023637 | 101 | 040 | 120 |
| | 023642 | 101 | 124 | 110 |
| | 023645 | 000 | | |
| 2222 | 023646 | 103 | 117 | 125 |
| | 023651 | 114 | 104 | 040 |
| | 023654 | 116 | 117 | 124 |
| | 023657 | 040 | 122 | 105 |
| | 023662 | 101 | 104 | 040 |
| | 023665 | 060 | 060 | 060 |
| | 023670 | 060 | 060 | 063 |
| | 023673 | 040 | 106 | 122 |
| | 023676 | 117 | 115 | 040 |
| | 023701 | 103 | 101 | 103 |
| | 023704 | 110 | 105 | 040 |
| | 023707 | 110 | 111 | 040 |
| | 023712 | 101 | 104 | 104 |
| | 023715 | 122 | 105 | 123 |
| | 023720 | 123 | 040 | 122 |
| | 023723 | 105 | 107 | 040 |
| | 023726 | 050 | 110 | 111 |
| | 023731 | 101 | 104 | 122 |
| | 023734 | 123 | 051 | 200 |
| 2223 | 023737 | 120 | 117 | 123 |
| | 023742 | 123 | 111 | 102 |
| | 023745 | 114 | 105 | 040 |
| | 023750 | 105 | 122 | 122 |
| | 023753 | 117 | 122 | 040 |
| | 023756 | 111 | 116 | 040 |
| | 023761 | 103 | 101 | 103 |
| | 023764 | 110 | 105 | 040 |
| | 023767 | 122 | 105 | 107 |
| | 023772 | 111 | 123 | 124 |
| | 023775 | 105 | 122 | 040 |

.ASCIZ ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?

EM22: .ASCII ?COULD NOT READ 000003 FROM CACHE HI ADDRESS REG (HIADRS)?<CRLF>

.ASCIZ ?POSSIBLE ERROR IN CACHE REGISTER DATA PATH?

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 024000 | 104 | 101 | 124 | |
| | 024003 | 101 | 040 | 120 | |
| | 024006 | 101 | 124 | 110 | |
| | 024011 | 000 | | | |
| 2224 | 024012 | 123 | 125 | 115 | EM23: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE CONTROL REGISTER? |
| | 024015 | 115 | 101 | 122 | |
| | 024020 | 131 | 040 | 117 | |
| | 024023 | 106 | 040 | 103 | |
| | 024026 | 117 | 125 | 116 | |
| | 024031 | 124 | 040 | 120 | |
| | 024034 | 101 | 124 | 124 | |
| | 024037 | 105 | 122 | 116 | |
| | 024042 | 040 | 106 | 101 | |
| | 024045 | 111 | 114 | 125 | |
| | 024050 | 122 | 105 | 123 | |
| | 024053 | 040 | 111 | 116 | |
| | 024056 | 040 | 103 | 101 | |
| | 024061 | 103 | 110 | 105 | |
| | 024064 | 040 | 103 | 117 | |
| | 024067 | 116 | 124 | 122 | |
| | 024072 | 117 | 114 | 040 | |
| | 024075 | 122 | 105 | 107 | |
| | 024100 | 111 | 123 | 124 | |
| | 024103 | 105 | 122 | 000 | |
| 2225 | 024106 | 123 | 125 | 115 | EM24: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES IN CACHE MAINTENANCE REGISTER? |
| | 024111 | 115 | 101 | 122 | |
| | 024114 | 131 | 040 | 117 | |
| | 024117 | 106 | 040 | 103 | |
| | 024122 | 117 | 125 | 116 | |
| | 024125 | 124 | 040 | 120 | |
| | 024130 | 101 | 124 | 124 | |
| | 024133 | 105 | 122 | 116 | |
| | 024136 | 040 | 106 | 101 | |
| | 024141 | 111 | 114 | 125 | |
| | 024144 | 122 | 105 | 123 | |
| | 024147 | 040 | 111 | 116 | |
| | 024152 | 040 | 103 | 101 | |
| | 024155 | 103 | 110 | 105 | |
| | 024160 | 040 | 115 | 101 | |
| | 024163 | 111 | 116 | 124 | |
| | 024166 | 105 | 116 | 105 | |
| | 024171 | 116 | 103 | 105 | |
| | 024174 | 040 | 122 | 105 | |
| | 024177 | 107 | 111 | 123 | |
| | 024202 | 124 | 105 | 122 | |
| | 024205 | 000 | | | |
| 2226 | 024206 | 122 | 105 | 106 | EM25: .ASCIZ ?REFERENCED MAP REGISTER 0 WITH ADDRESS ONE BIT DIFFERENT THAN 770200? |
| | 024211 | 105 | 122 | 105 | |
| | 024214 | 116 | 103 | 105 | |
| | 024217 | 104 | 040 | 115 | |
| | 024222 | 101 | 120 | 040 | |
| | 024225 | 122 | 105 | 107 | |
| | 024230 | 111 | 123 | 124 | |
| | 024233 | 105 | 122 | 040 | |
| | 024236 | 060 | 040 | 127 | |
| | 024241 | 111 | 124 | 110 | |
| | 024244 | 040 | 101 | 104 | |

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 024247 | 104 | 122 | 105 | |
| | 024252 | 123 | 123 | 040 | |
| | 024255 | 117 | 116 | 105 | |
| | 024260 | 040 | 102 | 111 | |
| | 024263 | 124 | 040 | 104 | |
| | 024266 | 111 | 106 | 106 | |
| | 024271 | 105 | 122 | 105 | |
| | 024274 | 116 | 124 | 040 | |
| | 024277 | 124 | 110 | 101 | |
| | 024302 | 116 | 040 | 067 | |
| | 024305 | 067 | 060 | 062 | |
| | 024310 | 060 | 060 | 000 | |
| 2227 | 024313 | 122 | 105 | 106 | EM26. .ASCII ?REFERENCED CACHE LOW ADDRESS REGISTER WITH ADDRESS ONE BIT?<CRLF> |
| | 024316 | 105 | 122 | 105 | |
| | 024321 | 116 | 103 | 105 | |
| | 024324 | 104 | 040 | 103 | |
| | 024327 | 101 | 103 | 110 | |
| | 024332 | 105 | 040 | 114 | |
| | 024335 | 117 | 127 | 040 | |
| | 024340 | 101 | 104 | 104 | |
| | 024343 | 122 | 105 | 123 | |
| | 024346 | 123 | 040 | 122 | |
| | 024351 | 105 | 107 | 111 | |
| | 024354 | 123 | 124 | 105 | |
| | 024357 | 122 | 040 | 127 | |
| | 024362 | 111 | 124 | 110 | |
| | 024365 | 040 | 101 | 104 | |
| | 024370 | 104 | 122 | 105 | |
| | 024373 | 123 | 123 | 040 | |
| | 024376 | 117 | 116 | 105 | |
| | 024401 | 040 | 102 | 111 | |
| | 024404 | 124 | 200 | | |
| 2228 | 024406 | 104 | 111 | 106 | .ASCII ?DIFFERENT THAN 777740? |
| | 024411 | 106 | 105 | 122 | |
| | 024414 | 105 | 116 | 124 | |
| | 024417 | 040 | 124 | 110 | |
| | 024422 | 101 | 116 | 040 | |
| | 024425 | 067 | 067 | 067 | |
| | 024430 | 067 | 064 | 060 | |
| | 024433 | 000 | | | |
| 2229 | 024434 | 103 | 101 | 116 | EM30: .ASCII ?CAN'T GET TO MAIN MEMORY FROM UNIBUS WITH THE MAP OFF?<CRLF> |
| | 024437 | 047 | 124 | 040 | |
| | 024442 | 107 | 105 | 124 | |
| | 024445 | 040 | 124 | 117 | |
| | 024450 | 040 | 115 | 101 | |
| | 024453 | 111 | 116 | 040 | |
| | 024456 | 115 | 105 | 115 | |
| | 024461 | 117 | 122 | 131 | |
| | 024464 | 040 | 106 | 122 | |
| | 024467 | 117 | 115 | 040 | |
| | 024472 | 125 | 116 | 111 | |
| | 024475 | 102 | 125 | 123 | |
| | 024500 | 040 | 127 | 111 | |
| | 024503 | 124 | 110 | 040 | |
| | 024506 | 124 | 110 | 105 | |
| | 024511 | 040 | 115 | 101 | |
| | 024514 | 120 | 040 | 117 | |

| | | | | |
|------|--------|-----|-----|-----|
| 2230 | 024517 | 106 | 106 | 200 |
| | 024522 | 123 | 117 | 040 |
| | 024525 | 111 | 047 | 114 |
| | 024530 | 114 | 040 | 112 |
| | 024533 | 125 | 115 | 120 |
| | 024536 | 040 | 124 | 117 |
| | 024541 | 040 | 124 | 110 |
| | 024544 | 105 | 040 | 123 |
| | 024547 | 111 | 132 | 105 |
| | 024552 | 040 | 112 | 125 |
| | 024555 | 115 | 120 | 105 |
| | 024560 | 122 | 040 | 124 |
| | 024563 | 105 | 123 | 124 |
| | 024566 | 040 | 106 | 117 |
| | 024571 | 122 | 040 | 126 |
| | 024574 | 105 | 122 | 111 |
| | 024577 | 106 | 111 | 103 |
| | 024602 | 101 | 124 | 111 |
| 2231 | 024605 | 117 | 116 | 000 |
| | 024610 | 123 | 125 | 115 |
| | 024613 | 115 | 101 | 122 |
| | 024616 | 131 | 040 | 117 |
| | 024621 | 106 | 040 | 103 |
| | 024624 | 117 | 125 | 116 |
| | 024627 | 124 | 040 | 120 |
| | 024632 | 101 | 124 | 124 |
| | 024635 | 105 | 122 | 116 |
| | 024640 | 040 | 106 | 101 |
| | 024643 | 111 | 114 | 125 |
| | 024646 | 122 | 105 | 123 |
| | 024651 | 040 | 117 | 116 |
| | 024654 | 040 | 124 | 110 |
| | 024657 | 105 | 040 | 125 |
| | 024662 | 116 | 111 | 102 |
| | 024665 | 125 | 123 | 040 |
| | 024670 | 104 | 101 | 124 |
| | 024673 | 101 | 040 | 120 |
| | 024676 | 101 | 124 | 110 |
| 2232 | 024701 | 000 | | |
| | 024702 | 125 | 116 | 111 |
| | 024705 | 102 | 125 | 123 |
| | 024710 | 040 | 115 | 101 |
| | 024713 | 120 | 040 | 111 |
| | 024716 | 123 | 040 | 122 |
| | 024721 | 105 | 114 | 117 |
| | 024724 | 103 | 101 | 124 |
| | 024727 | 111 | 116 | 107 |
| | 024732 | 040 | 127 | 110 |
| | 024735 | 105 | 116 | 040 |
| | 024740 | 116 | 117 | 124 |
| | 024743 | 040 | 105 | 116 |
| | 024746 | 101 | 102 | 114 |
| 2233 | 024751 | 105 | 104 | 000 |
| | 024754 | 103 | 101 | 116 |
| | 024757 | 116 | 117 | 124 |
| | 024762 | 040 | 125 | 123 |
| | 024765 | 105 | 040 | 101 |

.ASCIZ ?SO I'LL JUMP TO THE SIZE JUMPER TEST FOR VERIFICATION?

EM31: .ASCIZ ?SUMMARY OF COUNT PATTERN FAILURES ON THE UNIBUS DATA PATH?

EM32: .ASCIZ ?UNIBUS MAP IS RELOCATING WHEN NOT ENABLED?

EM33: .ASCII ?CANNOT USE ANY OF THE MAP REGISTERS.POSSIBLY DUE?<CRLF>

| | | | | |
|------|--------|-----|-----|-----|
| | 024770 | 116 | 131 | 040 |
| | 024773 | 117 | 106 | 040 |
| | 024776 | 124 | 110 | 105 |
| | 025001 | 040 | 115 | 101 |
| | 025004 | 120 | 040 | 122 |
| | 025007 | 105 | 107 | 111 |
| | 025012 | 123 | 124 | 105 |
| | 025015 | 122 | 123 | 056 |
| | 025020 | 120 | 117 | 123 |
| | 025023 | 123 | 111 | 102 |
| | 025026 | 114 | 131 | 040 |
| | 025031 | 104 | 125 | 105 |
| | 025034 | 200 | | |
| 2234 | 025035 | 124 | 117 | 040 |
| | 025040 | 125 | 116 | 111 |
| | 025043 | 102 | 125 | 123 |
| | 025046 | 040 | 115 | 105 |
| | 025051 | 115 | 117 | 122 |
| | 025054 | 131 | 040 | 111 |
| | 025057 | 116 | 040 | 120 |
| | 025062 | 114 | 101 | 103 |
| | 025065 | 105 | 040 | 117 |
| | 025070 | 122 | 040 | 120 |
| | 025073 | 110 | 131 | 123 |
| | 025076 | 111 | 103 | 101 |
| | 025101 | 114 | 040 | 101 |
| | 025104 | 104 | 104 | 122 |
| | 025107 | 105 | 123 | 123 |
| | 025112 | 040 | 102 | 111 |
| | 025115 | 124 | 040 | 061 |
| | 025120 | 064 | 040 | 123 |
| | 025123 | 124 | 125 | 103 |
| | 025126 | 113 | 040 | 114 |
| | 025131 | 117 | 127 | 056 |
| | 025134 | 200 | | |
| 2235 | 025135 | 124 | 110 | 105 |
| | 025140 | 040 | 120 | 122 |
| | 025143 | 117 | 107 | 122 |
| | 025146 | 101 | 115 | 040 |
| | 025151 | 127 | 111 | 114 |
| | 025154 | 114 | 040 | 122 |
| | 025157 | 105 | 123 | 124 |
| | 025162 | 101 | 122 | 124 |
| | 025165 | 000 | | |
| 2236 | 025166 | 124 | 110 | 105 |
| 2237 | 025171 | 040 | 123 | 111 |
| | 025174 | 132 | 105 | 040 |
| | 025177 | 112 | 125 | 115 |
| | 025202 | 120 | 105 | 122 |
| | 025205 | 123 | 040 | 117 |
| | 025210 | 116 | 040 | 124 |
| | 025213 | 110 | 105 | 040 |
| | 025216 | 125 | 116 | 111 |
| | 025221 | 102 | 125 | 123 |
| | 025224 | 040 | 115 | 101 |
| | 025227 | 120 | 040 | 101 |

.ASCII ?TO UNIBUS MEMORY IN PLACE OR PHYSICAL ADDRESS BIT 14 STUCK LOW?<CR LF>

.ASCII ?THE PROGRAM WILL RESTART?

EM35: .ASCII ?THE SIZE JUMPERS ON THE UNIBUS MAP ARE NOT SET IN?<CR LF>

ERROR MESSAGES AND DATA TABLES

| | | | | | |
|------|--------|-----|-----|-----|--|
| | 025232 | 122 | 105 | 040 | |
| | 025235 | 116 | 117 | 124 | |
| | 025240 | 040 | 123 | 105 | |
| | 025243 | 124 | 040 | 111 | |
| | 025246 | 115 | 200 | | |
| 2238 | 025250 | 124 | 110 | 105 | .ASCII ?THEIR DEFAULT POSITION FOR FULL TEST COVERAGE .?<CRLF> |
| | 025253 | 111 | 122 | 040 | |
| | 025256 | 104 | 105 | 106 | |
| | 025261 | 101 | 125 | 114 | |
| | 025264 | 124 | 040 | 120 | |
| | 025267 | 117 | 123 | 111 | |
| | 025272 | 124 | 111 | 117 | |
| | 025275 | 116 | 040 | 106 | |
| | 025300 | 117 | 122 | 040 | |
| | 025303 | 106 | 125 | 114 | |
| | 025306 | 114 | 040 | 124 | |
| | 025311 | 105 | 123 | 124 | |
| | 025314 | 040 | 103 | 117 | |
| | 025317 | 126 | 105 | 122 | |
| | 025322 | 101 | 107 | 105 | |
| | 025325 | 040 | 056 | 200 | |
| 2239 | 025330 | 124 | 110 | 105 | .ASCII ?THE DEFAULT POSITION ALLOWS UNIBUS ADDRESSES?<CRLF> |
| | 025333 | 040 | 104 | 105 | |
| | 025336 | 106 | 101 | 125 | |
| | 025341 | 114 | 124 | 040 | |
| | 025344 | 120 | 117 | 123 | |
| | 025347 | 111 | 124 | 111 | |
| | 025352 | 117 | 116 | 040 | |
| | 025355 | 101 | 114 | 114 | |
| | 025360 | 117 | 127 | 123 | |
| | 025363 | 040 | 125 | 116 | |
| | 025366 | 111 | 102 | 125 | |
| | 025371 | 123 | 040 | 101 | |
| | 025374 | 104 | 104 | 122 | |
| | 025377 | 105 | 123 | 123 | |
| | 025402 | 105 | 123 | 200 | |
| 2240 | 025405 | 060 | 060 | 060 | .ASCII ?000000 TO 757776 TO REFERENCE MAIN MEMORY.?<CRLF> |
| | 025410 | 060 | 060 | 060 | |
| | 025413 | 040 | 124 | 117 | |
| | 025416 | 040 | 067 | 065 | |
| | 025421 | 067 | 067 | 067 | |
| | 025424 | 066 | 040 | 124 | |
| | 025427 | 117 | 040 | 122 | |
| | 025432 | 105 | 106 | 105 | |
| | 025435 | 122 | 105 | 116 | |
| | 025440 | 103 | 105 | 040 | |
| | 025443 | 115 | 101 | 111 | |
| | 025446 | 116 | 040 | 115 | |
| | 025451 | 105 | 115 | 117 | |
| | 025454 | 122 | 131 | 056 | |
| | 025457 | 200 | | | |
| 2241 | 025460 | 124 | 110 | 105 | .ASCII ?THE DIAGNOSTIC, IF CONTINUED, WILL TEST WITH REDUCED COVERAGE?<CRLF> |
| | 025463 | 040 | 104 | 111 | |
| | 025466 | 101 | 107 | 116 | |
| | 025471 | 117 | 123 | 124 | |
| | 025474 | 111 | 103 | 054 | |
| | 025477 | 111 | 106 | 040 | |

| | | | | |
|------|--------|-----|-----|-----|
| | 025502 | 103 | 117 | 116 |
| | 025505 | 124 | 111 | 116 |
| | 025510 | 125 | 105 | 104 |
| | 025513 | 054 | 040 | 127 |
| | 025516 | 111 | 114 | 114 |
| | 025521 | 040 | 124 | 105 |
| | 025524 | 123 | 124 | 040 |
| | 025527 | 127 | 111 | 124 |
| | 025532 | 110 | 040 | 122 |
| | 025535 | 105 | 104 | 125 |
| | 025540 | 103 | 105 | 104 |
| | 025543 | 040 | 103 | 117 |
| | 025546 | 126 | 105 | 122 |
| | 025551 | 101 | 107 | 105 |
| | 025554 | 200 | | |
| 2242 | 025555 | 125 | 123 | 111 |
| | 025560 | 116 | 107 | 040 |
| | 025563 | 040 | 124 | 110 |
| | 025566 | 105 | 040 | 103 |
| | 025571 | 117 | 116 | 123 |
| | 025574 | 124 | 122 | 101 |
| | 025577 | 111 | 116 | 124 |
| | 025602 | 040 | 124 | 110 |
| | 025605 | 101 | 124 | 040 |
| | 025610 | 115 | 101 | 111 |
| | 025613 | 116 | 040 | 115 |
| | 025616 | 105 | 115 | 117 |
| | 025621 | 122 | 131 | 040 |
| | 025624 | 127 | 111 | 114 |
| | 025627 | 114 | 040 | 122 |
| | 025632 | 105 | 123 | 120 |
| | 025635 | 117 | 116 | 104 |
| | 025640 | 200 | | |
| 2243 | 025641 | 124 | 117 | 040 |
| | 025644 | 124 | 110 | 105 |
| | 025647 | 040 | 122 | 101 |
| | 025652 | 116 | 107 | 105 |
| | 025655 | 040 | 117 | 106 |
| | 025660 | 040 | 125 | 116 |
| | 025663 | 111 | 102 | 125 |
| | 025666 | 123 | 040 | 101 |
| | 025671 | 104 | 104 | 122 |
| | 025674 | 105 | 123 | 123 |
| | 025677 | 105 | 123 | 040 |
| | 025702 | 123 | 120 | 105 |
| | 025705 | 103 | 111 | 106 |
| | 025710 | 111 | 105 | 104 |
| | 025713 | 040 | 102 | 131 |
| | 025716 | 040 | 124 | 110 |
| | 025721 | 105 | 040 | 106 |
| | 025724 | 117 | 114 | 114 |
| | 025727 | 117 | 127 | 111 |
| | 025732 | 116 | 107 | 072 |
| | 025735 | 000 | | |
| 2244 | 025736 | 115 | 101 | 120 |
| | 025741 | 040 | 122 | 105 |
| | 025744 | 107 | 111 | 123 |

.ASCII USING THE CONSTRAINT THAT MAIN MEMORY WILL RESPOND?<CRLF>

.ASCII ?TO THE RANGE OF UNIBUS ADDRESSES SPECIFIED BY THE FOLLOWING:?

