

B01

EOF1DQKDCASEQ
PDP10 411

00010000

770323

PDP10 411

HDR1DQKKAASEQ

00010000

770323

.REPT 0

IDENTIFICATION

Product Code: MAINDEC-11-DQKKA-A-D
PRODUCT NAME: 11/6X CACHE DIAGNOSTIC
DATE: MARCH, 1977
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: WARREN SALTZ

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) 1977, BY DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	SYSTEM REQUIREMENTS
2.1	Hardware
2.2	Software
2.3	APT Setup
2.4	Execution Time
3.0	DIAGNOSTIC HIERARCHY PREREQUISITES
4.0	STARTING ADDRESS
5.0	PROGRAM CONTROL AND OPERATOR ACTION
6.0	SWITCH OPTIONS
7.0	PROGRAM DESCRIPTION
8.0	ERROR REPORTING AND FAULT ISOLATION
9.0	HANDLERS AND COMMON ROUTINES
9.1	End of Pass Routine
9.2	Scope Handler
9.3	Error Handler
9.4	Memory Size Routine
9.5	Trap Handler
9.6	Power Down and Up Routine
9.7	Trap Catcher
9.8	UPERR Routine
9.9	UT4 Routine
9.10	VIP Routine
9.11	TAG Routine
9.12	VEC Routine
9.13	HUBEN Routine
9.14	HUBEO Routine
9.15	HRK05 Routine
9.16	HRP03 Routine
9.17	HTU10 Routine
9.18	HAD Routine
9.19	Sweep Routine

1.0 ABSTRACT

The 11/6X Cache Diagnostic is comprised of a series of tests which were designed to check the cache's data paths on the Cache/KT board and its control logic on the Bus Control module. The tests are arranged in a logical order such that they build on one another. That is, the currently running test will depend on logic exercised by previous tests. Basic cache operations are exercised first followed by address and data functions. Those tests requiring extensive amounts of cache functioning are done near the end of the program. This testing procedure should provide a very effective degree of fault isolation.

2.0 SYSTEM REQUIREMENTS

2.1 Hardware

1. A working 11/6X CPU
2. A minimum of 13K to a max of 124K of memory. 124K is needed for complete check of TAG memory.
3. A console terminal (not mandatory under APT)
4. One of the following peripherals if NPR DATOs are to be tested (SW8=1).
 - a. Unibus Exercisor (M7885)
 - b. Bus Tester (old)
 - c. RK05
 - d. RPO3
 - e. TU10
5. When running under APT and either the NPR DATO tests (SW08=1) or the power up tests (SW07=1) are to be run, the diagnostic assumes a default peripheral of the Unibus Exercisor (M7885). In addition it assumes its data buffer address (BEDB) is 770000.

2.2 Software

This diagnostic will run under ACT/APT, XXDP and stand alone. When running under one of the various system testers, there should be no peripheral device doing any NPR DATO traffic on the bus (except those specifically chosen and under control of the diagnostic).

2.3 APT Setup

When running under APT and the NPR device tests or the power down tests are to be run, the APT software switch reg (switch 8 & 7 respectively) should be set (see sec. 6.0). The default APT device must be present when this is done (see

2.1.5).

2.4 Execution Time

For an error free, first run pass on a PDQ with core memory, it takes approximately 15 seconds.

3.0 DIAGNOSTIC HIERARCHY PREREQUISITES

It is assumed that CPU, memory, KT and stack limit are working properly for this program to give correct error reports. If not, their respective diagnostic should be run before the cache diagnostic. In addition, if one of the peripheral devices (see 2.1-4) is chosen, it is assumed to be error free. If not, further tests using the device are skipped.

4.0 STARTING ADDRESS

200 for normal startup

5.0 PROGRAM CONTROL AND OPERATOR ACTION

5.1 The standard diagnostic loading procedures are to be followed.

5.2 Load address 200

5.3 If the power up test is to be run set switch 07=1. If not running under APT after the test is started and the message "POWER MACHINE DOWN AND THEN UP" is typed, the machine should be powered down and up. The test will then continue. If running under APT & SW07=1, the program assumes the Unibus Exerciser is available. There is no type out when the exerciser is used in this manner.

5.4 If one of the peripheral devices is available (see 2.1-4) and the NPR DATO tests are to be done, set switch 8=1. Upon start of the program, the following beginning message will be typed (under APT message is not typed see sec. 6.8):

"TYPE WHICH DEVICE SHOULD BE USED:"

- 0 - [carriage return] - Unibus Exerciser (M7885)
- 1 - [carriage return] - Bus Tester Old
- 2 - [carriage return] - RK05
- 3 - [carriage return] - RPO3
- 4 - [carriage return] - TU10

Before any device is chosen, it should be powered up and in the Ready state. The device should be write enabled and a scratch disk or tape should be mounted if the corresponding peripheral is used. The operator should then choose one of the devices and indicate his choice with a carriage return. If an incorrect entry is made (<0 or >4) the message "?INVALID ENTRY, TRY AGAIN" is typed. The program then waits for a correct value to be chosen. A rubout feature is provided to delete a typing error.

Depending upon the operator's choice, different information will have to be supplied by the user. The dialogue for each device is as follows:

a. 0 - Unibus Exercisor new

The following message is printed:

"TYPE THE UBE'S DATA BUFFER ADDRESS"

The operator should then supply the requested information. If the data is valid, the program proceeds to the first test. If there is no response to the address, the following message is printed:

"DEVICE DOES NOT RESPOND;
REFERENCE TO IT TRAPS TO 4."

"?INVALID ENTRY, TRY AGAIN."

If the entry typed is not a valid data buffer address, the following message is printed:

"?INVALID ENTRY, TRY AGAIN"

In either case, the user should retype the correct data buffer address or restart the test and choose another device.

b. 1 - Unibus Exercisor old

No further operator action is needed if the device is present. If a reference to it times out, the following message is typed:

"DEVICE DOES NOT RESPOND
REFERENCE TO IT TRAPS TO 4"

The program then retypes the beginning message and the user must choose another device.

c. 2 - RKOS

If the RKOS is present, the following message is printed:

"WHICH DRIVE SHOULD BE USED?
TYPE 0-7 <CARRIAGE RETURN>"

The user should then type the device number he wishes to use and indicate his choice with a carriage return. If a valid drive is chosen (>0,=0 or <8) the program proceeds to the first test. If it is invalid, the following message is typed:

"?INVALID ENTRY, TRY AGAIN"

The operator should then choose a correct drive number or restart the test and choose another device.

If a reference to an RK05 register times out, the RK05 is assumed not present or inoperable. In this case the following message is typed:

"DEVICE DOES NOT RESPOND
REFERENCE TO IT TRAPS TO 4"

The program then retypes the beginning message and the user must choose another device.

d. 3 - RPO3

If the RPO3 is present the following message is printed:

"WHICH DRIVE SHOULD BE USED?
TYPE 0-7 <CARRIAGE RETURN>"

The user should then type the drive number he wishes to use and indicate his choice with a carriage return. If a valid drive is chosen (>0,=0 or <8), the program proceeds to the first test. If it is invalid, the following message is typed:

"?INVALID ENTRY, TRY AGAIN"

The operator should then choose a correct drive number or restart the test and choose another device.

If a reference to an RPO3 register times out, the RPO3 is assumed not present or inoperable. In this case the following message is typed:

"DEVICE DOES NOT RESPOND
REFERENCE TO IT TRAPS TO 4"

The program then retypes the beginning message and the user must choose another device.

e. 4 - TU10

If the TU10 is present, the following message is printed:

"WHICH DRIVE SHOULD BE USED?
TYPE 0-7 <CARRIAGE RETURN>"

A scratch tape should be mounted and the user should then type the drive number he wishes to use and indicate his choice with a carriage return. If a valid drive number is chosen, the device is selected properly, and the write protect is off, the program proceeds to the first test. If any of the above are false the proper message is typed. The operator should then correct the problem and then choose another drive number.

If in the initial set up of the tape drive the ready bit fails to set or the error bit sets, one of the following messages is then typed:

"DEVICE READY BIT DOES NOT SET"

or

"DEVICE ERROR BIT SET"

In either case the TUID is assumed defective and the beginning message is then typed. The user must then choose another device.

5.5 Start the Program

6.0 SWITCH OPTIONS

SW<15>=1=100000	Halt on Error
SW<14>=1=040000	Loop on Test
SW<13>=1=020000	Inhibit Error Typeouts
SW<12>=1=010000	Inhibit Tests Using Memory Management
SW<11>=1=004000	Inhibit Iterations
SW<10>=1=002000	Bell on Error
SW<09>=1=001000	Loop on Error
SW<08>=1=000400	Enable NPR Device Tests
SW<07>=1=000200	Enable Power up Test

6.1 SW<15>

When set, the program halts on encountering an error after printing out the error message. Pressing continue restores normal program operation.

6.2 SW<14>

The program loops on the subtest that is being executed when the switch is set.

6.3 SW<13>

When set, this switch inhibits all error typeouts.

6.4 SW<12>

When set, this switch inhibits those tests using memory management. This switch should only be used when there is reason to believe that the KT is failing. Significant portions of cache will not be tested when this switch is set.

6.5 SW<11>

When set, iterations of each test is inhibited.

6.6 SW<10>

When set, the bell is rung upon encountering an error.

6.7 SW<09>

When set, upon finding an error, the program will cycle from the point of error to the previous scope statement or error loop (SLPERR). (see sec. 9.2).

6.8 SW<08>

When set, the NPR device tests will be run. It also enables the user interactive questions at the start of the test (see sec. 5.4). These questions are only asked on the first pass of the program. This switch should only be set before the program is started. When running under APT a default NPR device (Unibus Exercisor) is assumed and no questions are asked.

6.9 SW<07>

When set, the power up test is run (see sec. 5.3). This switch should not be set when running under ACT since user intervention is required. When running under APT a default device (Unibus Exercisor) is assumed.

7.0 PROGRAM DESCRIPTION

Upon start of the program, the cache is immediately turned off (force miss is on for both halves of cache). The tests then proceed to selectively turn on only the half of cache that is to be exercised. The half of cache that is on is the half where the test locations reside. The half that is off always corresponds to the address space of the test instructions. This is to ensure that the instructions are not executed out of a possibly bad cache. In order to implement this scheme, the program was made non-contiguous between certain subtests.

The tests are structured on a half cache basis. That is several tests may be run on the low cache and then when the instruction address space has changed sufficiently to overlap the low cache addresses, the same tests will be repeated for the high cache addresses (low cache is defined as that portion of cache with physical address A10=0, high cache is defined as that portion of cache with physical address A10=1). This is done until cache is sufficiently checked out to assure that when all of it is turned on, there is a high probability that instructions can be executed out of it.

To facilitate the testing of cache, a 1K buffer is reserved at the end of the program for read and write operations. The starting address is BUFL corresponding to the first low cache address (A1-A9=0). The address BUFH corresponds to the first high cache address.

Immediately after the program is started the program identifies itself and then if SWB=1 it will interrogate the user about which peripheral device to use for the entire test (see sec. 5.4). This is only done on program start and not repeated for subsequent program loops. The interrogation is not done if running under APT. After this tests 1-47 are run.

8.0 ERROR REPORTING AND FAULT ISOLATION

Error calls are made via the EMT instruction. The lower byte of the instruction is encoded to indicate the error number. For example ERROR 1 would be (EMT+1) or 104001. Once an error instruction is executed, an error handler routine will then process the error call. The error message to be typed is determined from the item table at the end of the program. Item 1 corresponds to error 1 and so on. The item table contains a series of pointers to the message to be typed.

All error messages are identified by the words "ERROR: " or "FATAL ERROR: ".

A fatal error is a catastrophic failure which would cause all further printouts to be wrong or misleading. This is because fatal errors are only used to report failures in the hit reg and the cache control register. The entire diagnostic depends on this hardware functioning. A fatal error aborts the program and end of pass count is typed. In an "error" typeout only the individual test will be skipped. In some instances, the test will be continued until a max number of errors (usually 3) have been encountered. This is only done in cases where additional error information would aid in isolation.

The contents of the error reports identifies the hardware under test at the time of failure. Other pertinent information such as contents of cache control fields and

failing addresses are also reported. The address information is reported as physical address high (P ADDH) corresponding to address bits A17, A16 and physical address low (P ADDL) corresponding to A15-A0.

When trouble shooting a failing board, the first error reported should be the first one fixed. This is because the nature of the software and hardware can create additional, false or misleading error messages to appear after the first one. Since the tests build on one another and involve previously tested hardware, it will aid in the fault isolation to look up the tests previously run to know which hardware has been tested. It should be pointed out that the probability of the error lying on the bus control board will decrease after the basic cache tests are successfully completed. The bus control contains a great deal of cache's hardcore control logic which if not functioning will mean, many times, that the cache diagnostic or any program can not run out of cache. Because of this, if the diagnostic reports an error, there is a higher probability of it lying on the Cache/KT board than the Bus Control board.

9.0 HANDLERS AND COMMON ROUTINES

9.1 End of Pass Routine

This routine takes care of transferring control to the monitor (if one exists) or to the beginning of the program. It indicates the pass number each time it is executed.

9.2 Scope Handler

This handler is called via the 'IOT' trap. When 'scope' is executed an 'IOT' trap occurs to the memory location '\$SCOPE'. Depending on the switch settings, the handler then decides to loop on test, loop on error etc. The scope statement that is located at the first instruction of the following test is the one that enabled the desired action (looping etc.) for the present test.

9.3 Error Handler

This handler uses the 'EMT' trap. The lower byte of the instruction is encoded to indicate the error number. For example ERROR 1 would be (EMT+1) or 104001. Once an error instruction is executed the error handler determines the message to be typed. An item table at the end of the program contains pointers for each message to be typed. Each item corresponds to each error (Item 1 corresponds to error 1). The 'ERRTYP' routine then processes the table for the final error type out.

9.4 Memory Size Routine

This routine sizes memory to find the maximum memory size. If bit7 of location SKT11=1, before the routine is called, memory management will be used. SLSTAD contains the last virtual address of the last bank if memory management is used. Otherwise it contains the last absolute address of available memory. SLSTBK will contain the last bank as a page address register.

9.5 Trap Handler

This handler uses the trap instruction. The lower byte of the instruction is encoded differently for each of the different routines that use it. When a call for a routine is executed a trap occurs to the handler located at STRAP. The handler then determines by looking at the lower byte which address to go to for servicing the call. The following routines use this handler:

1. TYPE - this routine is used to type ASCIZ messages.
2. TYPOCT, TYPOS & TYPON - These routines are used to change a binary number to a 6 digit octal number and type it.
3. RDOCT - this routine will read an octal number from the TTY.
4. RDLIN - this routine will input an ASCII string from the TTY.
5. TYPDS - this routine converts a binary number to decimal and types it.

9.6 Power Down and Up Routines

When a power fail condition occurs, the contents of registers R0-R7 are saved on the stack. When the power returns, the same registers are restored.

9.7 Trap Catcher

This is a series of instructions starting in location 0 to detect unexpected traps and interrupts to the trap and interrupt vector area of memory.

Each vector PC address is loaded with the address of the next location. The next location is loaded with a halt. Thus an illegal trap or interrupt will cause a halt at the trap PSW location plus 2.

Once a halt occurs, by examining the contents of the address pointed to by the stack, the value of the PC when the trap or interrupt occurred can be determined.

9.8 UPERR

This subroutine is used to report unexpected parity errors while the program is running. At the beginning of each test a pointer to the next test is saved. Any spurious parity error is reported and then the test following the one with the error is started.

9.9 UT4

This subroutine reports unexpected traps to 4. After the error is reported, the machine will be halted. Pressing continue will restart the program.

9.10 VIP

This subroutine takes a virtual address stored in location STMPO and converts it to a physical address. The physical address bits A17, A16 are stored in SREG1 and bits A0 - A15 are stored in SREG2.

9.11 TAG

This subroutine calculates the tag field from a page address register's contents stored in STMPO.

9.12 VEC

This subroutine finds out if a new Unibus Exercisor module is being used and if so puts an RTI in its interrupt vector.

9.13 HUBEN

This subroutine sets up the new Unibus exercisor to do one NPR DATO to the address following the subroutine call.

9.14 HUBEO

This subroutine sets up the old Unibus Exercisor to do one NPR DATO to the address following the subroutine call.

9.15 HRK05

This subroutine sets up the RK05 to do NPR DATO's to the starting address following the subroutine call.

9.16 HRP03

This subroutine sets up the RPO3 to do NPR DATO's to the starting address following the subroutine call.

9.17 HTU10

This subroutine sets up the TU10 to do NPR DATO's to the starting address following the subroutine call.

9.18 HAD

This subroutine generates an address in a 1K test buffer at the end of the program. The address is (512)10 locations from the given address following this subroutine call.

9.19 SWEEP

This routine rids cache of bad parity. It is called after all cache has been turned off.

.ENDR

15	OPERATIONAL SWITCH SETTINGS
31	BASIC DEFINITIONS
141	MEMORY MANAGEMENT DEFINITIONS
340	TRAP CATCHER
353	STARTING ADDRESS(S)
360	APT PARAMETER BLOCK
382	ACT11 HOOKS
396	COMMON TAGS
457	APT MAILBOX-ETABLE
575	INITIALIZE THE COMMON TAGS
882	T1 TEST PA MUX AND PHYSICAL ADDRESS DRIVERS
1092	T2 TEST CACHE CAN BE TURNED OFF AND HIT REG CLEARED
1133	T3 TEST CAN GET A HIT ON A HIGH CACHE ADDRESS AND HIT REG CAN =1
1205	T4 TEST FORCE MISS ON HIGH ADDRESS
1229	T5 TEST CACHE TRACKS WHEN CACHE IS OFF
1263	T6 TEST DATOB OPERATION
1323	T7 TEST DATO ALLOCATES CACHE
1362	T10 TEST CAN GET HIT AND FORCE MISS ON LOW CACHE ADDRESS
1409	T11 TEST OF TAG ADDRESS COMPARATOR
1511	T12 TEST FORCE MISS LOCKS OUT PARITY ERRORS & CCR WWP CAN =1
1609	T13 TEST OF TAG PARITY GENERATOR/CHECKER
1824	T14 TEST OF DATA PARITY GENERATOR/CHECKER
2007	T15 TEST THE VALID BIT FOR LOW HALF OF CACHE
2213	T16 TEST TAG PARITY BIT FOR LOW CACHE ADDRESSES
2354	T17 TEST DATA PARITY BITS FOR LOW CACHE
2494	T20 TEST THE VALID BIT FOR HIGH HALF OF CACHE
2686	T21 TEST TAG PARITY BIT FOR HIGH CACHE ADDRESSES
2832	T22 TEST TAG ADDRESS BITS FOR LOW HALF OF CACHE
3018	T23 TEST OF CACHE DATA LOC WITH FLOAT 1 & 0 PATTERNS
3145	T24 TEST DATA PARITY BITS FOR HIGH CACHE
3280	T25 TEST TAG ADDRESS BITS FOR HIGH HALF OF CACHE
3472	T26 TEST DATA FIELD FOR LOW HALF OF CACHE
3643	T27 TEST DATA FIELD FOR HIGH HALF OF CACHE
3807	T30 TEST OF MSB ADDRESS (A10) TO VALID BIT
3943	T31 TEST OF MSB ADDRESS (A10) TO CACHE TAG FIELD
4092	T32 TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD
4233	T33 TEST CACHE IS NOT ALLOCATED DURING ODD ADDRESS TRAP
4281	T34 TEST CACHE NOT ALLOCATED DURING RED ZONE TRAP
4332	T35 TEST CACHE NOT ALLOCATED DURING KT ABORT
4395	T36 DYNAMIC TEST OF CACHE
4612	T37 TEST RETRIES TO BACKING STORE DONE ON CACHE PARITY TRAP
4679	T40 TEST DATO TO I/O LOC NOT WRITTEN IN CACHE AND I/O
4718	T41 TEST CONSOLE INITIATED SNEEP INVALIDATES ALL CACHE
4782	T42 TEST POWER UP INVALIDATES CACHE AND CLEARS CACHE CONTROL REG
4918	T43 TEST NPR DATO INVALIDATES CACHE FOR PHYSICAL ADDRESS BITS A1-A10
5049	T44 TEST NPR DATO INVALIDATES CACHE FOR PHYSICAL ADDRESS BITS A17-A11
5187	END OF PASS ROUTINE
5590	SCOPE HANDLER ROUTINE
5651	ERROR HANDLER ROUTINE
5707	ERROR MESSAGE TYPEOUT ROUTINE
5754	ROUTINE TO SIZE MEMORY
5846	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
5913	TYPE ROUTINE
5992	APT COMMUNICATIONS ROUTINE
6049	BINARY TO OCTAL (ASCII) AND TYPE
6126	TTY INPUT ROUTINE

C02

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC MACY11 27(1006) 09-FEB-77 15:33
DQKKA.P11 07-FEB-77 11:01 TABLE OF CONTENTS

6228	READ AN OCTAL NUMBER FROM THE TTY
6281	TRAP DECODER
6304	TRAP TABLE
6322	POWER DOWN AND UP ROUTINES
7500	ERROR POINTER TABLE

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

000001

.TITLE MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
: #COPYRIGHT (C) APRIL 11, 1975
: #DIGITAL EQUIPMENT CORP.
: #MAYNARD, MASS. 01754
:
: #PROGRAM BY WARREN L. SALTZ
:
: #THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
: #PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
:

\$TN=1
.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT TEST USING MEMORY MANAGEMENT
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	ENABLE NPR DEVICE TESTS
7	ENABLE POWER UP TEST

.SBTTL BASIC DEFINITIONS

001100

: #INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR ;; BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;; BASIC DEFINITION OF SCOPE CALL

000011
000012
000015
000200
177776
177774
177772
177570
177570

: #MISCELLANEOUS DEFINITIONS
HT= 11 ;; CODE FOR HORIZONTAL TAB
LF= 12 ;; CODE FOR LINE FEED
CR= 15 ;; CODE FOR CARRIAGE RETURN
CRLF= 200 ;; CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 ;; PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774 ;; STACK LIMIT REGISTER
PIRQ= 177772 ;; PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570 ;; HARDWARE SWITCH REGISTER
DDISP= 177570 ;; HARDWARE DISPLAY REGISTER

000000
000001
000002
000003
000004
000005
000006
000007

: #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

57 000006
 58 000007
 59
 60
 61 000000
 62 000040
 63 000100
 64 000140
 65 000200
 66 000240
 67 000300
 68 000340
 69
 70
 71 100000
 72 040000
 73 020000
 74 010000
 75 004000
 76 002000
 77 001000
 78 000400
 79 000200
 80 000100
 81 000040
 82 000020
 83 000010
 84 000004
 85 000002
 86 000001
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99 100000
 100 040000
 101 020000
 102 010000
 103 004000
 104 002000
 105 001000
 106 000400
 107 000200
 108 000100
 109 000040
 110 000020
 111 000010
 112 000004

SP= %6 ;;STACK POINTER
 PC= %7 ;;PROGRAM COUNTER

;;*PRIORITY LEVEL DEFINITIONS
 PR0= 0 ;;PRIORITY LEVEL 0
 PR1= 40 ;;PRIORITY LEVEL 1
 PR2= 100 ;;PRIORITY LEVEL 2
 PR3= 140 ;;PRIORITY LEVEL 3
 PR4= 200 ;;PRIORITY LEVEL 4
 PR5= 240 ;;PRIORITY LEVEL 5
 PR6= 300 ;;PRIORITY LEVEL 6
 PR7= 340 ;;PRIORITY LEVEL 7

;;*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000
 SW14= 40000
 SW13= 20000
 SW12= 10000
 SW11= 4000
 SW10= 2000
 SW09= 1000
 SW08= 400
 SW07= 200
 SW06= 100
 SW05= 40
 SW04= 20
 SW03= 10
 SW02= 4
 SW01= 2
 SW00= 1
 .EQUIV SW09, SW9
 .EQUIV SW08, SW8
 .EQUIV SW07, SW7
 .EQUIV SW06, SW6
 .EQUIV SW05, SW5
 .EQUIV SW04, SW4
 .EQUIV SW03, SW3
 .EQUIV SW02, SW2
 .EQUIV SW01, SW1
 .EQUIV SW00, SW0

;;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 BIT11= 4000
 BIT10= 2000
 BIT09= 1000
 BIT08= 400
 BIT07= 200
 BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4

```

113      000002      BIT01= 2
114      000001      BIT00= 1
115      .EQUIV BIT09,BIT9
116      .EQUIV BIT08,BIT8
117      .EQUIV BIT07,BIT7
118      .EQUIV BIT06,BIT6
119      .EQUIV BIT05,BIT5
120      .EQUIV BIT04,BIT4
121      .EQUIV BIT03,BIT3
122      .EQUIV BIT02,BIT2
123      .EQUIV BIT01,BIT1
124      .EQUIV BIT00,BIT0
125
126      ;#BASIC "CPU" TRAP VECTOR ADDRESSES
127      000004      ERRVEC= 4          ;: TIME OUT AND OTHER ERRORS
128      000010      RESVEC= 10         ;: RESERVED AND ILLEGAL INSTRUCTIONS
129      000014      TBITVEC=14         ;: "T" BIT
130      000014      TRTVEC= 14         ;: TRACE TRAP
131      000014      BPTVEC= 14         ;: BREAKPOINT TRAP (BPT)
132      000020      IOTVEC= 20         ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
133      000024      PWRVEC= 24         ;: POWER FAIL
134      000030      EMTVEC= 30         ;: EMULATOR TRAP (EMT) **ERROR**
135      000034      TRAPVEC=34         ;: "TRAP" TRAP
136      000060      TKVEC= 60          ;: TTY KEYBOARD VECTOR
137      000064      TPVEC= 64          ;: TTY PRINTER VECTOR
138      000240      PIRQVEC=240        ;: PROGRAM INTERRUPT REQUEST VECTOR
139      .SBTTL MEMORY MANAGEMENT DEFINITIONS
140
141      ;#KT11 VECTOR ADDRESS
142
143      000250      MMVEC= 250
144
145      ;#KT11 STATUS REGISTER ADDRESSES
146
147      177572      SR0= 177572
148      177574      SR1= 177574
149      177576      SR2= 177576
150      172516      SR3= 172516
151
152      ;#USER "I" PAGE DESCRIPTOR REGISTERS
153
154      177600      UIPDR0= 177600
155      177602      UIPDR1= 177602
156      177604      UIPDR2= 177604
157      177606      UIPDR3= 177606
158      177610      UIPDR4= 177610
159      177612      UIPDR5= 177612
160      177614      UIPDR6= 177614
161      177616      UIPDR7= 177616
162
163      ;#USER "D" PAGE DESCRIPTOR REGISTORS
164
165      177620      UDPDR0= 177620
166      177622      UDPDR1= 177622
167      177624      UDPDR2= 177624
168      177626      UDPDR3= 177626

```

169	177630	UDPDR4= 177630
170	177632	UDPDR5= 177632
171	177634	UDPDR6= 177634
172	177636	UDPDR7= 177636
173		
174		;*USER "I" PAGE ADDRESS REGISTERS
175		
176	177640	UIPAR0= 177640
177	177642	UIPAR1= 177642
178	177644	UIPAR2= 177644
179	177646	UIPAR3= 177646
180	177650	UIPAR4= 177650
181	177652	UIPAR5= 177652
182	177654	UIPAR6= 177654
183	177656	UIPAR7= 177656
184		
185		;*USER "D" PAGE ADDRESS REGISTERS
186		
187	177660	UDPAR0= 177660
188	177662	UDPAR1= 177662
189	177664	UDPAR2= 177664
190	177666	UDPAR3= 177666
191	177670	UDPAR4= 177670
192	177672	UDPAR5= 177672
193	177674	UDPAR6= 177674
194	177676	UDPAR7= 177676
195		
196		;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
197		
198	172200	SIPDR0= 172200
199	172202	SIPDR1= 172202
200	172204	SIPDR2= 172204
201	172206	SIPDR3= 172206
202	172210	SIPDR4= 172210
203	172212	SIPDR5= 172212
204	172214	SIPDR6= 172214
205	172216	SIPDR7= 172216
206		
207		;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
208		
209	172220	SDPDR0= 172220
210	172222	SDPDR1= 172222
211	172224	SDPDR2= 172224
212	172226	SDPDR3= 172226
213	172230	SDPDR4= 172230
214	172232	SDPDR5= 172232
215	172234	SDPDR6= 172234
216	172236	SDPDR7= 172236
217		
218		;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
219		
220	172240	SIPAR0= 172240
221	172242	SIPAR1= 172242
222	172244	SIPAR2= 172244
223	172246	SIPAR3= 172246
224	172250	SIPAR4= 172250

225	172252	SIPAR5= 172252
226	172254	SIPAR6= 172254
227	172256	SIPAR7= 172256
228		
229		;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
230		
231	172260	SDPAR0= 172260
232	172262	SDPAR1= 172262
233	172264	SDPAR2= 172264
234	172266	SDPAR3= 172266
235	172270	SDPAR4= 172270
236	172272	SDPAR5= 172272
237	172274	SDPAR6= 172274
238	172276	SDPAR7= 172276
239		
240		;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
241		
242	172300	KIPDR0= 172300
243	172302	KIPDR1= 172302
244	172304	KIPDR2= 172304
245	172306	KIPDR3= 172306
246	172310	KIPDR4= 172310
247	172312	KIPDR5= 172312
248	172314	KIPDR6= 172314
249	172316	KIPDR7= 172316
250		
251		;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
252		
253	172320	KDPDR0= 172320
254	172322	KDPDR1= 172322
255	172324	KDPDR2= 172324
256	172326	KDPDR3= 172326
257	172330	KDPDR4= 172330
258	172332	KDPDR5= 172332
259	172334	KDPDR6= 172334
260	172336	KDPDR7= 172336
261		
262		;*KERNEL "I" PAGE ADDRESS REGISTERS
263		
264	172340	KIPAR0= 172340
265	172342	KIPAR1= 172342
266	172344	KIPAR2= 172344
267	172346	KIPAR3= 172346
268	172350	KIPAR4= 172350
269	172352	KIPAR5= 172352
270	172354	KIPAR6= 172354
271	172356	KIPAR7= 172356
272		
273		;*KERNEL "D" PAGE ADDRESS REGISTERS
274		
275	172360	KDPAR0= 172360
276	172362	KDPAR1= 172362
277	172364	KDPAR2= 172364
278	172366	KDPAR3= 172366
279	172370	KDPAR4= 172370
280	172372	KDPAR5= 172372