EM36: .ASCII MAP REGISTER UNDER TEST DID NOT RESPOND IN DUAL MAPPING TEST?

| | | | | | |
|-----|--------|-----|-----|-----|---|
| | 025747 | 124 | 105 | 122 | |
| | 025752 | 040 | 125 | 116 | |
| | 025755 | 104 | 105 | 122 | |
| | 025760 | 040 | 124 | 105 | |
| | 025763 | 123 | 124 | 040 | |
| | 025766 | 104 | 111 | 104 | |
| | 025771 | 040 | 116 | 117 | |
| | 025774 | 124 | 040 | 122 | |
| | 025777 | 105 | 123 | 120 | |
| | 026002 | 117 | 116 | 104 | |
| | 026005 | 040 | 111 | 116 | |
| | 026010 | 040 | 104 | 125 | |
| | 026013 | 101 | 114 | 040 | |
| | 026016 | 115 | 101 | 120 | |
| | 026021 | 120 | 111 | 116 | |
| | 026024 | 107 | 040 | 124 | |
| | 026027 | 105 | 123 | 124 | |
| | 026032 | 000 | | | |
| 245 | 026033 | 123 | 125 | 115 | EM37: .ASCIZ ?SUMMARY OF UNIBUS ADDRESS ERRORS, WITH THE MAP RELOCATION DISABLED? |
| | 026036 | 115 | 101 | 122 | |
| | 026041 | 131 | 040 | 117 | |
| | 026044 | 106 | 040 | 125 | |
| | 026047 | 116 | 111 | 102 | |
| | 026052 | 125 | 123 | 040 | |
| | 026055 | 101 | 104 | 104 | |
| | 026060 | 122 | 105 | 123 | |
| | 026063 | 123 | 040 | 105 | |
| | 026066 | 122 | 122 | 117 | |
| | 026071 | 122 | 123 | 054 | |
| | 026074 | 040 | 127 | 111 | |
| | 026077 | 124 | 110 | 040 | |
| | 026102 | 124 | 110 | 105 | |
| | 026105 | 040 | 115 | 101 | |
| | 026110 | 120 | 040 | 122 | |
| | 026113 | 105 | 114 | 117 | |
| | 026116 | 103 | 101 | 124 | |
| | 026121 | 111 | 117 | 116 | |
| | 026124 | 040 | 104 | 111 | |
| | 026127 | 123 | 101 | 102 | |
| | 026132 | 114 | 105 | 104 | |
| | 026135 | 000 | | | |
| 246 | 026136 | 115 | 101 | 111 | EM40: .ASCIZ ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY? |
| | 026141 | 116 | 040 | 115 | |
| | 026144 | 105 | 115 | 117 | |
| | 026147 | 122 | 131 | 040 | |
| | 026152 | 124 | 111 | 115 | |
| | 026155 | 105 | 117 | 125 | |
| | 026160 | 124 | 040 | 117 | |
| | 026163 | 126 | 105 | 122 | |
| | 026166 | 040 | 124 | 110 | |
| | 026171 | 105 | 040 | 125 | |
| | 026174 | 116 | 111 | 102 | |
| | 026177 | 125 | 123 | 040 | |
| | 026202 | 104 | 111 | 104 | |
| | 026205 | 040 | 116 | 117 | |
| | 026210 | 124 | 040 | 117 | |
| | 026213 | 103 | 103 | 125 | |

| Address | Offset | Value | Value | Message |
|-------------|--------|-------|-------|---|
| 026216 | 122 | 040 | 120 | |
| 026221 | 122 | 117 | 120 | |
| 026224 | 105 | 122 | 114 | |
| 026227 | 131 | 000 | | |
| 2247 026231 | 122 | 105 | 114 | EM41: .ASCIZ ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD? |
| 026234 | 117 | 103 | 101 | |
| 026237 | 124 | 111 | 117 | |
| 026242 | 116 | 040 | 124 | |
| 026245 | 110 | 122 | 125 | |
| 026250 | 040 | 124 | 110 | |
| 026253 | 105 | 040 | 115 | |
| 026256 | 101 | 120 | 040 | |
| 026261 | 127 | 101 | 123 | |
| 026264 | 040 | 116 | 117 | |
| 026267 | 124 | 040 | 103 | |
| 026272 | 117 | 122 | 122 | |
| 026275 | 105 | 103 | 124 | |
| 026300 | 054 | 040 | 106 | |
| 026303 | 125 | 114 | 114 | |
| 026306 | 040 | 101 | 104 | |
| 026311 | 104 | 000 | | |
| 2248 026313 | 122 | 105 | 114 | EM42: .ASCIZ ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION? |
| 026316 | 117 | 103 | 101 | |
| 026321 | 124 | 111 | 117 | |
| 026324 | 116 | 040 | 124 | |
| 026327 | 110 | 122 | 125 | |
| 026332 | 040 | 124 | 110 | |
| 026335 | 105 | 040 | 115 | |
| 026340 | 101 | 120 | 040 | |
| 026343 | 127 | 101 | 123 | |
| 026346 | 040 | 116 | 117 | |
| 026351 | 124 | 040 | 103 | |
| 026354 | 117 | 122 | 122 | |
| 026357 | 105 | 103 | 124 | |
| 026362 | 054 | 040 | 103 | |
| 026365 | 101 | 122 | 122 | |
| 026370 | 131 | 040 | 120 | |
| 026373 | 122 | 117 | 120 | |
| 026376 | 101 | 107 | 101 | |
| 026401 | 124 | 111 | 117 | |
| 026404 | 116 | 000 | | |
| 2249 026406 | 124 | 110 | 105 | EM43: .ASCIZ ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS? |
| 026411 | 040 | 124 | 117 | |
| 026414 | 120 | 040 | 117 | |
| 026417 | 106 | 040 | 115 | |
| 026422 | 105 | 115 | 117 | |
| 026425 | 122 | 131 | 040 | |
| 026430 | 111 | 123 | 040 | |
| 026433 | 104 | 111 | 106 | |
| 026436 | 106 | 105 | 122 | |
| 026441 | 105 | 116 | 124 | |
| 026444 | 040 | 124 | 110 | |
| 026447 | 101 | 116 | 040 | |
| 026452 | 124 | 0 | 105 | |
| 026455 | 040 | 123 | 111 | |
| 026460 | 132 | 105 | 040 | |
| 026463 | 112 | 125 | 115 | |

| | | | | | |
|------|--------|-----|-----|-----|--|
| | 026466 | 120 | 105 | 122 | |
| | 026471 | 123 | 000 | | |
| 2250 | 026473 | 115 | 101 | 111 | EM45: .ASCII ?MAIN MEMORY TIMEOUT OVER THE UNIBUS DID NOT OCCUR PROPERLY?<CR LF> |
| | 026476 | 116 | 040 | 115 | |
| | 026501 | 105 | 115 | 117 | |
| | 026504 | 122 | 131 | 040 | |
| | 026507 | 124 | 111 | 115 | |
| | 026512 | 105 | 117 | 125 | |
| | 026515 | 124 | 040 | 117 | |
| | 026520 | 126 | 105 | 122 | |
| | 026523 | 040 | 124 | 110 | |
| | 026526 | 105 | 040 | 125 | |
| | 026531 | 116 | 111 | 102 | |
| | 026534 | 125 | 123 | 040 | |
| | 026537 | 104 | 111 | 104 | |
| | 026542 | 040 | 116 | 117 | |
| | 026545 | 124 | 040 | 117 | |
| | 026550 | 103 | 103 | 125 | |
| | 026553 | 122 | 040 | 120 | |
| | 026556 | 122 | 117 | 120 | |
| | 026561 | 105 | 122 | 114 | |
| 2251 | 026564 | 131 | 200 | | |
| | 026566 | 124 | 105 | 123 | .ASCII ?TEST CODE BEING RUN OVER UNIBUS? |
| | 026571 | 124 | 040 | 103 | |
| | 026574 | 117 | 104 | 105 | |
| | 026577 | 040 | 102 | 105 | |
| | 026602 | 111 | 116 | 107 | |
| | 026605 | 040 | 122 | 125 | |
| | 026610 | 116 | 040 | 117 | |
| | 026613 | 126 | 105 | 122 | |
| | 026616 | 040 | 125 | 116 | |
| | 026621 | 111 | 102 | 125 | |
| 2252 | 026624 | 123 | 000 | | |
| | 026626 | 122 | 105 | 114 | EM46: .ASCII ?RELOCATION THRU THE MAP WAS NOT CORRECT, FULL ADD?<CR LF> |
| | 026631 | 117 | 103 | 101 | |
| | 026634 | 124 | 111 | 117 | |
| | 026637 | 116 | 040 | 124 | |
| | 026642 | 110 | 122 | 125 | |
| | 026645 | 040 | 124 | 110 | |
| | 026650 | 105 | 040 | 115 | |
| | 026653 | 101 | 120 | 040 | |
| | 026656 | 127 | 101 | 123 | |
| | 026661 | 040 | 116 | 117 | |
| | 026664 | 124 | 040 | 103 | |
| | 026667 | 117 | 122 | 122 | |
| | 026672 | 105 | 103 | 124 | |
| | 026675 | 054 | 040 | 106 | |
| | 026700 | 125 | 114 | 114 | |
| | 026703 | 040 | 101 | 104 | |
| 2253 | 026706 | 104 | 200 | | |
| | 026710 | 124 | 105 | 123 | .ASCII ?TEST CODE BEING RUN OVER UNIBUS? |
| | 026713 | 124 | 040 | 103 | |
| | 026716 | 117 | 104 | 105 | |
| | 026721 | 040 | 102 | 105 | |
| | 026724 | 111 | 116 | 107 | |
| | 026727 | 040 | 122 | 125 | |
| | 026732 | 116 | 040 | 117 | |

| ERROR MESSAGES AND DATA TABLES | | | | |
|--------------------------------|--------|-----|-----|-----|
| | 026735 | 126 | 105 | 122 |
| | 026740 | 040 | 125 | 116 |
| | 026743 | 111 | 102 | 125 |
| | 026746 | 123 | 000 | |
| 2254 | 026750 | 122 | 105 | 114 |
| | 026753 | 117 | 103 | 101 |
| | 026756 | 124 | 111 | 117 |
| | 026761 | 116 | 040 | 124 |
| | 026764 | 110 | 122 | 125 |
| | 026767 | 040 | 124 | 110 |
| | 026772 | 105 | 040 | 115 |
| | 026775 | 101 | 120 | 040 |
| | 027000 | 127 | 101 | 123 |
| | 027003 | 040 | 116 | 117 |
| | 027006 | 124 | 040 | 103 |
| | 027011 | 117 | 122 | 122 |
| | 027014 | 105 | 103 | 124 |
| | 027017 | 054 | 040 | 103 |
| | 027022 | 101 | 122 | 122 |
| | 027025 | 131 | 040 | 120 |
| | 027030 | 122 | 117 | 120 |
| | 027033 | 101 | 107 | 101 |
| | 027036 | 124 | 111 | 117 |
| | 027041 | 116 | 200 | |
| 2255 | 027043 | 124 | 105 | 123 |
| | 027046 | 124 | 040 | 103 |
| | 027051 | 117 | 104 | 105 |
| | 027054 | 040 | 102 | 105 |
| | 027057 | 111 | 116 | 107 |
| | 027062 | 040 | 122 | 125 |
| | 027065 | 116 | 040 | 117 |
| | 027070 | 126 | 105 | 122 |
| | 027073 | 040 | 125 | 116 |
| | 027076 | 111 | 102 | 125 |
| | 027101 | 123 | 000 | |
| 2256 | 027103 | 124 | 110 | 105 |
| | 027106 | 040 | 124 | 117 |
| | 027111 | 120 | 040 | 117 |
| | 027114 | 106 | 040 | 115 |
| | 027117 | 105 | 115 | 117 |
| | 027122 | 122 | 131 | 040 |
| | 027125 | 111 | 123 | 040 |
| | 027130 | 104 | 111 | 106 |
| | 027133 | 106 | 105 | 122 |
| | 027136 | 105 | 116 | 124 |
| | 027141 | 040 | 124 | 110 |
| | 027144 | 101 | 116 | 040 |
| | 027147 | 124 | 110 | 105 |
| | 027152 | 040 | 123 | 111 |
| | 027155 | 132 | 105 | 040 |
| | 027160 | 112 | 125 | 115 |
| | 027163 | 120 | 105 | 122 |
| | 027166 | 123 | 200 | |
| 2257 | 027170 | 124 | 105 | 123 |
| | 027173 | 124 | 040 | 103 |
| | 027176 | 117 | 104 | 105 |
| | 027201 | 040 | 102 | 105 |

EM47: .ASCII ?RELOCATION THRU THE MAP WAS NOT CORRECT, CARRY PROPAGATION?<CRLF>

.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?

EM50: .ASCII ?THE TOP OF MEMORY IS DIFFERENT THAN THE SIZE JUMPERS?<CRLF>

.ASCIZ ?TEST CODE BEING RUN OVER UNIBUS?

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 027204 | 111 | 116 | *07 | |
| | 027207 | 040 | 122 | 125 | |
| | 027212 | 116 | 040 | 117 | |
| | 027215 | 126 | 105 | 122 | |
| | 027220 | 040 | 125 | 116 | |
| | 027223 | 111 | 102 | 125 | |
| | 027226 | 123 | 000 | | |
| 2258 | 027230 | 124 | 105 | 123 | .ASCIZ ?TEST CODE BEING RUN OVER UNIBUS? |
| | 027233 | 124 | 040 | 103 | |
| | 027236 | 117 | 104 | 105 | |
| | 027241 | 040 | 102 | 105 | |
| | 027244 | 111 | 116 | 107 | |
| | 027247 | 040 | 122 | 125 | |
| | 027252 | 116 | 040 | 117 | |
| | 027255 | 126 | 105 | 122 | |
| | 027260 | 040 | 125 | 116 | |
| | 027263 | 111 | 102 | 125 | |
| | 027266 | 123 | 000 | | |
| 2259 | 027270 | 123 | 125 | 115 | EM52: .ASCIZ ?SUMMARY OF DUAL MAPPING ERRORS? |
| | 027273 | 115 | 101 | 122 | |
| | 027276 | 131 | 040 | 117 | |
| | 027301 | 106 | 040 | 104 | |
| | 027304 | 125 | 101 | 114 | |
| | 027307 | 040 | 115 | 101 | |
| | 027312 | 120 | 120 | 111 | |
| | 027315 | 116 | 107 | 040 | |
| | 027320 | 105 | 122 | 122 | |
| | 027323 | 117 | 122 | 123 | |
| | 027326 | 000 | | | |
| 2260 | 027327 | 124 | 110 | 105 | EM201: .ASCIZ ?THE FOLLOWING REGISTERS TIMED OUT WHEN READ? |
| | 027332 | 040 | 106 | 117 | |
| | 027335 | 114 | 114 | 117 | |
| | 027340 | 127 | 111 | 116 | |
| | 027343 | 107 | 040 | 122 | |
| | 027346 | 105 | 107 | 111 | |
| | 027351 | 123 | 124 | 105 | |
| | 027354 | 122 | 123 | 040 | |
| | 027357 | 124 | 111 | 115 | |
| | 027362 | 105 | 104 | 040 | |
| | 027365 | 117 | 125 | 124 | |
| | 027370 | 040 | 127 | 110 | |
| | 027373 | 105 | 116 | 040 | |
| | 027376 | 122 | 105 | 101 | |
| | 027401 | 104 | 000 | | |
| 2261 | 027403 | 124 | 110 | 105 | EM202: .ASCIZ ?THE FOLLOWING MAP REGISTERS WILL NOT CLEAR? |
| | 027406 | 040 | 106 | 117 | |
| | 027411 | 114 | 114 | 117 | |
| | 027414 | 127 | 111 | 116 | |
| | 027417 | 107 | 040 | 115 | |
| | 027422 | 101 | 120 | 040 | |
| | 027425 | 122 | 105 | 107 | |
| | 027430 | 111 | 123 | 124 | |
| | 027433 | 105 | 122 | 123 | |
| | 027436 | 040 | 127 | 111 | |
| | 027441 | 114 | 114 | 040 | |
| | 027444 | 116 | 117 | 124 | |
| | 027447 | 040 | *03 | 114 | |

| | | | | | |
|------|--------|-----|-----|-----|--|
| | 027452 | 105 | 101 | 122 | |
| | 027455 | 000 | | | |
| 2262 | 027456 | 124 | 110 | 105 | EM203: .ASCIZ ?THE FOLLOWING ARE DUAL ADDRESSING ERRORS IN THE UNIBUS MAP? |
| | 027461 | 040 | 106 | 117 | |
| | 027464 | 114 | 114 | 117 | |
| | 027467 | 127 | 111 | 116 | |
| | 027472 | 107 | 040 | 101 | |
| | 027475 | 122 | 105 | 040 | |
| | 027500 | 104 | 125 | 101 | |
| | 027503 | 114 | 040 | 101 | |
| | 027506 | 104 | 104 | 122 | |
| | 027511 | 105 | 123 | 123 | |
| | 027514 | 111 | 116 | 107 | |
| | 027517 | 040 | 105 | 122 | |
| | 027522 | 122 | 117 | 122 | |
| | 027525 | 123 | 040 | 111 | |
| | 027530 | 116 | 040 | 124 | |
| | 027533 | 110 | 105 | 040 | |
| | 027536 | 125 | 116 | 111 | |
| | 027541 | 102 | 125 | 123 | |
| | 027544 | 040 | 115 | 101 | |
| | 027547 | 120 | 000 | | |
| 2263 | 027551 | 124 | 110 | 105 | EM204: .ASCIZ ?THE COUNT PATTERN THRU THE MAP REGISTERS FAILED? |
| | 027554 | 040 | 103 | 117 | |
| | 027557 | 125 | 116 | 124 | |
| | 027562 | 040 | 120 | 101 | |
| | 027565 | 124 | 124 | 105 | |
| | 027570 | 122 | 116 | 040 | |
| | 027573 | 124 | 110 | 122 | |
| | 027576 | 125 | 040 | 124 | |
| | 027601 | 110 | 105 | 040 | |
| | 027604 | 115 | 101 | 120 | |
| | 027607 | 040 | 122 | 105 | |
| | 027612 | 107 | 111 | 123 | |
| | 027615 | 124 | 105 | 122 | |
| | 027620 | 123 | 040 | 106 | |
| | 027623 | 101 | 111 | 114 | |
| | 027626 | 105 | 104 | 000 | |
| 2264 | 027631 | 125 | 116 | 111 | EM205: .ASCIZ ?UNIBUS DATA PATH COUNT PATTERN FAILURE? |
| | 027634 | 102 | 125 | 123 | |
| | 027637 | 040 | 104 | 101 | |
| | 027642 | 124 | 101 | 040 | |
| | 027645 | 120 | 101 | 124 | |
| | 027650 | 110 | 040 | 103 | |
| | 027653 | 117 | 125 | 116 | |
| | 027656 | 124 | 040 | 120 | |
| | 027661 | 101 | 124 | 124 | |
| | 027664 | 105 | 122 | 116 | |
| | 027667 | 040 | 106 | 101 | |
| | 027672 | 111 | 114 | 125 | |
| | 027675 | 122 | 105 | 000 | |
| 2265 | 027700 | 125 | 116 | 111 | EM206: .ASCIZ ?UNIBUS ADDRESSING ERRORS, MAP RELOCATION DISABLED? |
| | 027703 | 102 | 125 | 123 | |
| | 027706 | 040 | 101 | 104 | |
| | 027711 | 104 | 122 | 105 | |
| | 027714 | 123 | 123 | 111 | |
| | 027717 | 116 | 107 | 040 | |

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 027722 | 105 | 122 | 122 | |
| | 027725 | 117 | 122 | 123 | |
| | 027730 | 054 | 040 | 115 | |
| | 027733 | 101 | 120 | 040 | |
| | 027736 | 122 | 105 | 114 | |
| | 027741 | 117 | 103 | 101 | |
| | 027744 | 124 | 111 | 117 | |
| | 027747 | 116 | 040 | 104 | |
| | 027752 | 111 | 123 | 101 | |
| | 027755 | 102 | 114 | 105 | |
| | 027760 | 104 | 000 | | |
| 2266 | 027762 | 103 | 117 | 125 | EM207: .ASCIZ ?COUNT PATTERN FAILURES IN CACHE REGISTERS? |
| | 027765 | 116 | 124 | 040 | |
| | 027770 | 120 | 101 | 124 | |
| | 027773 | 124 | 105 | 122 | |
| | 027776 | 116 | 040 | 106 | |
| | 030001 | 101 | 111 | 114 | |
| | 030004 | 125 | 122 | 105 | |
| | 030007 | 123 | 040 | 111 | |
| | 030012 | 116 | 040 | 103 | |
| | 030015 | 101 | 103 | 110 | |
| | 030020 | 105 | 040 | 122 | |
| | 030023 | 105 | 107 | 111 | |
| | 030026 | 123 | 124 | 105 | |
| | 030031 | 122 | 123 | 000 | |
| 2267 | | | | | |
| 2268 | 030034 | 122 | 105 | 103 | DH1: .ASCIZ ?RECEIVD EXPECTD TESTNO PC AT ABORT? |
| | 030037 | 105 | 111 | 126 | |
| | 030042 | 104 | 040 | 105 | |
| | 030045 | 130 | 120 | 105 | |
| | 030050 | 103 | 124 | 104 | |
| | 030053 | 040 | 124 | 105 | |
| | 030056 | 123 | 124 | 116 | |
| | 030061 | 117 | 040 | 040 | |
| | 030064 | 120 | 103 | 040 | |
| | 030067 | 101 | 124 | 040 | |
| | 030072 | 101 | 102 | 117 | |
| | 030075 | 122 | 124 | 000 | |
| 2269 | 030100 | 122 | 105 | 103 | DH2: .ASCIZ ?RECEIVD TESTNO PC AT ABORT? |
| | 030103 | 105 | 111 | 126 | |
| | 030106 | 104 | 040 | 124 | |
| | 030111 | 105 | 123 | 124 | |
| | 030114 | 116 | 117 | 040 | |
| | 030117 | 040 | 120 | 103 | |
| | 030122 | 040 | 101 | 124 | |
| | 030125 | 040 | 101 | 102 | |
| | 030130 | 117 | 122 | 124 | |
| | 030133 | 000 | | | |
| 2270 | 030134 | 103 | 117 | 116 | DH3: .ASCII ?CONDITN ADDRESS MAINTEN CONTROL?<CRLF> |
| | 030137 | 104 | 111 | 124 | |
| | 030142 | 116 | 040 | 101 | |
| | 030145 | 104 | 104 | 122 | |
| | 030150 | 105 | 123 | 123 | |
| | 030153 | 040 | 040 | 040 | |
| | 030156 | 115 | 101 | 111 | |
| | 030161 | 116 | 124 | 105 | |
| | 030164 | 116 | 040 | 103 | |

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 030167 | 117 | 116 | 124 | |
| | 030172 | 122 | 117 | 114 | |
| | 030175 | 200 | | | |
| 2271 | 030176 | 122 | 105 | 103 | .ASCII ?RECEIVED REFERENCED REGISTER REGISTER TESTNO PC AT ABORT? |
| | 030201 | 105 | 111 | 126 | |
| | 030204 | 104 | 040 | 122 | |
| | 030207 | 105 | 106 | 105 | |
| | 030212 | 122 | 105 | 116 | |
| | 030215 | 103 | 104 | 040 | |
| | 030220 | 122 | 105 | 107 | |
| | 030223 | 111 | 123 | 124 | |
| | 030226 | 122 | 040 | 122 | |
| | 030231 | 105 | 107 | 111 | |
| | 030234 | 123 | 124 | 122 | |
| | 030237 | 040 | 124 | 105 | |
| | 030242 | 123 | 124 | 116 | |
| | 030245 | 117 | 040 | 040 | |
| | 030250 | 120 | 103 | 040 | |
| | 030253 | 101 | 124 | 040 | |
| | 030256 | 101 | 102 | 117 | |
| 2272 | 030261 | 122 | 124 | 000 | |
| | 030264 | 123 | 124 | 101 | DH5: .ASCII ?STATUS AUTOI/D VIRTUAL?<CRLF> |
| | 030267 | 124 | 125 | 123 | |
| | 030272 | 040 | 040 | 101 | |
| | 030275 | 125 | 124 | 117 | |
| | 030300 | 111 | 057 | 104 | |
| | 030303 | 040 | 126 | 111 | |
| | 030306 | 122 | 124 | 125 | |
| 2273 | 030311 | 101 | 114 | 200 | |
| | 030314 | 122 | 105 | 107 | .ASCII ?REGISTER REGISTER ADDRESS TESTNO PC AT ABORT? |
| | 030317 | 111 | 123 | 124 | |
| | 030322 | 122 | 040 | 122 | |
| | 030325 | 105 | 107 | 111 | |
| | 030330 | 123 | 124 | 122 | |
| | 030333 | 040 | 101 | 104 | |
| | 030336 | 104 | 122 | 105 | |
| | 030341 | 123 | 123 | 040 | |
| | 030344 | 124 | 105 | 123 | |
| | 030347 | 124 | 116 | 117 | |
| | 030352 | 040 | 040 | 120 | |
| | 030355 | 103 | 040 | 101 | |
| | 030360 | 124 | 040 | 101 | |
| | 030363 | 102 | 117 | 122 | |
| 2274 | 030366 | 124 | 000 | | |
| | 030370 | 122 | 105 | 107 | DH6: .ASCII ?REGADRS REGADRS?<CRLF> |
| | 030373 | 101 | 104 | 122 | |
| | 030376 | 123 | 040 | 122 | |
| | 030401 | 105 | 107 | 101 | |
| | 030404 | 104 | 122 | 123 | |
| 2275 | 030407 | 200 | | | |
| | 030410 | 040 | 042 | 117 | .ASCII ? 'OR' 'AND' #ERRORS TESTNO ERRORPC? |
| | 030413 | 122 | 042 | 040 | |
| | 030416 | 040 | 040 | 040 | |
| | 030421 | 042 | 101 | 116 | |
| | 030424 | 104 | 042 | 040 | |
| | 030427 | 040 | 043 | 105 | |
| | 030432 | 122 | 122 | 117 | |

ERROR MESSAGES AND DATA TABLES

| | | | | | |
|------|--------|-----|-----|-----|---|
| | 030435 | 122 | 123 | 040 | |
| | 030440 | 124 | 105 | 123 | |
| | 030443 | 124 | 116 | 117 | |
| | 030446 | 040 | 040 | 105 | |
| | 030451 | 122 | 122 | 117 | |
| | 030454 | 122 | 120 | 103 | |
| | 030457 | 000 | | | |
| 2276 | 030460 | 122 | 105 | 107 | DH10: .ASCII ?REGADRS REGADRS RECEIVD RECEIVD?<CRLF> |
| | 030463 | 101 | 104 | 122 | |
| | 030466 | 123 | 040 | 122 | |
| | 030471 | 105 | 107 | 101 | |
| | 030474 | 104 | 122 | 123 | |
| | 030477 | 040 | 122 | 105 | |
| | 030502 | 103 | 105 | 111 | |
| | 030505 | 126 | 104 | 040 | |
| | 030510 | 122 | 105 | 103 | |
| | 030513 | 105 | 111 | 126 | |
| 2277 | 030516 | 104 | 200 | | |
| | 030520 | 040 | 042 | 117 | .ASCIIZ ? 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO ERRORPC? |
| | 030523 | 122 | 042 | 040 | |
| | 030526 | 040 | 040 | 040 | |
| | 030531 | 042 | 101 | 116 | |
| | 030534 | 104 | 042 | 040 | |
| | 030537 | 040 | 040 | 042 | |
| | 030542 | 117 | 122 | 042 | |
| | 030545 | 040 | 040 | 040 | |
| | 030550 | 040 | 042 | 101 | |
| | 030553 | 116 | 104 | 042 | |
| | 030556 | 040 | 040 | 043 | |
| | 030561 | 105 | 122 | 122 | |
| | 030564 | 117 | 122 | 123 | |
| | 030567 | 040 | 124 | 105 | |
| | 030572 | 123 | 124 | 116 | |
| | 030575 | 117 | 040 | 040 | |
| | 030600 | 105 | 122 | 122 | |
| | 030603 | 117 | 122 | 120 | |
| 2278 | 030606 | 103 | 000 | | |
| | 030610 | 103 | 117 | 125 | DH12: .ASCII ?COUNT COUNT?<CRLF> |
| | 030613 | 116 | 124 | 040 | |
| | 030616 | 040 | 040 | 103 | |
| | 030621 | 117 | 125 | 116 | |
| 2279 | 030624 | 124 | 200 | | |
| | 030626 | 105 | 130 | 120 | .ASCIIZ ?EXPECTED RECEIVD TESTNO ERRORPC? |
| | 030631 | 105 | 103 | 124 | |
| | 030634 | 104 | 040 | 122 | |
| | 030637 | 105 | 103 | 105 | |
| | 030642 | 111 | 126 | 104 | |
| | 030645 | 040 | 124 | 105 | |
| | 030650 | 123 | 124 | 116 | |
| | 030653 | 117 | 040 | 040 | |
| | 030656 | 105 | 122 | 122 | |
| | 030661 | 117 | 122 | 120 | |
| 2280 | 030664 | 103 | 000 | | |
| | 030666 | 122 | 105 | 107 | DH14: .ASCII ?REGLOAD REGLOAD REGDUAL REGDUAL?<CRLF> |
| | 030671 | 114 | 117 | 101 | |
| | 030674 | 104 | 040 | 122 | |
| | 030677 | 105 | 107 | 114 | |

| | | | | | |
|------|--------|-----|-----|-----|--|
| | 031153 | 123 | 040 | 124 | |
| | 031156 | 105 | 123 | 124 | |
| 2284 | 031161 | 116 | 117 | 000 | |
| | 031164 | 122 | 105 | 103 | DM17: .ASCIZ ?RECEIVD TESTNO ERRORPC? |
| | 031167 | 105 | 111 | 126 | |
| | 031172 | 104 | 040 | 124 | |
| | 031175 | 105 | 123 | 124 | |
| | 031200 | 116 | 117 | 040 | |
| | 031203 | 040 | 105 | 122 | |
| | 031206 | 122 | 117 | 122 | |
| 2285 | 031211 | 120 | 103 | 000 | DM23: .ASCII ?EXPECTED EXPECTD RECEIVD RECEIVD<<RLF> |
| | 031214 | 105 | 130 | 120 | |
| | 031217 | 105 | 103 | 124 | |
| | 031222 | 104 | 040 | 105 | |
| | 031225 | 130 | 120 | 105 | |
| | 031230 | 103 | 124 | 104 | |
| | 031233 | 040 | 122 | 105 | |
| | 031236 | 103 | 105 | 111 | |
| | 031241 | 126 | 104 | 040 | |
| | 031244 | 122 | 105 | 103 | |
| | 031247 | 105 | 111 | 126 | |
| 2286 | 031252 | 104 | 200 | | |
| | 031254 | 040 | 042 | 117 | .ASCIZ ? 'OR' 'AND' 'OR' 'AND' #ERRORS TESTNO? |
| | 031257 | 122 | 042 | 040 | |
| | 031262 | 040 | 040 | 040 | |
| | 031265 | 042 | 101 | 116 | |
| | 031270 | 104 | 042 | 040 | |
| | 031273 | 040 | 040 | 042 | |
| | 031276 | 117 | 122 | 042 | |
| | 031301 | 040 | 040 | 040 | |
| | 031304 | 040 | 042 | 101 | |
| | 031307 | 116 | 104 | 042 | |
| | 031312 | 040 | 040 | 043 | |
| | 031315 | 105 | 122 | 122 | |
| | 031320 | 117 | 122 | 123 | |
| | 031323 | 040 | 124 | 105 | |
| | 031326 | 123 | 124 | 116 | |
| 2287 | 031331 | 117 | 000 | | |
| | 031333 | 101 | 104 | 104 | DM25: .ASCIZ ?ADDRUSED BITDIFF TESTNO ERRORPC? |
| | 031336 | 122 | 125 | 123 | |
| | 031341 | 105 | 104 | 040 | |
| | 031344 | 040 | 102 | 111 | |
| | 031347 | 124 | 104 | 111 | |
| | 031352 | 106 | 106 | 040 | |
| | 031355 | 124 | 105 | 123 | |
| | 031360 | 124 | 116 | 117 | |
| | 031363 | 040 | 040 | 105 | |
| | 031366 | 122 | 122 | 117 | |
| | 031371 | 122 | 120 | 103 | |
| 2288 | 031374 | 000 | | | |
| | 031375 | 101 | 104 | 104 | DM27: .ASCIZ ?ADDRUSED TESTING ERRORPC? |
| | 031400 | 122 | 125 | 123 | |
| | 031403 | 105 | 104 | 040 | |
| | 031406 | 040 | 124 | 105 | |
| | 031411 | 123 | 124 | 116 | |
| | 031414 | 117 | 040 | 040 | |
| | 031417 | 105 | 122 | 122 | |

| ERROR MESSAGES AND DATA TABLES | | | | | |
|--------------------------------|--------|-----|-----|-----|---|
| | 031422 | 117 | 122 | 120 | |
| | 031425 | 103 | 000 | | |
| 2289 | 031427 | 124 | 105 | 123 | DH30: .ASCIZ ?TESTNO ERRORPC? |
| | 031432 | 124 | 116 | 117 | |
| | 031435 | 040 | 040 | 105 | |
| | 031440 | 122 | 122 | 117 | |
| | 031443 | 122 | 120 | 103 | |
| | 031446 | 000 | | | |
| 2290 | 031447 | 122 | 105 | 115 | DH34: .ASCIZ ?REMOVED MISSING TESTNO ERRORPC? |
| | 031452 | 117 | 126 | 105 | |
| | 031455 | 104 | 040 | 115 | |
| | 031460 | 111 | 123 | 123 | |
| | 031463 | 111 | 116 | 107 | |
| | 031466 | 040 | 124 | 105 | |
| | 031471 | 123 | 124 | 116 | |
| | 031474 | 117 | 040 | 040 | |
| | 031477 | 105 | 122 | 122 | |
| | 031502 | 117 | 122 | 120 | |
| | 031505 | 103 | 000 | | |
| 2291 | 031507 | 114 | 117 | 127 | DH35: .ASCIZ ?LOWEST HIGHEST TESTNO ERRORPC? |
| | 031512 | 105 | 123 | 124 | |
| | 031515 | 040 | 040 | 110 | |
| | 031520 | 111 | 107 | 110 | |
| | 031523 | 105 | 123 | 124 | |
| | 031526 | 040 | 124 | 105 | |
| | 031531 | 123 | 124 | 116 | |
| | 031534 | 117 | 040 | 040 | |
| | 031537 | 105 | 122 | 122 | |
| | 031542 | 117 | 122 | 120 | |
| | 031545 | 103 | 000 | | |
| 2292 | 031547 | 124 | 105 | 123 | DH36: .ASCIZ ?TESTNO ERRORPC UNIBUS ADDRESS OF MAP REGISTER UNDER TEST? |
| | 031552 | 124 | 116 | 117 | |
| | 031555 | 040 | 040 | 105 | |
| | 031560 | 122 | 122 | 117 | |
| | 031563 | 122 | 120 | 103 | |
| | 031566 | 040 | 125 | 116 | |
| | 031571 | 111 | 102 | 125 | |
| | 031574 | 123 | 040 | 101 | |
| | 031577 | 104 | 104 | 122 | |
| | 031602 | 105 | 123 | 123 | |
| | 031605 | 040 | 117 | 106 | |
| | 031610 | 040 | 115 | 101 | |
| | 031613 | 120 | 040 | 122 | |
| | 031616 | 105 | 107 | 111 | |
| | 031621 | 123 | 124 | 105 | |
| | 031624 | 122 | 040 | 125 | |
| | 031627 | 116 | 104 | 105 | |
| | 031632 | 122 | 040 | 124 | |
| | 031635 | 105 | 123 | 124 | |
| | 031640 | 000 | | | |
| 2293 | 031641 | 103 | 117 | 116 | DH40: .ASCII ?CONDITN CONDITN?<CRLF> |
| | 031644 | 104 | 111 | 124 | |
| | 031647 | 116 | 040 | 103 | |
| | 031652 | 117 | 116 | 104 | |
| | 031655 | 111 | 124 | 116 | |
| | 031660 | 200 | | | |
| 2294 | 031661 | 105 | 130 | 120 | .ASCIZ ?EXPECTED RECEIVED TESTNO ERRORPC? |

| | | | | | |
|------|--------|-----|-----|-----|--|
| | 031664 | 105 | 103 | 124 | |
| | 031667 | 104 | 040 | 122 | |
| | 031672 | 105 | 103 | 105 | |
| | 031675 | 111 | 126 | 104 | |
| | 031700 | 040 | 124 | 105 | |
| | 031703 | 123 | 124 | 116 | |
| | 031706 | 117 | 040 | 040 | |
| | 031711 | 105 | 122 | 122 | |
| | 031714 | 117 | 122 | 120 | |
| | 031717 | 103 | 000 | | |
| 2295 | 031721 | 103 | 117 | 122 | DH41: .ASCII ?CORRECT ADDRESS?<CRLF> |
| | 031724 | 122 | 105 | 103 | |
| | 031727 | 124 | 040 | 101 | |
| | 031732 | 104 | 104 | 122 | |
| | 031735 | 105 | 123 | 123 | |
| | 031740 | 200 | | | |
| 2296 | 031741 | 101 | 104 | 104 | .ASCIZ ?ADDRESS FETCHED TESTNO ERRORPC? |
| | 031744 | 122 | 105 | 123 | |
| | 031747 | 123 | 040 | 106 | |
| | 031752 | 105 | 124 | 103 | |
| | 031755 | 110 | 105 | 104 | |
| | 031760 | 040 | 124 | 105 | |
| | 031763 | 123 | 124 | 116 | |
| | 031766 | 117 | 040 | 040 | |
| | 031771 | 105 | 122 | 122 | |
| | 031774 | 117 | 122 | 120 | |
| | 031777 | 103 | 000 | | |
| 2297 | 032001 | 103 | 117 | 122 | DH42: .ASCII ?CORRECT EXPECTD RECEIVD?<CRLF> |
| | 032004 | 122 | 105 | 103 | |
| | 032007 | 124 | 040 | 105 | |
| | 032012 | 130 | 120 | 105 | |
| | 032015 | 103 | 124 | 104 | |
| | 032020 | 040 | 122 | 105 | |
| | 032023 | 103 | 105 | 111 | |
| | 032026 | 126 | 104 | 200 | |
| 2298 | 032031 | 101 | 104 | 104 | .ASCIZ ?ADDRESS DATA FROM UB TESTNO ERRORPC? |
| | 032034 | 122 | 105 | 123 | |
| | 032037 | 123 | 040 | 104 | |
| | 032042 | 101 | 124 | 101 | |
| | 032045 | 040 | 040 | 040 | |
| | 032050 | 040 | 106 | 122 | |
| | 032053 | 117 | 115 | 040 | |
| | 032056 | 125 | 102 | 040 | |
| | 032061 | 124 | 105 | 123 | |
| | 032064 | 124 | 116 | 117 | |
| | 032067 | 040 | 040 | 105 | |
| | 032072 | 122 | 122 | 117 | |
| | 032075 | 122 | 120 | 103 | |
| | 032100 | 000 | | | |
| 2299 | 032101 | 122 | 105 | 107 | DH201: .ASCIZ ?REGADRS TESTNO ERRORPC? |
| | 032104 | 101 | 104 | 122 | |
| | 032107 | 123 | 040 | 124 | |
| | 032112 | 105 | 123 | 124 | |
| | 032115 | 116 | 117 | 040 | |
| | 032120 | 040 | 105 | 122 | |
| | 032123 | 122 | 117 | 122 | |
| | 032126 | 120 | 103 | 000 | |