281	172374	KDPA6= 172374
282	172376	KDPA7= 172376
283		
284		
285	177752	HMR=177752
286	177746	CCR=177746
287	000106	CDH=106
288	000106	CDL=106
289	000107	CTAG=107
290	177744	EREG=177744
291	177766	CER=177766
292	000101	HIADD=101
293	000102	LOADD=102
294	055016	BSD=55016
295	060000	BUFL=60000
296	062000	BUFH=BUFL+2000
297	076600	MED= 76600
298	000100	RJAM= 100
299	000101	RSER= 101
300	000102	RPBA= 102
301	000107	RTAG= 107
302	000106	RDAT= 106
303	000022	RLOG= 22
304	000222	WLOG= 222
305	000304	WFLI= 304
306	000226	WSM= 226
307	000352	WINIT= 352
308	177572	MMR0=SR0
309	177576	MMR2=SR2
310	000114	PVEC=114
311	177400	RKDS= 177400
312	177402	RKER= 177402
313	177404	RKCS= 177404
314	177406	RKWC= 177406
315	177410	RKBA= 177410
316	177412	RKDA= 177412
317	176710	RPDS= 176710
318	176712	RPER= 176712
319	176714	RPCS= 176714
320	176716	RPWC= 176716
321	176720	RPBA= 176720
322	176722	RPCA= 176722
323	176724	RPOA= 176724
324	172520	MTS= 172520
325	172522	MTC= 172522
326	172524	MTBRC= 172524
327	172526	MTCMA= 172526
328	000001	HMR0= 1
329	000002	HMR1= 2
330	000004	HMR2= 4
331	000010	HMR3= 10
332	000020	HMR4= 20
333	000040	HMR5= 40
334	000015	CR= 15
335	000012	LF= 12
336		

;*OTHER EQUATES

```

;HIT/MISS REG ADDRESS
;CACHE CONTROL REG ADDRESS
;CACHE DATA HIGH ADDRESS
;CACHE DATA LOW ADDRESS
;CACHE TAG ADDRESS
;MEMORY ERROR REG ADDRESS
;CPU ERROR REG ADDRESS
;HIGH UNIBUS ADDRESS OF ERROR
;LOW UNIBUS ADDRESS OF ERROR
;BACKING STORE DATA ADDRESS
;LOW ADDRESS BUFFER (A10=0)
;HIGH ADDRESS BUFFER (A10=1)
;MAINTENANCE INSTRUCTION
;LOG READ ADDRESS FOR JAM REG.
;LOG READ ADDRESS FOR SERVICE REG.
;LOG READ ADDRESS FOR PHYSICAL BUS ADDR.
;LOG READ ADDRESS FOR CACHE TAG
;LOG READ ADDRESS FOR CACHE DATA
;READ ADDRESS FOR CPU INTERNAL REG "WHAMI"
;WRITE ADDRESS FOR CPU INTERNAL REG "WHAMI"
;WRITE ADDRESS FOR CPU INTERNAL REG "FLAG/INT"
;WRITE ADDRESS FOR CPU INTERNAL REG "SWITCH REG"
;WRITE ADDRESS FOR CPU INTERNAL REG "INIT REG" (MOD FOR D
;KT11 STATUS REG
;KT11 STATUS REG
;PARITY TRAP VECTOR
;RK05 DRIVE STATUS REG
;RK05 ERROR REG
;RK05 CONTROL STATUS REG
;RK05 WORD COUNT REG
;RK05 CURRENT BUS ADDRESS REG
;RK05 DISK ADDRESS REG
;RPO3 DEVICE STATUS REG
;RPO3 ERROR REG
;RPO3 CONTROL STATUS REG
;RPO3 WORD COUNT REG
;RPO3 BUS ADDRESS REG
;RPO3 CYLINDER ADDRESS REG
;RPO3 DISK ADDRESS REG
;TU10 STATUS REG
;TU10 COMMAND REG
;TU10 BYTE RECORD COUNTER
;TU10 CURRENT MEMORY ADDRESS REG
;HIT MISS REG BIT 0
;HIT MISS REG BIT 1
;HIT MISS REG BIT 2
;HIT MISS REG BIT 3
;HIT MISS REG BIT 4
;HIT MISS REG BIT 5
;CARRIAGE RETURN
;LINE FEED
    
```

```

337 ;*****
338 .SBTTL TRAP CATCHER
339
340         .=0
341 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
342 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
343 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
344         .=174
345 000174 000000  DISPREG: .WORD 0          ;; SOFTWARE DISPLAY REGISTER
346 000176 000000  SWREG:   .WORD 0          ;; SOFTWARE SWITCH REGISTER
347 ;*****
348         LOC=.
349         .=200
350
351 .SBTTL STARTING ADDRESS(S)
352
353
354 000200 012737 000214 177746  MOV    #214,2#CCR    ;TURN CACHE OFF
355 000206 000137 001362          JMP    2#START      ;JUMP TO STARTING ADDRESS OF PROGRAM.
356         000200
357         001000
358         .=LOC
359         .=1000
360 .SBTTL APT PARAMETER BLOCK
361 ;*****
362 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
363 ;*****
364         .SX=.          ;; SAVE CURRENT LOCATION
365 000024 000024          .=24          ;; SET POWER FAIL TO POINT TO START OF PROGRAM
366         000200          200          ;; FOR APT START UP
367 000044 000044          .=44          ;; POINT TO APT INDIRECT ADDRESS PNTR.
368         001000          SAPTHDR ;; POINT TO APT HEADER BLOCK
369         001000          .=.SX      ;; RESET LOCATION COUNTER
370 ;*****
371 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
372 ;INTERFACE SPEC.
373 SAPTHD:
374 001000 000000  SHIBTS: .WORD 0          ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
375 001002 001236  SMAIL:  .WORD SMAIL     ;; ADDRESS OF APT MAILBOX (BITS 0-15)
376 001004 000060  $SYN:   .WORD 60        ;; RUN TIME OF LONGEST TEST
377 001006 000060  $PSTM:  .WORD 60        ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
378 001010 000000  $UNITH: .WORD          ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
379 001012 000052  .WORD SETEND-SMAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)
380 .SBTTL ACT11 HOOKS
381 ;*****
382 ;HOOKS REQUIRED BY ACT11
383
384         $$VPC=.          ;SAVE PC
385         .=46
386 000046 033106  $ENDAD          ;; 1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP
387         000052          .=52
388 000052 000000  .WORD 0          ;; 2)SET LOC.52 TO ZERO
389         001014          .=$$VPC      ;; RESTORE PC
390
391
392 ;*****

```

```

393
394
395
396
397
398
399      001100
400 001100 000000
401 001100 000000
402 001102 000
403 001103 000
404 001104 000000
405 001106 000000
406 001110 000000
407 001112 000000
408 001114 000
409 001115 001
410 001116 000000
411 001120 000000
412 001122 000000
413 001124 000000
414 001126 000000
415 001130 000000
416 001132 000000
417 001134 177570
418 001136 177570
419 001140 177560
420 001142 177562
421 001144 177564
422 001146 177566
423 001150 000
424 001151 002
425 001152 012
426 001153 000
427 001154 000000
428
429 001156 000000
430 001160 000000
431 001162 000000
432 001164 000000
433 001166 000000
434 001170 000000
435 001172 000000
436 001174 000000
437 001176 000000
438 001200 000000
439 001202 000000
440 001204 000000
441 001206 077
442 001207 015
443 001210 000012
444 001212 000000
445 001214 000000
446 001216 000000
447 001220 000000
448 001222 000000

```

.SBTTL COMMON TAGS

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

SCMTAG: .=1100

```

      .WORD 0
STSTNM: .BYTE 0
SERFLG: .BYTE 0
$ICNT:  .WORD 0
SLPADR: .WORD 0
SLPERR: .WORD 0
SERTTL: .WORD 0
$ITEMB: .BYTE 0
SERMAX: .BYTE 1
SERRPC: .WORD 0
SGDADR: .WORD 0
$BDADR: .WORD 0
SGDOAT: .WORD 0
SBDOAT: .WORD 0
      .WORD 0
SMR:     .WORD DSWR
DISPLAY: .WORD DDISP
$TKS:    177560
$TKB:    177562
$TPS:    177564
$TPB:    177566
$NULL:   .BYTE 0
$FILLS:  .BYTE 2
$FILLC:  .BYTE 12
$STPFLG: .BYTE 0
$REGAD:  .WORD 0
$REG0:   .WORD 0
$REG1:   .WORD 0
$REG2:   .WORD 0
$REG3:   .WORD 0
$REG4:   .WORD 0
$REG5:   .WORD 0
$TMP0:   .WORD 0
$TMP1:   .WORD 0
$TMP2:   .WORD 0
$TMP3:   .WORD 0
$TMP4:   .WORD 0
$TMP5:   .WORD 0
$QUES:   .ASCII 1?/
$CRLF:   .ASCII <15>
$LF:     .ASCII <12>
CREG1:   .WORD 0
CREG2:   .WORD 0
CREG3:   .WORD 0
CREG4:   .WORD 0
CREG5:   .WORD 0

```

;;START OF COMMON TAGS

```

: CONTAINS THE TEST NUMBER
: CONTAINS ERROR FLAG
: CONTAINS SUBTEST ITERATION COUNT
: CONTAINS SCOPE LOOP
: CONTAINS SCOPE RETURN FOR ERRORS
: CONTAINS TOTAL ERRORS DETECTED
: CONTAINS ITEM CONTROL BYTE
: CONTAINS MAX. ERRORS PER TEST
: CONTAINS PC OF LAST ERROR INSTRUCTION
: CONTAINS OF 'GOOD' DATA
: CONTAINS OF 'BAD' DATA
: CONTAINS 'GOOD' DATA
: CONTAINS 'BAD' DATA
: RESERVED--NOT TO BE USED

: OF SWITCH REGISTER
: OF DISPLAY REGISTER
: TTY KBD STATUS
: TTY KBD BUFFER
: TTY PRINTER STATUS REG.
: TTY PRINTER BUFFER REG.
: CONTAINS NULL CHARACTER FOR FILLS
: CONTAINS # OF FILLER CHARACTERS REQUIRED
: INSERT FILL CHARS. AFTER A "LINE FEED"
: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
: CONTAINS THE FROM
: WHICH ($REG0) WAS OBTAINED
: CONTAINS (( $REGAD)+0)
: CONTAINS(( $REGAD)+2)
: CONTAINS(( $REGAD)+4)
: CONTAINS(( $REGAD)+6)
: CONTAINS(( $REGAD)+10)
: CONTAINS(( $REGAD)+12)
: USER DEFINED
: USER DEFINED
: USER DEFINED
: USER DEFINED
: USER DEFINED
: USER DEFINED
: QUESTION MARK
: CARRIAGE RETURN
: LINE FEED
: CONTROL REG ADDR. FOR NPR DEVICE
: CONTROL REG ADDR. FOR NPR DEVICE
: CONTROL REG ADDR. FOR NPR DEVICE
: CONTROL REG ADDR. FOR NPR DEVICE
: CONTROL REG ADDR. FOR NPR DEVICE

```


449 001224 000000
 450 001226 000000
 451 001230 000000
 452 001232 000000
 453 001234 000000
 454
 455
 456
 457
 458
 459 001236
 460 001236 000000
 461 001240 000000
 462 001242 000000
 463 001244 000000
 464 001246 000000
 465 001250 000000
 466 001252 000000
 467 001254 000000
 468 001256
 469 001256 000
 470 001257 000
 471 001260 000000
 472 001262 000000
 473 001264 000000
 474
 475
 476
 477
 478
 479
 480 001266 000
 481 001267 000
 482
 483
 484
 485
 486 001270 000000
 487
 488 001272 000
 489 001273 000
 490 001274 000000
 491 001276 000
 492 001277 000
 493 001300 000000
 494 001302 000
 495 001303 000
 496 001304 000000
 497 001306 000000
 498 001310 000000
 499 001312 000000
 500 001314 000000
 501 001316 000000
 502 001320 000000
 503 001322 000000
 504 001324 000000

CREG6: .WORD 0 ;CONTROL REG ADDR. FOR NPR DEVICE
 IVEC: .WORD 0 ;ADDRESS OF DEVICE'S INTERRUPT VECTOR
 EAD: .WORD 0 ;ADDRESS OF DEVICE'S ERROR REG
 SETUP: .WORD 0 ;ADDRESS OF DEVICE'S HANDLER
 SKTST: .WORD 0 ;; POINTER TO TEST FOLLOWING ONE BEING EXECUTED

.SBTTL APT MAILBOX-ETABLE

```

;*****
;EVEN
SMAIL: ;APT MAILBOX
MSGTY: .WORD AMSGTY ;MESSAGE TYPE CODE
SFATAL: .WORD AFATAL ;FATAL ERROR NUMBER
STESTN: .WORD ATESTN ;TEST NUMBER
SPASS: .WORD APASS ;PASS COUNT
SDEVCT: .WORD ADEVCT ;DEVICE COUNT
SUNIT: .WORD AUNIT ;I/O UNIT NUMBER
MSGAD: .WORD AMSGAD ;MESSAGE ADDRESS
MSGLG: .WORD AMGLG ;MESSAGE LENGTH
SETABLE: ;APT ENVIRONMENT TABLE
SENV: .BYTE AENV ;ENVIRONMENT BYTE
SEVM: .BYTE AENVM ;ENVIRONMENT MODE BITS
SSWREG: .WORD ASWREG ;APT SWITCH REGISTER
SUSWR: .WORD AUSWR ;USER SWITCHES
SCPUOP: .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
SMAMS1: .BYTE AMAMS1 ;HIGH ADDRESS, M.S. BYTE
SMAMP1: .BYTE AMAMP1 ;MEM. TYPE, BLK#1
;MEM. TYPE BYTE -- (HIGH BYTE)
; 900 NSEC CORE=001
; 300 NSEC BIPOLAR=002
; 500 NSEC MOS=003
SMADR1: .WORD AMADR1 ;HIGH ADDRESS, BLK#1
;MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
SMAMS2: .BYTE AMAMS2 ;HIGH ADDRESS, M.S. BYTE
SMAMP2: .BYTE AMAMP2 ;MEM. TYPE, BLK#2
SMADR2: .WORD AMADR2 ;MEM. LAST ADDRESS, BLK#2
SMAMS3: .BYTE AMAMS3 ;HIGH ADDRESS, M.S. BYTE
SMAMP3: .BYTE AMAMP3 ;MEM. TYPE, BLK#3
SMADR3: .WORD AMADR3 ;MEM. LAST ADDRESS, BLK#3
SMAMS4: .BYTE AMAMS4 ;HIGH ADDRESS, M.S. BYTE
SMAMP4: .BYTE AMAMP4 ;MEM. TYPE, BLK#4
SMADR4: .WORD AMADR4 ;MEM. LAST ADDRESS, BLK#4
SVECT1: .WORD AVECT1 ;INTERRUPT VECTOR#1, BUS PRIORITY#1
SVECT2: .WORD AVECT2 ;INTERRUPT VECTOR#2, BUS PRIORITY#2
SBASE: .WORD ABASE ;BASE ADDRESS OF EQUIPMENT UNDER TEST
SDEVN: .WORD ADEVN ;DEVICE MAP
SCDW1: .WORD ACDW1 ;CONTROLLER DESCRIPTION WORD#1
SCDW2: .WORD ACDW2 ;CONTROLLER DESCRIPTION WORD#2
SDDWD: .WORD ADDWD ;DEVICE DESCRIPTOR WORD#0
SDDW1: .WORD ADDW1 ;DEVICE DESCRIPTOR WORD#1

```

505	001326	000000	SDDW2:	.WORD	ADDW2	::	DEVICE	DESCRIPTOR	WORD#2
506	001330	000000	SDDW3:	.WORD	ADDW3	::	DEVICE	DESCRIPTOR	WORD#3
507	001332	000000	SDDW4:	.WORD	ADDW4	::	DEVICE	DESCRIPTOR	WORD#4
508	001334	000000	SDDW5:	.WORD	ADDW5	::	DEVICE	DESCRIPTOR	WORD#5
509	001336	000000	SDDW6:	.WORD	ADDW6	::	DEVICE	DESCRIPTOR	WORD#6
510	001340	000000	SDDW7:	.WORD	ADDW7	::	DEVICE	DESCRIPTOR	WORD#7
511	001342	000000	SDDW8:	.WORD	ADDW8	::	DEVICE	DESCRIPTOR	WORD#8
512	001344	000000	SDDW9:	.WORD	ADDW9	::	DEVICE	DESCRIPTOR	WORD#9
513	001346	000000	SDDW10:	.WORD	ADDW10	::	DEVICE	DESCRIPTOR	WORD#10
514	001350	000000	SDDW11:	.WORD	ADDW11	::	DEVICE	DESCRIPTOR	WORD#11
515	001352	000000	SDDW12:	.WORD	ADDW12	::	DEVICE	DESCRIPTOR	WORD#12
516	001354	000000	SDDW13:	.WORD	ADDW13	::	DEVICE	DESCRIPTOR	WORD#13
517	001356	000000	SDDW14:	.WORD	ADDW14	::	DEVICE	DESCRIPTOR	WORD#14
518	001360	000000	SDDW15:	.WORD	ADDW15	::	DEVICE	DESCRIPTOR	WORD#15

519
520
521 001362
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560

SETEND:

561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616

001362

001362 012706 001100
001366 005026
001370 022706 001134
001374 001374
001376 012706 001100
001402 012737 035152 000020
001410 012737 000340 000022
001416 012737 035412 000030
001424 012737 000340 000032
001432 012737 040306 000034
001440 012737 000340 000036
001446 012737 040364 000024
001454 012737 000340 000026
001462 013737 033054 033046
001470 005037 035406
001474 005037 035606
001500 112737 000001 001115
001506 012737 001506 001106
001514 012737 001514 001110
001522 013746 000004
001526 012737 001562 000004
001534 012737 177570 001134
001542 012737 177570 001136
001550 022777 177777 177356
001556 001012
001560 000403
001562 012716 001570
001566 000002
001570 012737 000176 001134
001576 012737 000174 001136
001604 012637 000004
001610
001610 005037 001244
001614 132737 000200 001257
001622 001403
001624 012737 001260 001134
001632
001632 104401 040542

```
;;*****  
;;*****  
START:  
.SBTTL INITIALIZE THE COMMON TAGS  
;;CLEAR THE COMMON TAGS (SCMTAG) AREA  
MOV #SCMTAG,R6 ;;FIRST LOCATION TO BE CLEARED  
CLR (R6)+ ;;CLEAR MEMORY LOCATION  
CMP #SWR,R6 ;;DONE?  
BNE .-6 ;;LOOP BACK IF NO  
MOV #STACK,SP ;;SETUP THE STACK POINTER  
;;INITIALIZE A FEW VECTORS  
MOV #SSCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE  
MOV #340,@IOTVEC+2 ;;LEVEL 7  
MOV #SEERR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE  
MOV #340,@EMTVEC+2 ;;LEVEL 7  
MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS  
MOV #340,@TRAPVEC+2 ;;LEVEL 7  
MOV #SPWRON,@PWRVEC ;;POWER FAILURE VECTOR  
MOV #340,@PWRVEC+2 ;;LEVEL 7  
MOV #ENDCT,@SEOPCT ;;SETUP END-OF-PROGRAM COUNTER  
CLR STIMES ;;INITIALIZE NUMBER OF ITERATIONS  
CLR SESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS  
MOVB #1,SEMAX ;;ALLOW ONE ERROR PER TEST  
MOV #.,SLPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE  
MOV #.,SLPERR ;;SETUP THE ERROR LOOP ADDRESS  
;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS  
;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.  
MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR  
MOV #64$,@ERRVEC ;;SET UP ERROR VECTOR  
MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER  
MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER  
CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR  
BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED  
 ;;AND THE HARDWARE SWR IS NOT = -1  
BR 65$ ;;BRANCH IF NO TIMEOUT  
64$: MOV #65$, (SP) ;;SET UP FOR TRAP RETURN  
RTI  
65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR  
MOV #DISPREG,DISPLAY  
66$: MOV (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR  
CLR SPASS ;;CLEAR PASS COUNT  
BITB #APTSIZE,SENVN ;;TEST USER SIZE UNDER APT  
BEQ 67$ ;;YES, USE NON-APT SWITCH  
MOV #SSWREG,SWR ;;NO, USE APT SWITCH REGISTER  
67$: TYPE ,MSG1 ;TYPE 11/6X DIAGNOSTIC
```

```

617
618
619
620
621
622
623 001636 012700 170000      MOV      #170000,RO      ;SAVE UBE ADDRESS IF RUNNING UNDER APT
624 001642 105737 001256      TSTB     @#SENV         ;RUNNING UNDER APT?
625 001646 001410                BEQ      2$             ;BRANCH IF NO
626 001650 032777 000400 177256  BIT      #SW08,@SWR     ;ENABLE TESTS USING NPR DEVICES?
627 001656 001074                BNE     UBEAPT         ;BRANCH IF YES TO DEFAULT APT DEVICE:UBE
628 001660 032777 000200 177246  BIT      #SW07,@SWR     ;POWER DOWN TESTS TO BE RUN?
629 001666 001070                BNE     UBEAPT         ;BRANCH IF YES TO DEFAULT APT DEVICE:UBE
630
631 001670 032777 000400 177236 2$:  BIT      #SW08,@SWR     ;ENABLE TESTS USING NPR DEVICE?
632 001676 001002                BNE     Q2             ;BRANCH IF YES
633 001700 000137 003056                JMP      START1        ;GO TO BEGINNING OF TESTS
634
635 001704 104401 040670                Q2:  TYPE      ,MSG3    ;WHICH DEVICE SHOULD BE USED?
636 001710 104410                B2:  RDOCT                     ;WAIT FOR REPLY
637 001712 012600                MOV      (SP)+,RO      ;GET ANS OFF STACK
638 001714 020027 000005                CMP      RO,#5         ;WAS ANS VALID (<5)?
639 001720 002002                BGE     Q1             ;BRANCH IF NO
640 001722 005700                TST      RO            ;ANS VALID?
641 001724 002003                BGE     B1             ;BRANCH IF YES
642 001726 104401 041202                Q1:  TYPE      ,MSG4    ;?INVALID ENTRY TRY AGAIN
643 001732 000766                BR       B2            ;GO WAIT FOR NEW ANS
644
645 001734 000005                B1:  RESET                     ;INITIALIZE ALL DEVICES
646 001736 012737 000214 177746  MOV      #214,@#CCR     ;CACHE OFF
647 001744 006300                ASL      RO            ;ADJUST FOR WORD INDEXING
648 001746 000170 001752                JMP      @TAB(RO)      ;GO ASK FURTHER QUESTIONS ON DEVICE
649
650 001752 001764                TAB:  QUBEN                    ;POINTER TO UNIBUS EXERCISOR (NEW) QUESTIONS
651 001754 002156                QUBEO                    ;POINTER TO UNIBUS EXERCISOR (OLD) QUESTIONS
652 001756 002232                QRK05                    ;POINTER TO RK05 QUESTIONS
653 001760 002412                QRPO3                    ;POINTER TO RPO3 QUESTIONS
654 001762 002540                QTU10                    ;POINTER TO TU10 QUESTIONS
655
656
657
658
659
660 001764 104401 041242                QUBEN: TYPE      ,MSG5    ;TYPE THE UBE'S DATA BUFFER ADDRESS
661 001770 104410                3$:  RDOCT                     ;WAIT FOR ANS
662 001772 012737 002006 000004  MOV      #1$,@#4        ;SET UP FOR TIME OUTS
663 002000 012600                MOV      (SP)+,RO      ;SEE IF DEVOCE RESPONDS
664 002002 005710                TST      (RO)
665 002004 000413                BR       2$            ;BRANCH IF YES
666
667 002006 012737 000006 000004  1$:  MOV      #6,@#4        ;RESTORE TRAP CATCHER
668 002014 005037 000006                CLR      @#6           ;RESTORE TRAP CATCHER
669 002020 022626                CMP      (SP)+,(SP)+   ;RESTORE STACK
670 002022 104401 041312                TYPE      ,MSG6        ;DEVICE DOES NOT RESPOND; TRAPS TO 4
671 002026 104401 041202                4$:  TYPE      ,MSG4    ;?INVALID ENTRY, TRY AGAIN
672 002032 000756                BR       3$            ;WAIT FOR ANS

```

```

673
674 002034 032700 007417      2S:   BIT      #7417,RO      ; IS ADDRESS LEGAL?
675 002040 001372                BNE      4S      ; BRANCH IF NO
676 002042 022700 170000      CMP      #170000,RO ; IS ADDRESS LEGAL?
677 002046 003367                BGT      4S      ; BRANCH IF NO
678 002050 010001      UBEAPT: MOV      RO,R1      ; SAVE BUFFER ADDRESS
679 002052 042700 177000      BIC      #177000,RO ; CALCULATE DEVICE'S
680 002056 006200                ASR      RO      ; INTERRUPT VECTOR
681 002060 006200                ASR      RO
682 002062 062700 000510      ADD      #510,RO   ; RO=DEVICE INT VECTOR
683 002066 010037 001226      MOV      RO,I'VEC ; SAVE DEVICE INT VECTOR
684 002072 010137 001224      MOV      R1,CREG6 ; SAVE DEVICE BUFFER ADDR
685 002076 005721                TST      (R1)+    ; UPDATE ADDRESS
686 002100 010137 001222      MOV      R1,CREG5 ; SAVE UBE CYCLE COUNT REG ADDR.
687 002104 005721                TST      (R1)+    ; UPDATE ADDRESS
688 002106 010137 001220      MOV      R1,CREG4 ; SAVE UBE ADDRESS COUNTER ADDR.
689 002112 005721                TST      (R1)+    ; UPDATE ADDRESS
690 002114 010137 001212      MOV      R1,CREG1 ; SAVE UBE CONTROL REG 1 ADDR.
691 002120 005721                TST      (R1)+    ; UPDATE ADDRESS
692 002122 010137 001216      MOV      R1,CREG3 ; SAVE UBE ERROR CLEAR ADDR.
693 002126 005721                TST      (R1)+    ; UPDATE ADDRESS
694 002130 022121      CMP      (R1)+,(R1)+ ; UPDATE ADDRESS
695 002132 010137 001214      MOV      R1,CREG2 ; SAVE UBE CONTROL REG 2 ADDR.
696 002136 013737 001212 001230  MOV      CREG1,EAD ; SAVE UBE ERROR ADDRESS
697 002144 012737 034046 001232  MOV      #HUBEN,SETUP ; LOAD POINTER FOR UBE HANDLER
698 002152 000137 003056      JMP      START1   ; GO TO BEGINNING OF TEST
699
700 ;//////////////////////////////////////
701 ; UBE OLD INITIALIZE ROUTINE
702 ;//////////////////////////////////////
703
704 002156 012737 002172 000004  QUBEO: MOV      #15,2#4      ; SET UP FOR TIME OUTS
705 002164 005737 170000      TST      2#170000   ; SEE IF DATA BUFFER RESPONDS
706 002170 000405                BR        2S
707 002172 022626      1S:   CMP      (SP)+,(SP)+ ; RESTORE STACK
708 002174 104401 041312      TYPE     ,MSG6      ; DEVICE DOESN'T RESPOND
709 002200 000137 001726      JMP      Q1         ; GO CHOOSE ANOTHER DEVICE
710
711 002204 012737 170006 001212  2S:   MOV      #170006,CREG1 ; SAVE THE GO ADDRESS
712 002212 012737 034224 001230  MOV      #FAKE,EAD   ; SETUP FAKE ADDRESS FOR ERROR TEST
713 002220 012737 034174 001232  MOV      #HUBEO,SETUP ; LOAD PTER FOR UBE HANDLER
714 002226 000137 003056      JMP      START1     ; GO TO BEGINNING OF TEST
715
716 ;//////////////////////////////////////
717 ; RK05 QUESTION AND INITIALIZE ROUTINE
718 ;//////////////////////////////////////
719
720 002232 012737 002246 000004  QRK05: MOV      #15,2#4      ; SET UP FOR TIME OUTS
721 002240 005737 177404      TST      2#RKCS    ; SEE IF RK05 STATUS REG RESPONDS
722 002244 000405                BR        2S
723
724 002246 022626      1S:   CMP      (SP)+,(SP)+ ; RESTORE STACK
725 002250 104401 041312      TYPE     ,MSG5      ; DEVICE DOES NOT RESPOND
726 002254 000137 001726      JMP      Q1         ; GO CHOOSE ANOTHER DEVICE
727
728 002260 104401 041414      2S:   TYPE     ,MSG7      ; WHICH DRIVE SHOULD BE USED?