| | | | | | |
|------|--------|-----|-----|-----|--|
| 2300 | 032131 | 122 | 105 | 107 | DH202: .ASCIZ ?REGADRS DATAREC TESTNO ERRORPC? |
| | 032134 | 101 | 104 | 122 | |
| | 032137 | 123 | 040 | 104 | |
| | 032142 | 101 | 124 | 101 | |
| | 032145 | 122 | 105 | 103 | |
| | 032150 | 040 | 124 | 105 | |
| | 032153 | 123 | 124 | 116 | |
| | 032156 | 117 | 040 | 040 | |
| | 032161 | 105 | 122 | 122 | |
| | 032164 | 117 | 122 | 120 | |
| | 032167 | 103 | 000 | | |
| 2301 | 032171 | 115 | 101 | 120 | DH203: .ASCII ?MAPREG MAPREG?<CRLF> |
| | 032174 | 122 | 105 | 107 | |
| | 032177 | 040 | 040 | 115 | |
| | 032202 | 101 | 120 | 122 | |
| | 032205 | 105 | 107 | 200 | |
| 2302 | 032210 | 124 | 105 | 123 | .ASCIZ ?TESTING DUALFD TESTNO ERRORPC? |
| | 032213 | 124 | 111 | 116 | |
| | 032216 | 107 | 040 | 104 | |
| | 032221 | 125 | 101 | 114 | |
| | 032224 | 105 | 104 | 040 | |
| | 032227 | 040 | 124 | 105 | |
| | 032232 | 123 | 124 | 116 | |
| | 032235 | 117 | 040 | 040 | |
| | 032240 | 105 | 122 | 122 | |
| | 032243 | 117 | 122 | 120 | |
| | 032246 | 103 | 000 | | |
| 2303 | 032250 | 122 | 105 | 107 | DH204: .ASCIZ ?REGADRS PATTERN EXPECTD RECEIVD TESTNO ERRORPC? |
| | 032253 | 101 | 104 | 122 | |
| | 032256 | 123 | 040 | 120 | |
| | 032261 | 101 | 124 | 124 | |
| | 032264 | 105 | 122 | 116 | |
| | 032267 | 040 | 105 | 130 | |
| | 032272 | 120 | 105 | 103 | |
| | 032275 | 124 | 104 | 040 | |
| | 032300 | 122 | 105 | 103 | |
| | 032303 | 105 | 111 | 126 | |
| | 032306 | 104 | 040 | 124 | |
| | 032311 | 105 | 123 | 124 | |
| | 032314 | 116 | 117 | 040 | |
| | 032317 | 040 | 105 | 122 | |
| | 032322 | 122 | 117 | 122 | |
| | 032325 | 120 | 103 | 000 | |
| 2304 | 032330 | 105 | 130 | 120 | DH205: .ASCIZ ?EXPECTD RECEIVD ADDRSLD TESTNO ERRORPC? |
| | 032333 | 105 | 103 | 124 | |
| | 032336 | 104 | 040 | 122 | |
| | 032341 | 105 | 103 | 105 | |
| | 032344 | 111 | 126 | 104 | |
| | 032347 | 040 | 101 | 104 | |
| | 032352 | 104 | 122 | 123 | |
| | 032355 | 114 | 117 | 101 | |
| | 032360 | 104 | 040 | 124 | |
| | 032363 | 105 | 123 | 124 | |
| | 032366 | 116 | 117 | 040 | |
| | 032371 | 040 | 105 | 122 | |
| | 032374 | 122 | 117 | 122 | |
| | 032377 | 120 | 103 | 000 | |

| | | | | | |
|------|--------|-----|-----|-----|--|
| 2305 | 032402 | 101 | 104 | 104 | DM206: .ASCII ?ADDRESS ADDRESS?<CRLF> |
| | 032405 | 122 | 105 | 123 | |
| | 032410 | 123 | 040 | 101 | |
| | 032413 | 104 | 104 | 122 | |
| | 032416 | 105 | 123 | 123 | |
| | 032421 | 200 | | | |
| 2306 | 032422 | 105 | 130 | 120 | .ASCII ?EXPECTED RECEIVED TESTNO ERRORPC? |
| | 032425 | 105 | 103 | 124 | |
| | 032430 | 104 | 040 | 122 | |
| | 032433 | 105 | 103 | 105 | |
| | 032436 | 111 | 126 | 104 | |
| | 032441 | 040 | 124 | 105 | |
| | 032444 | 123 | 124 | 116 | |
| | 032447 | 117 | 040 | 040 | |
| | 032452 | 105 | 122 | 122 | |
| | 032455 | 117 | 122 | 120 | |
| | 032460 | 103 | 000 | | |
| 2307 | 032462 | 122 | 105 | 107 | DM207: .ASCII ?REGISTER EXPECTED RECEIVED?<CRLF> |
| | 032465 | 111 | 123 | 124 | |
| | 032470 | 122 | 040 | 105 | |
| | 032473 | 130 | 120 | 105 | |
| | 032476 | 103 | 124 | 104 | |
| | 032501 | 040 | 122 | 105 | |
| | 032504 | 103 | 105 | 111 | |
| | 032507 | 126 | 104 | 200 | |
| 2308 | 032512 | 101 | 104 | 104 | .ASCII ?ADDRESS DATA DATA TESTNO ERRORPC? |
| | 032515 | 122 | 105 | 123 | |
| | 032520 | 123 | 040 | 040 | |
| | 032523 | 104 | 101 | 124 | |
| | 032526 | 101 | 040 | 040 | |
| | 032531 | 040 | 040 | 104 | |
| | 032534 | 101 | 124 | 101 | |
| | 032537 | 040 | 040 | 040 | |
| | 032542 | 124 | 105 | 123 | |
| | 032545 | 124 | 116 | 117 | |
| | 032550 | 040 | 040 | 105 | |
| | 032553 | 122 | 122 | 117 | |
| | 032556 | 122 | 120 | 103 | |
| 2309 | 032561 | 000 | | | |

| | | | | | | | | | | |
|------|--------|--------|--------|--------|--------|-------|---|--|--|--|
| 2311 | | | | | | | | | | |
| 2312 | | | | | | .EVEN | | | | |
| 2313 | 032562 | 001270 | 001266 | 001264 | DT1: | .WORD | PCPUER,CPUEXP,TESTNO,BADPC,0 | | | |
| | 032570 | 001300 | 000000 | | | | | | | |
| 2314 | 032574 | 001270 | 001264 | 001300 | DT2: | .WORD | PCPUER,TESTNO,BADPC,0 | | | |
| | 032602 | 000000 | | | | | | | | |
| 2315 | 032604 | 001156 | 001172 | 001264 | DT4: | .WORD | \$REG0,\$TMP0,TESTNO,\$ERRPC,0 | | | |
| | 032612 | 001114 | 000000 | | | | | | | |
| 2316 | 032616 | 001310 | 001312 | 001314 | DT5: | .WORD | PMMR0,PMMR1,PMMR2,TESTNO,BADPC,0 | | | |
| | 032624 | 001264 | 001300 | 000000 | | | | | | |
| 2317 | 032632 | 001230 | 001226 | 001256 | DT6: | .WORD | ADDROR,ADRAND,ERRCNT,TESTNO,\$ERRPC,0 | | | |
| | 032640 | 001264 | 001114 | 000000 | | | | | | |
| 2318 | 032646 | 001230 | 001226 | 001234 | DT10: | .WORD | ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,\$ERRPC,0 | | | |
| | 032654 | 001232 | 001256 | 001264 | | | | | | |
| | 032662 | 001114 | 000000 | | | | | | | |
| 2319 | 032666 | 001162 | 001156 | 001264 | DT12: | .WORD | \$REG2,\$REG0,TESTNO,\$ERRPC,0 | | | |
| | 032674 | 001114 | 000000 | | | | | | | |
| 2320 | 032700 | 001164 | 001160 | 001264 | DT13: | .WORD | \$REG3,\$REG1,TESTNO,\$ERRPC,0 | | | |
| | 032706 | 001114 | 000000 | | | | | | | |
| 2321 | 032712 | 001230 | 001226 | 001234 | DT14: | .WORD | ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0 | | | |
| | 032720 | 001232 | 001256 | 001264 | | | | | | |
| | 032726 | 000000 | | | | | | | | |
| 2322 | 032730 | 001230 | 001226 | 001240 | DT15: | .WORD | ADDROR,ADRAND,PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0 | | | |
| | 032736 | 001236 | 001234 | 001232 | | | | | | |
| | 032744 | 001256 | 001264 | 000000 | | | | | | |
| 2323 | 032752 | 001156 | 001264 | 001114 | DT17: | .WORD | \$REG0,TESTNO,\$ERRPC,0 | | | |
| | 032760 | 000000 | | | | | | | | |
| 2324 | 032762 | 001160 | 001264 | 001114 | DT20: | .WORD | \$REG1,TESTNO,\$ERRPC,0 | | | |
| | 032770 | 000000 | | | | | | | | |
| 2325 | 032772 | 001240 | 001236 | 001234 | DT23: | .WORD | PATTOR,PATAND,DATAOR,DATAND,ERRCNT,TESTNO,0 | | | |
| | 033000 | 001232 | 001256 | 001264 | | | | | | |
| | 033006 | 000000 | | | | | | | | |
| 2326 | 033010 | 001166 | 001156 | 001264 | DT25: | .WORD | \$REG4,\$REG0,TESTNO,\$ERRPC,0 | | | |
| | 033016 | 001114 | 000000 | | | | | | | |
| 2327 | 033022 | 001160 | 001264 | 001114 | DT27: | .WORD | \$REG1,TESTNO,\$ERRPC,0 | | | |
| | 033030 | 000000 | | | | | | | | |
| 2328 | 033032 | 001264 | 001114 | 000000 | DT30: | .WORD | TESTNO,\$ERRPC,0 | | | |
| 2329 | 033040 | 001256 | 001260 | 001264 | DT34: | .WORD | ERRCNT,CNTR,TESTNO,\$ERRPC,0 | | | |
| | 033046 | 001114 | 000000 | | | | | | | |
| 2330 | 033052 | 001242 | 001244 | 001264 | DT35: | .WORD | LOWEST,HIGEST,TESTNO,\$ERRPC,0 | | | |
| | 033060 | 001114 | 000000 | | | | | | | |
| 2331 | 033064 | 001264 | 001114 | 001156 | DT36: | .WORD | TESTNO,\$ERRPC,\$REG0,0 | | | |
| | 033072 | 000000 | | | | | | | | |
| 2332 | 033074 | 001230 | 001226 | 001234 | DT37: | .WORD | ADDROR,ADRAND,DATAOR,DATAND,ERRCNT,TESTNO,0 | | | |
| | 033102 | 001232 | 001256 | 001264 | | | | | | |
| | 033110 | 000000 | | | | | | | | |
| 2333 | 033112 | 001266 | 001270 | 001264 | DT40: | .WORD | CPUEXP,PCPUER,TESTNO,\$ERRPC,0 | | | |
| | 033120 | 001114 | 000000 | | | | | | | |
| 2334 | 033124 | 001162 | 001160 | 001264 | DT41: | .WORD | \$REG2,\$REG1,TESTNO,\$ERRPC,0 | | | |
| | 033132 | 001114 | 000000 | | | | | | | |
| 2335 | 033136 | 001160 | 001164 | 001162 | DT42: | .WORD | \$REG1,\$REG3,\$REG2,TESTNO,\$ERRPC,0 | | | |
| | 033144 | 001264 | 001114 | 000000 | | | | | | |
| 2336 | 033152 | 001156 | 001264 | 001114 | DT201: | .WORD | \$REG0,TESTNO,\$ERRPC,0 | | | |
| | 033160 | 000000 | | | | | | | | |
| 2337 | 033162 | 001156 | 001172 | 001264 | DT202: | .WORD | \$REG0,\$TMP0,TESTNO,\$ERRPC,0 | | | |
| | 033170 | 001114 | 000000 | | | | | | | |
| 2338 | 033174 | 001156 | 001160 | 001264 | DT203: | .WORD | \$REG0,\$REG1,TESTNO,\$ERRPC,0 | | | |

| Line | Address | Offset | Value | Label | Mode | Fields |
|------|---------|--------|--------|--------|--------|--|
| 2339 | 033202 | 001114 | 000000 | | | |
| | 033206 | 001156 | 001162 | 001166 | DT204: | .WORD \$REG0,\$REG2,\$REG4,\$REG3,TESTNO,\$ERRPC,0 |
| | 033214 | 001164 | 001264 | 001114 | | |
| | 033222 | 000000 | | | | |
| 2340 | 033224 | 001160 | 001156 | 001162 | DT205: | .WORD \$REG1,\$REG0,\$REG2,TESTNO,\$ERRPC,0 |
| | 033232 | 001264 | 001114 | 000000 | | |
| 2341 | 033240 | 001156 | 001164 | 001264 | DT206: | .WORD \$REG0,\$REG3,TESTNO,\$ERRPC,0 |
| | 033246 | 001114 | 000000 | | | |
| 2342 | 033252 | 001156 | 001162 | 001164 | DT207: | .WORD \$REG0,\$REG2,\$REG3,TESTNO,\$ERRPC,0 |
| | 033260 | 001264 | 001114 | 000000 | | |
| 2343 | | | | | | |
| 2344 | | | | | | |
| 2345 | 033266 | 000 | 000 | 000 | DF1: | .BYTE 0,0,0,0 |
| | 033271 | 000 | | | | |
| 2346 | 033272 | 000 | 000 | 000 | DF2: | .BYTE 0,0,0 |
| 2347 | 033275 | 000 | 002 | 000 | DF3: | .BYTE 0,2,0,0,0,0 |
| | 033300 | 000 | 000 | 000 | | |
| 2348 | 033303 | 000 | 000 | 000 | DF5: | .BYTE 0,0,0,0,0 |
| | 033306 | 000 | 000 | | | |
| 2349 | 033310 | 000 | 000 | 001 | DF6: | .BYTE 0,0,1,0,0 |
| | 033313 | 000 | 000 | | | |
| 2350 | 033315 | 000 | 000 | 000 | DF10: | .BYTE 0,0,0,0,1,0,0 |
| | 033320 | 000 | 001 | 000 | | |
| | 033323 | 000 | | | | |
| 2351 | 033324 | 000 | 000 | 000 | DF12: | .BYTE 0,0,0,0 |
| | 033327 | 000 | | | | |
| 2352 | 033330 | 000 | 000 | 000 | DF14: | .BYTE 0,0,0,0,1,0 |
| | 033333 | 000 | 001 | 000 | | |
| 2353 | 033336 | 000 | 000 | 000 | DF15: | .BYTE 0,0,0,0,0,0,1,0 |
| | 033341 | 000 | 000 | 000 | | |
| | 033344 | 001 | 000 | | | |
| 2354 | 033346 | 000 | 000 | 000 | DF17: | .BYTE 0,0,0 |
| 2355 | 033351 | 000 | 000 | 000 | DF23: | .BYTE 0,0,0,0,1,0 |
| | 033354 | 000 | 001 | 000 | | |
| 2356 | 033357 | 003 | 004 | 000 | DF25: | .BYTE 3,4,0,0 |
| | 033362 | 000 | | | | |
| 2357 | 033363 | 003 | 000 | 000 | DF27: | .BYTE 3,0,0 |
| 2358 | 033366 | 000 | 000 | | DF30: | .BYTE 0,0 |
| 2359 | 033370 | 000 | 000 | 000 | DF34: | .BYTE 0,0,0,0 |
| | 033373 | 000 | | | | |
| 2360 | 033374 | 004 | 004 | 000 | DF35: | .BYTE 4,4,0,0 |
| | 033377 | 000 | | | | |
| 2361 | 033400 | 000 | 000 | 003 | DF36: | .BYTE 0,0,3 |
| 2362 | 033403 | 000 | 000 | 000 | DF37: | .BYTE 0,0,0,0,1,0 |
| | 033406 | 000 | 001 | 000 | | |
| 2363 | 033411 | 000 | 000 | 000 | DF40: | .BYTE 0,0,0,0 |
| | 033414 | 000 | | | | |
| 2364 | 033415 | 000 | 000 | 000 | DF41: | .BYTE 0,0,0,0 |
| | 033420 | 000 | | | | |
| 2365 | 033421 | 003 | 000 | 000 | DF42: | .BYTE 3,0,0,0,0 |
| | 033424 | 000 | 000 | | | |
| 2366 | 033426 | 000 | 000 | 000 | DF201: | .BYTE 0,0,0 |
| 2367 | 033431 | 000 | 000 | 000 | DF202: | .BYTE 0,0,0,0 |
| | 033434 | 000 | | | | |
| 2368 | 033435 | 000 | 000 | 000 | DF203: | .BYTE 0,0,0,0 |
| | 033440 | 000 | | | | |
| 2369 | 033441 | 000 | 000 | 000 | DF204: | .BYTE 0,0,0,0,0,0,0 |

| | | | | | | | |
|------|--------|-----|-----|-----|--------|-------|-----------|
| 2370 | 033444 | 000 | 000 | 000 | | | |
| | 033447 | 000 | 000 | 003 | DF205: | .BYTE | 0.0.3.0.0 |
| | 033452 | 000 | 000 | | | | |
| 2371 | 033454 | 000 | 000 | 000 | DF206: | .BYTE | 0.0.0.0 |
| | 033457 | 000 | 000 | | | | |
| 2372 | 033460 | 000 | 000 | 000 | DF207: | .BYTE | 0.0.0.0.0 |
| | 033463 | 000 | 000 | | | | |
| 2373 | | | | | | | |
| 2374 | | | | | | .EVEN | |
| 2375 | 000001 | | | | | .END | |

| | | | | |
|-----------------|-----------------|---------------|-----------------|-----------------|
| ABASE = 000000 | BIT01 = 000002 | DF30 033366 | DT30 033032 | EM7 022303 |
| ACDW1 = 000000 | BIT02 = 000004 | DF34 033370 | DT34 033040 | EREXIT 021002 |
| ACDW2 = 000000 | BIT03 = 000010 | DF35 033374 | DT35 033052 | ERRCNT 001256 |
| ACPUOP = 000000 | BIT04 = 000020 | DF36 033400 | DT36 033064 | ERROR = 104000 |
| ADDROR 001230 | BIT05 = 000040 | DF37 033403 | DT37 033074 | ERRVEC = 000004 |
| ADDW0 = 000000 | BIT06 = 000100 | DF40 033411 | DT4 032604 | ERTYPE 001736 |
| ADDW1 = 000000 | BIT07 = 000200 | DF41 033415 | DT40 033112 | ER200 001656 |
| ADDW10 = 000000 | BIT08 = 000400 | DF42 033421 | DT41 033124 | FLAG 001262 |
| ADDW11 = 000000 | BIT09 = 001000 | DF5 033303 | DT42 033136 | HIAGRS = 177742 |
| ADDW12 = 000000 | BIT1 = 000002 | DF6 033310 | DT5 032616 | HIGEST 001244 |
| ADDW13 = 000000 | BIT10 = 002000 | DH 030034 | DT6 032632 | HITMIS = 177752 |
| ADDW14 = 000000 | BIT11 = 004000 | DH10 030460 | DUALAD 004342 | HREGL 001252 |
| ADDW15 = 000000 | BIT12 = 010000 | DH12 030610 | EMTVEC = 000030 | HREGU 001254 |
| ADDW2 000000 | BIT13 = 020000 | DH14 030666 | EM1 021554 | HT = 000011 |
| ADDW3 000000 | BIT14 = 040000 | DH15 031005 | EM'0 022365 | IOTVEC = 000020 |
| ADDW4 000000 | BIT15 = 100000 | DH17 031164 | EM11 022456 | KDPAR0 = 172360 |
| ADDW5 = 000000 | BIT2 = 000004 | DH2 030100 | EM12 022550 | KDPAR1 = 172362 |
| ADDW6 = 000000 | BIT3 000010 | DH201 032101 | EM13 022636 | KDPAR2 = 172364 |
| ADDW7 = 000000 | BIT4 000020 | DH202 032131 | EM14 022730 | KDPAR3 = 172366 |
| ADDW8 = 000000 | BIT5 000040 | DH203 032171 | EM15 023023 | KDPAR4 = 172370 |
| ADDW9 = 000000 | BIT6 000100 | DH204 032250 | EM16 023127 | KDPAR5 = 172372 |
| ADEVCT = 000000 | BIT7 000200 | DH205 032330 | EM17 023232 | KDPAR6 = 172374 |
| ADEVN = 000000 | BIT8 = 000400 | DH206 032402 | EM2 021636 | KDPAR7 = 172376 |
| ADRAND 001226 | BIT9 = 001000 | DH207 032462 | EM20 023354 | KDPDR0 = 172320 |
| AENV 000000 | BPTVEC = 000014 | DH23 031214 | EM201 027327 | KDPDR1 = 172322 |
| AENVN = 000000 | CACHE = 000114 | DH25 031333 | EM202 027403 | KDPDR2 = 172324 |
| AFATAL = 000000 | CLRMAR 004120 | DH27 031375 | EM203 027456 | KDPDR3 = 172326 |
| AMADR1 = 000000 | ^NTR 001260 | DH3 030134 | EM204 027551 | KDPDR4 = 172330 |
| AMADR2 = 000000 | CONTRL = 177746 | DH30 031427 | EM205 027631 | KDPDR5 = 172332 |
| AMADR3 = 000000 | COUNT 004416 | DH34 031447 | EM206 027700 | KDPDR6 = 172334 |
| AMADR4 = 000000 | CPFLAG 004564 | DH35 031507 | EM207 027762 | KDPDR7 = 172336 |
| AMAMS1 = 000000 | CPUER 004562 | DH36 031547 | EM21 023502 | KERSTK = 001100 |
| AMAMS2 = 000000 | CPUEER = 177766 | DH40 031641 | EM22 023646 | KIPAR0 = 172340 |
| AMAMS3 = 000000 | CPUEXP 001266 | DH41 031721 | EM23 024012 | KIPAR1 = 172342 |
| AMAMS4 = 000000 | CR = 000015 | DH42 032001 | EM24 024106 | KIPAR2 = 172344 |
| AMSGAD = 000000 | CRLF = 000200 | DH5 030264 | EM25 024206 | KIPAR3 = 172346 |
| AMSGLG = 000000 | DATA 001324 | DH6 030370 | EM26 024313 | KIPAR4 = 172350 |
| AMSGTY = 000000 | DATAND 001232 | DISPLA 001152 | EM3 021705 | KIPAR5 = 172352 |
| AMTYP1 = 000000 | DATAOR 001234 | DT1 032562 | EM30 024434 | KIPAR6 = 172354 |
| AMTYP2 = 000000 | DF1 033266 | DT10 032646 | EM31 024610 | KIPAR7 = 172356 |
| AMTYP3 = 000000 | DF10 033315 | DT12 032666 | EM32 024702 | KIPDR0 = 172300 |
| AMTYP4 = 000000 | DF12 033324 | DT13 032700 | EM33 024754 | KIPDR1 = 172302 |
| APASS = 000000 | DF14 033330 | DT14 032712 | EM35 025166 | KIPDR2 = 172304 |
| APRIOR 000000 | DF15 033336 | DT15 032730 | EM36 025736 | KIPDR3 = 172306 |
| APTCSU = 000040 | DF17 033346 | DT17 032752 | EM37 026033 | KIPDR4 = 172310 |
| APTENV = 000001 | DF2 033272 | DT2 032574 | EM4 022006 | KIPDR5 = 172312 |
| APTSIZ = 000200 | DF201 033426 | DT20 032762 | EM40 026136 | KIPDR6 = 172314 |
| APTSPO = 000100 | DF202 033431 | DT201 033152 | EM41 026231 | KIPDR7 = 172316 |
| ASWREG = 000000 | DF203 033435 | DT202 033162 | EM42 026313 | KSP = 2000006 |
| ATESTN = 000000 | DF204 033441 | DT203 033174 | EM43 026406 | LF = 000012 |
| AUNIT = 000000 | DF205 033447 | DT204 033206 | EM45 026473 | LOADRS = 177740 |
| AUSWR = 000000 | DF206 033454 | DT205 033224 | EM46 026626 | LOOP 010332 |
| AVECT1 = 000000 | DF207 033460 | DT206 033240 | EM47 026750 | LOWEST 001242 |
| AVECT2 = 000000 | DF23 033351 | DT207 033252 | EM5 022115 | LREGL 001246 |
| BADPC 001300 | DF25 033357 | DT23 032772 | EM50 027103 | LREGU 001250 |
| BIT0 000001 | DF27 033363 | DT25 033010 | EM52 027270 | MAINT = 177750 |
| BIT00 000001 | DF3 033275 | DT27 033022 | EM6 022223 | MAPHO 170202 |

MAPH00= 170202
 MAPH01 170206
 MAPH02= 170212
 MAPH03= 170216
 MAPH04= 170222
 MAPH05= 170226
 MAPH06= 170232
 MAPH07= 170236
 MAPH1 = 170206
 MAPH10= 170242
 MAPH11= 170246
 MAPH12= 170252
 MAPH13= 170256
 MAPH14= 170262
 MAPH15= 170266
 MAPH16= 170272
 MAPH17 170276
 MAPH2 = 170212
 MAPH20= 170302
 MAPH21= 170306
 MAPH22= 170312
 MAPH23= 170316
 MAPH24= 170320
 MAPH25 170326
 MAPH26= 170332
 MAPH27= 170336
 MAPH3 = 170216
 MAPH30= 170342
 MAPH31= 170346
 MAPH32= 170352
 MAPH33= 170356
 MAPH34= 170362
 MAPH35= 170366
 MAPH36= 170372
 MAPH37= 170376
 MAPH4 = 170222
 MAPH5 = 170226
 MAPH6 170232
 MAPH7 170236
 MAPL0 170200
 MAPL00 170200
 MAPL01= 170204
 MAPL02= 170210
 MAPL03= 170214
 MAPL04= 170220
 MAPL05= 170224
 MAPL06= 170230
 MAPL07= 170234
 MAPL1 = 170204
 MAPL10= 170240
 MAPL11= 170244
 MAPL12= 170250
 MAPL13= 170254
 MAPL14= 170260
 MAPL15= 170264
 MAPL16= 170270
 MAPL17 170274

MAPL2 = 170210
 MAPL20= 170300
 MAPL21= 170304
 MAPL22= 170310
 MAPL23= 170314
 MAPL24= 170320
 MAPL25= 170324
 MAPL26= 170330
 MAPL27= 170334
 MAPL3 = 170214
 MAPL30= 170340
 MAPL31= 170344
 MAPL32= 170350
 MAPL33 170354
 MAPL34= 170360
 MAPL35= 170364
 MAPL36= 170370
 MAPL37= 170374
 MAPL4 = 170220
 MAPL5 = 170224
 MAPL6 = 170230
 MAPL7 = 170234
 MEMERR= 177744
 MMFLAG 004676
 MMR0 = 177572
 MMR1 = 177574
 MMR2 = 177576
 MMR3 = 172516
 MMTRAP 004674
 MMVEC = 000250
 NEXMEM= 000040
 NXTTST 001322
 OLDPC 001302
 OLDPS 001304
 OLDPSW 001306
 PADDRSH 001224
 PADDRSL 001222
 PATAND 001236
 PATTOR 001240
 PCONTR 001274
 PCPUER 001270
 PIRQ = 177772
 PIRQVE= 000240
 PMAINT 001276
 PMMR0 001310
 PMMR1 001312
 PMMR2 001314
 PPARER 001272
 PR0 = 000000
 PR1 = 000040
 PR2 = 000100
 PR3 = 000140
 PR4 = 000200
 PR5 = 000240
 PR6 = 000300
 PR7 = 000340
 PS = 177776

PSW = 177776
 PWRMSG 003670
 PWRVEC= 000024
 RELC22 012166
 RESREG= 104414
 RESVEC= 000010
 RETRY 001320
 RSIZE 001316
 R10 =%000000
 R11 =%000001
 R12 =%000002
 R13 =%000003
 R14 =%000004
 R15 =%000005
 R6 =%000006
 R7 =%000007
 SAVREG= 104412
 SCOPE = 000004
 SDPAR0= 172260
 SDPAR1= 172262
 SDPAR2= 172264
 SDPAR3= 172266
 SDPAR4= 172270
 SDPAR5= 172272
 SDPAR6= 172274
 SDPAR7= 172276
 SDPDR0= 172220
 SDPDR1= 172222
 SDPDR2= 172224
 SDPDR3= 172226
 SDPDR4= 172230
 SDPDR5= 172232
 SDPDR6= 172234
 SDPDR7= 172236
 SIPAR0= 172240
 SIPAR1= 172242
 SIPAR2= 172244
 SIPAR3= 172246
 SIPAR4= 172250
 SIPAR5= 172252
 SIPAR6= 172254
 SIPAR7= 172256
 SIPDR0= 172200
 SIPDR1= 172202
 SIPDR2= 172204
 SIPDR3= 172206
 SIPDR4= 172210
 SIPDR5= 172212
 SIPDR6= 172214
 SIPDR7= 172216
 SIZEH1= 177762
 SIZEJ 013064
 SIZELO= 177760
 SR0 = 177572
 SR1 = 177574
 SR2 = 177576
 SR3 = 172516

SSP =%000006
 STACK = 001100
 START 010000
 STKLMT= 177774
 SUPSTK= 000700
 SWR 001150
 SW0 = 000001
 SW00 = 000001
 SW01 = 000002
 SW02 = 000004
 SW03 = 000010
 SW04 = 000020
 SW05 = 000040
 SW06 = 000100
 SW07 = 000200
 SW08 = 000400
 SW09 = 001000
 SW1 = 000002
 SW10 = 002000
 SW11 = 004000
 SW12 = 010000
 SW13 = 020000
 SW14 = 040000
 SW15 = 100000
 SW2 = 000004
 SW3 = 000010
 SW4 = 000020
 SW5 = 000040
 SW6 = 000100
 SW7 = 000200
 SW8 = 000400
 SW9 = 001000
 SYSTID= 177764
 TBIT = 000020
 TBITO = 104416
 TBITOF 004054
 TBITR = 104420
 TBITRE 004102
 TBITVE= 000014
 TESTNO 001264
 TIMEOU 004136
 TIMOUT= 000020
 TKVEC = 000060
 TOFLAG 004140
 TPVEC = 000064
 TRAPVE= 000034
 TRTVEC= 000014
 TST1 010422
 TST10 011322
 TST11 011446
 TST12 011572
 TST13 011716
 TST14 012042
 TST15 012330
 TST16 012502
 TST17 012734
 TST2 010506

TST20 013046
 TST21 013420
 TST22 013614
 TST23 014074
 TST24 014206
 TST25 015114
 TST26 015450
 TST27 015602
 TST3 010604
 TST30 016516
 TST31 016774
 TST32 017324
 TST4 010674
 TST5 010764
 TST6 011054
 TST7 011200
 TYPDS = 104410
 TYPE = 104400
 TYPOC = 104402
 TYPON = 104406
 TYPOS = 104404
 TYPVAD 002244
 UBADDR 002352
 UBCOUN 004506
 UBMAP 015372
 UDPAR0= 177660
 UDPAR1= 177662
 UDPAR2= 177664
 UDPAR3= 177666
 UDPAR4= 177670
 UDPAR5= 177672
 UDPAR6= 177674
 UDPAR7= 177676
 UDPDR0= 177620
 UDPDR1= 177622
 UDPDR2= 177624
 UDPDR3= 177626
 UDPDR4= 177630
 UDPDR5= 177632
 UDPDR6= 177634
 UDPDR7= 177636
 UIPAR0= 177640
 UIPAR1= 177642
 UIPAR2= 177644
 UIPAR3= 177646
 UIPAR4= 177650
 UIPAR5= 177652
 UIPAR6= 177654
 UIPAR7= 177656
 UIPDR0= 177600
 UIPDR1= 177602
 UIPDR2= 177604
 UIPDR3= 177606
 UIPDR4= 177610
 UIPDR5= 177612
 UIPDR6= 177614
 UIPDR7= 177616

| | | | | | | | | | |
|---------|---------|---------|--------|---------|--------|----------|--------|----------|--------|
| USESTK= | 000600 | \$DDW3 | 020106 | \$GET42 | 021444 | \$OCTVL | 004036 | \$TKS | 001134 |
| USP = | 0000006 | \$DDW4 | 020110 | \$HD = | 000000 | \$OMODE | 003244 | \$TMP0 | 001172 |
| WRITEE | 004256 | \$DDW5 | 020112 | \$HIBTS | 020000 | \$OVER | 020452 | \$TMP1 | 001174 |
| \$APTHD | 020000 | \$DDW6 | 020114 | \$ICNT | 001102 | \$PASS | 020022 | \$TMP2 | 001176 |
| \$ATYC | 021030 | \$DDW7 | 020116 | \$ILLUP | 003662 | \$PASTM | 020006 | \$TMP3 | 001200 |
| \$ATY1 | 021004 | \$DDW8 | 020120 | \$ITEMB | 001112 | \$PWAD | 003656 | \$TMP4 | 001202 |
| \$ATY3 | 021012 | \$DDW9 | 020122 | \$LF | 001220 | \$PWADN | 003534 | \$TMP5 | 001204 |
| \$ATY4 | 021022 | \$DEVCT | 020024 | \$LFLG | 021247 | \$PWARMG | 003652 | \$TN = | 000033 |
| \$BASE | 020070 | \$DEVN | 020072 | \$LOOP | 021542 | \$PWRUP | 003602 | \$TPB | 001142 |
| \$BDADR | 001120 | \$DOAGN | 021502 | \$LPADR | 001104 | \$QUES | 001216 | \$TPFLG | 001147 |
| \$BDAT | 001124 | \$DTBL | 003452 | \$LPERR | 001106 | \$REGAD | 001154 | \$TPS | 001140 |
| \$BELL | 001212 | \$ENDAD | 021472 | \$MADR1 | 020046 | \$REG0 | 001156 | \$TRAP | 003472 |
| \$CDW1 | 020074 | \$ENDCT | 021320 | \$MADR2 | 020052 | \$REG1 | 001160 | \$TRP = | 000022 |
| \$CDW2 | 020076 | \$ENULL | 021550 | \$MADR3 | 020056 | \$REG2 | 001162 | \$TRPAD | 003512 |
| \$CHARC | 003014 | \$ENV | 020034 | \$MADR4 | 020062 | \$REG3 | 001164 | \$STSM | 020004 |
| \$CLR.T | 021462 | \$ENVN | 020035 | \$MAIL | 020014 | \$REG4 | 001166 | \$STSTM | 001100 |
| \$CMTAG | 001100 | \$EOP | 021252 | \$MAMS1 | 020044 | \$REG5 | 001170 | \$STPDS | 003246 |
| \$CM1 = | 000006 | \$EOPCT | 021312 | \$MAMS2 | 020050 | \$RESRE | 002466 | \$STPE | 002524 |
| \$CM2 = | 000014 | \$ERFLG | 001101 | \$MAMS3 | 020054 | \$RTRN | 021540 | \$STPEC | 002700 |
| \$CM3 = | 000006 | \$ERMAX | 001113 | \$MAMS4 | 020060 | \$SAVRE | 002430 | \$STPEX | 003016 |
| \$CM4 = | 000006 | \$ERROR | 020470 | \$MBADR | 020002 | \$SAVR6 | 003666 | \$STPOC | 003044 |
| \$CPUOP | 020042 | \$ERRPC | 001114 | \$MFLG | 021246 | \$SCOPE | 020140 | \$STPON | 003060 |
| \$CRLF | 001217 | \$ERRTB | 001326 | \$MSGAD | 020030 | \$SETUP= | 000037 | \$STPOS | 003020 |
| \$DBLK | 003462 | \$ERTTL | 001110 | \$MSGLG | 020032 | \$STUP = | 177777 | \$SUNIT | 020026 |
| \$DB20 | 003734 | \$ESCAP | 001210 | \$MSGTY | 020014 | \$SVLAD | 020416 | \$SUNITM | 020010 |
| \$DDW0 | 020100 | \$ETABL | 020034 | \$MTYP1 | 020045 | \$SWPC = | 000204 | \$SUSWR | 020040 |
| \$DDW1 | 020102 | \$ETEND | 020140 | \$MTYP2 | 020051 | \$SWR - | 177400 | \$VECT1 | 020064 |
| \$DDW10 | 020124 | \$FATAL | 020016 | \$MTYP3 | 020055 | \$SWREG | 020036 | \$VECT2 | 020066 |
| \$DDW11 | 020126 | \$FFLG | 021250 | \$MTYP4 | 020061 | \$SWRMK= | 000300 | \$XTSTR | 020214 |
| \$DDW12 | 020130 | \$FILLC | 001146 | \$MXCNT | 020466 | \$TBIT | 021546 | \$GET4= | 000001 |
| \$DDW13 | 020132 | \$FILLS | 001145 | \$NULL | 001144 | \$TESTN | 020020 | \$STRP = | 000002 |
| \$DDW14 | 020134 | \$GDADR | 001116 | \$NWTST | 000001 | \$TIMES | 001206 | \$OFILL | 003243 |
| \$DDW15 | 020136 | \$GDDAT | 001122 | \$OCNT | 003242 | \$TK9 | 001136 | \$.X | 020000 |
| \$DDW2 | 020104 | | | | | | | | |

. ABS. 033466 000
 000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 64784 WORDS (254 PAGES)

DYNAMIC MEMORY: 20434 WORDS (78 PAGES)