```

```

729
730 002264 104410
731 002266 012600
732 002270 002003
733 002272 104401 041202
734 002276 000772
735
736 002300 022700 000010
737 002304 003772
738 002306 012701 000015
739 002312 006300
740 002314 077102
741 002316 010037 001214
742
743 002322 012737 177404 001212
744 002330 012737 177404 001230
745 002336 012737 034226 001232
746 002344 013737 001214 177412
747 002352 012737 000015 177404
748 002360 005001
749 002362 032737 000100 177400
750 002370 001006
751 002372 005201
752 002374 001372
753 002376 104401 041645
754 002402 000137 001726
755
756 002406 000137 003056
757
758
759
760
761
762 002412 012737 002426 000004
763 002420 005737 176714
764 002424 000405
765
766 002426 022626
767 002430 104401 041312
768 002434 000137 001726
769
770 002440 104401 041414
771
772 002444 104410
773 002446 012600
774 002450 002003
775 002452 104401 041202
776 002456 000772
777
778 002460 022700 000010
779 002464 003772
780 002466 000300
781 002470 010037 001214
782 002474 052737 000004 001214
783 002502 005037 176722
784 002506 005037 176724

```

```

4$: RDOCT
MOV (SP)+,RO
BGE 3$
5$: TYPE MSG4
BR 4$
3$: CMP #10,RO
BLE 5$
MOV #15,R1
6$: ASL RO
SOB R1,6$
MOV RO,@CREG2
MOV #RKCS,CREG1
MOV #RKCS,EAD
MOV #RKDS,SETUP
MOV @CREG2,@RKDA
MOV #15,@RKCS
CLR R1
8$: BIT #100,@RKDS
BNE 7$
INC R1
BNE 8$
TYPE MSG13
JMP Q1
7$: JMP START1

```

```

;TYPE 0-7 <CARRIAGE RETURN>
;WAIT FOR ANS
;IS DRIVE VALID # = OR >0?
;BRANCH IF YES
;INVALID ENTRY, TRY AGAIN
;GO WAIT FOR REPLY
;IS DRIVE VALID # <?
;BRANCH IF NO
;PUT DRIVE #
;IN 3 MSB OF RO
;LOOP TILL DONE
;SAVE DISK ADDRESS REG CONTENTS WITH SELECTED
;DRIVE AND CYLINDER ADDR, SURFACE & SECTOR=0
;SAVE THE GO ADDRESS
;SAVE THE ERROR ADDRESS
;LOAD POINTER FOR RKDS HANDLER
;SET UP DRIVE #
;RESET DRIVE
;INIT COUNT
;DRIVE READY?
;BRANCH IF YES
;WAIT FOR
;DRIVE RDY
;DEVICE RDY BIT DOES NOT SET
;GO CHOOSE ANOTHER DEVICE
;GO TO FIRST TEST

```

```

;////////////////////////////////////
;RPO3 QUESTION AND INITIALIZE ROUTINE
;////////////////////////////////////
QRPO3: MOV #15,@4
TST @RPCS
BR 2$
1$: CMP (SP)+,(SP)+
TYPE MSG6
JMP Q1
2$: TYPE ,MSG7
;WHICH DRIVE SHOULD BE USED?
;TYPE 0-7 <CARRIAGE RETURN>
;WAIT FOR REPLY
;GET DRIVE # FROM STACK
;BRANCH IF DRIVE #>OR=0
;INVALID ENTRY, TRY AGAIN
;GO WAIT FOR REPLY
3$: CMP #10,RO
BLE 5$
SWAB RO
MOV RO,CREG2
BIS #4,CREG2
CLR @RPCA
CLR @RPDA
;IS DRIVE VALID #> OR=7
;BRANCH IF NO
;PUT DRIVE # IN HIGH BYTE
;SETUP CONTROL MASK WITH DRIVE # AND
;A READ OPERATION (NPR DATO)
;SETUP CYLINDER ADDRESS REG FOR 0
;SETUP DISK ADDRESS REG FOR 0 SECTOR AND TRACK

```

```

785 002512 012737 176714 001212      MOV      #RPCS,CREG1      ;SAVE THE GO ADDRESS
786 002520 012737 176714 001230      MOV      #RPCS,EAD        ;SAVE THE ERROR ADDRESS
787 002526 012737 034460 001232      MOV      #HRP03,SETUP    ;LOAD POINTER TO RPO3 HANDLER
788 002534 000137 003056                JMP      START1           ;GO TO FIRST TEST
789
790                                     ;////////////////////////////////////
791                                     ;TU10 QUESTION AND INITIALIZE ROUTINE
792                                     ;////////////////////////////////////
793
794 002540 012737 002554 000004  QTU10:  MOV      #15,2#4          ;SETUP FOR TIME OUT
795 002546 005737 172522                TST      2#MTC            ;SEE IF TU10 COMMAND REG RESPONDS
796 002552 000405                BR       2$              ;YES, BRANCH
797
798 002554 022626                1$:     CMP      (SP)+,(SP)+ ;RESTORE STACK
799 002556 104401 041312                TYPE    MSG6             ;DEVICE DOES NOT RESPOND
800 002562 000137 001726                JMP      01              ;GO CHOOSE ANOTHER DEVICE
801
802 002566 104401 041414                2$:     TYPE    ,MSG7     ;WHICH DRIVE SHOULD BE USED?
803                                     ;TYPE 0-7 (CARRIAGE RETURN)
804 002572 104410                4$:     RDOCT                ;WAIT FOR REPLY
805 002574 012600                MOV      (SP)+,RO        ;GET DRIVE # FROM STACK
806 002576 002003                BGE     3$               ;BRANCH IF DRIVE # > OR = 0
807 002600 104401 041202                5$:     TYPE    ,MSG4     ;INVALID ENTRY TRY AGAIN
808 002604 000772                BR       4$              ;WAIT FOR REPLY
809
810 002606 022700 000010                3$:     CMP      #10,RO    ;IS DRIVE VALID # < OR = 7
811 002612 003772                BLE     5$               ;BRANCH IF NO
812 002614 000300                SWAB    RO               ;PUT DRIVE # IN HIGH BYTE
813 002616 012737 010000 172522        MOV      #10000,2#MTC    ;POWER CLEAR CONTROLLER
814 002624 012701 000010                MOV      #10,R1          ;SET DELAY FOR POWER CLEAR
815 002630 077101                6$:     SOB     R1,6$     ;WAIT FOR POWER CLEAR
816 002632 012737 000016 172522        MOV      #16,2#MTC       ;SET UP TO REWIND
817 002640 050037 172522                BIS     RO,2#MTC         ;SET UP DRIVE # IN CONTROL
818 002644 012701 000777                MOV      #777,R1         ;SET UP DELAY COUNT
819 002650 077101                7$:     SOB     R1,7$     ;DELAY FOR SELECT REMOTE
820 002652 032737 000100 172520        BIT     #100,2#MTC       ;SEE IF DRIVE SELECTED
821 002660 001003                BNE     8$               ;BRANCH IF YES
822 002662 104401 041507                TYPE    ,MSG10          ;DRIVE NOT SELECTED PROPERLY
823 002666 000744                BR       5$              ;SELECT ANOTHER
824
825 002670 032737 000004 172520  8$:     BIT     #4,2#MTC    ;WRITE PROTECT ON?
826 002676 001403                BEQ     9$               ;BRANCH IF NO
827 002700 104401 041546                TYPE    ,MSG11          ;WRITE PROTECT ON
828 002704 000735                BR       5$              ;SELECT ANOTHER UNIT
829
830 002706 005237 172522                9$:     INC     2#MTC     ;REWIND TAPE
831 002712 032737 000001 172520 10$:    BIT     #1,2#MTC       ;TAPE UNIT RDY?
832 002720 001774                BEQ     10$              ;LOOP TILL IS
833 002722 012737 034714 001232        MOV      #HTU10,SETUP    ;LOAD PTER TO TU10 HANDLER
834 002730 012737 172522 001212        MOV      #MTC,CREG1     ;SAVE GO ADDRESS
835 002736 012737 172522 001230        MOV      #MTC,EAD        ;SAVE ERROR ADDRESS
836 002744 012737 040000 001214        MOV      #40000,CREG2   ;SET UP CONTROL MASK WITH DENSITY=800BPI, 7 CHANNEL
837 002752 050037 001214                BIS     RO,CREG2        ;SET DRIVE # IN MASK
838
839                                     ;NOW WRITE MIN # OF BYTES ON TAPE (24)8
840

```

```

841 002756 013737 001214 172522      MOV      CREG2,2#MTC      ;SET UP TO DO WRITE
842 002764 052737 000004 172522      BIS      #4,2#MTC        ;SET FUNCTION=WRITE
843 002772 012737 177760 172524      MOV      #20,2#MTBRC     ;WRITE (20)B BYTES
844 003000 012737 060000 172526      MOV      #BUFL,2#MTCMA   ;SETUP ADDRESS FOR XFER
845 003006 005237 172522                INC      2#MTC           ;START WRITE
846 003012 012701 177777                MOV      #177777,R1     ;SET UP FOR MAX DELAY
847 003016 032737 000001 172520 12$:  BIT      #1,2#MTS        ;UNIT DONE?
848 003024 001005                BNE     11$             ;BRANCH IF YES
849 003026 077105                SOB     R1,12$         ;LOOP TILL MAX COUNT DONE
850 003030 104401 041645                TYPE   MSG13           ;DEVICE RDY BIT DOES NOT SET
851 003034 000137 001704                JMP     Q2              ;TRY ANOTHER DEVICE
852
853 003040 005737 172522                11$:  TST      2#MTC        ;ERROR BIT SET?
854 003044 100004                BPL     START1         ;BRANCH IF NO TO FIRST TEST
855 003046 104401 041614                TYPE   MSG12           ;DEVICE ERROR BIT SET
856 003052 000137 001704                JMP     Q2              ;TRY ANOTHER DEVICE
857
858
859 003056 012737 033352 000004  START1: MOV      #UT4,2#4      ;SETUP FOR UNEXPECTED TRAPS TO VECTOR 4
860 003064 012737 033142 000114      MOV      #SUPERR,2#114  ;SET UP FOR UNEXPECTED PARITY ERRORS.
861 003072 042737 000001 177572      BIC      #1,2#MARD     ;KT OFF IF ON
862 003100 012706 001100                MOV      #STACK,SP     ;INIT STACK POINTER
863
864 003104 010046                MOV      RO,-(SP)      ;SAVE RO FOR MED INST
865 003106 076600                MED     ;GET CONTENTS OF LOG REG
866 003110 000022                .WORD   RLOG
867 003112 052700 100001                BIS      #100001,RO    ;ENABLE ERROR LOG & LOG FIRST MODE
868 003116 076600                MED     ;UNLOCK ERROR LOG
869 003120 000222                .WORD   WLOG
870 003122 012600                MOV      (SP)+,RO     ;RESTORE RO
871
872 003124 023727 001232 034046      CMP      SETUP,#HUBEN  ;IS THERE A UNIBUS EXERCISER DEVICE?
873 003132 001013                BNE     1$             ;BRANCH IF NO
874 003134 013737 001226 001172      MOV      IVEC,$TMPD    ;GET ITS VECTOR
875 003142 062737 000002 001172      ADD      #2,$TMPD      ;AND PUT A TRAP
876 003150 013777 001172 176050      MOV      $TMPD,$IVEC  ;CATCHER THERE
877 003156 005077 176010                CLR     2$TMPD
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896 003162 012737 000214 177746 1$:  TST1:  MOV      #214,2#CCR ;TURN OFF CACHE FOR SCOPE

```



```

897 003170 000004          SCOPE
898 003172 012737 004164 001234  MOV    #TST2,SKTST  ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
899 003200 032777 010000 175726  BIT    #SW12,SWR    ;INHIBIT TESTS USING KT?
900 003206 001402          BEQ    A15          ;BRANCH IF NO
901 003210 000137 004164          JMP    TST2         ;YES,GO TO NEXT TEST
902 003214 012737 004072 000004  A15:  MOV    #NSSYN,2#4   ;SET UP FOR TRAPS TO 4 DUE TO ADDRESSING ERRORS
903
904
905
906 003222 052737 000200 036034  BIS    #200,2#SKT11 ;TURN ON KT FOR $SIZE
907 003230 004737 035750          JSR    PC,$SIZE     ;SIZE MEMORY
908 003234 022737 001000 036322  CMP    #1000,$LSTBK ;IS THERE MORE THAN 16K OF MEM?
909 003242 003402          BLE    A16         ;BRANCH IF YES
910 003244 000137 004054          JMP    A17         ;NO GO EXIT TEST
911 003250 012700 100000          A16:  MOV    #100000,R0   ;SET UP R0 TO ADDRESS PAR4
912 003254 012701 077000          MOV    #77000,R1   ;INITIALIZE TEST DATA REG
913 003260 012737 077406 172310  MOV    #77406,2#KIPDR4 ;PAGE LENGTH=4K, EXPAND UP READ/WRITE
914 003266 012737 001000 172350  MOV    #1000,2#KIPAR4 ;SET UP TO TEST ADDRESS BIT 15
915 003274 005237 177572          INC    2#MMR0      ;TURN ON KT
916 003300 023737 036322 172350  A5:  CMP    $LSTBK,2#KIPAR4 ;TESTED ALL ADDRESSES?
917 003306 001401          BEQ    A3          ;BRANCH IF AT LAST ONE
918 003310 101411          BLOS  A4          ;BRANCH IF PAST LAST ADDRESS
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
003312 011046          A3:  MOV    (R0),-(SP)  ;SAVE DATA
003314 010110          MOV    R1,(R0)    ;WRITE TEST DATA IN LOC
003316 005201          INC    R1         ;CALC NEW TEST DATA
003320 006337 172350          ASL    2#KIPAR4   ;CALC NEXT TEST ADDRESS
003324 005737 172350          TST    2#KIPAR4   ;AT LAST ADDRESS?
003330 001401          BEQ    A4         ;GO TEST DATA IF PAST LAST ADDR.
003332 000762          BR    A5         ;GO SEE IF ADDR. TO BE TESTED
003334 012701 077000          A4:  MOV    #77000,R1   ;INIT. TEST DATA REG
003340 012737 001000 172350  MOV    #1000,2#KIPAR4 ;INIT PAR FOR LOWEST ADDR.
003346 023737 036322 172350  A8:  CMP    $LSTBK,2#KIPAR4 ;LOOKED AT LAST ADDRESS?
003354 001401          BEQ    A6         ;BRANCH IF AT LAST
003356 101474          BLOS  A77        ;BRANCH IF PAST ADDRESS
003360 020110          A6:  CMP    R1,(R0)    ;WAS DATA IN LOC?
003362 001007          BNE    A1         ;BRANCH IF NO TO ERROR
003364 005201          INC    R1         ;CALC. TEST DATA
003366 006337 172350          ASL    2#KIPAR4   ;CALC. NEXT TEST LOC.
003372 005737 172350          TST    2#KIPAR4   ;AT LAST ADDR.?
003376 001464          BEQ    A77        ;BRANCH IF DONE WITH HIGH ADDR.
003400 000762          BR    A8         ;LOOK AT NEXT LOCATION
003402 011037 001164          A1:  MOV    (R0),$REG3  ;SAVE BAD DATA
;ROUTINE TO CONVERT VIRTUAL ADDRESS IN R0 TO PHYSICAL ADDRESS IN R4,R5
003406 010002          MOV    R0,R2      ;GET VIRTUAL ADDRESS
003410 005003          CLR    R3         ;INIT SHIFT COUNTER
003412 006202          1$:  ASR    R2         ;SHIFT BLOCK NO. TO LSB 0-6
003414 005203          INC    R3         ;COUNT SHIFTS

```

H03

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 18
T1 TEST FA MUX AND PHYSICAL ADDRESS DRIVERS

953	003416	020327	000006				CMP	R3, #6	; ALL DONE?
954	003422	001373					BNE	1\$; BRANCH IF NO
955	003424	010204					MOV	R2, R4	; SAVE BLOCK #
956	003426	042704	177600				BIC	#177600, R4	; CALC. BLOCK #
957	003432	006202			2\$:		ASR	R2	; SHIFT ACTIVE PAGE FIELD TO LSB 1-3
958	003434	005203					INC	R3	; COUNT SHIFTS
959	003436	020327	000014				CMP	R3, #14	; ALL DONE?
960	003442	001373					BNE	2\$; BRANCH IF NO
961	003444	042702	177761				BIC	#177761, R2	; CALC. APFX2
962	003450	062702	172340				ADD	#KIPAR0, R2	; CALC. ADDR. OF PAR REFERENCING
963	003454	011202					MOV	(R2), R2	; GET (PAR)
964	003456	060204					ADD	R2, R4	; CALC. PHYSICAL BLOCK #
965	003460	010405					MOV	R4, R5	; START TO SAVE PHYSICAL ADDR. A17, A16
966	003462	005003					CLR	R3	; INIT. SHIFT COUNTER
967	003464	006205			3\$:		ASR	R5	; SHIFT ADDR BIT 17, 16 TO LSB 0, 1
968	003466	005203					INC	R3	; COUNT
969	003470	020327	000012				CMP	R3, #12	; DONE?
970	003474	001373					BNE	3\$; BRANCH IF NO
971	003476	005003					CLR	R3	; INIT SHIFT COUNTER
972	003500	006304			4\$:		ASL	R4	; SHIFT MSB TO BIT 16
973	003502	005203					INC	R3	; COUNT
974	003504	020327	000006				CMP	R3, #6	; ALL DONE?
975	003510	001373					BNE	4\$; BRANCH IF NO
976	003512	010002					MOV	R0, R2	; GET VIRTUAL ADDRESS
977	003514	042702	177700				BIC	#177700, R2	; LEAVE BLOCK COUNT IN REG
978	003520	060204					ADD	R2, R4	; HAVE R4 CONTAIN PHY. ADDR. 0-15
979	003522	010437	001162				MOV	R4, \$REG2	; SAVE LO ADD
980	003526	010537	001160				MOV	R5, \$REG1	; SAVE HI ADD
981	003532	010137	001166				MOV	R1, \$REG4	; SAVE CURRENT DATA
982	003536	012706	001100				MOV	# \$STACK, SP	; RESTORE STACK IF LOOP
983	003542	104020					ERROR	20	; ERROR: PHYSICAL ADDRESS LINE ERROR
984									; ADDRESS HELD WRONG DATA
985	003544	000137	004054				JMP	A17	; GO TO NEXT TEST
986									
987									
988									; RESTORE FLOATING '1' ADDRESSES
989	003550	012737	004000	172350	A77:		MOV	#4000, @#KIPAR4	; INIT. KIPAR1 TO RESTORE 3 LOC
990	003556	012700	100000				MOV	#100000, R0	; INIT R0 TO ADDRESS KIPAR4
991	003562	022737	004000	036322			CMP	#4000, @#SLSTBK	; WERE 3 LOC WRITTEN?
992	003570	101405					BLOS	A80	; BRANCH IF YES
993	003572	022737	002000	036322			CMP	#2000, @#SLSTBK	; WERE 2 LOC WRITTEN?
994	003600	101402					BLOS	A81	; BRANCH IF YES
995	003602	000405					BR	A82	; RESTORE LAST LOC ONLY
996	003604	012610			A80:		MOV	(SP)+, (R0)	
997	003606	012737	002000	172350	A81:		MOV	#2000, @#KIPAR4	; SET UP KIPAR4 TO RESTORE 2 LOC
998	003614	012610					MOV	(SP)+, (R0)	
999	003616	012737	001000	172350	A82:		MOV	#1000, @#KIPAR4	; SET UP KIPAR4 TO RESTORE LAST LOC
1000	003624	012610					MOV	(SP)+, (R0)	
1001									
1002									
1003									; NOW TEST ADDRESS 15, 16, 17 CAN FLOAT A '0'
1004	003626	022737	003740	036322			CMP	#3740, \$LSTBK	; ENOUGH MEM TO TEST A17?
1005	003634	003107					BGT	A17	; BRANCH IF NO
1006	003636	012701	177000				MOV	#177000, R1	; SET UP TEST DATA
1007	003642	012700	103776				MOV	#103776, R0	; ADDR. PAR4 & HAVE ALL LOW ADDRESS BITS=1
1008	003646	012737	003740	172350			MOV	#3740, @#KIPAR4	; SET UP PAR4 SO A17=0 A16, A15=1 & ALL HIGH ADDR. BITS =1

```

1009 003654 011046      MOV      (RO), -(SP)      ;SAVE DATA ON STACK
1010 003656 010110      MOV      R1, (RO)        ;LOAD TEST ADDRESS WITH DATA
1011 003660 005201      INC      R1              ;CHANGE DATA
1012 003662 022737 005740 036322  CMP      # 5740, $LSTBK   ;ENOUGH MEM TO TEST A16?
1013 003670 003006      BGT      A10            ;BRANCH IF NO
1014 003672 012737 005740 172350  MOV      # 5740, 2#KIPAR4 ;HAVE A17, A16, A15=101
1015 003700 011046      MOV      (RO), -(SP)     ;SAVE DATA
1016 003702 010110      MOV      R1, (RO)        ;LOAD TEST DATA
1017 003704 005201      INC      R1              ;CHANGE DATA
1018
1019 003706 022737 006740 036322 A10:  CMP      # 6740, $LSTBK   ;ENOUGH MEM TO TEST A15?
1020 003714 003005      BGT      A12            ;BRANCH IF NO
1021 003716 012737 006740 172350  MOV      # 6740, 2#KIPAR4
1022 003724 011046      MOV      (RO), -(SP)     ;SAVE DATA
1023 003726 010110      MOV      R1, (RO)        ;LOAD TEST DATA
1024
1025                                     ;SEE IF DATA WRITTEN PROPERLY
1026
1027 003730 012737 003740 172350 A12:  MOV      # 3740, 2#KIPAR4 ;SET UP ADDRESS
1028 003736 012701 177000      MOV      #177000, R1
1029 003742 020110      CMP      R1, (RO)        ;DATA OK?
1030 003744 001402      BEQ      A11            ;BRANCH IF YES
1031 003746 000137 003402      JMP      A1              ;REPORT ERROR
1032
1033 003752 022737 005740 036322 A11:  CMP      # 5740, $LSTBK   ;TESTING A16?
1034 003760 003034      BGT      A14            ;BRANCH IF NO TO RESTORE DATA
1035 003762 005201      INC      R1              ;UPDATE DATA
1036 003764 012737 005740 172350  MOV      # 5740, 2#KIPAR4 ;SETUP ADDRESS
1037 003772 020110      CMP      R1, (RO)        ;DATA OK?
1038 003774 001402      BEQ      A13            ;BRANCH YES
1039 003776 000137 003402      JMP      A1              ;REPORT ERROR
1040
1041 004002 022737 006740 036322 A13:  CMP      # 6740, $LSTBK   ;TESTING A15?
1042 004010 003014      BGT      A85            ;BRANCH NO TO RESTORE DATA
1043 004012 005201      INC      R1              ;UPDATE DATA
1044 004014 012737 006740 172350  MOV      # 6740, 2#KIPAR4 ;SETUP ADDRESS
1045 004022 020110      CMP      R1, (RO)        ;DATA OK?
1046 004024 001402      BEQ      A86            ;BRANCH YES
1047 004026 000137 003402      JMP      A1              ;REPORT ERROR
1048
1049                                     ;RESTORE DATA
1050
1051 004032 012610      MOV      (SP)+, (RO)      ;RESTORE 3 LOCS
1052 004034 012737 005740 172350  MOV      # 5740, 2#KIPAR4
1053 004042 012610      MOV      (SP)+, (RO)      ;RESTORE 2 LOCS
1054 004044 012737 003740 172350  MOV      # 3740, 2#KIPAR4
1055 004052 012610      MOV      (SP)+, (RO)      ;RESTORE 1 LOC
1056
1057                                     ;EXIT TEST
1058
1059 004054 042737 000001 177572 A17:  BIC      #1, 2#MMRO       ;TURN OFF KT IF ON
1060 004062 012737 033352 000004  MOV      #UT4, 2#4        ;RESTORE HANDLER FOR UNEXPECTED TRAPS
1061 004070 000435      BR       TST2            ;GO TO NEXT TEST
1062
1063                                     ;ROUTINE TO HANDLE NO SSYN ERRORS
1064

```

```

1065 004072 010046
1066 004074 076600
1067 004076 000101
1068 004100 010037 001160
1069 004104 076600
1070 004106 000102
1071 004110 010037 001162
1072 004114 012600
1073 004116 022626
1074 004120 013737 177744 001164
1075 004126 104021
1076 004130 012737 033352 000004
1077 004136 042737 000001 177572
1078
1079 004144 010046
1080 004146 076600
1081 004150 000022
1082 004152 052700 100001
1083 004156 076600
1084 004160 000222
1085 004162 012600
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103 004164 012737 000214 177746
1104 004172 000004
1105 004174 012737 000214 177746
1106 004202 012737 004320 001234
1107 004210 013737 177746 001160
1108 004216 022737 000214 001160
1109 004224 001406
1110 004226 012737 000214 001162
1111 004234 104005
1112 004236 000137 033020
1113
1114 004242 013737 177752 001160
1115 004250 001405
1116 004252 005037 001162
1117 004256 104006
1118 004260 000137 033020
1119
1120 004264 012700 060000
    
```

```

NSSYN: MOV RO,-(SP)
IS: MED
      .WORD HIADD
      MOV RO,@#SREG1
      MED
      .WORD LOADD
      MOV RO,@#SREG2
      MOV (SP)+,RO
      CMP (SP)+,(SP)+
      MOV @#EREG,@#SREG3
      ERROR 21 ;ERROR: TRAP TO VECTOR 4 WHEN TESTING PHYSICAL ADDR. LI
      MOV #UT4,@#4 ;RESTORE HANDLER FOR UNEXPECT. TRAPS
      BIC #1,@#MMRO ;TURN OFF KT
      MOV RO,-(SP) ;SAVE RO FOR MED INST
      MED ;GET CONTENTS OF LOG REG
      .WORD RLOG
      BIS #100001,RO ;ENABLE ERROR LOG & LOG FIRST MODE
      MED ;UNLOCK ERROR LOG
      .WORD WLOG
      MOV (SP)+,RO ;RESTORE RO
    
```

```

*****
*TEST 2 TEST CACHE CAN BE TURNED OFF AND HIT REG CLEARED
*
* THE CACHE IS TURNED OFF AND THE CACHE CONTROL REG
*IS CHECKED TO CONTAIN ALL 1'S FOR ALL SETTABLE BITS
*EXCEPT BIT6 (WMP).NEXT THE HIT REG (HMR) IS TESTED TO BE ALL 0'S. AFTER THIS,
*A LOW CACHE ADDRESS AND THEN A HIGH ADDRESS ARE TRIED TO
*BE MADE HITS AND THEN THE HMR IS CHECKED TO BE ALL 0'S.
*(LOW CACHE ADDRESS HAS PHYSICAL ADDRESS BIT 10=0).
*
*IF THIS TEST REPORTS A FATAL ERROR, ALL FOLLOWING TESTS
*ARE ABORTED
    
```

```

*****
†ST2: MOV #214,@#CCR ;CACHE OFF FOR SCOPE
      SCOPE
      MOV #214,@#CCR ;SET UP DATA
      MOV #TST3,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
      MOV @#CCR,@#SREG1 ;GET (CCR)
      CMP #214,@#SREG1 ;WERE BITS SET IN CCR?
      BEQ TO1L01 ;BRANCH IF YES
      MOV #214,@#SREG2 ;SAVE GOOD DATA
      ERROR 5 ;FATAL ERROR: CACHE CONTROL REG HELD WRONG DATA
      JMP $EOP ;ABORT TEST
TO1L01: MOV @#HMR,@#SREG1 ;SEE IF HIT MISS REG HAS ALL MISSES
        BEQ TO1L02 ;BRANCH IF YES
TO1L03: CLR @#SREG2 ;SAVE GOOD DATA
        ERROR 6 ;FATAL ERROR:HIT/MISS REG HELD WRONG DATA
        JMP $EOP ;ABORT TEST
TO1L02: MOV #BUFL,RO ;INITIALIZE RO TO LOW ADDRESS
    
```

K03

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 21
T2 TEST CACHE CAN BE TURNED OFF AND HIT REG CLEARED

```
1121 004270 021010
1122 004272 013737 177752 001160
1123 004300 001364
1124 004302 012700 062000
1125 004306 021010
1126 004310 013737 177752 001160
1127 004316 001355
```

```
CMP (R0), (R0) ; TRY TO MAKE LOC A HIT
MOV #HMR, SREG1 ; SEE IF MISS ON LOW ADDRESS SPACE
BNE T01L03 ; BRANCH IF GOT FALSE HIT
MOV #BUFH, R0 ; SET R0=TO HIGH ADDRESS SPACE
CMP (R0), (R0) ; TRY TO MAKE HIGH ADDRESS A HIT
MOV #HMR, SREG1 ; SEE IF MISS AT HIGH ADDRESS
BNE T01L03 ; BRANCH IF GET FALSE HIT
```

```
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
```

```
*****
*TEST 3 TEST CAN GET A HIT ON A HIGH CACHE ADDRESS AND HIT REG CAN =1
*
* THIS IS THE FIRST TEST WHERE THE HIGH HALF OF CACHE IS
*TURNED ON. THE CACHE CONTROL REG IS FIRST LOADED AND CHECKED
*TO CONTAIN THE PROPER VALUE. THEN ONE LOCATION IN CACHE
*IS MADE A HIT. THE HIT REG IS THEN TESTED TO MAKE SURE
*ITS 5 MSB CAN =1 AT THE CORRECT TIME.
*
*IF THIS TEST REPORTS A FATAL ERROR, ALL FOLLOWING TESTS
*ABORTED.
```

```
1142 004320 012737 000214 177746
1143 004326 000004
1144 004330 012737 004616 001234
1145 004336 012737 000204 177746
1146 004344 013700 177746
1147 004350 022700 000204
1148 004354 001413
1149 004356 042737 000014 177746
1150 004364 010037 001160
1151 004370 012737 000210 001162
1152 004376 104005
1153 004400 000137 033020
```

```
*****
TST3: MOV #214, #CCR ; CACHE OFF FOR SCOPE
SCOPE
MOV #TST4, SKTST ; SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
MOV #204, #CCR ; TURN ON HIGH ADDRESSES OF CACHE
MOV #CCR, R0 ; GET (CCR)
CMP #204, R0 ; WAS CACHE TURNED ON?
BEQ T02L01 ; BRANCH IF YES
BIC #14, #CCR ; TURN CACHE OFF
MOV R0, SREG1 ; SAVE BAD DATA
MOV #210, SREG2 ; SAVE GOOD DATA
15: ERROR 5 ; FATAL ERROR: CACHE CONTROL REG HELD WRONG DATA
JMP SEOP ; ABORT TEST
```

```
1155 004404 012701 177752
1156 004410 012700 062000
1157 004414 021010
1158 004416 011102
1159 004420 011103
1160 004422 011104
1161 004424 011105
1162 004426 052737 000014 177746
1163 004434 030227 000002
1164 004440 001010
1165 004442 010237 001160
1166 004446 012737 000002 001162
1167 004454 104013
1168 004456 000137 033020
```

```
T02L01: MOV #HMR, R1 ; SAVE HIT/MISS ADDRESS
MOV #BUFH, R0 ; INITIALIZE R0 TO HIGH ADDRESS
CMP (R0), (R0) ; MAKE ADDRESS A HIT
MOV (R1), R2 ; SAVE HIT-MISS REG SHIFTED ONE
MOV (R1), R3 ; SAVE HIT MISS REG SHIFTED TWO
MOV (R1), R4 ; SAVE HIT MISS REG SHIFTED THREE
MOV (R1), R5 ; SAVE HIT MISS REG SHIFTED FOUR
BIS #14, #CCR ; TURN OFF CACHE
BIT R2, #HMR1 ; DID WE GET A HIT AND WAS IT SHIFTED?
BNE T02L02 ; BRANCH IF YES
MOV R2, SREG1 ; SAVE BAD DATA
MOV #2, SREG2 ; SAVE GOOD DATA
T02L06: ERROR 13 ; FATAL ERROR:HIT/MISS REG HELD WRONG DATA
JMP SEOP ; ABORT TEST
```

```
1170 004462 030327 000004
1171 004466 001006
1172 004470 010337 001160
1173 004474 012737 000004 001162
1174 004502 000764
1175
1176 004504 030427 000010
```

```
T02L02: BIT R3, #HMR2 ; WAS DATA SHIFTED?
BNE T02L03 ; BRANCH IF YES
MOV R3, SREG1 ; SAVE BAD DATA
MOV #4, SREG2 ; SAVE GOOD DATA
BR T02L06 ; REPORT ERROR

T02L03: BIT R4, #HMR3 ; WAS DATA SHIFTED?
```