ELAPSED TIME: 00:08:47

CKKUA0,CKKUA0/NL:TOC/CRF/-SP-CKKUA0.SML,CKKUA0.P11

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|--------|--|
| ABASE | = | 000000 | 62-2195 62-2195 |
| ACDW1 | = | 000000 | 62-2195 62-2195 |
| ACDW2 | = | 000000 | 62-2195 62-2195 |
| ACPUOP | = | 000000 | 62-2195 62-2195 |
| ADDROR | | 001230 | #52-226 *57-742 *57-766 *57-789 *57-811 *59-1591 *62-2196 64-2317 64-2318 64-2321 64-2322 64-2332 |
| ADDW0 | = | 000000 | 62-2195 62-2195 |
| ADDW1 | = | 000000 | 62-2195 62-2195 |
| ADDW10 | = | 000000 | 62-2195 62-2195 |
| ADDW11 | = | 000000 | 62-2195 62-2195 |
| ADDW12 | = | 000000 | 62-2195 62-2195 |
| ADDW13 | = | 000000 | 62-2195 62-2195 |
| ADDW14 | = | 000000 | 62-2195 62-2195 |
| ADDW15 | = | 000000 | 62-2195 62-2195 |
| ADDW2 | = | 000000 | 62-2195 62-2195 |
| ADDW3 | = | 000000 | 62-2195 62-2195 |
| ADDW4 | = | 000000 | 62-2195 62-2195 |
| ADDW5 | = | 000000 | 62-2195 62-2195 |
| ADDW6 | = | 000000 | 62-2195 62-2195 |
| ADDW7 | = | 000000 | 62-2195 62-2195 |
| ADDW8 | = | 000000 | 62-2195 62-2195 |
| ADDW9 | = | 000000 | 62-2195 62-2195 |
| ADEVCT | = | 000000 | 62-2195 62-2195 |
| ADEVM | = | 000000 | 62-2195 62-2195 |
| ADRAND | | 001226 | #52-226 *57-743 *57-767 *57-790 *57-812 *59-1589 *62-2196 64-2317 64-2318 64-2321 64-2322 64-2332 |
| AENV | = | 000000 | 62-2195 62-2195 |
| AENVM | = | 000000 | 62-2195 62-2195 |
| AFATAL | = | 000000 | 62-2195 62-2195 |
| AMADR1 | = | 000000 | 62-2195 62-2195 |
| AMADR2 | = | 000000 | 62-2195 62-2195 |
| AMADR3 | = | 000000 | 62-2195 62-2195 |
| AMADR4 | = | 000000 | 62-2195 62-2195 |
| AMAMS1 | = | 000000 | 62-2195 62-2195 |
| AMAMS2 | = | 000000 | 62-2195 62-2195 |
| AMAMS3 | = | 000000 | 62-2195 62-2195 |
| AMAMS4 | = | 000000 | 62-2195 62-2195 |
| AMSGAD | = | 000000 | 62-2195 62-2195 |
| AMSGLG | = | 000000 | 62-2195 62-2195 |
| AMSGTY | = | 000000 | 62-2195 62-2195 |
| AMTYP1 | = | 000000 | 62-2195 62-2195 |
| AMTYP2 | = | 000000 | 62-2195 62-2195 |
| AMTYP3 | = | 000000 | 62-2195 62-2195 |
| AMTYP4 | = | 000000 | 62-2195 62-2195 |
| APASS | = | 000000 | 62-2195 62-2195 |
| APRIOR | = | 000000 | 62-2195 |
| APTCSU | = | 000040 | #62-2198 |
| APTENV | = | 000001 | 62-2198 #62-2198 |
| APTSIZ | = | 000200 | #62-2198 |
| APTSPO | = | 000100 | 62-2198 #62-2198 |
| ASWREG | = | 000000 | 62-2195 62-2195 |
| ATESTN | = | 000000 | 62-2195 62-2195 |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|--------|--|
| AUNIT | = | 000000 | 62-2195 62-2195 |
| AUSWR | = | 000000 | 62-2195 62-2195 |
| AVECT1 | = | 000000 | 62-2195 62-2195 |
| AVECT2 | = | 000000 | 62-2195 62-2195 |
| BADPC | = | 001300 | #52-226 *57-877 *57-910 64-2313 64-2314 64-2316 |
| BIT0 | = | 000001 | #51-222 58-1293 |
| BIT00 | = | 000001 | #51-222 51-222 |
| BIT01 | = | 000002 | #51-222 51-222 |
| BIT02 | = | 000004 | #51-222 51-222 |
| BIT03 | = | 000010 | #51-222 51-222 |
| BIT04 | = | 000020 | #51-222 51-222 |
| BIT05 | = | 000040 | #51-222 51-222 |
| BIT06 | = | 000100 | #51-222 51-222 |
| BIT07 | = | 000200 | #51-222 51-222 |
| BIT08 | = | 000400 | #51-222 51-222 62-2196 |
| BIT09 | = | 001000 | #51-222 51-222 62-2196 62-2197 |
| BIT1 | = | 000002 | #51-222 |
| BIT10 | = | 002000 | #51-222 62-2197 |
| BIT11 | = | 004000 | #51-222 58-1313 62-2196 |
| BIT12 | = | 010000 | #51-222 62-2199 |
| BIT13 | = | 020000 | #51-222 62-2197 |
| BIT14 | = | 040000 | #51-222 |
| BIT15 | = | 100000 | #51-222 |
| BIT2 | = | 000004 | #51-222 |
| BIT3 | = | 000010 | #51-222 |
| BIT4 | = | 000020 | #51-222 56-672 58-1294 58-1307 |
| BIT5 | = | 000040 | #51-222 58-1440 58-1552 59-1614 59-1714 |
| BIT6 | = | 000100 | #51-222 |
| BIT7 | = | 000200 | #51-222 |
| BIT8 | = | 000400 | #51-222 |
| BIT9 | = | 001000 | #51-222 53-498 |
| BPTVEC | = | 000014 | #51-222 |
| CACHE | = | 000114 | #51-222 |
| CLRMAR | = | 004120 | #57-710 58-1188 58-134e 58-1505 |
| CNTR | = | 001260 | #52-226 64-2329 |
| CONTRL | = | 177746 | #51-222 |
| COUNT | = | 004416 | #57-810 58-1224 58-1237 58-1251 58-1264 |
| CPFLAG | = | 004564 | #57-865 *57-886 *57-947 *62-2197 |
| CPUER | = | 004562 | #57-864 57-941 57-989 |
| CPUERP | = | 177766 | #51-222 57-736 *57-737 57-876 *57-885 57-950 62-2197 |
| CPUEXP | = | 001266 | #52-226 57-738 57-878 57-380 *57-943 *58-1308 *58-1331 *58-1347 *58-1385 |
| | | | *58-1402 *58-1420 *58-1434 *58-1463 *59-1615 *59-1633 *59-1724 *59-1735 *59-1761 |
| | | | *59-1811 *59-1829 *59-1925 *59-1936 *59-1962 64-2313 64-2333 |
| CR | = | 000015 | #51-222 53-642 53-642 |
| CRLF | = | 000200 | #51-222 53-642 53-642 57-936 57-936 63-2216 63-2218 63-2220 63-2222 |
| | | | 63-2227 63-2229 63-2233 63-2234 63-2237 63-2238 63-2239 63-2240 63-2241 |
| | | | 63-2242 63-2250 63-2252 63-2254 63-2256 63-2270 63-2272 63-2274 63-2276 |
| | | | 63-2278 63-2280 63-2282 63-2285 63-2293 63-2295 63-2297 63-2301 63-2305 |
| | | | 63-2307 |
| DATA | = | 001324 | #52-226 59-1726 *59-1750 *59-1757 *59-1758 *59-1759 59-1927 *59-1951 *59-1958 |
| | | | *59-1959 *59-1960 |
| DATAND | = | 001232 | #52-226 *57-769 *57-792 *57-814 *57-839 *59-1590 *62-2196 64-2318 64-2321 |

| SYMBOL CROSS REFERENCE | | REFERENCES | | | | | | | | | |
|------------------------|--------|------------|----------|----------|----------|---------|----------|----------|---------|---------|--|
| SYMBOL | VALUE | 64-2322 | 64-2325 | 64-2332 | •57-813 | •57-838 | •59-1592 | •62-2196 | 64-2318 | 64-2321 | |
| DATAOR | 001234 | #52-226 | •57-768 | •57-791 | | | | | | | |
| DF 1 | 033266 | 64-2322 | 64-2325 | 64-2332 | | | | | | | |
| DF 10 | 033315 | 53-231 | #64-2345 | | | | | | | | |
| DF 12 | 033324 | 53-258 | 53-265 | #64-2350 | | | | | | | |
| DF 14 | 033330 | 53-272 | 53-279 | #64-2351 | | | | | | | |
| DF 15 | 033336 | 53-286 | #64-2352 | | | | | | | | |
| DF 17 | 033346 | 53-293 | 53-300 | #64-2353 | | | | | | | |
| DF 2 | 033272 | #64-2354 | | | | | | | | | |
| DF 201 | 033426 | 53-237 | #64-2346 | | | | | | | | |
| DF 202 | 033431 | 53-425 | #64-2366 | | | | | | | | |
| DF 203 | 033435 | 53-431 | #64-2367 | | | | | | | | |
| DF 204 | 033441 | 53-438 | #64-2368 | | | | | | | | |
| DF 205 | 033447 | 53-444 | #64-2369 | | | | | | | | |
| DF 206 | 033454 | 53-450 | #64-2370 | | | | | | | | |
| DF 207 | 033460 | 53-457 | #64-2371 | | | | | | | | |
| DF 23 | 033351 | #64-2372 | | | | | | | | | |
| DF 25 | 033357 | 53-321 | #64-2355 | | | | | | | | |
| DF 27 | 033363 | 53-307 | #64-2356 | | | | | | | | |
| DF 3 | 033275 | #64-2357 | | | | | | | | | |
| DF 30 | 033366 | #64-2347 | | | | | | | | | |
| DF 34 | 033370 | 53-314 | 53-327 | 53-335 | #64-2358 | | | | | | |
| DF 35 | 033374 | #64-2359 | | | | | | | | | |
| DF 36 | 033400 | 53-349 | #64-2360 | | | | | | | | |
| DF 37 | 033403 | 53-355 | #64-2361 | | | | | | | | |
| DF 40 | 033411 | 53-362 | 53-415 | #64-2362 | | | | | | | |
| DF 41 | 033415 | 53-369 | 53-391 | #64-2363 | | | | | | | |
| DF 42 | 033421 | 53-376 | 53-399 | #64-2364 | | | | | | | |
| DF 5 | 033303 | 53-383 | 53-407 | #64-2365 | | | | | | | |
| DF 6 | 033310 | 53-244 | #64-2348 | | | | | | | | |
| DH1 | 030034 | 53-251 | #64-2349 | | | | | | | | |
| DH10 | 030460 | 53-229 | #63-2268 | | | | | | | | |
| DH12 | 030610 | 53-255 | 53-262 | #63-2276 | | | | | | | |
| DH14 | 030666 | 53-269 | 53-276 | #63-2278 | | | | | | | |
| DH15 | 031005 | 53-283 | #63-2280 | | | | | | | | |
| DH17 | 031164 | 53-290 | 53-297 | #63-2282 | | | | | | | |
| DH2 | 030100 | #63-2284 | | | | | | | | | |
| DH201 | 032101 | 53-235 | #63-2269 | | | | | | | | |
| DH202 | 032131 | 53-423 | #63-2299 | | | | | | | | |
| DH203 | 032171 | 53-429 | #63-2300 | | | | | | | | |
| DH204 | 032250 | 53-435 | #63-2301 | | | | | | | | |
| DH205 | 032330 | 53-442 | #63-2303 | | | | | | | | |
| DH206 | 032402 | 53-448 | #63-2304 | | | | | | | | |
| DH207 | 032462 | 53-454 | #63-2305 | | | | | | | | |
| DH23 | 031214 | #63-2307 | | | | | | | | | |
| DH25 | 031333 | 53-318 | 53-359 | 53-412 | #63-2285 | | | | | | |
| DH27 | 031375 | 53-305 | #63-2287 | | | | | | | | |
| DH3 | 030134 | #63-2288 | | | | | | | | | |
| DH30 | 031427 | #63-2270 | | | | | | | | | |
| DH34 | 031447 | 53-312 | 53-325 | 53-333 | #63-2289 | | | | | | |
| DH35 | 031507 | #63-2290 | | | | | | | | | |
| | | 53-347 | #63-2291 | | | | | | | | |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|--------|---|
| DM36 | | 031547 | 53-353 #63-2292 |
| DM40 | | 031641 | 53-366 53-388 #63-2293 |
| DM41 | | 031721 | 53-373 53-396 #63-2295 |
| DM42 | | 032001 | 53-380 53-404 #63-2297 |
| DM5 | | 030264 | 53-241 #63-2272 |
| DM6 | | 030370 | 53-248 #63-2274 |
| DISPLA | | 001152 | #52-226 58-1306 58-1433 59-1809 62-2196 62-2197 |
| DT1 | | 032562 | 53-230 #64-2313 |
| DT10 | | 032646 | 53-257 53-264 #64-2318 |
| DT12 | | 032666 | 53-271 #64-2319 |
| DT13 | | 032700 | 53-278 #64-2320 |
| DT14 | | 032712 | 53-285 #64-2321 |
| DT15 | | 032730 | 53-292 53-299 #64-2322 |
| DT17 | | 032752 | #64-2323 |
| DT2 | | 032574 | 53-236 #64-2314 |
| DT20 | | 032762 | #64-2324 |
| DT201 | | 033152 | 53-424 #64-2336 |
| DT202 | | 033162 | 53-430 #64-2337 |
| DT203 | | 033174 | 53-437 #64-2338 |
| DT204 | | 033206 | 53-443 #64-2339 |
| DT205 | | 033224 | 53-449 #64-2340 |
| DT206 | | 033240 | 53-456 #64-2341 |
| DT207 | | 033252 | #64-2342 |
| DT23 | | 032772 | 53-320 #64-2325 |
| DT25 | | 033010 | 53-306 #64-2326 |
| DT27 | | 033022 | #64-2327 |
| DT30 | | 033032 | 53-313 53-326 53-334 #64-2328 |
| DT34 | | 033040 | #64-2329 |
| DT35 | | 033052 | 53-348 #64-2330 |
| DT36 | | 033064 | 53-354 #64-2331 |
| DT37 | | 033074 | 53-361 53-414 #64-2332 |
| DT4 | | 032604 | #64-2315 |
| DT40 | | 033112 | 53-368 53-390 #64-2333 |
| DT41 | | 033124 | 53-375 53-398 #64-2334 |
| DT42 | | 033136 | 53-382 53-406 #64-2335 |
| DT5 | | 032616 | 53-243 #64-2316 |
| DT6 | | 032632 | 53-250 #64-2317 |
| DLALAD | | 004342 | #57-787 58-1196 58-1521 |
| EMTVEC | | 000030 | #51-222 57-935 57-935 |
| EM1 | | 021554 | 53-228 #63-2202 |
| EM10 | | 022365 | 53-254 #63-2209 |
| EM11 | | 022456 | 53-261 #63-2210 |
| EM12 | | 022550 | 53-268 #63-2211 |
| EM13 | | 022636 | 53-275 #63-2212 |
| EM14 | | 022730 | 53-282 #63-2213 |
| EM15 | | 023023 | 53-289 #63-2214 |
| EM16 | | 023127 | 53-296 #63-2215 |
| EM17 | | 023232 | #63-2216 |
| EM2 | | 021636 | 53-234 #63-2203 |
| EM20 | | 023354 | #63-2218 |
| EM201 | | 027327 | 53-422 #63-2260 |
| EM202 | | 027403 | 53-428 #63-2261 |