L03

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 22
T3 TEST CAN GET A HIT ON A HIGH CACHE ADDRESS AND HIT REG CAN =1

```

1177 004510 001006          BNE      T02L04      ;BRANCH IF YES
1178 004512 010437 001160    MOV      R4,$REG1    ;SAVE BAD DATA
1179 004516 012737 000010 001162  MOV      #10,$REG2   ;SAVE GOOD DATA
1180 004524 000753          BR       T02L06      ;REPORT ERROR
1181
1182 004526 030527 000020    T02L04: BIT      R5,#HMR4 ;WAS DATA SHIFTED?
1183 004532 001006          BNE      T02L05      ;BRANCH IF YES
1184 004534 010537 001160    MOV      R5,$REG1    ;SAVE BAD DATA
1185 004540 012737 000020 001162  MOV      #20,$REG2   ;SAVE GOOD DATA
1186 004546 000742          BR       T02L06      ;REPORT ERROR
1187
1188 004550 012737 000204 177746 T02L05: MOV      #204,#CCR ;TURN HALF CACHE ON
1189 004556 021010          CMP      (R0),(R0)   ;MAKE ADDRESS A HIT
1190 004560 021010          CMP      (R0),(R0)   ;SHIFT HIT 3 TIMES
1191 004562 000240          NOP
1192 004564 011102          MOV      (R1),R2    ;SHIFT HIT FOURTH TIME
1193 004566 030227 000040    BIT      R2,#MRS    ;SHIFT HIT FIFTH TIME AND SAVE
1194 004572 001011          BNE      TST4        ;WAS DATA SHIFTED?
1195 004574 052737 000014 177746  BIS      #14,#CCR    ;BRANCH IF YES TO NEXT TEST
1196 004602 010237 001160    MOV      R2,$REG1    ;TURN CACHE OFF
1197 004606 012737 000054 001162  MOV      #54,$REG2   ;SAVE BAD DATA
1198 004614 000717          BR       T02L06      ;SAVE GOOD DATA
1199
1200          ;*****
1201          ;*TEST 4      TEST FORCE MISS ON HIGH ADDRESS
1202          ;*
1203          ;*A LOCATION IS PUT IN CACHE. CACHE IS THEN TURNED OFF
1204          ;*AND THE LOCATION IS CHECKED TO BE A MISS.
1205
1206          ;*****
1207 004616 012737 000214 177746  TST4:  MOV      #214,#CCR ;TURN OFF CACHE FOR SCOPE
1208 004624 000004          SCOPE
1209 004626 012737 004712 001234    MOV      #TST5,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1210 004634 012737 000204 177746    MOV      #204,#CCR   ;TURN ON HIGH ADDRESS OF CACHE
1211 004642 012700 062000    MOV      #BUFH,R0    ;INITIALIZE R0=HIGH ADDRESS
1212 004646 021010          CMP      (R0),(R0)   ;MAKE LOC A HIT
1213 004650 052737 000014 177746  BIS      #14,#CCR    ;TURN OFF CACHE
1214 004656 005710          TST      (R0)        ;SEE IF LOC STILL A HIT
1215 004660 033727 177752 000004  BIT      #HMR,#HMR2 ;WAS IT A MISS?
1216 004666 001411          BEQ      TST5        ;BRANCH IF YES
1217 004670 013737 177746 001160  MOV      #CCR,$REG1  ;SAVE (CCR)
1218 004676 012737 000000 001162  MOV      #0,$REG2    ;SAVE PHYSICAL ADDRESS HIGH
1219 004704 010037 001164          MOV      R0,$REG3    ;SAVE PHYSICAL ADDRESS LOW
1220 004710 104012          1$:      ERROR      12 ;ERROR:FORCE MISS BIT FAILED TO CAUSE MISS.
1221
1222          ;*****
1223          ;*TEST 5      TEST CACHE TRACKS WHEN CACHE IS OFF
1224          ;*
1225          ;* A LOC IS MADE A HIT IN CACHE. CACHE IS THEN TURNED OFF
1226          ;*AND A SECOND LOC IS REFERENCED WHICH HAS AN OVERLAPPING
1227          ;*CACHE ADDRESS WITH THE FIRST ONE. CACHE IS TURNED ON
1228          ;*AND THE SECOND LOC IS TESTED TO BE A HIT (IMPLYING
1229          ;*CACHE HAS TRACKED).
1230
1231          ;*****
1232

```

```

1233 004712 012737 000214 177746
1234 004720 000004
1235 004722 012737 005044 001234
1236 004730 012737 000204 177746
1237 004736 023737 002000 002000
1238 004744 033727 177752 000004
1239 004752 001423
1240 004754 052737 000014 177746
1241 004762 005737 062000
1242 004766 012737 000204 177746
1243 004774 005737 062000
1244 005000 033727 177752 000004
1245 005006 001016
1246
1247 005010 052737 000014 177746
1248 005016 104107
1249 005020 000411
1250
1251 005022 052737 000014 177746
1252 005030 005037 001160
1253 005034 012737 002000 001162
1254 005042 104043
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266 005044 012737 000214 177746
1267 005052 000004
1268 005054 012737 005364 001234
1269 005062 012737 000204 177746
1270 005070 005737 002000
1271 005074 112737 000377 062000
1272 005102 005737 062000
1273 005106 033727 177752 000004
1274 005114 001413
1275 005116 052737 000014 177746
1276 005124 012737 000000 001160
1277 005132 012737 062000 001162
1278 005140 104007
1279 005142 000510
1280
1281 005144 005037 062000
1282 005150 112737 177777 062001
1283 005156 005737 062000
1284 005162 033727 177752 000004
1285 005170 001013
1286 005172 052737 000014 177746
1287 005200 012737 000000 001160
1288 005206 012737 062000 001162

```

```

TST5:  MOV      #214,2#CCR      ;CACHE OFF FOR SCOPE
        SCOPE
        MOV      #TST6,SKTST    ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        MOV      #204,2#CCR      ;HALF CACHE ON
        CMP      2#2000,2#2000    ;PUT DATA IN CACHE
        BIT      2#HMR,2#HMR2    ;DATA IN CACHE?
        BEQ      15              ;BRANCH IF NO TO ERROR
        BIS      #14,2#CCR        ;CACHE OFF
        TST      2#BUFH          ;REFERENCE LOC NOT IN CACHE AND SEE IF TRACK
        MOV      #204,2#CCR      ;HALF CACHE ON
        TST      2#BUFH          ;SEE IF CACHE TRACKED
        BIT      2#HMR,2#HMR2    ;HIT?
        BNE      TST6            ;;YES, GO TO NEXT TEST

        BIS      #14,2#CCR        ;CACHE OFF
        ERROR    107            ;ERROR: CACHE DID NOT TRACK WHEN FORCE MISS ON
        BR       TST6            ;;GO TO NEXT TEST

15:     BIS      #14,2#CCR        ;CACHE OFF
        CLR      $REG1          ;SAVE BAD ADDR.
        MOV      #2000,$REG2     ;SAVE BAD ADDR.
        ERROR    43            ;ERROR: ADDRESS COULD NOT BE MADE A HIT

```

```

*****
;TEST 6      TEST DATOB OPERATION
;
; A DATOB IS DONE TO AN ADDRESS NOT IN CACHE AND THEN
;THE LOC IS REFERENCED TO SEE THAT CACHE WAS NOT ALLOCATED.
;NEXT A DATOB IS DONE TO AN ODD LOC IN CACHE AND THE
;CORRECT BYTE IS CHECKED TO BE MODIFIED. THIS IS RE-
;PEATED FOR AN EVEN ADDRESS.

```

```

*****
TST6:  MOV      #214,2#CCR      ;TURN OFF CACHE FOR SCOPE
        SCOPE
        MOV      #TST7,SKTST    ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        MOV      #204,2#CCR      ;TURN ON CACHE HIGH ADDRESS
        TST      2#2000          ;MAKE LOC BUFH IN NEXT INST. A MISS
        MOV      #377,2#BUFH    ;DO DATOB TO NON-HIT LOC TO SEE IT DOESN'T GET CACHED
        TST      2#BUFH          ;SEE IF DATA PUT IN CACHE
        BIT      2#HMR,2#HMR2    ;WAS DATA A HIT?
        BEQ      T04L01         ;BRANCH IF NO
        BIS      #14,2#CCR        ;TURN OFF CACHE
        MOV      #0,$REG1        ;SAVE PHYSICAL ADDRESS HIGH
        MOV      #BUFH,$REG2     ;SAVE NO HIT PHYSICAL ADDRESS LOW
        ERROR    7              ;ERROR:DATA CACHED ON DATOB TO NO 'HIT' ADD.
        BR       TST7            ;;GO TO NEXT TEST

T04L01: CLR      2#BUFH          ;INITIALIZE LOC BUFH
        MOV      #177777,2#BUFH+1 ;DO DATOB TO A HIT LOC
        TST      2#BUFH          ;SEE IF DATA PUT IN CACHE
        BIT      2#HMR,2#HMR2    ;WAS DATA A HIT ?
        BNE      T04L02         ;BRANCH IF YES
        BIS      #14,2#CCR        ;TURN OFF CACHE
        MOV      #0,$REG1        ;SAVE PHYSICAL ADDRESS HIGH
        MOV      #BUFH,$REG2     ;SAVE PHYSICAL ADDRESS LOW

```

```

1289 005214 104010
1290 005216 000462
1291
1292 005220 022737 177400 062000 T04L02: CMP #177400,2#BUFH ;WAS DATA WRITTEN CORRECTLY?
1293 005226 001424 BEQ T04L03 ;BRANCH IF YES
1294 005230 013700 062000 MOV 2#BUFH,RO ;GET BAD DATA
1295 005234 052737 000014 177746 BIS #14,2#CCR ;TURN OFF CACHE
1296 005242 012737 000000 001160 MOV #0,$REG1 ;SAVE PHYSICAL ADDRESS HIGH
1297 005250 012737 062000 001162 MOV #BUFH,$REG2 ;SAVE PHYSICAL ADDRESS LOW
1298 005256 010037 001164 MOV RO,$REG3 ;SAVE BAD DATA
1299 005262 012737 177400 001166 MOV #177400,$REG4 ;SAVE GOOD DATA
1300 005270 104011 1S: ERROR 11 ;ERROR: CACHE DID NOT CONTAIN PROPER DATA ON DATOB
1301 005272 042737 000010 177746 BIC #10,2#CCR ;TURN CACHE ON
1302
1303 005300 005037 062000 T04L03: CLR 2#BUFH ;INITIALIZE LOCATION
1304 005304 112737 000377 062000 MOVB #377,2#BUFH ;DO DATOB TO EVEN ADDRESS
1305 005312 022737 000377 062000 CMP #377,2#BUFH ;WAS DATA WRITTEN CORRECTLY?
1306 005320 001421 BEQ TST7 ;BRANCH IF YES TO NEXT TEST
1307 005322 013700 062000 MOV 2#BUFH,RO ;GET BAD DATA
1308 005326 052737 000014 177746 BIS #14,2#CCR ;TURN CACHE OFF
1309 005334 012737 000000 001160 MOV #0,$REG1 ;SAVE PHYSICAL ADDRESS HIGH
1310 005342 012737 062000 001162 MOV #BUFH,$REG2 ;SAVE PHYSICAL ADDRESS LOW
1311 005350 010037 001164 MOV RO,$REG3 ;SAVE BAD DATA
1312 005354 012737 000377 001166 MOV #377,$REG4 ;SAVE GOOD DATA
1313 005362 104011 1S: ERROR 11 ;ERROR: CACHE DID NOT CONTAIN PROPER DATA ON DATOB.
1314
1315 ;*****
1316 ;*TEST 7 TEST DATO ALLOCATES CACHE
1317 ;*
1318 ;* A LOC IS MADE A HIT IN CACHE, THEN A DATO IS DONE TO
1319 ;*A SECOND CACHE ADDRESS WITH ADDRESS BITS A0-A10 THE SAME.
1320 ;*THE SECOND ADDRESS IS THEN CHECKED TO BE ALLOCATED IN
1321 ;*CACHE.
1322
1323 ;*****
1324 005364 012737 000214 177746 TST7: MOV #214,2#CCR ;CACHE OFF FOR SCOPE
1325 005372 000004 SCOPE
1326 005374 012737 006000 001234 MOV #TST10,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1327 005402 012737 000204 177746 MOV #204,2#CCR ;HALF CACHE ON
1328 005410 023737 002000 002000 CMP 2#2000,2#2000 ;PUT LOC IN CACHE TO MAKE NEXT REF A MISS
1329 005416 033727 177752 000004 BIT 2#HMR,#HMR2 ;HIT?
1330 005424 001422 BEQ T05L01 ;BRANCH TO ERROR IF NO
1331 005426 005037 062000 CLR 2#BUFH ;DO DATO TO A MISS ADDRESS
1332 005432 005737 062000 TST 2#BUFH ;LOC IN CACHE?
1333 005436 033727 177752 000004 BIT 2#HMR,#HMR2 ;HIT?
1334 005444 001023 BNE T05L02 ;YES, GO TO END OF TEST
1335 005446 052737 000014 177746 BIS #14,2#CCR ;CACHE OFF
1336 005454 005037 001160 CLR $REG1 ;SAVE FAILING ADDRESS
1337 005460 012737 062000 001162 MOV #BUFH,$REG2 ;SAVE FAILING ADDRESS
1338 005466 104014 ERROR 14 ;ERROR: ADDR. NOT A HIT AFTER DATO TO IT
1339 005470 000411 BR T05L02 ;GO TO END OF TEST
1340
1341 005472 052737 000014 177746 T05L01: BIS #14,2#CCR ;CACHE OFF
1342 005500 005037 001160 CLR $REG1 ;SAVE FAILING ADDR
1343 005504 012737 002000 001162 MOV #2000,$REG2 ;SAVE FAILING ADDR
1344 005512 104043 ERROR 43 ;ERROR: ADDR. COULD NOT BE MADE A HIT

```



```

1345
1346 005514 052737 000014 177746 T05L02: BIS #14, @CCR ;CACHE OFF WHEN CROSS CACHE ADDRESS BOUNDARY
1347 005522 000526 BR TST10 ;;GO TO NEXT TEST
1348
1349
1350 006000 . =6000 ;ADJUST ADDRESS SPACE FOR NEXT TEST
1351
1352

```

```

*****
*TEST 10 TEST CAN GET HIT AND FORCE MISS ON LOW CACHE ADDRESS
*
* THIS IS THE FIRST TEST WHERE LOW CACHE IS TURNED
*ON. THE CACHE CONTROL REG IS FIRST LOADED AND CHECKED
*TO CONTAIN THE PROPER VALUE. THEN ONE LOC IN LOW
*CACHE IS MADE A HIT. THE HIT IS CHECKED FOR AND THEN
*CACHE IS TURNED OFF AND THE LOC IS RETESTED TO NOW BE
*A MISS.
*
*IF THIS TEST REPORTS A FATAL ERROR, ALL FOLLOWING TESTS
*ARE ABORTED.

```

```

1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367 006000 012737 000214 177746 TST10: MOV #214, @CCR ;CACHE OFF FOR SCOPE
1368 006006 000004 SCOPE
1369 006010 012737 006166 001234 MOV #TST11, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1370 006016 012737 000210 177746 MOV #210, @CCR ;TURN ON LOW CACHE
1371 006024 013700 177746 MOV @CCR, R0 ;GET (CCR)
1372 006030 022700 000210 CMP #210, R0 ;(CCR) OK?
1373 006034 001413 BEQ 25 ;BRANCH IF YES
1374 006036 052737 000014 177746 BIS #14, @CCR ;CACHE OFF
1375 006044 010037 001160 MOV R0, $REG1 ;SAVE BAD DATA
1376 006050 012737 000210 001162 MOV #210, $REG2 ;SAVE GOOD DATA
1377 006056 104005 ERROR 5 ;FATAL ERROR: CCR HELD WRONG DATA
1378 006060 000137 033020 JMP SEOP ;ABORT PROGRAM
1379
1380 006064 012700 060000 25: MOV #BUFL, R0 ;INIT R0=LOW ADDRESS
1381 006070 021010 CMP (R0), (R0) ;MAKE LOC A HIT
1382 006072 033727 177752 000004 BIT @HMR, #HMR2 ;WAS IT A HIT?
1383 006100 001012 BNE 45 ;BRANCH IF YES
1384 006102 052737 000014 177746 BIS #14, @CCR ;CACHE OFF
1385 006110 005037 001160 CLR $REG1 ;SAVE ADDRESS
1386 006114 012737 060000 001162 MOV #BUFL, $REG2 ;SAVE ADDRESS
1387 006122 104043 ERROR 43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT
1388 006124 000420 BR TST11 ;;GO TO NEXT TEST
1389
1390 006126 052737 000014 177746 45: BIS #14, @CCR ;CACHE OFF
1391 006134 005710 TST (R0) ;SEE IF LOC STILL A HIT
1392 006136 033727 177752 000004 BIT @HMR, #HMR2 ;WAS IT A MISS?
1393 006144 001410 BEQ TST11 ;BRANCH IF YES
1394 006146 013737 177746 001160 MOV @CCR, $REG1 ;SAVE (CCR)
1395 006154 005037 001162 CLR $REG2 ;SAVE ADDRESS
1396 006160 010037 001164 MOV R0, $REG3 ;SAVE ADDRESS
1397 006164 104012 ERROR 12 ;ERROR: FORCE MISS BIT FAILED TO CAUSE MISS
1398
1399
1400

```

```

*****
*TEST 11 TEST OF TAG ADDRESS COMPATOR

```

1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456

*** THIS TEST USES ONE LOC IN CACHE AND LOADS IT WITH
*** VARIOUS TAG ADDRESSES. A GROUP OF MEMORY REFERENCES
*** ARE MADE FOR EACH TAG ADDRESS AND IT IS DETERMINED
*** WHETHER EACH REFERENCE WILL BE A HIT OR A MISS. THE
*** LOW ADDRESS COMPARATOR FOR BITS A11-A14 IS TESTED FIRST.
*** A TAG ADDRESS IS LOADED AND THEN ALL POSSIBLE COMBINATIONS
*** OF MEMORY ADDRESSES TO THAT CHIP ARE MADE. ALL TAG
*** COMBINATIONS ARE TRIED IN THIS MANNER FOR THESE LOW
*** ADDRESSES. THE HIGH ADDRESS COMPARATOR FOR BITS A15-
*** A17 IS HELD CONSTANT DURING THIS TIME. THE SAME PRO-
*** CEDURE IS REPEATED FOR THIS HIGH ADDRESS COMPARATOR
*** WHILE THE LOW ONE IS HELD CONSTANT. THE COMPARATOR
*** TEST IS LIMITED TO THE MEMORY SIZE AVAILABLE.
*** KIPAR4 CONTAINS THE TAG ADDRESS BEING TESTED. KIPAR5
*** CONTAINS THE MEMORY REFERENCE ADDRESS BEING MADE. IF
*** INHIBIT TESTS USING KT SWITCH IS SET (SW12), THIS TEST
*** IS INHIBITED.

```
*****  
TST11: MOV      #214, @CCR          ; TURN OFF CACHE FOR SCOPE  
        SCOPE  
        MOV      @TST12, SKTST     ; SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR  
        BIT      @SW12, @SWR      ; INHIBIT TESTS USING KT?  
        BEQ     2$                ; BRANCH IF NO  
        JMP     @TST12             ; GO TO NEXT TEST  
2$:     BIS      @200, @SKT11      ; TURN ON KT FOR MEM SIZING  
        JSR     PC, @SIZE         ; SIZE MEM  
        MOV     @KIPAR4, R0        ; SET UP TO  
        MOV     @KIPDR4, R1        ; INIT KIPDR4, 5 & KIPAR4, 5  
1$:     CLR     (R0)+              ; FOR TESTING  
        MOV     @77406, (R1)+     ; PAGE LENGTH=4K, EXPAND UP, READ/WRITE  
        CMP     R1, @KIPDR6       ; KT SET UP?  
        BNE    1$                 ; BRANCH IF NO  
        BR     T06L12             ; GO TO START OF TEST  
  
T06L01: TST     @KIPAR5            ; PAST MAX PARS?  
        BEQ     T06L03            ; BRANCH IF YES TO CHOOSE NEXT TAG ADDRESS  
T06L12: CMP     @KIPAR5, @SLSTBK  ; REFERENCED ALL POSSIBLE ADDRS. FOR THIS COMPARATOR?  
        BLE    T06L02            ; BRANCH IF NO  
T06L03: CMP     @KIPAR4, #1000    ; TESTED COMPARATOR FOR ADDRESS BITS 15,16,17?  
        BLT    T06L05            ; BRANCH IF NO  
        ADD     @1000, @KIPAR4    ; TEST NEXT ADDRESS BIT OF HIGH ADDR. COMP.  
        BR     T06L06  
  
T06L05: ADD     #40, @KIPAR4      ; TEST NEXT ADDRESS BIT OF LOW ADDR. COMP.  
T06L06: TST     @KIPAR4          ; PAST MAX TAG ADDRESS?  
        BEQ     T06L04            ; GO TO END OF TEST IF YES  
        CMP     @KIPAR4, @SLSTBK  ; HAVE ALL POSSIBLE TAG INPUTS TO COMPARATOR BEEN DONE  
        BGE    T06L04            ; GO TO END OF TEST IF YES  
        CMP     @KIPAR4, #1000    ; ARE WE TESTING THE HIGH ADDRESS COMPARATOR?  
        BGE    T06L07            ; BRANCH IF YES  
        CLR     @KIPAR5          ; INIT PARS TO TEST LOW ADDR. COMP.  
        BR     T06L02            ; GO TEST COMPARATOR  
  
T06L07: MOV     #1000, @KIPAR5   ; INIT. PARS TO TEST HIGH ADDR. COMP.
```

```

1457 006366 052737 000014 177746 T06L02: BIS #14,2#CCR ;TURN CACHE OFF
1458 006374 012737 120000 001172 MOV #120000,$TMP0 ;START CALC. OF PHYSICAL
1459 006402 004737 033434 JSR PC,VIP ;ADDRESS REFERENCING AND
1460 006406 013700 172350 MOV 2#KIPAR4,RO ;START CALC OF TAG ADDRESS TESTING
1461 006412 005001 CLR R1 ;GET TAG FIELD TO 7 LSB RO
1462 006414 006200 1S: ASR RO ;GET TAG FIELD TO 7 LSB RO
1463 006416 005201 INC R1 ;GET TAG FIELD TO 7 LSB RO
1464 006420 020127 000005 CMP R1,#5 ;GET TAG FIELD TO 7 LSB RO
1465 006424 001373 BNE 1S ;GET TAG FIELD TO 7 LSB RO
1466 006426 010037 001164 MOV RO,$REG3 ;SAVE TAG IN CASE OF ERROR
1467
1468 006432 052737 000001 177572 T06L08: BIS #1,2#MMRO ;TURN ON KT
1469 006440 012737 000210 177746 MOV #210,2#CCR ;TURN ON HALF OF CACHE ON
1470 006446 023737 172350 172352 CMP 2#KIPAR4,2#KIPARS ;WILL REFERENCE BE A HIT
1471 006454 001422 BEQ T06L09 ;BRANCH IF YES
1472 006456 023737 100000 120000 CMP #100000,2#120000 ;LOAD ADDRESS IN TAG FIELD & THEN REFERENCE IT
1473 006464 033727 177752 000004 BIT 2#MMR,2#MMR2 ;WAS REFERENCE A MISS?
1474 006472 001435 BEQ T06L10 ;BRANCH IF YES
1475 006474 052737 000014 177746 BIS #14,2#CCR ;TURN OFF CACHE
1476 006502 012737 006440 001110 MOV #T06L08,2#SLPERR ;INIT. FOR LOOP ON ERROR
1477 006510 104022 ERROR 22 ;ERROR: TEST OF ADDR. COMP. FAILED TO BE MISS
1478 006512 042737 000001 177572 BIC #1,2#MMRO ;TURN OFF KT
1479 006520 000442 BR T5T12 ;GO TO NEXT TEST
1480
1481 006522 023737 100000 120000 T06L09: CMP #100000,2#120000 ;LOAD ADDRESS IN TAG FIELD & THEN REFERENCE IT
1482 006530 033727 177752 000004 BIT 2#MMR,2#MMR2 ;WAS REF. A HIT?
1483 006536 001013 BNE T06L10 ;BRANCH IF YES
1484 006540 052737 000014 177746 BIS #14,2#CCR ;TURN OFF CACHE FOR ERROR REPORT
1485 006546 012737 006440 001110 MOV #T06L08,2#SLPERR ;SETUP RETURN FOR LOOP ON ERROR
1486 006554 104023 ERROR 23 ;ERROR: TEST OF ADDR. COMP. FAILED TO BE HIT
1487 006556 042737 000001 177572 BIC #1,2#MMRO ;TURN OFF KT
1488 006564 000420 BR T5T12 ;GO TO NEXT TEST
1489
1490 006566 023727 172352 000740 T06L10: CMP 2#KIPARS,#740 ;REFERENCED ADDRESSES OF LOWER ADDR. COMP.?
1491 006574 001640 BEQ T06L03 ;BRANCH IF YES
1492 006576 002404 BLT T06L11 ;BRANCH IF PARS STILL REF. LOW ADDR. COMP.
1493 006600 062737 001000 172352 ADD #1000,2#KIPARS ;ADDRESS NEXT LOC FOR HIGH ADDR. COMPARATOR
1494 006606 000624 BR T06L01 ;SEE IF DONE
1495 006610 062737 000040 172352 T06L11: ADD #40,2#KIPARS ;ADDRESS NEXT LOC FOR LOW ADDR. COMP.
1496 006616 000620 BR T06L01 ;SEE IF DONE
1497
1498 006620 042737 000001 177572 T06L04: BIC #1,2#MMRO ;TURN KT OFF
1499
1500 ;*****
1501 ;*TEST 12 TEST FORCE MISS LOCKS OUT PARITY ERRORS & CCR WWP CAN =1
1502 ;*
1503 ;* THIS IS THE FIRST TEST WHERE WRITE WRONG PARITY AND
1504 ;*THE CACHE PARITY TRAP IS EXERCISED. FIRST THE WWP IS
1505 ;*SET AND THE CACHE CONTROL REG IS CHECKED TO CONTAIN THE
1506 ;*PROPER VALUE. A PARITY TRAP IS THEN FORCED AND TESTED
1507 ;*FOR. THE LOCATION IS REMITTEN WITH WRONG PARITY AND
1508 ;*THEN THE CACHE IS TURNED OFF. THE LOCATION IS REFERENCED
1509 ;*AND NO PARITY TRAP WHEN FORCE MISS IS ON IS CHECKED FOR.
1510 ;*****
1511
1512 006626 012737 000214 177746 T5T12: MOV #214,2#CCR ;TURN OFF CACHE

```

E04

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 28
T12 TEST FORCE MISS LOCKS OUT PARITY ERRORS & CCR WWP CAN =1

1513	006634	000004			SCOPE		
1514	006636	012737	007140	001234	MOV	#TST13,SKTST	;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1515	006644	012737	006770	000114	MOV	#T09L01,@PVEC	;SETUP PARITY TRAP HANDLER
1516	006652	012737	000310	177746	MOV	#310,@CCR	;TURN ON HALF OF CACHE & WWP
1517	006660	013700	177746		MOV	@CCR,RO	
1518	006664	020027	000310		CMP	RO,#310	;WERE BITS SET IN CCR?
1519	006670	001414			BEG	T09L02	;BRANCH IF YES
1520	006672	012737	000014	177746	MOV	#14,@CCR	;TURN CACHE OFF
1521	006700	010037	001160		MOV	RO,\$REG1	;SAVE BAD DATA
1522	006704	012737	000310	001162	MOV	#310,\$REG2	;SAVE GOOD DATA
1523	006712	104026			ERROR	26	;ERROR: CACHE CONTROL REG HELD WRONG DATA
1524	006714	012737	000310	177746	MOV	#310,@CCR	;TURN ON HALF OF CACHE & WWP
1525							
1526	006722	005037	060000		T09L02: CLR	@BUFL	;WRITE WRONG PARITY IN 1 LOC
1527	006726	012737	000210	177746	MOV	#210,@CCR	;WWP OFF
1528	006734	005737	060000		TST	@BUFL	;SEE IF GET PARITY TRAP
1529							
1530							
1531							;RID CACHE OF BAD PARITY
1532	006740	012737	000214	177746	MOV	#214,@CCR	;CACHE OFF IF ON
1533	006746	004737	035134		JSR	PC,SWEEP	;GO PURGE CACHE
1534							
1535							
1536	006752	005037	001160		CLR	\$REG1	;SAVE ADDRESS
1537	006756	012737	060000	001162	MOV	@BUFL,\$REG2	;SAVE ADDRESS
1538	006764	104042			ERROR	42	;ERROR: NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PARIT
1539	006766	000450			BR	T09L06	;GO TO END OF TEST
1540							
1541	006770				T09L01:		
1542							
1543							;RID CACHE OF BAD PARITY
1544	006770	012737	000214	177746	MOV	#214,@CCR	;CACHE OFF IF ON
1545	006776	004737	035134		JSR	PC,SWEEP	;GO PURGE CACHE
1546							
1547							
1548							
1549	007002	010046			MOV	RO,-(SP)	;SAVE RO FOR MED INST
1550	007004	076600			MED		;GET CONTENTS OF LOG REG
1551	007006	000022			.WORD	RLOG	
1552	007010	052700	100001		BIS	#100001,RO	;ENABLE ERROR LOG & LOG FIRST MODE
1553	007014	076600			MED		;UNLOCK ERROR LOG
1554	007016	000222			.WORD	WLOG	
1555	007020	012600			MOV	(SP)+,RO	;RESTORE RO
1556							
1557	007022	022626			CMP	(SP)+,(SP)+	;RESTORE STACK
1558	007024	012737	007072	000114	MOV	#T09L03,@PVEC	;SET UP PARITY TRAP HANDLER
1559	007032	012737	000310	177746	MOV	#310,@CCR	;TURN HALF OF CACHE ON & WWP
1560	007040	005037	060000		CLR	@BUFL	;WRITE WRONG PARITY IN ONE LOC
1561	007044	012737	000214	177746	MOV	#214,@CCR	;CACHE OFF
1562	007052	005737	060000		TST	@BUFL	;SEE IF SEE GET PARITY TRAP
1563							
1564	007056				T09L04:		
1565							
1566							;RID CACHE OF BAD PARITY
1567	007056	012737	000214	177746	MOV	#214,@CCR	;CACHE OFF IF ON
1568	007064	004737	035134		JSR	PC,SWEEP	;GO PURGE CACHE

```

1569
1570
1571 007070 000407          BR      T09L06          ;GO TO END OF TEST
1572
1573 007072          T09L03:
1574
1575          ;RID CACHE OF BAD PARITY
1576 007072 012737 000214 177746  MOV     #214,2#CCR      ;CACHE OFF IF ON
1577 007100 004737 035134          JSR     PC,SWEEP        ;GO PURGE CACHE
1578
1579
1580 007104 022626          CMP     (SP)+,(SP)+     ;RESTORE STACK
1581 007106 104024          ERROR   24             ;ERROR: FORCE MISS DID NOT INHIBIT PARITY ERRORS
1582
1583 007110          T09L06:
1584
1585 007110 010046          MOV     RO,-(SP)        ;SAVE RO FOR MED INST
1586 007112 076600          MED     ;GET CONTENTS OF LOG REG
1587 007114 000022          .WORD  RLOG
1588 007116 052700 100001     BIS     #100001,RO      ;ENABLE ERROR LOG & LOG FIRST MODE
1589 007122 076600          MED     ;UNLOCK ERROR LOG
1590 007124 000222          .WORD  WLOG
1591 007126 012600          MOV     (SP)+,RO       ;RESTORE RO
1592
1593 007130 012737 033142 000114  MOV     #UPERR,2#PVEC   ;RESTORE HANDLER FOR UNEXPECTED PARITY ERRORS
1594 007136 000400          BR      TST13          ;;GO TO NEXT TEST
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617

```

```

*****
*TEST 13      TEST OF TAG PARITY GENERATOR/CHECKER
*
* THIS TEST INITIALLY SIZES MEMORY TO DETERMINE THE
*MAXIMUM TESTABLE ADDRESS. KIPAR4 IS SETUP TO WRITE ALL
*TAG COMBINATIONS UP TO THE MAX ADDRESS INTO ONE CACHE
*LOCATION. FIRST, THE LOCATION IS WRITTEN WITH WRONG
*PARITY FOR ALL THE TAG COMBINATIONS AND A PARITY TRAP
*IS FORCED AND TESTED FOR. AFTER EACH TRAP, THE PROGRAM
*CHECKS THAT THE TRAP WAS FROM THE TAG FIELD AND THAT
*THE TAG CONTENTS (FROM ERROR LOG) WAS WHAT WAS WRITTEN.
*THIS LATTER CHECK IS DONE PRIMARILY TO ENSURE THAT THE
*TRAP WAS BECAUSE WRONG PARITY WAS WRITTEN AND NOT DUE
*TO A FAILING LOCATION.
* SECOND, THE LOCATION IS WRITTEN WITH GOOD PARITY FOR
*ALL TAG COMBINATIONS. THE LOC IS REFERENCED AND ANY
*PARITY ERROR IS DETECTED AND REPORTED.
* IF INHIBIT TESTS USING KT SWITCH (SW12) IS SET,
*THIS TEST IS INHIBITED.
*****