SYMBOL CROSS REFERENCE

| SYMBOL | VALUE | REFERENCES | | | | | | | | |
|--------|----------|------------|----------|----------|----------|---------|---------|---------|---------|---------|
| EM203 | 027456 | 53-434 | #63-2262 | | | | | | | |
| EM204 | 027551 | 53-441 | #63-2263 | | | | | | | |
| EM205 | 027631 | 53-447 | #63-2264 | | | | | | | |
| EM206 | 027700 | 53-453 | #63-2265 | | | | | | | |
| EM207 | 027762 | #63-2266 | | | | | | | | |
| EM21 | 023502 | #63-2220 | | | | | | | | |
| EM22 | 023646 | #63-2222 | | | | | | | | |
| EM23 | 024012 | #63-2224 | | | | | | | | |
| EM24 | 024106 | #63-2225 | | | | | | | | |
| EM25 | 024206 | 53-303 | #63-2226 | | | | | | | |
| EM26 | 024313 | #63-2227 | | | | | | | | |
| EM3 | 021705 | #63-2204 | | | | | | | | |
| EM30 | 024434 | 53-310 | #63-2229 | | | | | | | |
| EM31 | 024610 | 53-317 | #63-2231 | | | | | | | |
| EM32 | 024702 | 53-324 | #63-2232 | | | | | | | |
| EM33 | 024754 | 53-330 | #63-2233 | | | | | | | |
| EM35 | 025166 | 53-343 | #63-2237 | | | | | | | |
| EM36 | 025736 | 53-352 | #63-2244 | | | | | | | |
| EM37 | 026033 | 53-358 | #63-2245 | | | | | | | |
| EM4 | 022006 | #63-2205 | | | | | | | | |
| EM40 | 026136 | 53-365 | #63-2246 | | | | | | | |
| EM41 | 026231 | 53-372 | #63-2247 | | | | | | | |
| EM42 | 026313 | 53-379 | #63-2248 | | | | | | | |
| EM43 | 026406 | #63-2249 | | | | | | | | |
| EM45 | 026473 | 53-386 | #63-2250 | | | | | | | |
| EM46 | 026626 | 53-394 | #63-2252 | | | | | | | |
| EM47 | 026750 | 53-402 | #63-2254 | | | | | | | |
| EM5 | 022115 | 53-240 | #63-2206 | | | | | | | |
| EM50 | 027103 | #63-2256 | | | | | | | | |
| EM52 | 027270 | 53-411 | #63-2259 | | | | | | | |
| EM6 | 022223 | 53-247 | #63-2207 | | | | | | | |
| EM7 | 022303 | #63-2208 | | | | | | | | |
| EREXIT | 021002 | 62-2197 | #62-2197 | | | | | | | |
| ERRCN1 | 001256 | #52-226 | 57-744 | 57-770 | 57-793 | 57-817 | 57-840 | 57-973 | 57-1002 | 58-1033 |
| | | 58-1063 | 58-1093 | 58-1205 | 58-1224 | 58-1237 | 58-1251 | 58-1264 | 58-1380 | 58-1536 |
| | | 59-1593 | 59-1599 | *62-2196 | *62-2197 | 64-2317 | 64-2318 | 64-2321 | 64-2322 | 64-2325 |
| | | 64-2329 | 64-2332 | | | | | | | |
| ERROR | = 104000 | #51-222 | | | | | | | | |
| ERRVEC | = 000004 | #51-222 | *57-941 | *57-942 | *57-963 | *57-989 | 62-2196 | 62-2196 | 62-2196 | 62-2196 |
| ERTYPE | 001736 | #53-488 | 62-2197 | | | | | | | |
| ER200 | 001656 | #53-417 | 53-508 | 53-511 | | | | | | |
| FLAG | 001262 | #52-226 | | | | | | | | |
| GNS | = ***** | 51-224 | 51-224 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 |
| | | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-646 | 53-646 |
| | | 53-647 | 53-647 | 57-936 | 62-2199 | 62-2199 | | | | |
| HIADRS | 177742 | #51-222 | | | | | | | | |
| HIGEST | 001244 | #52-226 | *58-1455 | 58-1468 | 58-1475 | 58-1525 | 58-1534 | 64-2330 | | |
| HITMIS | = 177752 | #51-222 | | | | | | | | |
| HREGL | 001252 | #52-226 | *58-1486 | | | | | | | |
| HREGU | 001254 | #52-226 | *58-1488 | | | | | | | |
| HT | = 000011 | #51-222 | 53-642 | 53-642 | | | | | | |
| :OTVE | = 000020 | #51-222 | 57-935 | 57-935 | | | | | | |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|---------|--|
| KDPAR0 | = | 172360 | #51-222 |
| KDPAR1 | = | 172362 | #51-222 |
| KDPAR2 | = | 172364 | #51-222 |
| KDPAR3 | = | 172366 | #51-222 |
| KDPAR4 | = | 172370 | #51-222 |
| KDPAR5 | = | 172372 | #51-222 |
| KDPAR6 | = | 172374 | #51-222 |
| KDPAR7 | = | 172376 | #51-222 |
| KDPDR0 | = | 172320 | #51-222 |
| KDPDR1 | = | 172322 | #51-222 |
| KDPDR2 | = | 172324 | #51-222 |
| KDPDR3 | = | 172326 | #51-222 |
| KDPDR4 | = | 172330 | #51-222 |
| KDPDR5 | = | 172332 | #51-222 |
| KDPDR6 | = | 172334 | #51-222 |
| KDPDR7 | = | 172336 | #51-222 |
| KERSTK | = | 001100 | #51-222 |
| KIPAR0 | = | 172340 | #51-222 57-874 |
| KIPAR1 | = | 172342 | #51-222 53-600 *58-1285 *59-1795 |
| KIPAR2 | = | 172344 | #51-222 *58-1286 *59-1793 |
| KIPAR3 | = | 172346 | #51-222 *58-1287 |
| KIPAR4 | = | 172350 | #51-222 *58-1288 |
| | | | #51-222 *58-1289 *58-1442 *58-1449 58-1450 58-1454 58-1455 *58-1456 58-1457 |
| | | | *58-1515 *58-1553 *58-1560 58-1561 *59-1616 *59-1721 59-1749 *59-1751 *59-1812 |
| | | | *59-1922 59-1950 *59-1952 |
| KIPAR5 | = | 172352 | #51-222 *58-1290 *58-1360 *58-1361 |
| KIPAR6 | = | 172354 | #51-222 *58-1291 *58-1310 *58-1315 *58-1326 *58-1349 *58-1355 58-1356 58-1360 |
| | | | 58-1404 58-1568 58-1571 *59-1584 59-1585 59-1651 59-1656 59-1661 59-1666 |
| | | | 59-1671 59-1676 59-1681 59-1686 59-1691 59-1696 59-1701 *59-1722 59-1852 |
| | | | 59-1857 59-1862 59-1867 59-1872 59-1877 59-1882 59-1887 59-1892 59-1897 |
| | | | 59-1902 *59-1923 |
| KIPAR7 | = | 172356 | #51-222 *58-1292 |
| KIPDR0 | = | 172300 | #51-222 *58-1277 |
| KIPDR1 | = | 172302 | #51-222 *58-1278 |
| KIPDR2 | = | 172304 | #51-222 *58-1279 |
| KIPDR3 | = | 172306 | #51-222 *58-1280 |
| KIPDR4 | = | 172310 | #51-222 *58-1281 |
| KIPDR5 | = | 172312 | #51-222 *58-1282 |
| KIPDR6 | = | 172314 | #51-222 *58-1283 |
| KIPDR7 | = | 172316 | #51-222 *58-1284 |
| KSP | = | 0000006 | #51-222 *53-488 *53-492 53-504 *53-532 *53-539 *53-545 *53-551 53-553 |
| | | | *53-554 *53-562 *53-567 *53-575 *53-576 53-593 *53-608 53-610 *53-611 |
| | | | *53-616 53-616 53-625 *53-630 53-632 *53-633 *53-637 53-637 56-674 |
| | | | 56-676 56-677 56-693 *57-734 *57-735 *57-750 *57-751 *57-765 *57-788 |
| | | | *57-810 *57-835 *57-872 *57-873 *57-874 *57-887 *57-888 57-910 *57-911 |
| | | | *57-912 *57-919 *57-920 |
| LF | = | 000012 | #51-222 53-642 53-642 |
| LOADRS | = | 177740 | #51-222 |
| LOOP | = | 010332 | #57-937 62-2199 |
| LOWEST | = | 001242 | #52-226 *58-1454 58-1466 58-1476 58-1510 58-1511 59-1616 59-1651 59-1656 |
| | | | 59-1661 59-1666 59-1671 59-1676 59-1681 59-1686 59-1691 59-1696 59-1701 |
| | | | 59-1722 59-1791 59-1812 59-1852 59-1857 59-1862 59-1867 59-1872 59-1877 |
| | | | 59-1882 59-1887 59-1892 59-1897 59-1902 59-1923 64-2330 |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|--------|---|
| LREG1 | | 001246 | #52-226 *58-1483 58-1509 59-1618 59-1651 59-1656 59-1661 59-1666 59-1671 59-1676 59-1681 59-1686 59-1691 59-1696 59-1701 59-1718 59-1752 59-1814 59-1852 59-1857 59-1862 59-1867 59-1872 59-1877 59-1882 59-1887 59-1892 59-1897 59-1902 59-1919 59-1953 |
| LREG2 | | 001250 | #52-226 *58-1485 59-1617 59-1651 59-1656 59-1661 59-1666 59-1671 59-1676 59-1681 59-1686 59-1691 59-1696 59-1701 59-1717 59-1754 59-1755 59-1785 59-1813 59-1852 59-1857 59-1862 59-1867 59-1872 59-1877 59-1882 59-1887 59-1892 59-1897 59-1902 59-1918 59-1955 59-1956 |
| MAINT | = | 177750 | #51-222 |
| MAPH0 | = | 170202 | #51-222 58-1021 58-1150 |
| MAPH00 | = | 170202 | #51-222 51-222 58-1151 58-1237 |
| MAPH01 | = | 170206 | #51-222 51-222 |
| MAPH02 | = | 170212 | #51-222 51-222 |
| MAPH03 | = | 170216 | #51-222 51-222 |
| MAPH04 | = | 170222 | #51-222 51-222 |
| MAPH05 | = | 170226 | #51-222 51-222 |
| MAPH06 | = | 170232 | #51-222 51-222 |
| MAPH07 | = | 170236 | #51-222 51-222 |
| MAPH1 | = | 170206 | #51-222 |
| MAPH10 | = | 170242 | #51-222 |
| MAPH11 | = | 170246 | #51-222 |
| MAPH12 | = | 170252 | #51-222 |
| MAPH13 | = | 170256 | #51-222 |
| MAPH14 | = | 170262 | #51-222 |
| MAPH15 | = | 170266 | #51-222 |
| MAPH16 | = | 170272 | #51-222 |
| MAPH17 | = | 170276 | #51-222 58-1029 58-1237 |
| MAPH2 | = | 170212 | #51-222 |
| MAPH20 | = | 170302 | #51-222 58-1081 58-1162 58-1163 58-1264 |
| MAPH21 | = | 170306 | #51-222 |
| MAPH22 | = | 170312 | #51-222 |
| MAPH23 | = | 170316 | #51-222 |
| MAPH24 | = | 170320 | #51-222 |
| MAPH25 | = | 170326 | #51-222 |
| MAPH26 | = | 170332 | #51-222 |
| MAPH27 | = | 170336 | #51-222 |
| MAPH3 | = | 170216 | #51-222 |
| MAPH30 | = | 170342 | #51-222 |
| MAPH31 | = | 170346 | #51-222 |
| MAPH32 | = | 170352 | #51-222 |
| MAPH33 | = | 170356 | #51-222 |
| MAPH34 | = | 170362 | #51-222 |
| MAPH35 | = | 170366 | #51-222 |
| MAPH36 | = | 170372 | #51-222 |
| MAPH37 | = | 170376 | #51-222 57-712 57-969 58-1089 58-1187 58-1201 58-1264 |
| MAPH4 | = | 170222 | #51-222 |
| MAPH5 | = | 170226 | #51-222 |
| MAPH6 | = | 170232 | #51-222 |
| MAPH7 | = | 170236 | #51-222 |
| MAPL0 | = | 170200 | #51-222 57-710 57-964 57-990 58-1185 58-1198 58-1312 58-1321 58-1435 |
| MAPL00 | = | 170200 | #51-222 51-222 *58-1113 58-1114 58-1224 |
| MAPL01 | = | 170204 | #51-222 51-222 |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|--------|--|
| MAPL02 | = | 170210 | #51-222 51-222 |
| MAPL03 | = | 170214 | #51-222 51-222 |
| MAPL04 | = | 170220 | #51-222 51-222 |
| MAPL05 | = | 170224 | #51-222 51-222 |
| MAPL06 | = | 170230 | #51-222 51-222 |
| MAPL07 | = | 170234 | #51-222 51-222 |
| MAPL1 | = | 170204 | #51-222 |
| MAPL10 | = | 170240 | #51-222 |
| MAPL11 | = | 170244 | #51-222 |
| MAPL12 | = | 170250 | #51-222 |
| MAPL13 | = | 170254 | #51-222 |
| MAPL14 | = | 170260 | #51-222 |
| MAPL15 | = | 170264 | #51-222 |
| MAPL16 | = | 170270 | #51-222 |
| MAPL17 | = | 170274 | #51-222 57-998 58-1224 |
| MAPL2 | = | 170210 | #51-222 |
| MAPL20 | = | 170300 | #51-222 58-1051 *58-1125 58-1126 58-1251 |
| MAPL21 | = | 170304 | #51-222 |
| MAPL22 | = | 170310 | #51-222 |
| MAPL23 | = | 170314 | #51-222 |
| MAPL24 | = | 170320 | #51-222 |
| MAPL25 | = | 170324 | #51-222 |
| MAPL26 | = | 170330 | #51-222 |
| MAPL27 | = | 170334 | #51-222 |
| MAPL3 | = | 170214 | #51-222 |
| MAPL30 | = | 170340 | #51-222 |
| MAPL31 | = | 170344 | #51-222 |
| MAPL32 | = | 170350 | #51-222 |
| MAPL33 | = | 170354 | #51-222 |
| MAPL34 | = | 170360 | #51-222 |
| MAPL35 | = | 170364 | #51-222 |
| MAPL36 | = | 170370 | #51-222 |
| MAPL37 | = | 170374 | #51-222 58-1059 58-1251 58-1438 |
| MAPL4 | = | 170220 | #51-222 |
| MAPL5 | = | 170224 | #51-222 |
| MAPL6 | = | 170230 | #51-222 |
| MAPL7 | = | 170234 | #51-222 |
| MEMERR | = | 177744 | #51-222 |
| MMFLAG | = | 004676 | #57-903 *57-918 *57-949 *62-2197 |
| MMRO | = | 177572 | #51-222 51-222 57-913 57-917 57-944 *58-1293 62-2197 *62-2199 |
| MMR1 | = | 177574 | #51-222 51-222 57-914 |
| MMR2 | = | 177576 | #51-222 51-222 57-915 |
| MMR3 | = | 172516 | #51-222 51-222 57-945 *58-1294 58-1440 *58-1552 *59-1614 *62-2199 |
| MMTRAP | = | 004674 | #57-902 57-938 |
| MMVEC | = | 000250 | #51-222 *57-938 *57-940 |
| NEXMEM | = | 000040 | #59-1714 59-1735 59-1737 59-1936 59-1938 |
| NKTTST | = | 001322 | *52-226 *57-962 *57-988 *58-1020 *58-1050 *58-1080 *58-1108 *58-1145 *58-1184 |
| | | | *58-1223 *58-1236 *58-1250 *58-1263 *58-1345 *58-1401 *58-1432 *58-1504 *58-1551 |
| | | | *59-1613 *59-1646 *59-1713 *59-1810 *59-1845 *59-1916 *59-1990 *60-2097 |
| OLDPC | = | 001302 | #52-226 *57-734 57-751 *57-765 57-775 *57-788 57-798 *57-810 57-822 |
| | | | *57-835 57-845 *57-872 57-877 57-888 *57-911 57-920 |
| OLDPS | = | 001304 | #52-226 *57-735 57-750 *57-873 57-887 *57-912 57-919 |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|---------|---|
| OLDPSW | | 001306 | #52-226 *56-676 56-693 *57-695 |
| PADRSW | | 001224 | #52-226 *53-606 *53-629 |
| PADRSL | | 001222 | #52-226 *53-607 53-608 *53-628 53-630 |
| PATAND | | 001236 | #52-226 *57-816 *57-837 *62-2196 64-2322 64-2325 |
| PATTOR | | 001240 | #52-226 *57-815 *57-836 *62-2196 64-2322 64-2325 |
| PCONTR | | 001274 | #52-226 |
| PCPUFR | | 001270 | #52-226 *57-736 57-737 57-738 *57-876 57-880 57-885 *58-1350 58-1353 *59-1620 59-1629 *59-1725 59-1733 59-1737 *59-1816 59-1825 *59-1926 59-1934 59-1938 64-2313 64-2314 64-2333 |
| PIRO | = | 177772 | #51-222 |
| PIRQVE | = | 000240 | #51-222 |
| PMANT | | 001276 | #52-226 |
| PPPRO | | 001310 | #52-226 *57-913 64-2316 |
| PPPR1 | | 001312 | #52-226 *57-914 64-2316 |
| PPPR2 | | 001314 | #52-226 *57-915 64-2316 |
| PPARER | | 001272 | #52-226 |
| PRO | = | 000000 | #51-222 |
| PR1 | = | 000040 | #51-222 |
| PR2 | = | 000100 | #51-222 |
| PR3 | = | 000140 | #51-222 |
| PR4 | = | 000200 | #51-222 |
| PR5 | = | 000240 | #51-222 |
| PR6 | = | 000300 | #51-222 |
| PR7 | | 000340 | #51-222 |
| PS | = | 177776 | #51-222 51-222 57-935 62-2199 |
| PSW | = | 177776 | #51-222 |
| PURMSG | | 003670 | 53-649 #53-650 |
| PURVEC | = | 000024 | #51-222 53-649 53-649 53-649 53-649 53-649 57-935 57-935 |
| RELC22 | | 012166 | 58-1264 #58-1273 |
| RESREG | = | 104414 | 53-615 53-636 #53-645 53-653 |
| RESVEC | = | 000010 | #51-222 57-935 57-935 57-935 |
| RETRY | | 001320 | #52-226 *62-2196 |
| RSIZE | | 001316 | #52-226 *59-1749 *59-1950 |
| R10 | = | %000000 | #51-222 |
| R11 | = | %000001 | #51-222 |
| R12 | = | %000002 | #51-222 |
| R13 | = | %000003 | #51-222 |
| R14 | = | %000004 | #51-222 |
| R15 | = | %000005 | #51-222 |
| R6 | = | %000006 | #51-222 51-222 *57-935 *57-935 57-935 *60-2042 *60-2042 *60-2156 *60-2156 |
| R7 | = | %000007 | #51-222 51-222 |
| SAVREG | = | 104412 | 53-592 53-624 #53-645 53-653 |
| SCOPE | = | 000004 | #51-222 57-962 57-988 58-1020 58-1050 58-1080 58-1108 58-1145 58-1184 58-1223 58-1236 58-1250 58-1263 58-1345 58-1401 58-1432 58-1504 58-1551 59-1613 59-1646 59-1713 59-1772 59-1845 59-1916 59-1990 60-2096 62-2199 |
| SDPAR0 | = | 172260 | #51-222 |
| SDPAR1 | = | 172262 | #51-222 |
| SDPAR2 | = | 172264 | #51-222 |
| SDPAR3 | = | 172266 | #51-222 |
| SDPAR4 | = | 172270 | #51-222 |
| SDPAR5 | = | 172272 | #51-222 |
| SDPAR6 | = | 172274 | #51-222 |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|---------|---|
| SDPAR7 | = | 172276 | #51-222 |
| SDPDR0 | = | 172220 | #51-222 |
| SDPDR1 | = | 172222 | #51-222 |
| SDPDR2 | = | 172224 | #51-222 |
| SDPDR3 | = | 172226 | #51-222 |
| SDPDR4 | = | 172230 | #51-222 |
| SDPDR5 | = | 172232 | #51-222 |
| SDPDR6 | = | 172234 | #51-222 |
| SDPDR7 | = | 172236 | #51-222 |
| SIPAR0 | = | 172240 | #51-222 |
| SIPAR1 | = | 172242 | #51-222 |
| SIPAR2 | = | 172244 | #51-222 |
| SIPAR3 | = | 172246 | #51-222 |
| SIPAR4 | = | 172250 | #51-222 |
| SIPAR5 | = | 172252 | #51-222 |
| SIPAR6 | = | 172254 | #51-222 |
| SIPAR7 | = | 172256 | #51-222 |
| SIPDR0 | = | 172200 | #51-222 |
| SIPDR1 | = | 172202 | #51-222 |
| SIPDR2 | = | 172204 | #51-222 |
| SIPDR3 | = | 172206 | #51-222 |
| SIPDR4 | = | 172210 | #51-222 |
| SIPDR5 | = | 172212 | #51-222 |
| SIPDR6 | = | 172214 | #51-222 |
| SIPDR7 | = | 172216 | #51-222 |
| SIZEHI | = | 177762 | #51-222 |
| SIZEJ | = | 013064 | 58-1359 #58-1433 |
| SIZELO | = | 177760 | #51-222 |
| SRO | = | 177572 | #51-222 |
| SR1 | = | 177574 | #51-222 |
| SR2 | = | 177576 | #51-222 |
| SR3 | = | 172516 | #51-222 |
| SSP | = | X000006 | #51-222 |
| STACK | = | 001100 | #51-222 51-222 51-222 51-222 57-935 |
| START | = | 010000 | 51-224 53-649 #57-930 58-1453 |
| STKLMT | = | 177774 | #51-222 |
| SUPSTK | = | 000700 | #51-222 |
| SWR | = | 001150 | #52-226 53-501 *57-932 62-2196 62-2196 62-2196 62-2196 62-2196 62-2197 62-2197 62-2197 62-2197 62-2197 62-2199 |
| SW0 | = | 000001 | #51-222 |
| SW00 | = | 000001 | #51-222 51-222 |
| SW01 | = | 000002 | #51-222 51-222 |
| SW02 | = | 000004 | #51-222 51-222 |
| SW03 | = | 000010 | #51-222 51-222 |
| SW04 | = | 000020 | #51-222 51-222 |
| SW05 | = | 000040 | #51-222 51-222 |
| SW06 | = | 000100 | #51-222 51-222 |
| SW07 | = | 000200 | #51-222 51-222 |
| SW08 | = | 000400 | #51-222 51-222 |
| SW09 | = | 001000 | #51-222 51-222 |
| SW1 | = | 000002 | #51-222 |
| SW10 | = | 002000 | #51-222 |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES | | | | | | | |
|--------|-----------------|---------|------------|----------|----------|---------|---------|---------|---------|---------|
| TST32 | | 017324 | 59-1990 | #6J-2096 | | | | | | |
| TST4 | | 010674 | 58-1020 | 58-1034 | #58-1050 | | | | | |
| TST5 | | 010764 | 58-1050 | 58-1064 | #58-1080 | | | | | |
| TST6 | | 011054 | 58-1080 | 58-1094 | #58-1108 | | | | | |
| TST7 | | 011200 | 58-1108 | #58-1145 | | | | | | |
| TYPDS | = | 104410 | 53-546 | #53-645 | 62-2199 | 62-2199 | | | | |
| TYPE | = | 104400 | 53-518 | 53-520 | 53-525 | 53-527 | 53-555 | 53-571 | 53-612 | 53-634 |
| | | | 53-643 | 53-644 | #53-645 | 53-649 | 57-936 | 62-2197 | 62-2197 | 62-2199 |
| | | | 62-2199 | | | | | | | 53-642 |
| TYPOC | = | 104402 | 53-493 | 53-540 | #53-645 | | | | | |
| TYPON | = | 104406 | #53-645 | | | | | | | |
| TYPOS | = | 104404 | #53-645 | | | | | | | |
| TYPVAD | | 002244 | 53-563 | #53-592 | | | | | | |
| UBADDR | | 002352 | 53-568 | #53-624 | | | | | | |
| UBCOUN | | 004506 | #57-835 | 58-1377 | | | | | | |
| UBMAP | | 015372 | #59-1772 | | | | | | | |
| UDPAR0 | = | 177660 | #51-222 | | | | | | | |
| UDPAR1 | = | 177662 | #51-222 | | | | | | | |
| UDPAR2 | = | 177664 | #51-222 | | | | | | | |
| UDPAR3 | = | 177666 | #51-222 | | | | | | | |
| UDPAR4 | = | 177670 | #51-222 | | | | | | | |
| UDPAR5 | = | 177672 | #51-222 | | | | | | | |
| UDPAR6 | = | 177674 | #51-222 | | | | | | | |
| UDPAR7 | = | 177676 | #51-222 | | | | | | | |
| UDPDR0 | = | 177620 | #51-222 | | | | | | | |
| UDPDR1 | = | 177622 | #51-222 | | | | | | | |
| UDPDR2 | = | 177624 | #51-222 | | | | | | | |
| UDPDR3 | = | 177626 | #51-222 | | | | | | | |
| UDPDR4 | = | 177630 | #51-222 | | | | | | | |
| UDPDR5 | = | 177632 | #51-222 | | | | | | | |
| UDPDR6 | = | 177634 | #51-222 | | | | | | | |
| UDPDR7 | = | 177636 | #51-222 | | | | | | | |
| UIPAR0 | = | 177640 | #51-222 | | | | | | | |
| UIPAR1 | = | 177642 | #51-222 | | | | | | | |
| UIPAR2 | = | 177644 | #51-222 | | | | | | | |
| UIPAR3 | = | 177646 | #51-222 | | | | | | | |
| UIPAR4 | = | 177650 | #51-222 | | | | | | | |
| UIPAR5 | = | 177652 | #51-222 | | | | | | | |
| UIPAR6 | = | 177654 | #51-222 | | | | | | | |
| UIPAR7 | = | 177656 | #51-222 | | | | | | | |
| UIPDR0 | = | 177600 | #51-222 | | | | | | | |
| UIPDR1 | = | 177602 | #51-222 | | | | | | | |
| UIPDR2 | = | 177604 | #51-222 | | | | | | | |
| UIPDR3 | = | 177606 | #51-222 | | | | | | | |
| UIPDR4 | = | 177610 | #51-222 | | | | | | | |
| UIPDR5 | = | 177612 | #51-222 | | | | | | | |
| UIPDR6 | = | 177614 | #51-222 | | | | | | | |
| UIPDR7 | = | 177616 | #51-222 | | | | | | | |
| USESTK | = | 000600 | #51-222 | | | | | | | |
| USP | = | %000006 | #51-222 | | | | | | | |
| WRITEE | | 004256 | #57-764 | 57-996 | 58-1027 | 58-1057 | 58-1087 | | | |
| SAPTHD | | 020000 | 62-2194 | #62-2194 | | | | | | |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|---------|-----------------|--------|--|
| SENV | | 020034 | #62-2195 62-2197 62-2198 62-2198 |
| SEVM | | 020035 | 53-642 57-930 59-1994 60-2101 #62-2195 62-2198 |
| SEOP | | 021252 | 60-2097 61-2186 #62-2199 |
| SEOPT | | 021312 | *57-935 #62-2199 62-2199 |
| SEFLG | | 001101 | #52-226 62-2196 62-2196 62-2196 *62-2196 62-2196 62-2196 *62-2197 62-2197 |
| SEMAX | | 001113 | #52-226 *57-935 62-2196 *62-2196 62-2196 62-2196 |
| SEERR | | 020470 | 57-935 #62-2197 |
| SERPC | | 001114 | #52-226 53-492 *62-2197 *62-2197 62-2197 62-2197 62-2197 64-2315 64-2317 |
| | | | 64-2318 64-2319 64-2320 64-2323 64-2324 64-2326 64-2327 64-2328 64-2329 |
| | | | 64-2330 64-2331 64-2333 64-2334 64-2335 64-2336 64-2337 64-2338 64-2339 |
| | | | 64-2340 64-2341 64-2342 |
| SERTB | | 001326 | #53-226 53-511 53-515 |
| SERTL | | 001110 | #52-226 *62-2197 62-2197 62-2197 62-2199 *62-2199 |
| SESCAP | | 001210 | #52-226 *57-935 *62-2196 62-2197 62-2197 *62-2197 62-2197 |
| SETABL | | 020034 | #62-2195 |
| SETEND | | 020140 | 62-2194 #62-2195 |
| SEFATAL | | 020016 | #62-2195 *62-2198 |
| SEFFLG | | 021250 | *62-2198 *62-2198 62-2198 *62-2198 #62-2198 |
| SEFILLC | | 001146 | #52-226 53-642 53-642 53-642 |
| SEFILLS | | 001145 | #52-226 53-642 53-642 |
| SEGDADR | | 001116 | #52-226 |
| SEGDAT | | 001122 | #52-226 |
| SEGET42 | | 021444 | #62-2199 |
| SEHD | | 000000 | 51-219 51-219 51-219 |
| SEHIBTS | | 020000 | #62-2194 |
| SEICNT | | 001102 | #52-226 *62-2196 62-2196 *62-2196 62-2196 62-2196 |
| SEILLUP | | 003662 | 53-649 #53-649 |
| SEITEMB | | 001112 | #52-226 53-490 *62-2197 62-2197 62-2197 62-2197 |
| SELF | | 001220 | #52-226 53-642 53-642 62-2197 62-2197 |
| SEFLG | | 021247 | *62-2198 #62-2198 |
| SELOOP | | 021542 | 62-2199 #62-2199 |
| SELPADR | | 001104 | #52-226 *57-935 *58-1306 *58-1433 *58-1566 *59-1809 *59-1991 *60-2098 *62-2196 |
| | | | *62-2196 62-2196 62-2196 *57-971 *57-991 *57-1000 *58-1022 *58-1031 *58-1052 |
| SELPERR | | 001106 | #52-226 *57-935 *57-965 *57-971 *57-991 *57-1000 *58-1022 *58-1031 *58-1052 |
| | | | *58-1061 *58-1082 *58-1091 *58-1111 *58-1123 *58-1132 *58-1148 *58-1160 *58-1170 |
| | | | *58-1186 *58-1203 *58-1224 *58-1224 *58-1237 *58-1237 *58-1251 *58-1251 *58-1264 |
| | | | *58-1264 *58-1306 *58-1309 *58-1330 *58-1348 *58-1370 *58-1384 *58-1403 *58-1419 |
| | | | *58-1433 *58-1508 *58-1567 *59-1602 *59-1619 *59-1632 *59-1651 *59-1656 *59-1661 |
| | | | *59-1666 *59-1671 *59-1676 *59-1681 *59-1686 *59-1691 *59-1696 *59-1701 *59-1702 |
| | | | *59-1723 *59-1760 *59-1809 *59-1815 *59-1828 *59-1852 *59-1857 *59-1862 *59-1867 |
| | | | *59-1872 *59-1877 *59-1882 *59-1887 *59-1892 *59-1897 *59-1902 *59-1903 *59-1924 |
| | | | *59-1961 *59-1992 *60-2099 62-2196 *62-2196 62-2196 62-2196 |
| SEMADR1 | | 020046 | #62-2195 |
| SEMADR2 | | 020052 | #62-2195 |
| SEMADR3 | | 020056 | #62-2195 |
| SEMADR4 | | 020062 | #62-2195 |
| SEMAIL | | 020074 | 62-2194 62-2194 #62-2195 |
| SEMAMS1 | | 020044 | #62-2195 |
| SEMAMS2 | | 020050 | #62-2195 |
| SEMAMS3 | | 020054 | #62-2195 |
| SEMAMS4 | | 020060 | #62-2195 |