```

```

1618 007140 012737 000214 177746  TST13: MOV     #214,2#CCR      ;TURN CACHE OFF FOR SCOPE
1619 007146 000004          SCOPE
1620 007150 012737 010230 001234  MOV     #TST14,SKTST    ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1621 007156 032777 010000 171750  BIT     #SW12,2#SWR     ;INHIBIT TEST USING KT11?
1622 007164 001402          BEQ     15             ;BRANCH IF NO
1623 007166 000137 010230          JMP     2#TST14        ;GO TO NEXT TEST
1624 007172 052737 000200 036034  15:    BIS     #200,2#SKT11    ;TURN ON KT FOR $SIZE

```

```

1625 007200 004737 035750          JSR    PC,$SIZE          ;SIZE MEMORY
1626 007204 012737 007354 000114  MOV    #T07L01,#PVEC    ;SET UP TO HANDLE PARITY TRAPS
1627 007212 012737 077406 172310  MOV    #77406,#KIPDR4  ;PAGE LENGTH=4K, EXPAND UP, READ/WRITE
1628 007220 005037 172350          CLR    #KIPAR4          ;INIT PAR
1629 007224 052737 000001 177572  BIS    #1,#MMRO         ;TURN KT ON
1630 007232 023737 172350 036322  T07L04: CMP    #KIPAR4,#$SLSTBK ;TESTED ALL POSSIBLE ADDRESSES?
1631 007240 003402          BLE    1$              ;BRANCH IF NO TO CONTINUE
1632 007242 000137 007650          JMP    T07L02          ;TEST GOOD PARITY GEN.
1633 007246 012737 000310 177746  1$:   MOV    #310,#CCR      ;TURN HALF OF CACHE ON & WMP
1634
1635 007254 013737 100000 100000  T07L03: MOV    #100000,#100000 ;WRITE WRONG PARITY IN LOC
1636 007262 012737 000210 177746  MOV    #210,#CCR       ;WMP OFF
1637 007270 005737 100000          TST    #100000         ;FORCE A PARITY ERROR
1638
1639
1640          ;RID CACHE OF BAD PARITY
1641 007274 012737 000214 177746  MOV    #214,#CCR       ;CACHE OFF IF ON
1642 007302 004737 035134          JSR    PC,$SWEEP        ;GO PURGE CACHE
1643
1644
1645 007306 012737 100000 001172  MOV    #100000,$TMP0    ;GET ADDRESS JUST TESTED
1646 007314 004737 033434          JSR    PC,$VIP          ;CALC ITS PHYSICAL ADDRESS
1647 007320 013737 172350 001172  MOV    #KIPAR4,$TMP0    ;GET PAR FOR TAG CALC.
1648 007326 004737 033606          JSR    PC,$TAG          ;CALC WHAT TAG CONTENTS SHOULD BE
1649 007332 013737 001172 001164  MOV    $TMP0,$REG3      ;SAVE (TAG) SHOULD BE
1650 007340 012737 007232 001110  MOV    #T07L04,#$SLPERR ;SET UP RETURN FOR LOOP ON ERROR
1651 007346 104027          ERROR  27              ;ERROR: TEST OF TAG PARITY GENERATOR/CHECKER FAILED
1652          ;
1653          ;
1654 007350 000137 010214          JMP    #T07L05         ;GO TO END OF TEST
1655
1656 007354          T07L01:
1657
1658          ;RID CACHE OF BAD PARITY
1659 007354 012737 000214 177746  MOV    #214,#CCR       ;CACHE OFF IF ON
1660 007362 004737 035134          JSR    PC,$SWEEP        ;GO PURGE CACHE
1661
1662
1663
1664 007366 010046          MOV    RO,-($P)        ;SAVE RO FOR MED INST
1665 007370 076600          MED          ;GET CONTENTS OF LOG REG
1666 007372 000022          .WORD  RLOG
1667 007374 052700 100001  BIS    #100001,RO      ;ENABLE ERROR LOG & LOG FIRST MODE
1668 007400 076600          MED          ;UNLOCK ERROR LOG
1669 007402 000222          .WORD  WLOG
1670 007404 012600          MOV    ($P)+,RO       ;RESTORE RO
1671
1672 007406 022626          CMP    ($P)+,($P)+     ;RESTORE STACK
1673 007410 032737 000040 177744  BIT    #40,#$REG       ;TRAP DUE TO PARITY ERROR IN TAG?
1674 007416 001040          BNE    T07L06         ;BRANCH IF YES
1675 007420 076600          MED          ;GET LOG INFORMATION
1676 007422 000102          .WORD  LOADD
1677 007424 010037 001162  MOV    RO,$REG2        ;SAVE INFORMATION
1678 007430 076600          MED          ;GET LOG INFOR FOR PHY. ADDR. A17,A16
1679 007432 000101          .WORD  RSER
1680 007434 000300          SWAB   RO             ;PUT PHY. ADDR A17, A16 IN LOW BYTE

```

```

1681 007436 042700 177776      BIC      #177776,RO      ;ONLY LOOK AT A17, A16
1682 007442 010037 001160      MOV      RO,$REG1    ;SAVE ADDRESS
1683 007446 076600                MED                ;GET TAG LOG INFO.
1684 007450 000107                .WORD      RTAG
1685 007452 000300                SWAB      RO        ;PUT TAG IN LOW BYTE
1686 007454 042700 177400      BIC      #177400,RO  ;LOOK AT TAG ONLY
1687 007460 010037 001164      MOV      RO,$REG3    ;SAVE TAG
1688 007464 013737 172350 001172      MOV      @#KIPAR4,$STMP0 ;GET PAR FOR TAG CALC.
1689 007472 004737 033606      JSR      PC,TAG      ;FIND GOOD CONTENTS OF TAG
1690 007476 013737 001172 001166      MOV      $STMP0,$REG4 ;SAVE GOOD DATA
1691 007504 012737 007232 001110      MOV      @T07L04,@#SLPERR ;SET UP RETURN FOR ERROR LOOP
1692 007512 104030                ERROR      30        ;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
1693                                ;DID NOT GET PARITY TRAP FROM TAG FIELD
1694                                ;WHEN WROTE WRONG PARITY
1695 007514 000137 010214                JMP      @#T07L05    ;GO TO END OF TEST
1696
1697 007520 013737 172350 001172 T07L06: MOV      @#KIPAR4,$STMP0 ;GET PAR FOR TAG CALC.
1698 007526 004737 033606      JSR      PC,TAG      ;CALC WHAT TAG SHOULD BE
1699 007532 076600                MED                ;GET TAG LOG INFO.
1700 007534 000107                .WORD      RTAG
1701 007536 000300                SWAB      RO        ;PUT TAG IN LOW BYTE
1702 007540 042700 177400      BIC      #177400,RO  ;LOOK AT TAG ONLY
1703 007544 020037 001172      CMP      RO,$STMP0   ;DATA OK?
1704 007550 001432                BEQ      T07L07      ;BRANCH IF YES
1705 007552 010037 001164      MOV      RO,$REG3    ;SAVE TAG
1706 007556 076600                MED                ;GET LOG INFORMATION
1707 007560 000102                .WORD      LOADD
1708 007562 010037 001162      MOV      RO,$REG2    ;SAVE INFORMATION
1709 007566 076600                MED                ;GET LOG INFOR FOR PHY. ADDR. A17,A16
1710 007570 000101                .WORD      RSER
1711 007572 000300                SWAB      RO        ;PUT PHY. ADDR A17, A16 IN LOW BYTE
1712 007574 042700 177776      BIC      #177776,RO  ;ONLY LOOK AT A17, A16
1713 007600 010037 001160      MOV      RO,$REG1    ;SAVE ADDRESS
1714 007604 013737 001172 001166      MOV      $STMP0,@#SREG4 ;SAVE GOOD DATA
1715 007612 012737 007232 001110      MOV      @T07L04,@#SLPERR ;SET UP RETURN FOR ERROR LOOP
1716 007620 104031                ERROR      31        ;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
1717                                ;TAG FIELD HELD WRONG DATA ON PARITY TRAP
1718 007622 123727 001103 000003      CMPB     @#SERFLG,#3 ;MORE THAN THREE ERRORS?
1719 007630 101402                BLOS     T07L07      ;BRANCH IF NO
1720 007632 000137 010214                JMP      T07L05      ;GO TO END OF TEST
1721
1722 007636 062737 000040 172350 T07L07: ADD      #40,@#KIPAR4   ;CALC NEXT TAG ADDRESS TO TEST
1723 007644 000137 007232                JMP      T07L04      ;CONTINUE TEST
1724
1725                                T07L02:
1726
1727                                ;RID CACHE OF BAD PARITY
1728 007650 012737 000214 177746      MOV      #214,@#CCR  ;CACHE OFF IF ON
1729 007656 004737 035134      JSR      PC,$SWEEP   ;GO PURGE CACHE
1730
1731
1732 007662 012737 007734 000114      MOV      @T07L08,@#PVEC ;SET UP FOR PARITY ERRORS
1733 007670 005037 172350                CLR      @#KIPAR4    ;INIT ADDRESSES
1734 007674 023737 172350 036322 T07L09: CMP      @#KIPAR4,@#SLSTBK ;TESTED ALL POSSIBLE ADDRESSES?
1735 007702 003144                BGT      T07L05      ;YES GO TO END OF TEST
1736 007704 012737 000210 177746      MOV      #210,@#CCR  ;TURN HALF CACHE ON

```

1737	007712	013737	100000	100000		MOV	2#100000,2#100000	;GENERATE PARITY IN CACHE
1738	007720	005737	102000			TST	2#102000	;CHECK PARITY IN CACHE
1739	007724	062737	000040	172350		ADD	2#40,2#KIPAR4	;CALC NEXT TAG ADDRESS TO TEST
1740	007732	000760				BR	T07L09	;CONTINUE TEST
1741								
1742	007734				T07L08:			
1743								
1744								;RID CACHE OF BAD PARITY
1745	007734	012737	000214	177746		MOV	2#214,2#CCR	;CACHE OFF IF ON
1746	007742	004737	035134			JSR	PC,SWEEP	;GO PURGE CACHE
1747								
1748								
1749								
1750	007746	010046				MOV	RO,-(SP)	;SAVE RO FOR MED INST
1751	007750	076600				MED		;GET CONTENTS OF LOG REG
1752	007752	000022				.WORD	RLOG	
1753	007754	052700	100001			BIS	2#100001,RO	;ENABLE ERROR LOG & LOG FIRST MODE
1754	007760	076600				MED		;UNLOCK ERROR LOG
1755	007762	000222				.WORD	WLOG	
1756	007764	012600				MOV	(SP)+,RO	;RESTORE RO
1757								
1758	007766	022626				CMP	(SP)+,(SP)+	;RESTORE STACK
1759	007770	076600				MED		;GET LOG INFOR FOR PHY. ADDR. A17,A16
1760	007772	000101				.WORD	RSER	
1761	007774	000300				SWAB	RO	;PUT PHY. ADDR A17, A16 IN LOW BYTE
1762	007776	042700	177776			BIC	2#177776,RO	;ONLY LOOK AT A17, A16
1763	010002	010037	001160			MOV	RO,\$REG1	;SAVE ADDRESS
1764	010006	076600				MED		;GET LOG INFORMATION
1765	010010	000102				.WORD	LOADD	
1766	010012	010037	001162			MOV	RO,\$REG2	;SAVE INFORMATION
1767	010016	032737	000040	177744		BIT	2#40,2#EREG	;ERROR DUE TO TAG ERROR?
1768	010024	001424				BEQ	T07L10	;BRANCH IF NO
1769	010026	076600				MED		;GET TAG LOG INFO.
1770	010030	000107				.WORD	RTAG	
1771	010032	000300				SWAB	RO	;PUT TAG IN LOW BYTE
1772	010034	042700	177400			BIC	2#177400,RO	;LOOK AT TAG ONLY
1773	010040	010037	001164			MOV	RO,\$REG3	;SAVE TAG
1774	010044	013737	172350	001172		MOV	2#KIPAR4,\$TMP0	;GET PAR FOR TAG CALC.
1775	010052	004737	033606			JSR	PC,TAG	;CALC GOOD DATA
1776	010056	013737	001172	001166		MOV	\$TMP0,2#\$REG4	;SAVE GOOD DATA
1777	010064	012737	007674	001110		MOV	2#T07L09,2#\$SLPERR	;SET UP FOR ERROR LOOP
1778	010072	104034				ERROR	34	;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
1779								;PARITY ERROR OCCURRED IN TAG FIELD
1780	010074	000447				BR	T07L05	;GO TO END OF TEST
1781								
1782	010076	032737	000100	177744	T07L10:	BIT	2#100,2#EREG	;ERROR IN LOW BYTE?
1783	010104	001414				BEQ	T07L11	;BRANCH IF NO
1784	010106	076600				MED		;GET LOG INFORMATION
1785	010110	000106				.WORD	CDL	
1786	010112	010037	001164			MOV	RO,\$REG3	;SAVE INFORMATION
1787	010116	013737	102000	001166		MOV	2#102000,2#\$REG4	;SAVE GOOD DATA
1788	010124	012737	007674	001110		MOV	2#T07L09,2#\$SLPERR	;INIT LOOP ON ERROR RETURN
1789	010132	104033				ERROR	33	;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
1790								;PARITY ERROR IN LOW BYTE OF DATA
1791	010134	000427				BR	T07L05	;GO TO END OF TEST
1792								


```

1793 010136 032737 000200 177744 T07L11: BIT      #200,2#EREG      ;ERROR IN HIGH BYTE?
1794 010144 001414                BEQ      T07L12      ;BRANCH IF NO
1795 010146 076600                MED                        ;GET LOG INFORMATION
1796 010150 000106                .WORD   CDH
1797 010152 010037 001164                MOV     RD,$REG3      ;SAVE INFORMATION
1798 010156 013737 102000 001166                MOV     2#102000,$REG4 ;SAVE GOOD DATA
1799 010164 012737 007674 001110                MOV     #T07L09,2#SLPERR ;SET UP LOOP ON ERROR
1800 010172 104032                ERROR   32           ;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
1801                                ;PARITY ERROR IN HIGH BYTE OF DATA
1802 010174 000407                BR      T07L05      ;GO TO END OF TEST
1803
1804 010176 016637 177774 001164 T07L12: MOV     -4(SP),$REG3 ;SAVE PC OF ERROR
1805 010204 012737 007674 001110                MOV     #T07L09,2#SLPERR ;SET UP FOR ERROR LOOP
1806 010212 104001                ERROR   1           ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
1807
1808 010214 042737 000001 177572 T07L05: BIC     #1,2#MMRD ;TURN KT OFF
1809 010222 012737 033142 000114                MOV     #SUPERR,2#114 ;RESTORE UNEXPECTED PARITY ERROR HANDLER
1810

```

```

1811 ;*****
1812 ;TEST 14 TEST OF DATA PARITY GENERATOR/CHECKER
1813 ;
1814 ;WRONG PARITY IS WRITTEN INTO ONE BYTE OF ONE LOCATION
1815 ;IN THE CACHE DATA FIELD VIA A DATOB. THE LOC IS REFERENCED
1816 ;AND THE PARITY TRAP IS CHECKED FOR. THE TRAP FROM THE
1817 ;CORRECT BYTE IS THEN TESTED. THIS PROCEDURE IS REPEATED
1818 ;FOR THE OTHER BYTE. AFTER THIS, WRONG PARITY IS WRITTEN
1819 ;FOR ALL 8 BIT COMBINATIONS IN BOTH THE LOW AND HIGH
1820 ;BYTE SIMULTANEOUSLY FOR ONE LOC. AFTER EACH DATA PATTERN
1821 ;IS WRITTEN (RD CONTAINS DATA PATTERN) A TRAP IS FORCED
1822 ;AND THE PROGRAM CHECKS THAT THE TRAP WAS FROM BOTH HIGH
1823 ;& LOW BYTES.
1824 ;FOLLOWING THIS ALL 8 BIT DATA PATTERNS FOR BOTH THE
1825 ;HIGH & LOW BYTE ARE WRITTEN WITH GOOD PARITY IN ONE
1826 ;CACHE LOC. THE LOCATION IS REFERENCED AND ANY DATA
1827 ;PARITY ERROR IS REPORTED.
1828

```

```

1829 ;*****
1830 010230 012737 000214 177746 TST14: MOV     #214,2#CCR ;TURN CACHE OFF FOR SCOPE
1831 010236 000004                SCOPE
1832 010240 012737 012000 001234                MOV     #TST15,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1833 010246 012737 010350 000114                MOV     #T08L01,2#PVEC ;SET UP PARITY TRAP HANDLER
1834 010254 012700 062000                MOV     #BUFH,RD      ;GET TEST ADDRESS
1835 010260 005001                CLR     R1           ;INIT FLAG TO INDIC. TESTING LOW BYTE
1836 010262 005037 001166 T08L06: CLR     2#$REG4      ;SAVE DATA IF ERROR
1837 010266 005037 001160                CLR     2#$REG1      ;SAVE ADDRESS IF ERROR
1838 010272 010037 001162                MOV     RD,2#$REG2   ;SAVE ADDRESS IF ERROR
1839 010276 012737 000204 177746                MOV     #204,2#CCR   ;TURN ON HALF OF CACHE
1840 010304 005737 062000                TST    2#BUFH       ;PUT LOC IN CACHE
1841 010310 052737 000100 177746                BIS    #100,2#CCR    ;ENABLE WRITE WRONG PARITY
1842 010316 112710 000000                MOV8   #D,(RD)      ;DO DATOB TO LOC & WWP
1843 010322 042737 000100 177746                BIC    #100,2#CCR    ;WWP OFF
1844 010330 005737 062000                TST    2#BUFH       ;FORCE PARITY TRAP
1845 010334 012737 000214 177746                MOV     #214,2#CCR   ;CACHE OFF
1846
1847 010342 104035                ERROR   35           ;ERROR: TEST OF DATA PARITY GENERATOR/CKER FAILED
1848                                ;DID NOT GET PARITY TRAP WHEN WROTE WRONG PARITY

```

K04

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01MACY11 27(1006) 09-FEB-77 15:33 PAGE 34
T14 TEST OF DATA PARITY GENERATOR/CHECKER

```

1849 010344 000137 011052          JMP      T08L02          ;GO TO NEXT TEST
1850
1851 010350 012737 000214 177746 T08L01: MOV      #214, @#CCR          ;CACHE OFF
1852
1853 010356 010046          MOV      RO, -(SP)      ;SAVE RO FOR MED INST
1854 010360 076600          MED          ;GET CONTENTS OF LOG REG
1855 010362 000022          .WORD    RLOG
1856 010364 052700 100001      BIS      #100001, RO    ;ENABLE ERROR LOG & LOG FIRST MODE
1857 010370 076600          MED          ;UNLOCK ERROR LOG
1858 010372 000222          .WORD    WLOG
1859 010374 012600          MOV      (SP)+, RO      ;RESTORE RO
1860
1861 010376 022626          CMP      (SP)+, (SP)+  ;RESTORE STACK
1862 010400 005701          TST     R1              ;TESTING HIGH BYTE?
1863 010402 001013          BNE     T08L03         ;BRANCH IF YES
1864 010404 032737 000100 177744      BIT      #100, @#EREG   ;WAS TRAP FROM LOW BYTE?
1865 010412 001022          BNE     T08L04         ;BRANCH IF YES
1866
1867 010414 076600          MED          ;GET LOG INFORMATION
1868 010416 000106          .WORD    CDL
1869 010420 010037 001164      MOV     RO, $REG3      ;SAVE INFORMATION
1870 010424 104036          ERROR   36            ;ERROR: TEST OF DATA PARITY GENERATOR/CKER FAILED
1871                                     ;DID NOT GET PARITY TRAP FROM LOW BYTE WHEN WWP
1872 010426 000137 011052          JMP      T08L02          ;GO TO NEXT TEST
1873
1874 010432 032737 000200 177744 T08L03: BIT      #200, @#EREG   ;WAS TRAP FROM HIGH BYTE?
1875 010440 001012          BNE     T08L05         ;BRANCH IF YES TO CONTINUE TEST
1876 010442 076600          MED          ;GET LOG INFORMATION
1877 010444 000106          .WORD    CDH
1878 010446 010037 001164      MOV     RO, $REG3      ;SAVE INFORMATION
1879 010452 104037          ERROR   37            ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
1880                                     ;DID NOT GET PARITY TRAP FROM HIGH BYTE WHEN WWP
1881 010454 000137 011052          JMP      T08L02          ;GO TO NEXT TEST
1882
1883 010460 005200          T08L04: INC     RO          ;TEST HIGH BYTE
1884 010462 005201          INC     R1              ;SET FLAG INDICATING HIGH BYTE TEST
1885 010464 000676          BR      T08L06         ;GO TEST IT
1886
1887 010466 012737 010546 000114 T08L05: MOV     #T08L07, @#PVEC ;SET UP PARITY TRAP HANDLER
1888 010474 012737 062000 001162      MOV     #BUFH, @#SREG2 ;SAVE ADDRESS IF ERROR
1889 010502 005000          CLR     RO              ;INIT. TEST DATA REG
1890 010504 010037 001166          T08L10: MOV     RO, $REG4   ;SAVE DATA IF ERROR
1891 010510 012737 000304 177746      MOV     #304, @#CCR     ;TURN HALF OF CACHE ON & WWP
1892 010516 010037 062000          MOV     RO, @#BUFH     ;GENERATE BAD PARITY AND WRITE IN CACHE
1893 010522 042737 000100 177746      BIC     #100, @#CCR    ;WWP OFF
1894 010530 005737 062000          TST     @#BUFH         ;FORCE PARITY TRAP
1895
1896 010534 012737 000214 177746      MOV     #214, @#CCR    ;TURN CACHE OFF FOR ERROR
1897 010542 104035          ERROR   35            ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
1898                                     ;NO PARITY TRAP WHEN WROTE WRONG PARITY
1899 010544 000542          BR      T08L02         ;GO TO NEXT TEST
1900
1901 010546 012737 000214 177746 T08L07: MOV     #214, @#CCR    ;TURN CACHE OFF AFTER TRAP
1902
1903 010554 010046          MOV     RO, -(SP)      ;SAVE RO FOR MED INST
1904 010556 076600          MED          ;GET CONTENTS OF LOG REG

```

1905	010560	000022			.WORD	RLOG		
1906	010562	052700	100001		BIS	#100001,RO		;ENABLE ERROR LOG & LOG FIRST MODE
1907	010566	076600			MED			;UNLOCK ERROR LOG
1908	010570	000222			.WORD	WLOG		
1909	010572	012600			MOV	(SP)+,RO		;RESTORE RO
1910								
1911	010574	022626			CMP	(SP)+,(SP)+		;RESTORE STACK
1912	010576	032737	000100	177744	BIT	#100,2#EREG		;TRAP FROM LOW BYTE?
1913	010604	001011			BNE	TOBL09		;BRANCH IF YES
1914								
1915	010606	076600			MED			;GET LOG INFORMATION
1916	010610	000106			.WORD	CDL		
1917	010612	010037	001164		MOV	RO,SREG3		;SAVE INFORMATION
1918	010616	012737	010504	001110	MOV	#TOBL10,2#SLPERR		;INIT FOR ERROR LOOP
1919	010624	104036			ERROR	36		;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
1920								NO PARITY TRAP FROM LOW BYTE WHEN WWP
1921	010626	000511			BR	TOBL02		;GO TO END OF TEST
1922								
1923	010630	032737	000200	177744	TOBL09: BIT	#200,2#EREG		;TRAP FROM HIGH BYTE?
1924	010636	001011			BNE	TOBL11		;BRANCH IF YES
1925								
1926	010640	076600			MED			;GET LOG INFORMATION
1927	010642	000106			.WORD	CDH		
1928	010644	010037	001164		MOV	R7,SREG3		;SAVE INFORMATION
1929	010650	012737	010504	001110	MOV	#TOBL10,2#SLPERR		;INIT FOR ERROR LOOP
1930	010656	104037			ERROR	37		;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
1931								NO PARITY TRAP FROM HIGH BYTE WHEN WWP
1932	010660	000474			BR	TOBL02		;GO TO NEXT TEST
1933								
1934	010662	022700	177777		TOBL11: CMP	#177777,RO		;ALL WRITE WRONG PARITY PATTERNS CKED?
1935	010666	001403			BEQ	TOBL12		;BRANCH IF YES
1936	010670	062700	000401		ADD	#401,RO		;GENERATE DATA FOR HIGH AND LOW BYTE
1937	010674	000703			BR	TOBL10		;GO TEST IT
1938								
1939	010676	012737	010740	000114	TOBL12: MOV	#TOBL13,2#PVEC		;SET UP FOR PARITY ERRORS
1940	010704	005000			CLR	RO		;INIT TEST DATA REG
1941	010706	012737	000204	177746	TOBL14: MOV	#204,2#CCR		;TURN HALF OF CACHE ON
1942	010714	010037	062000		MOV	RO,2#BUFH		;GEN PARITY AND STORE IN CACHE
1943	010720	005737	062000		TST	2#BUFH		;TEST PARITY
1944	010724	022700	177777		TOBL16: CMP	#177777,RO		;ALL GOOD PARITY PATTERNS CKED?
1945	010730	001450			BEQ	TOBL02		;BRANCH YES TO END OF TEST
1946	010732	062700	000401		ADD	#401,RO		;GENERATE DATA FOR HIGH & LOW BYTE
1947	010736	000763			BR	TOBL14		;TEST IT
1948								
1949	010740	052737	000014	177746	TOBL13: BIS	#14,2#CCR		;TURN CACHE OFF
1950								
1951	010746	010046			MOV	RO,-(SP)		;SAVE RO FOR MED INST
1952	010750	076600			MED			;GET CONTENTS OF LOG REG
1953	010752	000022			.WORD	RLOG		
1954	010754	052700	100001		BIS	#100001,RO		;ENABLE ERROR LOG & LOG FIRST MODE
1955	010760	076600			MED			;UNLOCK ERROR LOG
1956	010762	000222			.WORD	WLOG		
1957	010764	012600			MOV	(SP)+,RO		;RESTORE RO
1958								
1959	010766	022626			CMP	(SP)+,(SP)+		;RESTORE STACK
1960	010770	010037	001166		MOV	RO,SREG4		;SAVE GOOD DATA

```

1961 010774 076600 MED ;GET LOG INFORMATION
1962 010776 000106 .WORD RDAT
1963 011000 010037 001164 MOV RO,SREG3 ;SAVE INFORMATION
1964 011004 013700 001166 MOV SREG4,RO ;RESTORE RO
1965 011010 032737 000100 177744 BIT #100,2#EREG ;PARITY ERROR LOW BYTE?
1966 011016 001405 BEQ TOBL15 ;BRANCH IF NO
1967 011020 012737 010706 001110 MOV #TOBL14,2#SLPERR ;INIT ERROR LOOP
1968 011026 104040 ERROR 40 ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
1969 ; ;PARITY ERROR IN LOW BYTE
1970 011030 000410 BR TOBL02 ;GO TO END OF TEST
1971
1972 011032 032737 000200 177744 TOBL15: BIT #200,2#EREG ;PARITY ERROR HIGH BYTE?
1973 011040 001731 BEQ TOBL16 ;TEST NEXT PATTERN IF NO
1974 011042 012737 010706 001110 MOV #TOBL14,2#SLPERR ;INIT RETURN FOR LOOP ON ERROR
1975 011050 104041 ERROR 41 ;ERROR: TEST OF DATA PARITY GEN//CKER FAILED
1976 ; ;PARITY ERROR IN HIGH BYTE
1977
1978 011052 TOBL02:
1979
1980 ;RID CACHE OF BAD PARITY
1981 011052 012737 000214 177746 MOV #214,2#CCR ;CACHE OFF IF ON
1982 011060 004737 035134 JSR PC,SWEEP ;GO PURGE CACHE
1983
1984
1985 011064 012737 033142 000114 MOV #UPERR,2#114 ;RESTORE UNEXPECTED PARITY ERROR HANDLER
1986 011072 000137 012000 JMP 2#TST15 ;GO TO NEXT TEST
1987
1988
1989 012000 .=12000 ;ADJUST ADDRESS SPACE FOR NEXT TEST
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016

```

```

*****
:TEST 15 TEST THE VALID BIT FOR LOW HALF OF CACHE
:
:* THE TEST OF THE VALID BIT IS NOT COMPLETE UNTIL THE
:*VALID TEST FOR THE SECOND HALF OF CACHE IS RUN. THIS
:*IS THE FIRST TEST WHERE THIS ENTIRE HALF OF CACHE ADDRESSES ARE
:*EXERCISED.
:* DURING THE ENTIRE TEST ONLY ONE TAG AND DATA VALUE IS
:*USED. INITIALLY, THE ENTIRE HALF OF CACHE WHICH IS
:*ENABLED (FORCE MISS OFF) IS WRITTEN AND CHECKED THAT ALL
:*ITS ADDRESSES CAN BE MADE HITS. FOLLOWING THIS, A WRITE/
:*READ PROCEDURE IS DONE WHICH VERIFIES THAT THE LOCATIONS
:*CAN BE VALIDATED/INVALIDATED AND THAT THERE IS NO DUAL
:*ADDRESSING PROBLEM FOR THE V BIT. FIRST THE VALID BIT
:*IS SET FOR HALF OF CACHE, THEN STARTING AT THE LOWEST
:*HALF CACHE ADDRESS, EACH LOC IS TESTED TO BE A HIT (VALID
:*SET) AND THEN INVALIDATED VIA WRITING WRONG PARITY AND
:*FORCING A TRAP. THIS IS DONE INCREASING THE ADDRESS
:*UNTIL HALF OF CACHE IS READ AND WRITTEN. NEXT, STARTING
:*AT THE HIGH HALF CACHE ADDRESS, EACH LOC IS READ, TESTED
:*TO BE A MISS (VALID=0) AND THEN WRITTEN TO SET THE VALID
:*BIT. THIS IS DONE, DECREASING THE ADDRESS EACH TIME,
:*TILL THE LOW ADDRESS IS REACHED. THIS PROCEDURE IS THEN
:*REPEATED FOR A SECOND PASS WITH THE PATTERN REVERSED.

```

;(I.E. STARTING WITH ALL LOC INVALIDATED AND THEN READING
 ;AND WRITING THE V BIT.)
 ;*
 ;*RD CONTAINS THE CACHE ADDRESS BEING TESTED.