| SYMBOL | CROSS REFERENCE | VALUE | REFERENCES |
|--------|-----------------|--------|--|
| SMBADR | | 020002 | #62-2194 |
| SMFLG | | 021246 | *62-2198 62-2198 *62-2198 #62-2198 |
| SMSGAD | | 020030 | #62-2195 *62-2198 62-2198 |
| SMSGLG | | 020032 | #62-2195 *62-2198 |
| SMSGTY | | 020014 | #62-2195 62-2198 *62-2198 62-2198 *62-2198 |
| SM*YP1 | | 020045 | #62-2195 |
| SMTYP2 | | 020051 | #62-2195 |
| SMTYP3 | | 020055 | #62-2195 |
| SMTYP4 | | 020061 | #62-2195 |
| SMXCNT | | 020466 | 62-2196 62-2196 62-2196 #62-2196 |
| SNULL | | 001144 | #52-226 53-642 53-642 53-642 |
| SNWTST | - | 000001 | #57-962 57-962 #57-962 57-962 #57-988 57-988 #57-988 57-988 #58-1020 |
| | | | 58-1020 #58-1020 58-1020 #58-1050 58-1050 #58-1050 58-1080 #58-1080 |
| | | | #58-1080 58-1080 #58-1108 58-1108 #58-1108 58-1145 #58-1145 58-1145 #58-1145 |
| | | | 58-1145 #58-1184 58-1184 #58-1184 58-1184 #58-1223 58-1223 #58-1223 58-1223 |
| | | | #58-1236 58-1236 #58-1236 58-1236 #58-1250 58-1250 #58-1250 58-1250 #58-1263 |
| | | | 58-1263 #58-1263 58-1263 #58-1306 58-1306 #58-1306 58-1306 #58-1345 58-1345 |
| | | | #58-1345 58-1345 #58-1401 58-1401 #58-1401 58-1401 #58-1432 58-1432 #58-1432 |
| | | | 58-1432 #58-1504 58-1504 #58-1504 58-1504 #58-1551 58-1551 #58-1551 58-1551 |
| | | | #59-1613 59-1613 #59-1613 59-1613 #59-1646 59-1646 #59-1646 59-1646 #59-1713 |
| | | | 59-1713 #59-1713 59-1713 #59-1809 59-1809 #59-1809 59-1809 #59-1845 59-1845 |
| | | | #59-1845 59-1845 #59-1916 59-1916 #59-1916 59-1916 #59-1990 59-1990 #59-1990 |
| | | | 59-1990 #60-2096 60-2096 #60-2096 60-2096 |
| SOCNT | | 003247 | *53-643 *53-643 #53-643 |
| SOCTVL | | 004036 | 53-653 #53-653 |
| SOMODE | | 003244 | *53-643 *53-643 53-643 *53-643 *53-643 #53-643 |
| SOVER | | 020452 | 62-2196 62-2196 62-2196 62-2196 #62-2196 |
| SPASS | | 020022 | *57-935 58-1464 #62-2195 62-2196 62-2196 62-2196 *62-2199 *62-2199 62-2199 |
| | | | 62-2199 62-2199 |
| SPASTM | | 020006 | #62-2194 |
| SPWRAD | | 003656 | #53-649 |
| SPWRDN | | 003534 | #53-649 53-649 57-935 |
| SPWRMG | | 003652 | #53-649 |
| SPWRUP | | 003602 | 53-649 #53-649 |
| SOUES | | 001216 | #52-226 62-2197 62-2197 |
| SRDCHR | - | ***** | 53-645 |
| SRDDEC | - | ***** | 53-645 |
| SRDLIN | - | ***** | 53-645 |
| SRDOCT | - | ***** | 53-645 |
| SREGAD | | 001154 | #52-226 |
| SREGO | | 001156 | #52-226 *62-2197 64-2315 64-2319 64-2323 64-2326 64-2331 64-2336 64-2337 |
| | | | 64-2338 64-2339 64-2340 64-2341 64-2342 |
| SREG1 | | 001160 | #52-226 *62-2197 64-2320 64-2324 64-2327 64-2334 64-2335 64-2338 64-2340 |
| SREG2 | | 001162 | #52-226 *62-2197 64-2319 64-2334 64-2335 64-2339 64-2340 64-2342 |
| SREG3 | | 001164 | #52-226 *62-2197 64-2320 64-2335 64-2339 64-2341 64-2342 |
| SREG4 | | 001166 | #52-226 *62-2197 64-2326 64-2339 |
| SREG5 | | 001170 | #52-226 *62-2197 |
| SRESRE | | 002466 | #53-641 53-645 |
| SRTRN | | 021540 | 57-935 *57-935 *57-935 62-2199 #62-2199 |
| SSAVRE | | 002430 | #53-641 53-645 53-645 |
| SSAVR6 | | 003666 | *53-649 53-649 *53-649 *53-649 #53-649 |
| SSCOPE | | 020140 | 57-935 #62-2196 |

SYMBOL CROSS REFERENCE

SYMBOL VALUE
\$TRAP 003472
\$TRP = 000022

REFERENCES

| | | | | | | | | | | |
|---------|----------|----------|----------|---------|----------|---------|----------|---------|---------|---------|
| \$TRAP | 003472 | #53-645 | 57-935 | | | | | | | |
| \$TRP | = 000022 | #53-645 | 53-645 | 53-645 | 53-645 | 53-645 | #53-645 | 53-645 | 53-645 | 53-645 |
| | | 53-645 | #53-645 | 53-645 | 53-645 | 53-645 | 53-645 | #53-645 | 53-645 | 53-645 |
| | | 53-645 | 53-645 | #53-645 | 53-645 | 53-645 | 53-645 | 53-645 | #53-645 | 53-645 |
| | | 53-645 | 53-645 | 53-645 | #53-645 | 53-645 | 53-645 | 53-645 | 53-645 | #53-645 |
| | | 53-646 | 53-646 | 53-646 | 53-646 | #53-646 | 53-647 | 53-647 | 53-647 | 53-647 |
| | | #53-647 | | | | | | | | |
| \$TRPAD | 003512 | 53-645 | #53-645 | | | | | | | |
| \$TSTM | 020004 | #62-2194 | | | | | | | | |
| \$TSTNM | 001100 | #52-226 | *58-1306 | 58-1306 | *58-1433 | 58-1433 | *59-1809 | 59-1809 | 62-2196 | 62-2196 |
| | | *62-2196 | 62-2196 | 62-2196 | 62-2196 | 62-2196 | 62-2197 | 62-2197 | 62-2197 | 62-2197 |
| | | *62-2199 | | | | | | | | |
| | | 53-645 | | | | | | | | |
| \$TYPBN | ***** | #53-644 | 53-645 | 53-645 | | | | | | |
| \$TYPDS | 003246 | #53-642 | 53-645 | 53-645 | 62-2198 | | | | | |
| \$TYPE | 002524 | 53-642 | 53-642 | 53-642 | #53-642 | 53-642 | | | | |
| \$TYPEC | 002700 | 53-642 | 53-642 | #53-642 | | | | | | |
| \$TYPEX | 003016 | #53-643 | 53-645 | 53-645 | | | | | | |
| \$TYPOC | 003044 | 53-643 | #53-643 | 53-645 | | | | | | |
| \$TYPON | 003060 | #53-643 | 53-645 | | | | | | | |
| \$TYPOS | 003020 | #62-2195 | | | | | | | | |
| \$UNIT | 020026 | #62-2194 | | | | | | | | |
| \$UNITM | 020010 | #62-2195 | | | | | | | | |
| \$USWP | 020040 | #62-2195 | | | | | | | | |
| \$VECT1 | 020064 | #62-2195 | | | | | | | | |
| \$VECT2 | 020066 | #62-2195 | | | | | | | | |
| \$XTSTR | 020214 | #62-2196 | | | | | | | | |
| \$GET4 | 000001 | #62-2199 | #62-2199 | 62-2199 | | | | | | |
| \$STRP | - 000002 | #53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-646 |
| | | 53-647 | | | | | | | | |
| \$CFILL | 003243 | *53-643 | *53-643 | 53-643 | #53-643 | | | | | |
| .\$ASTA | - ***** | 62-2198 | 62-2198 | | | | | | | |
| .\$X | 020000 | #62-2194 | 62-2194 | | | | | | | |

| MACRO NAME | REFERENCES | | | | | | | | | |
|------------|------------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| SAVST | #51-54 | 62-2197 | | | | | | | | |
| SETTRA | #53-645 | #53-645 | #53-645 | #53-645 | #53-645 | #53-645 | #53-645 | #53-645 | #53-646 | #53-647 |
| SETUP | #10-1188 | #51-222 | 57-935 | | | | | | | |
| SKIP | #16-1505 | #51-222 | 57-974 | 57-1003 | 58-1034 | 58-1064 | 58-1094 | 58-1206 | 58-1224 | 58-1237 |
| | 58-1251 | 58-1264 | 58-1465 | 58-1537 | 58-1570 | | | | | |
| SLASH | #12-1309 | #51-222 | | | | | | | | |
| SPACE | #51-9 | #51-222 | | | | | | | | |
| SSCOPE | #51-32 | #62-2196 | | | | | | | | |
| STARS | #11-1278 | #51-222 | #51-225 | #52-226 | #53-226 | #53-641 | #53-642 | #53-643 | #53-644 | #53-645 |
| | #53-649 | #53-653 | #57-962 | #57-962 | #57-988 | #57-988 | #58-1020 | #58-1020 | #58-1050 | #58-1050 |
| | #58-1080 | #58-1080 | #58-1108 | #58-1108 | #58-1145 | #58-1145 | #58-1184 | #58-1184 | #58-1223 | #58-1223 |
| | #58-1236 | #58-1236 | #58-1250 | #58-1250 | #58-1263 | #58-1263 | #58-1306 | #58-1306 | #58-1345 | #58-1345 |
| | #58-1401 | #58-1401 | #58-1432 | #58-1432 | #58-1504 | #58-1504 | #58-1551 | #58-1551 | #59-1613 | #59-1613 |
| | #59-1646 | #59-1646 | #59-1713 | #59-1713 | #59-1809 | #59-1809 | #59-1845 | #59-1845 | #59-1916 | #59-1916 |
| | #59-1990 | #59-1990 | #60-2096 | #60-2096 | #62-2194 | #62-2194 | #62-2194 | #62-2194 | #62-2195 | #62-2197 |
| | #62-2198 | #62-2199 | | | | | | | | |
| TRAK | #51-27 | 57-962 | 57-988 | 58-1020 | 58-1050 | 58-1080 | 58-1108 | 58-1145 | 58-1184 | 58-1223 |
| | 58-1236 | 58-1250 | 58-1263 | 58-1345 | 58-1401 | 58-1432 | 58-1504 | 58-1551 | 59-1613 | 59-1646 |
| | 59-1713 | 59-1845 | 59-1916 | 59-1990 | | | | | | |
| TRAK1 | #51-20 | 57-962 | 57-988 | 58-1020 | 58-1050 | 58-1080 | 58-1108 | 58-1145 | 58-1184 | 58-1223 |
| | 58-1236 | 58-1250 | 58-1263 | 58-1345 | 58-1401 | 58-1432 | 58-1504 | 58-1551 | 59-1613 | 59-1646 |
| | 59-1713 | 59-1845 | 59-1916 | 59-1990 | | | | | | |
| TRMTRP | #53-645 | | | | | | | | | |
| TYPBIN | #19-1812 | #51-222 | | | | | | | | |
| TYPDEC | #19-1782 | #51-222 | 62-2199 | 62-2199 | | | | | | |
| TYPNAM | #17-1558 | #51-222 | 57-936 | | | | | | | |
| TYPNUM | #19-1749 | #51-222 | | | | | | | | |
| TYPOCS | #19-1702 | #51-222 | | | | | | | | |
| TYPOCT | #19-1665 | #51-222 | | | | | | | | |
| TYPTXT | #18-1619 | #51-222 | 62-2199 | 62-2199 | | | | | | |
| USER | #51-78 | 52-226 | | | | | | | | |
| ZEROER | #51-66 | 62-2197 | | | | | | | | |
| SSCMRE | #51-226 | #52-226 | #52-226 | #52-226 | #52-226 | #52-226 | #52-226 | #52-226 | | |
| SSCMTM | #51-226 | #52-226 | #52-226 | #52-226 | #52-226 | #52-226 | #52-226 | | | |
| SSESCA | #15-1484 | #51-222 | | | | | | | | |
| SSNEW1 | #14-1438 | #51-222 | 57-962 | 57-988 | 58-1020 | 58-1050 | 58-1080 | 58-1108 | 58-1145 | 58-1184 |
| | 58-1223 | 58-1236 | 58-1250 | 58-1263 | 58-1306 | 58-1345 | 58-1401 | 58-1432 | 58-1504 | 58-1551 |
| | 59-1613 | 59-1646 | 59-1713 | 59-1809 | 59-1845 | 59-1916 | 59-1990 | 60-2096 | | |
| SSSET | #53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-645 | 53-646 | 53-647 |
| SSSETU | #57-935 | 57-935 | | | | | | | | |
| SSSKIP | #16-1518 | #51-222 | 57-974 | 57-1003 | 58-1034 | 58-1064 | 58-1094 | 58-1206 | 58-1224 | 58-1237 |
| | 58-1251 | 58-1465 | 58-1537 | 58-1570 | | | | | | |
| .EQUAT | #4-183 | | | | | | | | | |
| .EQUIV | #51-6 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 |
| | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 |
| | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 |
| | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 |
| | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 |
| | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 |
| | #51-222 | #51-222 | #51-222 | #51-222 | #51-222 | #56-672 | #58-1307 | #59-1714 | | |
| .HEADI | #2-58 | #51-210 | 51-219 | | | | | | | |
| .KT11 | #5-326 | | | | | | | | | |
| .SETUP | #9-1134 | #51-210 | 51-223 | | | | | | | |

| MACRO NAME | REFERENCES | | |
|------------|------------|---------|----------|
| .SWRHI | #3-100 | #51-210 | 51-220 |
| .SWRLO | #51-220 | 51-221 | |
| .SACT1 | #45-4374 | #51-210 | #51-225 |
| .SAPT8 | #45-4428 | #51-213 | 62-2195 |
| .SAPTH | #46-4684 | #51-213 | #62-2194 |
| .SAPTY | #49-4859 | #51-213 | 62-2198 |
| .SASTA | #47-4730 | | |
| .SCATC | #7-929 | #51-210 | 51-224 |
| .SCMTA | #8-986 | #51-210 | 51-226 |
| .SDB2D | #40-4111 | | |
| .SDB2O | #42-4236 | #51-212 | 53-653 |
| .SDIV | #39-4013 | | |
| .SEOP | #21-1935 | #51-211 | #62-2199 |
| .SERRO | #23-2352 | #51-211 | 62-2197 |
| .SERRT | #24-2534 | | |
| .SMULT | #38-3949 | | |
| .SPOWE | #34-3657 | #51-212 | 53-649 |
| .SRAND | #35-3717 | | |
| .SRDDE | #31-3353 | | |
| .SRDOC | #30-3261 | | |
| .SREAD | #29-3025 | | |
| .SSAVE | #32-3429 | #51-211 | #53-641 |
| .SSB2D | #41-4196 | | |
| .SSB2O | #43-4299 | | |
| .SSCOP | #22-2153 | #51-211 | 62-2196 |
| .SSIZE | #36-3771 | | |
| .SSUPR | #44-4338 | | |
| .STRAP | #33-3530 | #51-212 | #53-645 |
| .STYP8 | #28-2928 | | |
| .STYPD | #27-2850 | #51-211 | 53-644 |
| .STYPE | #25-2622 | #51-211 | 53-642 |
| .STYPC | #26-2753 | #51-211 | 53-643 |
| .1170 | #6-509 | #51-210 | 51-222 |