;*NOTE:TEST FOR DUAL ADDRESSING FOR LOCATIONS WHICH OVERLAP
 ;* THE PARITY TRAP ADDRESSES 114,116 IS NOT DONE

2017									
2018									
2019									
2020									
2021									
2022									
2023									
2024									
2025									
2026	012000	012737	000214	177746	TST15:	MOV	#214,#CCR		;CACHE OFF FOR SCOPE
2027	012006	000004				SCOPE			
2028	012010	012737	012734	001234		MOV	#TST16,SKTST		;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
2029	012016	012706	020000			MOV	#20000,SP		;ADJUST STACK FOR ADDRESSES OUT OF TEST AREA
2030	012022	012737	000210	177746		MOV	#210,#CCR		;HALF CACHE ON
2031	012030	012700	060000			MOV	#BUFL,R0		;INIT STARTING ADDRESS
2032	012034	012701	001000			MOV	#1000,R1		;INIT COUNT FOR 1/2 K
2033	012040	005020			1S:	CLR	(R0)+		;WRITE CACHE
2034	012042	077102				S0B	R1,1S		;LOOP TILL HALF CACHE WRITTEN
2035	012044	005740			T24L20:	TST	-(R0)		;SEE IF DATA IN CACHE
2036	012046	033727	177752	000004		BIT	#HMR,#HMR2		;HIT? (VALID BIT SET?)
2037	012054	001002				BNE	T24L19		;BRANCH IF YES
2038	012056	000137	012624			JMP	T24L01		;REPORT ERROR
2039	012062	020027	060000		T24L19:	CMP	R0,#BUFL		;HALF CACHE TESTED?
2040	012066	001366				BNE	T24L20		;BRANCH IF NO
2041									
2042									
2043	012070	012737	012154	000114		MOV	#T24L02,#PVEC		;SET UP PARITY HANDLER
2044									
2045	012076	020027	060114		T24L05:	CMP	R0,#BUFL!114		;TESTING PARITY AREA?
2046	012102	001412				BEQ	T24L22		;DON'T TEST ADDRESS IF YES
2047	012104	020027	060116			CMP	R0,#BUFL!116		;TESTING PARITY AREA?
2048	012110	001407				BEQ	T24L22		;DON'T TEST ADDRESS IF YES
2049	012112	005710				TST	(R0)		;SEE IF VALID BIT SET
2050	012114	033727	177752	000004		BIT	#HMR,#HMR2		;HIT? (VALID BIT SET?)
2051	012122	001002				BNE	T24L22		;BRANCH IF YES
2052	012124	000137	012646			JMP	T24L03		;REPORT ERROR
2053									
2054	012130	012737	000310	177746	T24L22:	MOV	#310,#CCR		;CACHE ON IF OFF AND WRITE WRONG PARITY
2055	012136	005010				CLR	(R0)		;WRITE LOC WITH WRONG PARITY
2056	012140	012737	000210	177746		MOV	#210,#CCR		;WMP OFF
2057	012146	005710				TST	(R0)		;FORCE PARITY TRAP
2058	012150	000137	012670			JMP	T24L04		;REPORT ERROR IF DID NOT TRAP
2059									
2060	012154				T24L02:				
2061									
2062	012154	010046				MOV	R0,-(SP)		;SAVE R0 FOR MED INST
2063	012156	076600				MED			;GET CONTENTS OF LOG REG
2064	012160	000022				.WORD	RLOG		
2065	012162	052700	100001			BIS	#100001,R0		;ENABLE ERROR LOG & LOG FIRST MODE
2066	012166	076600				MED			;UNLOCK ERROR LOG
2067	012170	000222				.WORD	WLOG		
2068	012172	012600				MOV	(SP)+,R0		;RESTORE R0
2069									
2070	012174	062700	000002			ADD	#2,R0		;LOOK AT NEXT ADDR.
2071	012200	062706	000004			ADD	#4,SP		;RESTORE STACK
2072	012204	020027	062000			CMP	R0,#BUFL+2000		;HALF ADDRESSES TESTED?

```

2073 012210 001332          BNE      T24L05          ;BRANCH IF NO
2074
2075 012212 012737 033142 000114      MOV      #UPERR, @PVEC      ;RESTORE UNEXP. PARITY ERROR HANDLER
2076 012220 005740          TST      -(RO)              ;WAS LOC INVALIDATED?
2077 012222 033727 177752 000004      BIT      @#HMR, #HMR2      ;LOC A MISS? (INVALIDATED?)
2078 012230 001402          BEQ      T24L09            ;BRANCH IF YES
2079 012232 000137 012712          JMP      T24L06            ;REPORT ERROR
2080 012236 005010          CLR      (RO)              ;WRITE LOC
2081 012240 020027 060000          CMP      RO, #BUFL         ;AT LAST LOC?
2082 012244 001365          BNE      1$               ;BRANCH IF NO
2083
2084                                ;NOW WRITE/READ VALID BIT WITH PATTERN REVERSED
2085
2086 012246 012737 012316 000114 T24L10: MOV      #T24L07, @PVEC      ;SET UP FOR PARITY TRAP
2087 012254 012700 061776          MOV      #BUFL+1776, RO    ;INIT TEST ADDR.
2088 012260 012737 000310 177746 T24L08: MOV      #310, @CCR      ;WRITE WRONG PARITY & CACHE ON
2089 012266 005010          CLR      (RO)              ;WRITE WRONG PARITY
2090 012270 012737 000210 177746          MOV      #210, @CCR        ;WMP OFF
2091 012276 005710          TST      (RO)              ;FORCE TRAP
2092 012300 012737 000214 177746          MOV      #214, @CCR        ;CACHE OFF
2093 012306 012737 012260 001110          MOV      #T24L08, @SLPERR ;INIT RETURN FOR ERROR LOOP
2094 012314 000570          BR       T24L15           ;REPORT ERROR IF DID NOT TRAP
2095
2096 012316          T24L07:
2097
2098 012316 010046          MOV      RO, -(SP)         ;SAVE RO FOR MED INST
2099 012320 076600          MED      ;GET CONTENTS OF LOG REG
2100 012322 000022          .WORD   RLOG              ;
2101 012324 052700 100001          BIS      #100001, RO      ;ENABLE ERROR LOG & LOG FIRST MODE
2102 012330 076600          MED      ;UNLOCK ERROR LOG
2103 012332 000222          .WORD   WLOG              ;
2104 012334 012600          MOV      (SP)+, RO        ;RESTORE RO
2105
2106 012336 162700 000002          SUB      #2, RO           ;LOOK AT NEXT ADDRESS
2107 012342 062706 000004          ADD      #4, SP           ;ADJUST STACK
2108 012346 020027 057776          CMP      RO, #BUFL-2      ;HALF CACHE WRITTEN?
2109 012352 001342          BNE      T24L08           ;BRANCH IF NO
2110
2111 012354 012737 033142 000114      MOV      #UPERR, @PVEC      ;
2112 012362 062700 000002          T24L12: ADD      #2, RO           ;ADJUST ADDRESS
2113 012366 005710          TST      (RO)              ;READ LOC
2114 012370 033727 177752 000004      BIT      @#HMR, #HMR2      ;MISS? (LOC INVALIDATED?)
2115 012376 001407          BEQ      T24L09            ;BRANCH IF YES
2116 012400 012737 000214 177746          MOV      #214, @CCR        ;CACHE OFF
2117 012406 012737 012246 001110          MOV      #T24L10, @SLPERR ;INIT RETURN FOR ERROR LOOP
2118 012414 000536          BR       T24L06           ;REPORT ERROR
2119
2120 012416 005010          T24L09: CLR      (RO)              ;WRITE LOC
2121 012420 020027 061776          CMP      RO, #BUFL+1776    ;HALF CACHE WRITTEN?
2122 012424 001356          BNE      T24L12           ;BRANCH IF NO
2123
2124                                ;NOW READ LOC TO SEE IF VALID STILL SET
2125
2126 012426 012737 012536 000114      MOV      #T24L16, @PVEC      ;SET UP PARITY HANDLER
2127 012434 020027 060114          T24L17: CMP      RO, #BUFL!114    ;TESTING PARITY AREA?
2128 012440 001417          BEQ      T24L13           ;DON'T TEST ADDRESS IF YES

```

```

2129 012442 020027 060116      CMP      RO,#BUFL!116      ;TESTING PARITY AREA?
2130 012446 001414      BEQ      T24L13           ;DON'T TEST ADDRESS IF YES
2131
2132 012450 005710      TST      (RO)             ;LOC IN CACHE?
2133 012452 033727 177752 000004      BIT      @#HMR,#HMR2      ;HIT?
2134 012460 001007      BNE      T24L13           ;BRANCH IF YES
2135 012462 012737 000214 177746      MOV      #214,@#CCR       ;CACHE OFF
2136 012470 012737 012246 001110      MOV      @T24L10,@#SLPERR ;INIT RETURN FOR ERROR LOOP
2137 012476 000466      BR       T24L14           ;REPORT ERROR
2138
2139 012500 052737 000100 177746 T24L13: BIS      #100,@#CCR       ;SET WRITE WRONG PARITY
2140 012506 005010      CLR      (RO)             ;WRITE WRONG PARITY
2141 012510 012737 000210 177746      MOV      #210,@#CCR       ;WMP OFF
2142 012516 005710      TST      (RO)             ;FORCE TRAP
2143 012520 012737 000214 177746      MOV      #214,@#CCR       ;CACHE OFF
2144 012526 012737 012246 001110      MOV      @T24L10,@#SLPERR ;
2145 012534 000460      BR       T24L15           ;REPORT ERROR
2146
2147 012536 062706 000004      T24L16: ADD      #4,SP      ;RESTORE STACK
2148 012542 162700 000002      SUB      #2,RO            ;LOOK AT NEXT ADDR.
2149
2150 012546 010046      MOV      RO,-(SP)         ;SAVE RO FOR MED INST
2151 012550 076600      MED      ;GET CONTENTS OF LOG REG
2152 012552 000022      .WORD   RLOG
2153 012554 052700 100001      BIS      #100001,RO       ;ENABLE ERROR LOG & LOG FIRST MODE
2154 012560 076600      MED      ;UNLOCK ERROR LOG
2155 012562 000222      .WORD   WLOG
2156 012564 012600      MOV      (SP)+,RO        ;RESTORE RO
2157
2158 012566 020027 057776      CMP      RO,#BUFL-2      ;ALL ADDR TESTED?
2159 012572 001320      BNE      T24L17           ;BRANCH IF NO
2160
2161 012574      T24L18:
2162
2163      ;RID CACHE OF BAD PARITY
2164 012574 012737 000214 177746      MOV      #214,@#CCR       ;CACHE OFF IF ON
2165 012602 004737 035134      JSR      PC,SWEEP         ;GO PURGE CACHE
2166
2167
2168 012606 012737 033142 000114      MOV      #UPERR,@#PVEC
2169 012614 012706 001100      MOV      #STACK,SP       ;RESTORE STACK
2170 012620 000137 012734      JMP      @TST16           ;GO TO NEXT TEST
2171
2172 012624 012737 000214 177746 T24L01: MOV      #214,@#CCR       ;CACHE OFF
2173 012632 005037 001160      CLR      $REG1            ;SAVE FAILING ADDR
2174 012636 010037 001162      MOV      RO,$REG2         ;SAVE FAILING ADDR
2175 012642 104043      ERROR   43                ;ERROR: ADDRESS COULD NOT BE MADE A HIT
2176 012644 000753      BR       T24L18           ;GO TO END OF TEST
2177
2178 012646 012737 000214 177746 T24L03: MOV      #214,@#CCR       ;CACHE OFF
2179 012654 005037 001160 T24L14: CLR      $REG1            ;SAVE FAILING ADDRESS
2180 012660 010037 001162      MOV      RO,$REG2         ;SAVE FAILING ADDRESS
2181 012664 104111      ERROR   111               ;ERROR: TEST OF VALID BIT FAILED
2182      ;LOC COULD NOT BE MADE A HIT
2183 012666 000742      BR       T24L18           ;GO TO END OF TEST
2184

```

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 40
T15 TEST THE VALID BIT FOR LOW HALF OF CACHE

```

2185 012670 012737 000214 177746 T24L04: MOV #214, @#CCR ;CACHE OFF
2186 012676 005037 001160 T24L15: CLR $REG1 ;SAVE FAILING ADDRESS
2187 012702 010037 001162 MOV R0, $REG2 ;SAVE FAILING ADDRESS
2188 012706 104042 ERROR 42 ;ERROR: NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PARIT
2189 012710 000731 BR T24L18 ;GO TO END OF TEST
2190
2191 012712 012737 000214 177746 T24L06: MOV #214, @#CCR ;CACHE OFF
2192 012720 005037 001160 CLR $REG1 ;SAVE FAILING ADDR
2193 012724 010037 001162 MOV R0, $REG2 ;SAVE FAILING ADDR
2194 012730 104112 ERROR 112 ;ERROR: TEST OF VALID BIT FAILED
2195 ; LOCATION NOT INVALIDATED BY PARITY TRAP
2196 012732 000720 BR T24L18 ;GO TO END OF TEST
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226

```

```

*****
;TEST 16 TEST TAG PARITY BIT FOR LOW CACHE ADDRESSES
;
;
; THE TEST OF THE TAG PARITY BIT IS NOT COMPLETE UNTIL
;THE TAG P BIT TEST FOR THE SECOND HALF OF CACHE AND THE
;MSB ADDRESS (A10) TO CACHE TAG FIELD TEST ARE RUN. TWO
;TAG ADDRESSES ARE USED TO GENERATE A PARITY BIT OF 1 AND
;#0. THE FIRST ADDRESS IS CHOSEN FROM A TEST BUFFER AREA
;AND THE SECOND IS CHOSEN TO LIE 1K AWAY. A WRITE/READ
;PROCEDURE IS DONE WHICH CHECKS THE P BIT AND DUAL ADD-
;RESSING FOR HALF OF CACHE. INITIALLY THE P BIT IS WRITTEN
;WITH ONE PARITY PATTERN IN HALF OF CACHE. THEN STARTING
;AT THE LOW HALF CACHE ADDRESS, THE LOC IS READ AND THEN
;WRITTEN WITH THE OPPOSITE PARITY. THIS IS SEQUENTIALLY
;REPEATED WITH INCREASING ADDRESSES UNTIL THE HIGH HALF
;CACHE ADDRESS IS REACHED. THEN STARTING AT THE HIGH ADDR,
;THE SECOND PARITY PATTERN IS READ AND THE LOC IS REWRITTEN
;WITH THE FIRST. THIS IS SEQUENTIALLY REPEATED, DECREASING
;THE ADDRESS, UNTIL THE LOW HALF CACHE ADDRESS IS REACHED.
;A SECOND PASS IS THEN MADE WITH THE PARITY PATTERN RE-
;VERSED. A PARITY ERROR HANDLER IS SETUP TO DETECT PARITY
;ERRORS. ALSO, LOCS WHICH SHOULD BE HITS ARE CHECKED FOR
;AND REPORTED IF NO HIT OCCURRED.
;
;#R0, R1 CONTAIN ADDRESSES TO GENERATE COMPLIMENTARY TAG
;#PARITY BITS.

```

```

2227 012734 012737 000214 177746 TST16: MOV #214, @#CCR ;CACHE OFF FOR SCOPE
2228 012742 000004 SCOPE
2229 012744 012737 013406 001234 MOV #TST17, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
2230 012752 012737 013126 000114 MOV #T11L01, @#PVEC ;SET UP FOR PARITY ERRORS
2231 012760 005003 CLR R3 ;INIT FLAG=FIRST PASS
2232 012762 012700 060000 MOV #BUFL, R0 ;SET UP ADDR. FOR FIRST PASS
2233 012766 012737 000210 177746 MOV #210, @#CCR ;TURN HALF CACHE ON
2234 012774 012701 001000 T11L02: MOV #1000, R1 ;INIT COUNTER
2235 013000 005720 15: TST (R0)+ ;PUT PARITY PATTERN IN TAG FIELD
2236 013002 077102 SOB R1, 15 ;LOAD HALF OF CACHE
2237
2238 013004 012701 001000 MOV #1000, R1 ;INIT. COUNTER
2239 013010 012700 060000 MOV #BUFL, R0 ;SET UP ADDR. FOR FIRST PASS
2240 013014 012702 054000 MOV #BUFL-4000, R2 ;SET UP ADDR. FOR FIRST PASS

```


E05

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 41
T16 TEST TAG PARITY BIT FOR LOW CACHE ADDRESSES

2241	013020	005703				TST	R3		; FIRST PASS?
2242	013022	001404				BEQ	T11L03		; BRANCH IF YES
2243	013024	012700	054000			MOV	#BUFL-4000,R0		; SET UP ADDR. FOR SECOND PASS
2244	013030	012702	060000			MOV	#BUFL,R2		; SET UP ADDR. FOR SECOND PASS
2245	013034	005720			T11L03:	TST	(R0)+		; READ CACHE TO SEE IF PARITY OK; NO-TRAPS
2246	013036	033727	177752	000004		BIT	2#HMR,#HMR2		; WAS ADDRESS A HIT?
2247	013044	001533				BEQ	T11L04		; BRANCH TO ERROR IF NO
2248	013046	005722				TST	(R2)+		; WRITE DIFFERENT PARITY PATTERN IN TAG FIELD
2249	013050	077107				SOB	R1,T11L03		; LOOK AT HALF OF CACHE
2250									
2251	013052	012701	001000			MOV	#1000,R1		; INIT COUNTER
2252	013056	005742			T11L11:	TST	-(R2)		; READ SECOND PARITY PATTERN
2253	013060	033727	177752	000004		BIT	2#HMR,#HMR2		; WAS ADDRESS A HIT?
2254	013066	001532				BEQ	T11L05		; BRANCH IF NO TO ERROR
2255	013070	005740				TST	-(R0)		; PUT NEW PARITY PATTERN IN TAG
2256	013072	077107				SOB	R1,T11L11		; LOOK AT HALF OF CACHE
2257									
2258	013074	005703				TST	R3		; FIRST PASS?
2259	013076	001140				BNE	T11L06		; NO GO TO END OF TEST
2260	013100	052703	000001			BIS	#1 R3		; SET FLAG TO INDIC. SECOND PASS
2261	013104	012737	000210	177746	T11L12:	MOV	#210,2#CCR		; HALF CACHE ON IF OFF
2262	013112	012737	013104	001110		MOV	#T11L12,2#SLPERR		; SETUP RETURN FOR ERROR IF ONE OCCURS
2263	013120	012700	054000			MOV	#BUFL-4000,R0		; SET UP FOR SECOND PASS.
2264	013124	000723				BR	T11L02		; GO TEST SECOND PASS
2265									
2266	013126				T11L01:				
2267									
2268									
2269	013126	012737	000214	177746					
2270	013134	004737	035134			MOV	#214,2#CCR		; CACHE OFF IF ON
2271						JSR	PC,SWEEP		; GO PURGE CACHE
2272									
2273									
2274	013140	010046				MOV	R0,-(SP)		; SAVE R0 FOR MED INST
2275	013142	076600				MED			; GET CONTENTS OF LOG REG
2276	013144	000022				.WORD	RLOG		
2277	013146	052700	100001			BIS	#100001,R0		; ENABLE ERROR LOG & LOG FIRST MODE
2278	013152	076600				MED			; UNLOCK ERROR LOG
2279	013154	000222				.WORD	WLOG		
2280	013156	012600				MOV	(SP)+,R0		; RESTORE R0
2281									
2282	013160	076600				MED			; GET LOG INFOR FOR PHY. ADDR. A17,A16
2283	013162	000101				.WORD	RSER		
2284	013164	000300				SWAB	R0		; PUT PHY. ADDR A17, A16 IN LOW BYTE
2285	013166	042700	177776			BIC	#177776,R0		; ONLY LOOK AT A17, A16
2286	013172	010037	001160			MOV	R0,\$REG1		; SAVE ADDRESS
2287	013176	076600				MED			; GET LOG INFORMATION
2288	013200	000102				.WORD	LOADD		
2289	013202	010037	001162			MOV	R0,\$REG2		; SAVE INFORMATION
2290	013206	076600				MED			; GET LOG INFORMATION
2291	013210	000100				.WORD	RJAM		
2292	013212	032700	000400			BIT	#400,R0		; ERROR IN BACKING STORE?
2293	013216	001410				BEQ	T11L07		; BRANCH IF NO
2294	013220	011637	001164			MOV	(SP),\$REG3		; GET PC+2 WHERE ERROR OCCURRED
2295	013224	162737	000002	001166		SUB	#2,\$REG4		; SAVE PC WHERE ERROR OCCURRED
2296	013232	022626				CMP	(SP)+,(SP)+		; RESTORE STACK

F05

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 42
T16 TEST TAG PARITY BIT FOR LOW CACHE ADDRESSES

```

2297 013234 104001          ERROR 1          ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
2298 013236 000460          BR      T11L06   ;GO TO NEXT TEST
2299
2300 013240 022626          T11L07: CMP      (SP)+,(SP)+ ;RESTORE STACK
2301 013242 032737 000040 177744 BIT      #40,2#EREG ;ERROR IN TAG?
2302 013250 001411          BEQ     T11L08   ;BRANCH NO
2303 013252 076600          MED          ;GET TAG LOG INFO.
2304 013254 000107          .WORD  RTAG
2305 013256 000300          SWAB   RO
2306 013260 042700 177400 BIC     #177400,RO ;PUT TAG IN LOW BYTE
2307 013264 010037 001164 MOV     RO,$REG3 ;LOOK AT TAG ONLY
2308 013270 104045          ERROR  45       ;SAVE BAD DATA
2309 013272 000442          BR      T11L06   ;ERROR: TAG PARITY ERROR WHEN TESTING TAG P BIT
2310
2311 013274 032737 000100 177744 T11L08: BIT      #100,2#EREG ;ERROR IN LOW BYTE?
2312 013302 001406          BEQ     T11L09   ;BRANCH IF NO
2313 013304 076600          MED          ;GET LOG INFORMATION
2314 013306 000106          .WORD  CDL
2315 013310 010037 001164 MOV     RO,$REG3 ;SAVE INFORMATION
2316 013314 104046          ERROR  46       ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING TAG P BIT
2317 013316 000430          BR      T11L06   ;NEXT TEST
2318
2319 013320          T11L09:
2320 013320 076600          MED          ;GET LOG INFORMATION
2321 013322 000106          .WORD  CDH
2322 013324 010037 001164 MOV     RO,$REG3 ;SAVE INFORMATION
2323 013330 104047          ERROR  47       ;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG P BIT
2324 013332 000422          BR      T11L06   ;NEXT TEST
2325
2326 013334 052737 000014 177746 T11L04: BIS      #14,2#CCR ;CACHE OFF
2327 013342 162700 000002          SUB     #2,RO    ;GET BAD ADDRESS
2328 013346 010037 001162          MOV     RO,$REG2 ;SAVE BAD ADDRESS
2329 013352 000407          BR      T11L10   ;REPORT ERROR
2330 013354 052737 000014 177746 T11L05: BIS      #14,2#CCR ;CACHE OFF
2331 013362 010237 001162          MOV     R2,$REG2 ;SAVE BAD ADDRESS
2332 013366 062702 000002          ADD     #2,R2    ;RESTORE R2 TO FAILING ADDR.+2
2333 013372 005037 001160          T11L10: CLR     $REG1 ;SAVE BAD ADDRESS
2334 013376 104043          ERROR  43       ;ERROR: ADDRESS COULD NOT BE MADE A HIT
2335
2336 013400 012737 033142 000114 T11L06: MOV     #UPERR,2#PVEC ;RESTORE PARITY TRAP HANDLER
2337
2338 ;*****
2339 ;*TEST 17          TEST DATA PARITY BITS FOR LOW CACHE
2340 ;*
2341 ;* THE TEST OF THE DATA PARITY BITS ARE NOT COMPLETE
2342 ;*UNTIL THE DATA P BIT TEST FOR THE SECOND HALF OF CACHE
2343 ;*AND THE MSB ADDRESS (A10) TO CACHE DATA FIELD ARE RUN.
2344 ;*A WRITE/READ PROCEDURE IS DONE WHICH SIMULTANEOUSLY
2345 ;*CHECKS THE DATA P BIT FOR BOTH BYTES AND DUAL ADDRESSING
2346 ;*IN HALF OF CACHE FOR IT. INITIALLY THE P BIT IS WRITTEN
2347 ;*WITH ONE PARITY PATTERN IN HALF OF CACHE. THEN STARTING
2348 ;*AT THE LOW HALF CACHE ADDRESS, THE LOC IS READ AND THEN
2349 ;*WRITTEN WITH THE OPPOSITE PARITY. THIS IS SEQUEN-
2350 ;*TIALLY REPEATED WITH INCREASING ADDRESSES UNTIL THE HIGH
2351 ;*HALF CACHE ADDRESS IS REACHED. THEN STARTING AT THE
2352 ;*HIGH ADDR, THE SECOND PARITY PATTERN IS READ AND THE LOC

```

G05

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 43
T17 TEST DATA PARITY BITS FOR LOW CACHE

```

2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364 013406 012737 000214 177746 T17: MOV #214, @CCR ;CACHE OFF FOR SCOPE
2365 013414 000004 SCOPE
2366 013416 012737 014100 001234 MOV #TST20, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
2367 013424 012737 013646 000114 MOV #T12L01, @PVEC ;SET UP PARITY ERROR HANDLER
2368 013432 005003 CLR R3 ;INIT FLAG FOR FIRST PASS
2369 013434 005000 CLR R0 ;SET UP PARITY PATTERN A FOR FIRST PASS
2370 013436 012737 000210 177746 T12L02: MOV #210, @CCR ;HALF CACHE ON
2371 013444 012701 001000 MOV #1000, R1 ;INIT ADDR. COUNTER
2372 013450 012705 060000 MOV #BUFL, R5 ;INIT. TEST ADDRESS
2373 013454 010025 1S: MOV R0, (R5)+ ;WRITE DATA PARITY PATTERN
2374 013456 077102 SOB R1, 1S ;HALF ADDR. WRITTEN? BRANCH IF NO
2375
2376 013460 012701 001000 MOV #1000, R1 ;INIT ADDR. COUNTER
2377 013464 012705 060000 MOV #BUFL, R5 ;INIT. TEST ADDR
2378 013470 012700 000401 MOV #401, R0 ;SET UP PATTERN B FOR FIRST PASS
2379 013474 005703 TST R3 ;FIRST PASS?
2380 013476 001401 BEQ 2S ;BRANCH IF YES
2381 013500 005000 CLR R0 ;SET UP PARITY PATTERN A FOR SECOND PASS
2382 013502 005715 2S: TST (R5) ;SEE IF PARITY UNCHANGED
2383 013504 033727 177752 000004 BIT @HMR, #HMR2 ;DATA FROM CACHE?
2384 013512 001444 BEQ T12L07 ;BRANCH TO ERROR IF NO
2385 013514 010025 MOV R0, (R5)+ ;WRITE NEW DATA PARITY PATTERN
2386 013516 077107 SOB R1, 2S ;HALF ADDR. SPACE EXAMINED & WRITTEN?
2387
2388 013520 012701 001000 MOV #1000, R1 ;INIT ADDR. COUNTER
2389 013524 005000 CLR R0 ;SET UP PARITY PATTERN A FOR FIRST PASS
2390 013526 005703 TST R3 ;FIRST PASS?
2391 013530 001402 BEQ T12L06 ;BRANCH IF YES
2392 013532 012700 000401 MOV #401, R0 ;SET UP PARITY PATTERN B FOR SECOND PASS
2393 013536 012737 000210 177746 T12L06: MOV #210, @CCR ;HALF CACHE ON IF OFF FROM ERROR
2394 013544 005745 1S: TST -(R5) ;SEE IF PARITY UNCHANGED
2395 013546 033727 177752 000004 BIT @HMR, #HMR2 ;DATA FROM CACHE
2396 013554 001423 BEQ T12L07 ;BRANCH IF NO TO ERROR
2397 013556 010015 MOV R0, (R5) ;WRITE NEW PARITY PATTERN IN CACHE
2398 013560 077107 SOB R1, 1S ;HALF OF ADDRESS SPACE READ & WRITTEN? BRANCH IF NO
2399
2400 013562 005703 TST R3 ;SECOND PASS?
2401 013564 001010 BNE T12L08 ;GO TO END OF TEST IF YES
2402 013566 012700 000401 T12L13: MOV #401, R0 ;SET UP PARITY PATTERN B FOR SECOND PASS
2403 013572 052703 000001 BIS #1, R3 ;SET FLAG FOR PASS 2
2404 013576 012737 013566 001110 MOV #T12L13, @SLPERR ;INIT RETURN FOR ERROR LOOP IF ERROR OCCURS
2405 013604 000714 BR T12L02 ;TEST DATA
2406
2407
2408 013606 012737 033142 000114 T12L08: MOV #UPERR, @PVEC ;RESTORE PARITY ERROR HANDLER

```

H05

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 44
T17 TEST DATA PARITY BITS FOR LOW CACHE

2409	013614	052737	000014	177746		BIS	#14,2#CCR	: CACHE OFF WHEN CROSS CACHE ADDRESS BOUNDARY
2410	013622	000526				BR	TST20	:: GO TO NEXT TEST
2411								
2412	013624	052737	000014	177746	T12L07:	BIS	#14,2#CCR	: CACHE OFF
2413	013632	010537	001162			MOV	R5,\$REG2	: SAVE BAD ADDRESS
2414	013636	005037	001160			CLR	\$REG1	: SAVE BAD ADDRESS
2415	013642	104043				ERROR	43	: ERROR: ADDRESS COULD NOT BE MADE A HIT
2416	013644	000760				BR	T12L08	: GO TO END OF TEST
2417								
2418	013646	052737	000014	177746	T12L01:	BIS	#14,2#CCR	: CACHE OFF
2419								
2420	013654	010046				MOV	RO,-(SP)	: SAVE RO FOR MED INST
2421	013656	076600				MED		: GET CONTENTS OF LOG REG
2422	013660	000022				.WORD	RLOG	
2423	013662	052700	100001			BIS	#100001,RO	: ENABLE ERROR LOG & LOG FIRST MODE
2424	013666	076600				MED		: UNLOCK ERROR LOG
2425	013670	000222				.WORD	WLOG	
2426	013672	012600				MOV	(SP)+,RO	: RESTORE RO
2427								
2428	013674	076600				MED		: GET LOG INFOR FOR PHY. ADDR. A17,A16
2429	013676	000101				.WORD	RSER	
2430	013700	000300				SWAB	RO	: PUT PHY. ADDR A17, A16 IN LOW BYTE
2431	013702	042700	177776			BIC	#177776,RO	: ONLY LOOK AT A17, A16
2432	013706	010037	001160			MOV	RO,\$REG1	: SAVE ADDRESS
2433	013712	076600				MED		: GET LOG INFORMATION
2434	013714	000102				.WORD	LOADD	
2435	013716	010037	001162			MOV	RO,\$REG2	: SAVE INFORMATION
2436	013722	032737	000040	177744		BIT	#40,2#EREG	: ERROR IN TAG?
2437	013730	001417				BEQ	T12L09	: BRANCH IF NO
2438	013732	011637	001166			MOV	(SP),\$REG4	: GET PC+2 OF ERROR
2439	013736	162737	000002	001166		SUB	#2,\$REG4	: GET PC OF ERROR
2440	013744	076600				MED		: GET TAG LOG INFO.
2441	013746	000107				.WORD	RTAG	
2442	013750	000300				SWAB	RO	: PUT TAG IN LOW BYTE
2443	013752	042700	177400			BIC	#177400,RO	: LOOK AT TAG ONLY
2444	013756	010037	001164			MOV	RO,\$REG3	: SAVE BAD DATA
2445	013762	022626				CMP	(SP)+,(SP)+	: RESTORE THE STACK
2446	013764	104002				ERROR	2	: ERROR: UNEXPECTED PARITY ERROR IN TAG FIELD
2447	013766	000707				BR	T12L08	: GO TO END OF TEST
2448								
2449	013770	022626			T12L09:	CMP	(SP)+,(SP)+	: RESTORE STACK
2450	013772	005037	001166			CLR	\$REG4	: SAVE GOOD DATA
2451	013776	005700				TST	RO	: WAS TEST DATA =0?
2452	014000	001003				BNE	T12L11	: BRANCH IF NO
2453	014002	012737	000401	001166		MOV	#401,\$REG4	: SAVE GOOD DATA
2454	014010	032737	000200	177744	T12L11:	BIT	#200,2#EREG	: ERROR IN HIGH BYTE?
2455	014016	001406				BEQ	T12L12	: BRANCH IF NO
2456	014020	076600				MED		: GET LOG INFORMATION
2457	014022	000106				.WORD	CDH	
2458	014024	010037	001164			MOV	RO,\$REG3	: SAVE INFORMATION
2459	014030	104050				ERROR	50	: ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA P BITS
2460	014032	000665				BR	T12L08	: GO TO END OF TEST
2461								
2462	014034	032777	000100	163702	T12L12:	BIT	#100,2EREG	: ERROR IN LOW BYTE?
2463	014042	001406				BEQ	T12L14	: BRANCH IF NO
2464	014044	076600				MED		: GET LOG INFORMATION

465	014046	000106			.WORD	CDL		
466	014050	010037	001164		MOV	RO,\$REG3		;SAVE INFORMATION
467	014054	104051			ERROR	51		;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BITS
468	014056	000653			BR	T12L08		;GO TO END OF TEST
469								
470	014060	016637	177774	001164	T12L14: MOV	-4(SP),\$REG3		;GET PC+2 OF TRAP
471	014066	162737	000002	001164	SUB	#2,\$REG3		;SAVE PC OF TRAP
472	014074	104001			ERROR	1		;ERROR: UNEXP. PARITY ERROR IN BACKING STORE
473	014076	000643			BR	T12L08		;GO TO END OF TEST

474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520

```

*****
*TEST 20      TEST THE VALID BIT FOR HIGH HALF OF CACHE
*
*  THE TEST OF THE VALID BIT IS NOT COMPLETE UNTIL THE
*VALID TEST FOR THE SECOND HALF OF CACHE IS RUN.  THIS
*IS THE FIRST TEST WHERE THIS ENTIRE HALF OF CACHE ADDRESSES ARE
*EXERCISED.
*  DURING THE ENTIRE TEST ONLY ONE TAG AND DATA VALUE IS
*USED.  INITIALLY, THE ENTIRE HALF OF CACHE WHICH IS
*ENABLED (FORCE MISS OFF) IS WRITTEN AND CHECKED THAT ALL
*ITS ADDRESSES CAN BE MADE HITS.  FOLLOWING THIS, A WRITE/
*READ PROCEDURE IS DONE WHICH VERIFIES THAT THE LOCATIONS
*CAN BE VALIDATED/INVALIDATED AND THAT THERE IS NO DUAL
*ADDRESSING PROBLEM FOR THE V BIT.  FIRST THE VALID BIT
*IS SET FOR HALF OF CACHE, THEN STARTING AT THE LOWEST
*HALF CACHE ADDRESS, EACH LOC IS TESTED TO BE A HIT (VALID
*SET) AND THEN INVALIDATED VIA WRITING WRONG PARITY AND
*FORCING A TRAP.  THIS IS DONE INCREASING THE ADDRESS
*UNTIL HALF OF CACHE IS READ AND WRITTEN.  NEXT, STARTING
*AT THE HIGH HALF CACHE ADDRESS, EACH LOC IS READ, TESTED
*TO BE A MISS (VALID=0) AND THEN WRITTEN TO SET THE VALID
*BIT.  THIS IS DONE, DECREASING THE ADDRESS EACH TIME
*TILL THE LOW ADDRESS IS REACHED.  THIS PROCEDURE IS THEN
*REPEATED FOR A SECOND PASS WITH THE PATTERN REVERSED.
*(I.E. STARTING WITH ALL LOC INVALIDATED AND THEN READING
*AND WRITING THE V BIT.)
*
*RO CONTAINS THE CACHE ADDRESS BEING TESTED.

```

507	014100	012737	000214	177746	TST20: MOV	#214,#CCR		;CACHE OFF FOR SCOPE
508	014106	000004			SCOPE			
509	014110	012737	015000	001234	MOV	#TST21,SKTST		;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
510	014116	012737	000204	177746	MOV	#204,#CCR		;HALF CACHE ON
511	014124	012700	062000		MOV	#BUFH,RO		;INIT STARTING ADDRESS
512	014130	012701	001000		MOV	#1000,R1		;INIT COUNT FOR 1/2 K
513	014134	005020			15: CLR	(RO)+		;WRITE CACHE
514	014136	077102			SQB	R1,15		;LOOP TILL HALF CACHE WRITTEN
515								
516	014140	005740			T24H20: TST	-(RO)		;SEE IF DATA IN CACHE
517	014142	033727	177752	000004	BIT	#HMR,#HMR2		;HIT? (VALID BIT SET?)
518	014150	001002			BNE	T24H19		;BRANCH IF YES
519	014152	000137	014670		JMP	T24H01		;REPORT ERROR
520	014156	020027	062000		T24H19: CMP	RO,#BUFH		;HALF CACHE TESTED?

```

2521 014162 001366 BNE T24H20 ;BRANCH IF NO
2522
2523 014164 012737 014242 000114 MOV #T24H02,#PVEC ;SET UP PARITY HANDLER
2524 014172 012737 000204 177746 T24H21: MOV #204,#CCR ;CACHE ON IF OFF
2525 014200 005710 T24H05: TST (R0) ;SEE IF VALID BIT SET
2526 014202 033727 177752 000004 BIT #HMR,#HMR2 ;HIT? (VALID BIT SET?)
2527 014210 001002 BNE T24H22 ;BRANCH IF YES
2528 014212 000137 014712 JMP T24H03 ;REPORT ERROR
2529
2530 014216 012737 000304 177746 T24H22: MOV #304,#CCR ;CACHE ON IF OFF AND WRITE WRONG PARITY
2531 014224 005010 CLR (R0) ;WRITE LOC WITH WRONG PARITY
2532 014226 012737 000204 177746 MOV #204,#CCR ;WMP OFF
2533 014234 005710 TST (R0) ;FORCE PARITY TRAP
2534 014236 000137 014734 JMP T24H04 ;REPORT ERROR IF DID NOT TRAP
2535
2536 014242 T24H02:
2537
2538 014242 010046 MOV RO,-(SP) ;SAVE RO FOR MED INST
2539 014244 076600 MED ;GET CONTENTS OF LOG REG
2540 014246 000022 .WORD RLOG
2541 014250 052700 100001 BIS #100001,RO ;ENABLE ERROR LOG & LOG FIRST MODE
2542 014254 076600 MED ;UNLOCK ERROR LOG
2543 014256 000222 .WORD WLOG
2544 014260 012600 MOV (SP)+,RO ;RESTORE RO
2545
2546 014262 062700 000002 ADD #2,RO ;LOOK AT NEXT ADDR.
2547 014266 062706 000004 ADD #4,SP ;RESTORE STACK
2548 014272 020027 064000 CMP RO,#BUFH+2000 ;HALF ADDRESSES TESTED?
2549 014276 001340 BNE T24H05 ;BRANCH IF NO
2550
2551 014300 012737 033142 000114 MOV #UPERR,#PVEC ;RESTORE UNEXP. PARITY ERROR HANDLER
2552 014306 005740 1S: TST -(R0) ;WAS LOC INVALIDATED?
2553 014310 033727 177752 000004 BIT #HMR,#HMR2 ;LOC A MISS? (INVALIDATED?)
2554 014316 001402 BEQ 2S ;BRANCH IF YES
2555 014320 000137 014756 JMP T24H06 ;REPORT ERROR
2556 014324 005010 2S: CLR (R0) ;WRITE LOC
2557 014326 020027 062000 CMP RO,#BUFH ;AT LAST LOC?
2558 014332 001365 BNE 1S ;BRANCH IF NO
2559
2560 ;NOW WRITE/READ VALID BIT WITH PATTERN REVERSED
2561
2562 014334 012737 014404 000114 T24H10: MOV #T24H07,#PVEC ;SET UP FOR PARITY TRAP
2563 014342 012700 063776 MOV #BUFH+1776,RO ;INIT TEST ADDR.
2564 014346 012737 000304 177746 T24H08: MOV #304,#CCR ;WRITE WRONG PARITY & CACHE ON
2565 014354 005010 CLR (R0) ;WRITE WRONG PARITY & CACHE ON
2566 014356 012737 000204 177746 MOV #204,#CCR ;WMP OFF
2567 014364 005710 TST (R0) ;FORCE TRAP
2568 014366 012737 000214 177746 MOV #214,#CCR ;CACHE OFF
2569 014374 012737 014346 001110 MOV #T24H08,#SLPERR ;INIT RETURN FOR ERROR LOOP
2570 014402 000557 BR T24H15 ;REPORT ERROR IF DID NOT TRAP
2571
2572 014404 T24H07:
2573
2574 014404 010046 MOV RO,-(SP) ;SAVE RO FOR MED INST
2575 014406 076600 MED ;GET CONTENTS OF LOG REG
2576 014410 000022 .WORD RLOG

```

K05

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 47
T20 TEST THE VALID BIT FOR HIGH HALF OF CACHE

```

2577 014412 052700 100001      BIS      #100001,RO      ;ENABLE ERROR LOG & LOG FIRST MODE
2578 014416 076600              MED              ;UNLOCK ERROR LOG
2579 014420 000222      .WORD     WLOG
2580 014422 012600      MOV      (SP)+,RO      ;RESTORE RO
2581
2582 014424 162700 000002      SUB      #2,RO        ;LOOK AT NEXT ADDRESS
2583 014430 062706 000004      ADD      #4,SP        ;ADJUST STACK
2584 014434 020027 061776      CMP      RO,#BUFH-2   ;HALF CACHE WRITTEN?
2585 014440 001342      BNE     T24H08        ;BRANCH IF NO
2586
2587 014442 012737 033142 000114      MOV      #UPERR,#PVEC
2588 014450 062700 000002      T24H12: ADD      #2,RO        ;ADJUST ADDRESS
2589 014454 005710              TST      (RO)         ;READ LOC
2590 014456 033727 177752 000004      BIT      @#HMR,#HMR2  ;MISS? (LOC INVALIDATED?)
2591 014464 001407              BEQ      T24H09        ;BRANCH IF YES
2592 014466 012737 000214 177746      MOV      #214,#CCR    ;CACHE OFF
2593 014474 012737 014334 001110      MOV      #T24H10,#SLPERR ;INIT RETURN FOR ERROR LOOP
2594 014502 000525              BR       T24H06        ;REPORT ERROR
2595
2596 014504 005010              T24H09: CLR      (RO)         ;WRITE LOC
2597 014506 020027 063776      CMP      RO,#BUFH+1776 ;HALF CACHE WRITTEN?
2598 014512 001356              BNE     T24H12        ;BRANCH IF NO
2599
2600              ;NOW READ LOC TO SEE IF VALID STILL SET
2601
2602 014514 012737 014610 000114      MOV      #T24H16,#PVEC ;SET UP PARITY HANDLER
2603 014522 005710              T24H17: TST      (RO)         ;LOC IN CACHE?
2604 014524 033727 177752 000004      BIT      @#HMR,#HMR2  ;HIT?
2605 014532 001007              BNE     T24H13        ;BRANCH IF YES
2606 014534 012737 000214 177746      MOV      #214,#CCR    ;CACHE OFF
2607 014542 012737 014334 001110      MOV      #T24H10,#SLPERR ;INIT RETURN FOR ERROR LOOP
2608 014550 000463              BR       T24H14        ;REPORT ERROR
2609
2610 014552 052737 000100 177746      T24H13: BIS      #100,#CCR ;SET WRITE WRONG PARITY
2611 014560 005010              CLR      (RO)         ;WRITE WRONG PARITY
2612 014562 012737 000204 177746      MOV      #204,#CCR    ;WMP OFF
2613 014570 005710              TST      (RO)         ;FORCE TRAP
2614 014572 012737 000214 177746      MOV      #214,#CCR    ;CACHE OFF
2615 014600 012737 014334 001110      MOV      #T24H10,#SLPERR ;
2616 014606 000455              BR       T24H15        ;REPORT ERROR
2617
2618 014610 062706 000004              T24H16: ADD      #4,SP        ;RESTORE STACK
2619 014614 162700 000002      SUB      #2,RO        ;LOOK AT NEXT ADDR.
2620
2621 014620 010046              MOV      RO,-(SP)     ;SAVE RO FOR MED INST
2622 014622 076600              MED              ;GET CONTENTS OF LOG REG
2623 014624 000022      .WORD     RLOG
2624 014626 052700 100001      BIS      #100001,RO   ;ENABLE ERROR LOG & LOG FIRST MODE
2625 014632 076600              MED              ;UNLOCK ERROR LOG
2626 014634 000222      .WORD     WLOG
2627 014636 012600      MOV      (SP)+,RO    ;RESTORE RO
2628
2629 014640 020027 061776      CMP      RO,#BUFH-2   ;ALL ADDR TESTED?
2630 014644 001326      BNE     T24H17        ;BRANCH IF NO
2631
2632 014646              T24H18:

```

```

2633
2634
2635 014646 012737 000214 177746      :RID CACHE OF BAD PARITY
2636 014654 004737 035134      MOV      #214, @#CCR      ;CACHE OFF IF ON
2637                                     JSR      PC, SWEEP      ;GO PURGE CACHE
2638
2639 014660 012737 033142 000114      MOV      #UPERR, @#PVEC
2640 014666 000444                                     BR      TST21          ;;GO TO NEXT TEST
2641
2642 014670 012737 000214 177746 T24H01: MOV      #214, @#CCR      ;CACHE OFF
2643 014676 005037 001160      CLR      $REG1          ;SAVE FAILING ADDR
2644 014702 010037 001162      MOV      R0, $REG2      ;SAVE FAILING ADDR
2645 014706 104043      ERROR   43              ;ERROR: ADDRESS COULD NOT BE MADE A HIT
2646 014710 000756      BR      T24H18          ;GO TO END OF TEST
2647
2648 014712 012737 000214 177746 T24H03: MOV      #214, @#CCR      ;CACHE OFF
2649 014720 005037 001160 T24H14: CLR      $REG1          ;SAVE FAILING ADDRESS
2650 014724 010037 001162      MOV      R0, $REG2      ;SAVE FAILING ADDRESS
2651 014730 104111      ERROR   111            ;ERROR: TEST OF VALID BIT FAILED
2652                                     ;LOC COULD NOT BE MADE A HIT
2653 014732 000745      BR      T24H18          ;GO TO END OF TEST
2654
2655 014734 012737 000214 177746 T24H04: MOV      #214, @#CCR      ;CACHE OFF
2656 014742 005037 001160 T24H15: CLR      $REG1          ;SAVE FAILING ADDRESS
2657 014746 010037 001162      MOV      R0, $REG2      ;SAVE FAILING ADDRESS
2658 014752 104042      ERROR   42              ;ERROR: NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PARIT
2659 014754 000734      BR      T24H18          ;GO TO END OF TEST
2660
2661 014756 012737 000214 177746 T24H06: MOV      #214, @#CCR      ;CACHE OFF
2662 014764 005037 001160      CLR      $REG1          ;SAVE FAILING ADDR
2663 014770 010037 001162      MOV      R0, $REG2      ;SAVE FAILING ADDR
2664 014774 104112      ERROR   112            ;ERROR: TEST OF VALID BIT FAILED
2665                                     ;LOCATION NOT INVALIDATED BY PARITY TRAP
2666 014776 000723      BR      T24H18          ;GO TO END OF TEST
2667

```

```

2668 *****
2669 *TEST 21      TEST TAG PARITY BIT FOR HIGH CACHE ADDRESSES
2670 *
2671 *
2672 * THE TEST OF THE TAG PARITY BIT IS NOT COMPLETE UNTIL
2673 *THE TAG P BIT TEST FOR THE SECOND HALF OF CACHE AND THE
2674 *MSB ADDRESS (A10) TO CACHE TAG FIELD TEST ARE RUN. TWO
2675 *TAG ADDRESSES ARE USED TO GENERATE A PARITY BIT OF 1 AND
2676 *0. THE FIRST ADDRESS IS CHOSEN FROM A TEST BUFFER AREA
2677 *AND THE SECOND IS CHOSEN TO LIE 1K AWAY. A WRITE/READ
2678 *PROCEDURE IS DONE WHICH CHECKS THE P BIT AND DUAL ADD-
2679 *RESSING FOR HALF OF CACHE. INITIALLY THE P BIT IS WRITTEN
2680 *WITH ONE PARITY PATTERN IN HALF OF CACHE. THEN STARTING
2681 *AT THE LOW HALF CACHE ADDRESS, THE LOC IS READ AND THEN
2682 *WRITTEN WITH THE OPPOSITE PARITY. THIS IS SEQUENTIALLY
2683 *REPEATED WITH INCREASING ADDRESSES UNTIL THE HIGH HALF
2684 *CACHE ADDRESS IS REACHED. THEN STARTING AT THE HIGH ADDR,
2685 *THE SECOND PARITY PATTERN IS READ AND THE LOC IS REWRITTEN
2686 *WITH THE FIRST. THIS IS SEQUENTIALLY REPEATED, DECREASING
2687 *THE ADDRESS, UNTIL THE LOW HALF CACHE ADDRESS IS REACHED.
2688 *A SECOND PASS IS THEN MADE WITH THE PARITY PATTERN RE-

```


M05

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
 DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 49
 T21 TEST TAG PARITY BIT FOR HIGH CACHE ADDRESSES

```

;#VERSED. A PARITY ERROR HANDLER IS SETUP TO DETECT PARITY
;#ERRORS. ALSO, LOCS WHICH SHOULD BE HITS ARE CHECKED FOR
;#AND REPORTED IF NO HIT OCCURRED.
;#
;#RO, R1 CONTAIN ADDRESSES TO GENERATE COMPLIMENTARY TAG
;#PARITY BITS.
    
```

```

2689
2690
2691
2692
2693
2694
2695
2696
2697 015000 012737 000214 177746 TST21: MOV #214, @#CCR ;CACHE OFF FOR SCOPE
2698 015006 000004 SCOPE
2699 015010 012737 016000 001234 MOV #TST22 SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
2700 015016 012737 015172 000114 MOV #T11H01, @#PVEC ;SET UP FOR PARITY ERRORS
2701 015024 005003 CLR R3 ;INIT FLAG=FIRST PASS
2702 015026 012700 062000 MOV #BUFH, R0 ;SET UP ADDR. FOR FIRST PASS
2703 015032 012737 000204 177746 MOV #204, @#CCR ;TURN HALF CACHE ON
2704 015040 012701 001000 T11H02: MOV #1000, R1 ;INIT COUNTER
2705 015044 005720 IS: TST (R0)+ ;PUT PARITY PATTERN IN TAG FIELD
2706 015046 077102 SOB R1, IS ;LOAD HALF OF CACHE
2707
2708 015050 012701 001000 MOV #1000, R1 ;INIT. COUNTER
2709 015054 012700 062000 MOV #BUFH, R0 ;SET UP ADDR. FOR FIRST PASS
2710 015060 012702 056000 MOV #BUFH-4000, R2 ;SET UP ADDR. FOR FIRST PASS
2711 015064 005703 TST R3 ;FIRST PASS?
2712 015066 001404 BEQ T11H03 ;BRANCH IF YES
2713 015070 012700 056000 MOV #BUFH-4000, R0 ;SET UP ADDR. FOR SECOND PASS
2714 015074 012702 062000 MOV #BUFH, R2 ;SET UP ADDR. FOR SECOND PASS
2715 015100 005720 T11H03: TST (R0)+ ;READ CACHE TO SEE IF PARITY OK; NO-TRAPS
2716 015102 033727 177752 000004 BIT @#HMR, #HMR2 ;WAS ADDRESS A HIT?
2717 015110 001533 BEQ T11H04 ;BRANCH TO ERROR IF NO
2718 015112 005722 TST (R2)+ ;WRITE DIFFERENT PARITY PATTERN IN TAG FIELD
2719 015114 077107 SOB R1, T11H03 ;LOOK AT HALF OF CACHE
2720
2721 015116 012701 001000 T11H11: MOV #1000, R1 ;INIT COUNTER
2722 015122 005742 TST -(R2) ;READ SECOND PARITY PATTERN
2723 015124 033727 177752 000004 BIT @#HMR, #HMR2 ;WAS ADDRESS A HIT?
2724 015132 001532 BEQ T11H05 ;BRANCH IF NO TO ERROR
2725 015134 005740 TST -(R0) ;PUT NEW PARITY PATTERN IN TAG
2726 015136 077107 SOB R1, T11H11 ;LOOK AT HALF OF CACHE
2727
2728 015140 005703 TST R3 ;FIRST PASS?
2729 015142 001140 BNE T11H06 ;NO GO TO END OF TEST
2730 015144 052703 000001 BIS #1, R3 ;SET FLAG TO INDIC. SECOND PASS
2731 015150 012737 000204 177746 T11H12: MOV #204, @#CCR ;HALF CACHE ON IF OFF
2732 015156 012737 015150 001110 MOV #T11H12, @#SLPERR ;SETUP RETURN FOR ERROR IF ONE OCCURS
2733 015164 012700 056000 MOV #BUFH-4000, R0 ;SET UP FOR SECOMO PASS.
2734 015170 000723 BR T11H02 ;GO TEST SECOND PASS
2735
2736 015172 T11H01:
2737
2738 ;RID CACHE OF BAD PARITY
2739 015172 012737 000214 177746 MOV #214, @#CCR ;CACHE OFF IF ON
2740 015200 004737 035134 JSR PC, SWEEP ;GO PURGE CACHE
2741
2742
2743
2744 015204 010046 MOV RO, -(SP) ;SAVE RO FOR MED INST
    
```

2745	015206	076600				MED			;GET CONTENTS OF LOG REG
2746	015210	000022				.WORD	RLOG		
2747	015212	052700	100001			BIS	#100001,RO		;ENABLE ERROR LOG & LOG FIRST MODE
2748	015216	076600				MED			;UNLOCK ERROR LOG
2749	015220	000222				.WORD	WLOG		
2750	015222	012600				MOV	(SP)+,RO		;RESTORE RO
2751									
2752	015224	076600				MED			;GET LOG INFOR FOR PHY. ADDR. A17,A16
2753	015226	000101				.WORD	RSER		
2754	015230	000300				SWAB	RO		;PUT PHY. ADDR A17, A16 IN LOW BYTE
2755	015232	042700	177776			BIC	#177776,RO		;ONLY LOOK AT A17, A16
2756	015236	010037	001160			MOV	RO,\$REG1		;SAVE ADDRESS
2757	015242	076600				MED			;GET LOG INFORMATION
2758	015244	000102				.WORD	LOADD		
2759	015246	010037	001162			MOV	RO,\$REG2		;SAVE INFORMATION
2760	015252	076600				MED			;GET LOG INFORMATION
2761	015254	000100				.WORD	RJAM		
2762	015256	032700	000400			BIT	#400,RO		;ERROR IN BACKING STORE?
2763	015262	001410				BEQ	T11H07		;BRANCH IF NO
2764	015264	011637	001164			MOV	(SP),\$REG3		;GET PC+2 WHERE ERROR OCCURRED
2765	015270	162737	000002	001166		SUB	#2,\$REG4		;SAVE PC WHERE ERROR OCCURRED
2766	015276	022626				CMP	(SP)+,(SP)+		;RESTORE STACK
2767	015300	104001				ERROR	1		;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
2768	015302	000460				BR	T11H06		;GO TO NEXT TEST
2769									
2770	015304	022626			T11H07:	CMP	(SP)+,(SP)+		;RESTORE STACK
2771	015306	032737	000040	177744		BIT	#40,\$REG		;ERROR IN TAG?
2772	015314	001411				BEQ	T11H08		;BRANCH NO
2773	015316	076600				MED			;GET TAG LOG INFO.
2774	015320	000107				.WORD	RTAG		
2775	015322	000300				SWAB	RO		;PUT TAG IN LOW BYTE
2776	015324	042700	177400			BIC	#177400,RO		;LOOK AT TAG ONLY
2777	015330	010037	001164			MOV	RO,\$REG3		;SAVE BAD DATA
2778	015334	104045				ERROR	45		;ERROR: TAG PARITY ERROR WHEN TESTING TAG P BIT
2779	015336	000442				BR	T11H06		;GO TO NEXT TEST
2780									
2781	015340	032737	000100	177744	T11H08:	BIT	#100,\$REG		;ERROR IN LOW BYTE?
2782	015346	001406				BEQ	T11H09		;BRANCH IF NO
2783	015350	076600				MED			;GET LOG INFORMATION
2784	015352	000106				.WORD	CDL		
2785	015354	010037	001164			MOV	RO,\$REG3		;SAVE INFORMATION
2786	015360	104046				ERROR	46		;ERROR: LOW BYTE PARITY ERROR WHEN TESTING TAG P BIT
2787	015362	000430				BR	T11H06		;NEXT TEST
2788									
2789	015364				T11H09:				
2790	015364	076600				MED			;GET LOG INFORMATION
2791	015366	000106				.WORD	CDH		
2792	015370	010037	001164			MOV	RO,\$REG3		;SAVE INFORMATION
2793	015374	104047				ERROR	47		;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG P BIT
2794	015376	000422				BR	T11H06		;NEXT TEST
2795									
2796	015400	052737	000014	177746	T11H04:	BIS	#14,\$CCR		;CACHE OFF
2797	015406	162700	000002			SUB	#2,RO		;GET BAD ADDRESS
2798	015412	010037	001162			MOV	RO,\$REG2		;SAVE BAD ADDRESS
2799	015416	000407				BR	T11H10		;REPORT ERROR
2800	015420	052737	000014	177746	T11H05:	BIS	#14,\$CCR		;CACHE OFF

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 51
T21 TEST TAG PARITY BIT FOR HIGH CACHE ADDRESSES

2801	015426	010237	001162		MOV	R2,\$REG2	;SAVE BAD ADDRESS	
2802	015432	062702	000002		ADD	#2,R2	;RESTORE R2 TO FAILING ADDR.+2	
2803	015436	005037	001160	T11H10:	CLR	\$REG1	;SAVE BAD ADDRESS	
2804	015442	104043			ERROR	43	;ERROR: ADDRESS COULD NOT BE MADE A HIT	
2805								
2806	015444	012737	033142	000114	T11H06:	MOV	#UPERR,#PVEC	;RESTORE PARITY TRAP HANDLER
2807	015452	052737	000014	177746	BIS	#14,#CCR	;CACHE OFF WHEN CROSS CACHE ADDR. BOUNDARY	
2808	015460	000547			BR	TST22	;GO TO NEXT TEST	

2809							
2810							
2811	016000				.=16000		;ADJUST ADDRESS SPACE FOR NEXT TEST
2812							

```

*****
*TEST 22 TEST TAG ADDRESS BITS FOR LOW HALF OF CACHE
*
* THE TEST OF THE TAG BITS IS NOT COMPLETE UNTIL THE
*TAG ADDRESS TEST FOR THE OTHER HALF OF CACHE AND THE
*TEST OF THE MSB ADDRESS (A10) TO THE CACHE TAG FIELD
*ARE RUN. A WRITE/READ PROCEDURE IS DONE WHICH CHECKS
*THE TAG FIELD BITS AND DUAL ADDRESSING ON THEM FOR HALF
*OF CACHE. MEMORY IS FIRST SIZED TO DETERMINE THE MAX-
*IMUM TESTABLE ADDRESS. THE TAG ADDRESS BITS OF THIS
*ADDRESS ARE USED AS PATTERN A AND STORED IN KIPAR4. A
*PATTERN B IS NOW GENERATED WHICH HAS 'COMPLEMENT' TAG
*BITS AND STORED IN KIPAR5. ON THE FIRST PASS, PATTERN
*A IS WRITTEN THROUGH HALF OF CACHE. NEXT, STARTING AT
*THE HIGH HALF CACHE ADDRESS, THE LOCATION IS READ,
*CHECKED TO BE A HIT AND THEN WRITTEN WITH PATTERN B.
*THIS IS SEQUENTIALLY REPEATED WITH DECREASING ADDRESSES
*UNTIL THE LOW HALF CACHE ADDRESS IS REACHED. AT THE
*LOW ADDRESS, THE SECOND PATTERN IS READ, CHECKED TO BE A
*HIT AND REWRITTEN WITH THE FIRST PATTERN. THIS IS SE-
*QUENTIALLY REPEATED WITH INCREASING ADDRESSES UNTIL THE
*HIGH HALF CACHE ADDRESS IS REACHED. A SECOND PASS IS
*THEN MADE WITH THE PATTERNS REVERSED.
* ANY PARITY ERROR OR HIT ERROR IS REPORTED.
* DURING THE PASSES, R0, R1 CONTAIN ADDRESSES WHICH
*REFERENCE KIPAR5.
* R3 INDICATES THE PASS NUMBER.
*IF THE INHIBIT TESTS USING KT SWITCH (SW12) IS SET, THIS
*TEST IS SKIPPED.

```

2844	016000	012737	000214	177746	TST22:	MOV	#214,#CCR	;CACHE OFF FOR SCOPE
2845	016006	000004				SCOPE		
2846	016010	012737	016646	001234		MOV	#TST23,SKTST	;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
2847	016016	032777	010000	163110		BIT	#SW12,#SWR	;INHIBIT TESTS USING KT?
2848	016024	001402				BEQ	35	;CONTINUE TEST IF NO
2849	016026	000137	016646			JMP	#TST23	;GO TO NEXT TEST
2850	016032	052737	000200	036034	35:	BIS	#200,#SKT11	;KT ON FOR SSIZE
2851	016040	004737	035750			JSR	PC,\$SIZE	;SIZE MEMORY
2852	016044	012737	016266	000114		MOV	#T13L01,#PVEC	;SET UP PARITY ERROR HANDLER
2853	016052	013737	036322	172350		MOV	#SLSTBK,#KIPAR4	;SET UP PAR4 FOR ADDRESS PATTERN A
2854								
2855								
2856								

;CALC COMPLEMENT TAG PATTERN B

2857	016060	013700	036322		MOV	%SLSTBK,R0	;GET TEST PATTERN A AND	
2858	016064	005100			COM	R0	;CALC PATTERN B	
2859	016066	005001			CLR	R1		
2860	016070	005201		1S:	INC	R1		
2861	016072	006300			ASL	R0		
2862	016074	100775			BMI	1S		
2863	016076	006200		2S:	ASR	R0		
2864	016100	077102			SQB	R1,2S		
2865	016102	042700	000037		BIC	%37,R0	;ONLY COMPLEMENT TAG ADDR. BITS	
2866								
2867	016106	010037	172352		MOV	R0,%KIPARS	;SET UP PARS FOR ADDRESS PATTERN B	
2868								
2869	016112	012700	100000		MOV	%100000,R0	;INIT R0 TO ADD PATTERN A	
2870	016116	012701	122000		MOV	%122000,R1	;INIT R1 TO ADD PATTERN B	
2871	016122	005003			CLR	R3	;INIT FLAG FOR PASS 1	
2872	016124	005004		T13L02:	CLR	R4	;INIT INDICATOR FOR ERROR LOOP 1	
2873	016126	012702	001000		MOV	%1000,R2	;INIT ADDR. COUNTER	
2874	016132	052737	000001	177572	BIS	%1,%MMR0	;TURN KT ON	
2875	016140	012737	000210	177746	MOV	%210,%CCR	;TURN HALF OF CACHE ON	
2876								
2877	016146	005720		1S:	TST	(R0)+	;WRITE PATTERN IN CACHE	
2878	016150	077202			SQB	R2,1S	;ALL DONE? BRANCH IF NO	
2879								
2880	016152	012702	001000		MOV	%1000,R2	;INIT. ADDR. COUNTER	
2881	016156	005740		T13L03:	TST	-(R0)	;READ CACHE TAG BITS	
2882	016160	033727	177752	000004	BIT	%MMR,%MMR2	;HIT?	
2883	016166	001002			BNE	2S	;BRANCH IF YES	
2884	016170	000137	016540		JMP	T13L04	;REPORT ERROR	
2885	016174	005741		2S:	TST	-(R1)	;WRITE NEW PATTERN IN TAG	
2886	016176	077211			SQB	R2,T13L03	;HALF ADDR. TESTED? BRANCH IF NO	
2887								
2888	016200	005204			INC	R4	;SET INDICATOR FOR ERROR LOOP 2	
2889	016202	012702	001000		MOV	%1000,R2	;INIT. ADDR. COUNTER	
2890	016206	005711		T13L05:	TST	(R1)	;READ CACHE TAG BITS	
2891	016210	033727	177752	000004	BIT	%MMR,%MMR2	;HIT?	
2892	016216	001002			BNE	3S	;BRANCH IF YES	
2893	016220	000137	016606		JMP	T13L06	;REPORT ERROR	
2894	016224	005721		3S:	TST	(R1)+	;UPDATE FOR NEXT ADDRESS	
2895	016226	005720			TST	(R0)+	;WRITE NEW PATTERN IN TAG	
2896	016230	077212			SQB	R2,T13L05		
2897								
2898	016232	005703			TST	R3	;SECOND PASS?	
2899	016234	001402			BEQ	2S	;CONTINUE TEST IF NO	
2900	016236	000137	016634		JMP	T13L07	;GO TO END OF TEST	
2901	016242	052703	000001	001110	2S:	BIS	%1,R3	;SET FLAG FOR SECOND PASS
2902	016246	012737	016254		MOV	%T13L15,%SLPERR	;INIT RETURN FOR ERROR LOOP IF ERROR OCCURS	
2903	016254	012700	120000	T13L15:	MOV	%120000,R0	;INIT. R0 TO ADDR. PATTERN B	
2904	016260	012701	102000		MOV	%102000,R1	;INIT. R1 TO ADDR. PATTERN A	
2905	016264	000717			BR	T13L02	;GO TEST SECOND PASS	
2906								
2907	016266	052737	000014	177746	T13L01:	BIS	%14,%CCR	;CACHE OFF
2908								
2909	016274	010046			MOV	R0,-(SP)	;SAVE R0 FOR MED INST	
2910	016276	076600			MED		;GET CONTENTS OF LOG REG	
2911	016300	000022			.WORD	RLOG		
2912	016302	052700	100001		BIS	%100001,R0	;ENABLE ERROR LOG & LOG FIRST MODE	

2913	016306	076600				MED			;UNLOCK ERROR LOG
2914	016310	000222				.WORD	WLOG		
2915	016312	012600				MOV	(SP)+,RO		;RESTORE RO
2916									
2917	016314	011637	001164			MOV	(SP), \$REG3		;GET PC+2 OF TRAP
2918	016320	162737	000002	001164		SUB	#2, \$REG3		;SAVE PC FOR MAIN PARITY ERROR
2919	016326	022626				CMP	(SP)+, (SP)+		;RESTORE STACK
2920	016330	010046				MOV	RO, -(SP)		;SAVE RO ON STACK FOR MED INST.
2921	016332	076600				MED			;GET LOG INFOR FOR PHY. ADDR. A17,A16
2922	016334	000101				.WORD	RSER		
2923	016336	000300				SWAB	RO		;PUT PHY. ADDR A17, A16 IN LOW BYTE
2924	016340	042700	177776			BIC	#177776, RO		;ONLY LOOK AT A17, A16
2925	016344	010037	001160			MOV	RO, \$REG1		;SAVE ADDRESS
2926	016350	076600				MED			;GET LOG INFORMATION
2927	016352	000102				.WORD	LOADD		
2928	016354	010037	001162			MOV	RO, \$REG2		;SAVE INFORMATION
2929	016360	076600				MED			;GET LOG INFORMATION
2930	016362	000100				.WORD	RJAM		
2931	016364	012600				MOV	(SP)+, RO		;RESTORE RO
2932	016366	032700	000400			BIT	#400, RO		;ERROR BACKING STORE?
2933	016372	001402				BEQ	T13L08		;BRANCH IF NO
2934	016374	104001				ERROR	1		;ERROR: UNEXPECT. PARITY ERROR IN BACKING STORE
2935	016376	000516				BR	T13L07		;GO TO END OF TEST
2936									
2937	016400	011137	001166		T13L08:	MOV	(R1), \$REG4		;SAVE GOOD DATA
2938	016404	005704				TST	R4		;ERROR IN LOOP 2?
2939	016406	001002				BNE	T13L09		;BRANCH IF YES
2940	016410	011037	001166			MOV	(RO), \$REG4		;SAVE GOOD DATA
2941									
2942	016414	032737	000040	177744	T13L09:	BIT	#40, 2#EREG		;TAG PARITY ERROR?
2943	016422	001426				BEQ	T13L10		;BRANCH IF NO
2944	016424	004737	033634			JSR	PC, PAR		;GET PAR USED
2945	016430	000000				.WORD	0		;INDICATOR FOR RO
2946	016432	005704				TST	R4		;ERROR FROM LOOP 1?
2947	016434	001403				BEQ	T13L11		;BRANCH IF YES
2948	016436	004737	033634			JSR	PC, PAR		;GET PAR USED
2949	016442	000001				.WORD	1		;INDICATOR FOR R1
2950	016444	004737	033606		T13L11:	JSR	PC, TAG		;CALC TAG CONTENTS
2951	016450	013737	001172	001166		MOV	STMPD, \$REG4		;SAVE GOOD DATA
2952	016456	076600				MED			;GET TAG LOG INFO.
2953	016460	000107				.WORD	RTAG		
2954	016462	000300				SWAB	RO		;PUT TAG IN LOW BYTE
2955	016464	042700	177400			BIC	#177400, RO		;LOOK AT TAG ONLY
2956	016470	010037	001164			MOV	RO, \$REG3		;SAVE BAD DATA
2957	016474	104052				ERROR	52		;ERROR: TAG PARITY ERROR ON TEST OF TAG ADDRESS BITS
2958	016476	000456				BR	T13L07		;GO TO END OF TEST
2959									
2960	016500	032737	000100	177744	T13L10:	BIT	#100, 2#EREG		;LOW BYTE P.E.?
2961	016506	001406				BEQ	T13L12		;BRANCH IF NO
2962	016510	076600				MED			;GET LOG INFORMATION
2963	016512	000106				.WORD	CDL		
2964	016514	010037	001164			MOV	RO, \$REG3		;SAVE INFORMATION
2965	016520	104053				ERROR	53		;ERROR: LOW BYTE PARITY ERROR ON TEST OF TAG ADDR. BITS
2966	016522	000444				BR	T13L07		;GO TO END OF TEST
2967									
2968	016524				T13L12:				

E06

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 54
T22 TEST TAG ADDRESS BITS FOR LOW HALF OF CACHE

```

2969 016524 076600 MED ;GET LOG INFORMATION
2970 016526 000106 .WORD CDH
2971 016530 010037 001164 MOV RO,$REG3 ;SAVE INFORMATION
2972 016534 104054 ERROR 54 ;ERROR: HIGH BYTE PARITY ERROR ON TEST OF TAG ADDR. BITS
2973 016536 000436 BR T13L07 ;GO TO END OF TEST
2974
2975 016540 052737 000014 177746 T13L04: BIS #14,$CCR ;CACHE OFF
2976 016546 010037 001172 MOV RO,$TMP0 ;GET VIRTUAL ADDRESS TESTED
2977 016552 004737 033434 JSR PC,VIP ;SAVE ADDRESS TESTED
2978 016556 062700 000002 ADD #2,RO ;ADJUST ADDRESS WHEN LOOP
2979 016562 004737 033634 JSR PC,PAR ;GET PAR TESTED
2980 016566 000000 .WORD 0 ;INDICATOR FOR RO
2981 016570 004737 033606 T13L13: JSR PC,TAG ;CALC TAG FROM PAR
2982 016574 013737 001172 001164 MOV $TMP0,$REG3 ;SAVE TAG
2983 016602 104055 ERROR 55 ;ERROR: TEST OF TAG ADDRESS BITS FAILED
2984 ; ADDR. COULD NOT BE MADE A HIT
2985 016604 000413 BR T13L07 ;GO TO NEXT TEST
2986
2987 016606 052737 000014 177746 T13L06: BIS #14,$CCR ;CACHE OFF
2988 016614 010137 001172 MOV R1,$TMP0 ;GET VIRTUAL ADDRESS TESTED
2989 016620 004737 033434 JSR PC,VIP ;SAVE PHYSICAL ADDRESS TESTED
2990 016624 004737 033634 JSR PC,PAR ;GET PAR TESTED
2991 016630 000001 .WORD 1 ;INDICATOR FOR R1
2992 016632 000756 BR T13L13 ;REPORT ERROR
2993
2994
2995 016634 005037 177572 T13L07: CLR $MMRO ;KT OFF
2996 016640 012737 033142 000114 MOV $UPERR,$PVEC ;RESTORE UNEXP. PARITY ERROR HANDLER
2997
2998
2999 ;*****
3000 ;*TEST 23 TEST OF CACHE DATA LOC WITH FLOAT 1 & 0 PATTERNS
3001 ;*
3002 ;* THIS TEST MAKES TWO PASSES. ON THE FIRST, A FLOAT
3003 ;*'1' PATTERN IS WRITTEN/READ FROM ONE CACHE LOC. ON THE
3004 ;*SECOND, A FLOAT '0' PATTERN IS WRITTEN/READ FROM ONE
3005 ;*CACHE LOC. THERE IS A HANDLER FOR PARITY ERRORS. IF
3006 ;*THERE ARE LESS THAN 4 PARITY ERRORS THE TEST CONTINUES.
3007 ;*IF THERE ARE 4 OR MORE PARITY ERRORS THE TEST IS STOPPED.
3008 ;* RO CONTAINS THE DATA PATTERN
3009 ;* R2 CONTAINS THE TEST ADDRESS
3010 ;* R4 IS THE PASS INDICATOR
3011 ;*****
3012 016646 012737 000214 177746 TST23: MOV #214,$CCR ;CACHE OFF FOR SCOPE
3013 016654 000004 SCOPE
3014 016656 012737 020000 001234 MOV #TST24,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3015 016664 012737 016760 000114 MOV #T14L01,$PVEC ;SET UP PARITY ERROR HANDLER
3016 016672 005004 CLR R4 ;CLEAR PASS INDICATOR FOR FIRST PASS
3017 016674 012700 000001 MOV #1,RO ;SET UP FLOAT 1 PATTERN
3018 016700 012702 060000 MOV $BUFL,R2 ;SET UP TEST ADDRESS
3019 016704 012737 000210 177746 T14L02: MOV #210,$CCR ;HALF CACHE ON
3020 016712 010012 T14L06: MOV RO,(R2) ;WRITE CACHE
3021 016714 020012 CMP RO,(R2) ;READ CACHE
3022 016716 001151 BNE T14L03 ;BRANCH TO ERROR IF DATA BAD
3023 016720 005704 T14L10: TST R4 ;FIRST PASS?
3024 016722 001011 BNE T14L04 ;BRANCH IF NO

```

F06

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 55
T23 TEST OF CACHE DATA LOC WITH FLOAT 1 & 0 PATTERNS

```

3025 016724 005700          TST      RO          ;ALL SHIFTS FOR FLOAT 1 PATTERN DONE?
3026 016726 100402          BMI      T14L05      ;BRANCH IF YES
3027 016730 006300          ASL      RO          ;SHIFT FLOAT 1 PATTERN
3028 016732 000767          BR       T14L06      ;TEST IT
3029
3030 016734 052704 000001    T14L05: BIS      #1,R4          ;SET FLAG FOR SECOND PASS
3031 016740 012700 177776    MOV      #177776,RO   ;SET UP FLOAT 0 PATTERN
3032 016744 000762          BR       T14L06      ;GO TEST IT
3033
3034 016746 005700          T14L04: TST      RO          ;ALL SHIFTS FOR FLOAT 0 PATTERN DONE?
3035 016750 100155          BPL      T14L07      ;GO TO END OF TEST IF YES
3036 016752 000261          SEC          ;SET CARRY BIT FOR ROTATE
3037 016754 006100          ROL      RO          ;ROTATE FLOAT 0 PATTERN
3038 016756 000755          BR       T14L06      ;TEST IT
3039
3040 016760 052737 000014 177746  T14L01: BIS      #14,#CCR      ;CACHE OFF
3041
3042 016766 010046          MOV      RO,-(SP)     ;SAVE RO FOR MED INST
3043 016770 076600          MED          ;GET CONTENTS OF LOG REG
3044 016772 000022          .WORD    RLOG
3045 016774 052700 100001    BIS      #100001,RO   ;ENABLE ERROR LOG & LOG FIRST MODE
3046 017000 076600          MED          ;UNLOCK ERROR LOG
3047 017002 000222          .WORD    WLOG
3048 017004 012600          MOV      (SP)+,RO     ;RESTORE RO
3049
3050 017006 011637 001164          MOV      (SP),SREG3   ;GET PC+2 OF ERROR
3051 017012 162737 000002 001164    SUB      #2,SREG3     ;SAVE PC OF ERROR
3052 017020 022626          CMP      (SP)+,(SP)+ ;RESTORE STACK
3053 017022 010046          MOV      RO,-(SP)     ;SAVE RO FOR MED INST
3054 017024 076600          MED          ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3055 017026 000101          .WORD    RSER
3056 017030 000300          SWAB     RO          ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3057 017032 042700 177776    BIC      #177776,RO   ;ONLY LOOK AT A17, A16
3058 017036 010037 001160    MOV      RO,SREG1     ;SAVE ADDRESS
3059 017042 076600          MED          ;GET LOG INFORMATION
3060 017044 000102          .WORD    LOADD
3061 017046 010037 001162    MOV      RO,SREG2     ;SAVE INFORMATION
3062 017052 076600          MED          ;GET LOG INFORMATION
3063 017054 000100          .WORD    RJAM
3064 017056 032700 000400    BIT      #400,RO      ;ERROR IN BACKING STORE?
3065 017062 001403          BEQ      T14L08      ;BRANCH IF NO
3066 017064 010026          MOV      RO,(SP)+     ;RESTORE RO
3067 017066 104001          ERROR    1           ;ERROR: UNEXPECT. PARITY ERROR IN BACKING STORE
3068 017070 000505          BR       T14L07      ;GO TO END OF TEST
3069
3070 017072 011637 001166          T14L08: MOV      (SP),SREG4   ;SAVE GOOD DATA
3071 017076 012737 016704 001110    MOV      #T14L02,#SLPERR ;INIT RETURN FOR ERROR LOOP
3072 017104 032737 000100 177744    BIT      #100,#EREG   ;LOW BYTE PARITY ERROR?
3073 017112 001416          BEQ      T14L09      ;BRANCH IF NO
3074 017114 076600          MED          ;GET LOG INFORMATION
3075 017116 000106          .WORD    CDL
3076 017120 010037 001164    MOV      RO,SREG3     ;SAVE INFORMATION
3077 017124 012600          MOV      (SP)+,RO     ;RESTORE RO
3078 017126 104056          ERROR    56          ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD
3079 017130 123727 001103 000003  T14L12: CMPB     #SERFLG,#3 ;MORE THAN 3 ERRORS?
3080 017136 101062          BHI      T14L07      ;STOP TESTING IF YES

```

```

3081 017140 012737 000210 177746      MOV      #210,2#CCR      ;HALF CACHE ON
3082 017146 000664                      BR        T14L10        ;CONTINUE TEST
3083
3084 017150 033737 000200 177744  T14L09: BIT      200,2#EREG ;HIGH BYTE P.E.?
3085 017156 001407                      BEQ      T14L11        ;BRANCH IF NO
3086 017160 076600                      MED                      ;GET LOG INFORMATION
3087 017162 000106                      .WORD   CDH
3088 017164 010037 001164      MOV      RO,$REG3      ;SAVE INFORMATION
3089 017170 012600                      MOV      (SP)+,RO      ;RESTORE RO
3090 017172 104057                      ERROR   57             ;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA FIELD
3091 017174 000755                      BR        T14L12        ;SEE IF SHOULD CONTINUE TESTING
3092
3093 017176                      T14L11:
3094 017176 076600                      MED                      ;GET LOG INFORMATION
3095 017200 000107                      .WORD   CTAG
3096 017202 010037 001164      MOV      RO,$REG3      ;SAVE INFORMATION
3097 017206 012600                      MOV      (SP)+,RO      ;RESTORE RO
3098 017210 012737 060000 001166      MOV      #BUFL,$REG4   ;GET TESTED ADDRESS
3099 017216 012705 000013                      MOV      #13,R5        ;SETUP COUNTER
3100 017222 006237 001166      25:    ASR      $REG4    ;PUT TAG ADDRESS BITS IN LSB 6-0
3101 017226 077503                      SOB      R5,25         ;SHIFT NINE PLACES
3102 017230 052737 000200 001166      BIS      #200,$REG4    ;SET VALID BIT
3103 017236 104060                      ERROR   60             ;ERROR: TAG PARITY ERROR WHEN TESTING CACHE DATA FIELD
3104 017240 000733                      BR        T14L12        ;SEE IF WANT TO CONTINUE TEST
3105
3106 017242 011205                      T14L03: MOV      (R2),R5 ;GET BAD DATA
3107 017244 052737 000014 177746      BIS      #14,2#CCR     ;CACHE OFF
3108 017252 005037 001160      CLR      $REG1         ;SAVE ADDRESS
3109 017256 010237 001162      MOV      R2,$REG2     ;SAVE ADDRESS
3110 017262 010537 001164      MOV      R5,$REG3     ;SAVE BAD DATA
3111 017266 010037 001166      MOV      RO,$REG4     ;SAVE GOOD DATA
3112 017272 012737 016704 001110      MOV      #T14L02,2#SLPERR ;INIT RETURN FOR ERROR LOOP
3113 017300 104061                      ERROR   61             ;ERROR: CACHE DATA LOC HELD WRONG DATA
3114 017302 000712                      BR        T14L12        ;SEE IF TEST TO BE CONTINUED
3115
3116 017304 012737 033142 000114  T14L07: MOV      #UPERR,2#PVEC ;RESTORE HANDLER FOR UNEXP. PARITY ERRORS
3117 017312 052737 000014 177746      BIS      #14,2#CCR     ;CACHE OFF WHEN CROSS CACHE ADDR. BOUNDARY
3118 017320 000137 020000      JMP      2#TST24      ;GO TO NEXT TEST
3119
3120
3121                      020000                      .=-20000                ;ADJUST ADDRESS SPACE FOR NEXT TEST
3122
3123
3124
3125
3126
3127
3128
3129
3130
3131
3132
3133
3134
3135
3136

```

```

*****
;TEST 24      TEST DATA PARITY BITS FOR HIGH CACHE
;
; THE TEST OF THE DATA PARITY BITS ARE NOT COMPLETE
;UNTIL THE DATA P BIT TEST FOR THE SECOND HALF OF CACHE
;AND THE MSB ADDRESS (A10) TO CACHE DATA FIELD ARE RUN.
;A WRITE/READ PROCEDURE IS DONE WHICH SIMULTANEOUSLY
;CHECKS THE DATA P BIT FOR BOTH BYTES AND DUAL ADDRESSING
;IN HALF OF CACHE FOR IT.  INITIALLY THE P BIT IS WRITTEN
;WITH ONE PARITY PATTERN IN HALF OF CACHE.  THEN STARTING
;AT THE LOW HALF CACHE ADDRESS, THE LOC IS READ AND THEN
;WRITTEN WITH THE OPPOSITE PARITY.  THIS IS SEQUEN-
;TIALY REPEATED WITH INCREASING ADDRESSES UNTIL THE HIGH

```


H06

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 57
T24 TEST DATA PARITY BITS FOR HIGH CACHE

3137
3138
3139
3140
3141
3142
3143
3144
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164
3165
3166
3167
3168
3169
3170
3171
3172
3173
3174
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
3192

;*HALF CACHE ADDRESS IS REACHED. THEN STARTING AT THE
;*HIGH ADDR. THE SECOND PARITY PATTERN IS READ AND THE LOC
;*IS REWRITTEN WITH THE FIRST. THIS IS SEQUENTIALLY RE-
;*PEATED DECREASING THE ADDRESS UNTIL THE LOW HALF CACHE
;*ADDRESS IS REACHED. A SECOND PASS IS THEN MADE WITH
;*THE PARITY PATTERN REVERSED. A PARITY ERROR HANDLER IS
;*SETUP TO DETECT PARITY ERRORS. ALSO, LOCS WHICH SHOULD
;*BE HITS ARE CHECKED FOR AND REPORTED IF NO HIT OCCURRED.
;*:
;*R0, R1 CONTAIN DATA WHICH GENERATE OPPOSITE PARITY. R3
;*INDICATES WHICH PASS IS BEING DONE.

::*****

```
TST24: MOV #214,2#CCR ;CACHE OFF FOR SCOPE
        CLR SCOPE
        MOV #TST25 SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        MOV #T12H01,2#PVEC ;SET UP PARITY ERROR HANDLER
        CLR R3 ;INIT FLAG FOR FIRST PASS
        CLR R0 ;SET UP PARITY PATTERN A FOR FIRST PASS
T12H02: MOV #204,2#CCR ;HALF CACHE ON
        MOV #1000,R1 ;INIT ADDR. COUNTER
        MOV #BUFH,R5 ;INIT. TEST ADDRESS
1S: MOV R0,(R5)+ ;WRITE DATA PARITY PATTERN
     SOB R1,1S ;HALF ADDR. WRITTEN? BRANCH IF NO

        MOV #1000,R1 ;INIT ADDR. COUNTER
        MOV #BUFH,R5 ;INIT. TEST ADDR
        MOV #401,R0 ;SET UP PATTERN B FOR FIRST PASS
        TST R3 ;FIRST PASS?
        BEQ 2S ;BRANCH IF YES
        CLR R0 ;SET UP PARITY PATTERN A FOR SECOND PASS
2S: TST (R5) ;SEE IF PARITY UNCHANGED
     BIT 2#HMR,#HMR2 ;DATA FROM CACHE?
     BEQ T12H07 ;BRANCH TO ERROR IF NO
     MOV R0,(R5)+ ;WRITE NEW DATA PARITY PATTERN
     SOB R1,2S ;HALF ADDR. SPACE EXAMINED & WRITTEN?

        MOV #1000,R1 ;INIT ADDR. COUNTER
        CLR R0 ;SET UP PARITY PATTERN A FOR FIRST PASS
        TST R3 ;FIRST PASS?
        BEQ T12H06 ;BRANCH IF YES
        MOV #401,R0 ;SET UP PARITY PATTERN B FOR SECOND PASS
T12H06: MOV #204,2#CCR ;HALF CACHE ON IF OFF FROM ERROR
1S: TST -(R5) ;SEE IF PARITY UNCHANGED
     BIT 2#HMR,#HMR2 ;DATA FROM CACHE
     BEQ T12H07 ;BRANCH IF NO TO ERROR
     MOV R0,(R5) ;WRITE NEW PARITY PATTERN IN CACHE
     SOB R1,1S ;HALF OF ADDRESS SPACE READ & WRITTEN? BRANCH IF NO

        TST R3 ;SECOND PASS?
        BNE T12H08 ;GO TO END OF TEST IF YES
T12H13: MOV #401,R0 ;SET UP PARITY PATTERN B FOR SECOND PASS
        BIS #1,R3 ;SET FLAG FOR PASS 2
        MOV #T12H13,2#SLPERR ;INIT RETURN FOR ERROR LOOP IF ERROR OCCURS
        BR T12H02 ;TEST DATA
```

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 58
T24. TEST DATA PARITY BITS FOR HIGH CACHE

```

3193 020200 052737 000014 177746 T12H01: BIS #14,2#CCR ;CACHE OFF
3194
3195 020206 010046 MOV RO,-(SP) ;SAVE RO FOR MED INST
3196 020210 076600 MED ;GET CONTENTS OF LOG REG
3197 020212 000022 .WORD RLOG
3198 020214 052700 100001 BIS #100001,RO ;ENABLE ERROR LOG & LOG FIRST MODE
3199 020220 076600 MED ;UNLOCK ERROR LOG
3200 020222 000222 .WORD WLOG
3201 020224 012600 MOV (SP)+,RO ;RESTORE RO
3202
3203 020226 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3204 020230 000101 .WORD RSER
3205 020232 000300 SWAB RO ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3206 020234 042700 177776 BIC #177776,RO ;ONLY LOOK AT A17, A16
3207 020240 010037 001160 MOV RO,$REG1 ;SAVE ADDRESS
3208 020244 076600 MED ;GET LOG INFORMATION
3209 020246 000102 .WORD LOADD
3210 020250 010037 001162 MOV RO,$REG2 ;SAVE INFORMATION
3211 020254 032737 000040 177744 BIT #40,2#EREG ;ERROR IN TAG?
3212 020262 001417 BEQ T12H09 ;BRANCH IF NO
3213 020264 011637 001166 MOV (SP),$REG4 ;GET PC+2 OF ERROR
3214 020270 162737 000002 001166 SUB #2,$REG4 ;GET PC OF ERROR
3215 020276 076600 MED ;GET TAG LOG INFO.
3216 020300 000107 .WORD RTAG
3217 020302 000300 SWAB RO ;PUT TAG IN LOW BYTE
3218 020304 042700 177400 BIC #177400,RO ;LOOK AT TAG ONLY
3219 020310 010037 001164 MOV RO,$REG3 ;SAVE BAD DATA
3220 020314 022626 CMP (SP)+,(SP)+ ;RESTORE THE STACK
3221 020316 104002 ERROR ;ERROR: UNEXPECTED PARITY ERROR IN TAG FIELD
3222 020320 000453 BR T12H08 ;GO TO END OF TEST
3223
3224 020322 022626 T12H09: CMP (SP)+,(SP)+ ;RESTORE STACK
3225 020324 005037 001166 CLR $REG4 ;SAVE GOOD DATA
3226 020330 005700 TST RO ;WAS TEST DATA =0?
3227 020332 001003 BNE T12H11 ;BRANCH IF NO
3228 020334 012737 000401 001166 MOV #401,$REG4 ;SAVE GOOD DATA
3229 020342 032737 000200 177744 T12H11: BIT #200,2#EREG ;ERROR IN HIGH BYTE?
3230 020350 001406 BEQ T12H12 ;BRANCH IF NO
3231 020352 076600 MED ;GET LOG INFORMATION
3232 020354 000106 .WORD CDH
3233 020356 010037 001164 MOV RO,$REG3 ;SAVE INFORMATION
3234 020362 104050 ERROR 50 ;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA P BITS
3235 020364 000431 BR T12H08 ;GO TO END OF TEST
3236
3237 020366 032777 000100 157350 T12H12: BIT #100,2#EREG ;ERROR IN LOW BYTE?
3238 020374 001406 BEQ T12H14 ;BRANCH IF NO
3239 020376 076600 MED ;GET LOG INFORMATION
3240 020400 000106 .WORD CDL
3241 020402 010037 001164 MOV RO,$REG3 ;SAVE INFORMATION
3242 020406 104051 ERROR 51 ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BITS
3243 020410 000417 BR T12H08 ;GO TO END OF TEST
3244
3245 020412 016637 177774 001164 T12H14: MOV -4(SP),$REG3 ;GET PC+2 OF TRAP
3246 020420 162737 000002 001164 SUB #2,$REG3 ;SAVE PC OF TRAP
3247 020426 104001 ERROR 1 ;ERROR: UNEXP. PARITY ERROR IN BACKING STORE
3248

```

```

3249 020430 052737 000014 177746 T12H07: BIS #14,0#CCR ;CACHE OFF
3250 020436 010537 001162 MOV R5,$REG2 ;SAVE BAD ADDRESS
3251 020442 005037 001160 CLR $REG1 ;SAVE BAD ADDRESS
3252 020446 104043 ERROR 43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT
3253
3254 020450 012737 033142 000114 T12H08: MOV #UPERR,0#PVEC ;RESTORE PARITY ERROR HANDLER
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271
3272
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282
3283
3284
3285
3286
3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304

```

```

*****
*TEST 25 TEST TAG ADDRESS BITS FOR HIGH HALF OF CACHE
*
* THE TEST OF THE TAG BITS IS NOT COMPLETE UNTIL THE
*TAG ADDRESS TEST FOR THE OTHER HALF OF CACHE AND THE
*TEST OF THE MSB ADDRESS (A10) TO THE CACHE TAG FIELD
*ARE RUN. A WRITE/READ PROCEDURE IS DONE WHICH CHECKS
*THE TAG FIELD BITS AND DUAL ADDRESSING ON THEM FOR HALF
*OF CACHE. MEMORY IS FIRST SIZED TO DETERMINE THE MAX-
*IMUM TESTABLE ADDRESS. THE TAG ADDRESS BITS OF THIS
*ADDRESS ARE USED AS PATTERN A AND STORED IN KIPAR4. A
*PATTERN B IS NOW GENERATED WHICH HAS 'COMPLEMENT' TAG
*BITS AND STORED IN KIPAR5. ON THE FIRST PASS, PATTERN
*A IS WRITTEN THROUGH HALF OF CACHE. NEXT, STARTING AT
*THE HIGH HALF CACHE ADDRESS, THE LOCATION IS READ,
*CHECKED TO BE A HIT AND THEN WRITTEN WITH PATTERN B.
*THIS IS SEQUENTIALLY REPEATED WITH DECREASING ADDRESSES
*UNTIL THE LOW HALF CACHE ADDRESS IS REACHED. AT THE
*LOW ADDRESS, THE SECOND PATTERN IS READ, CHECKED TO BE A
*HIT AND REWRITTEN WITH THE FIRST PATTERN. THIS IS SE-
*QUENTIALLY REPEATED WITH INCREASING ADDRESSES UNTIL THE
*HIGH HALF CACHE ADDRESS IS REACHED. A SECOND PASS IS
*THEN MADE WITH THE PATTERNS REVERSED.
* ANY PARITY ERROR OR HIT ERROR IS REPORTED.
* DURING THE PASSES, R0, R1 CONTAIN ADDRESSES WHICH
*REFERENCE KIPAR5.
* R3 INDICATES THE PASS NUMBER.
*IF THE INHIBIT TESTS USING KT SWITCH (SW12) IS SET, THIS
*TEST IS SKIPPED.

```

```

3289 020456 012737 000214 177746 TST25: MOV #214,0#CCR ;CACHE OFF FOR SCOPE
3290 020464 000004 SCOPE
3291 020466 012737 022000 001234 MOV #TST26,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3292 020474 032777 010000 160432 BIT #SW12,$SWR ;INHIBIT TESTS USING KT?
3293 020502 001402 BEQ 3$ ;CONTINUE TEST IF NO
3294 020504 000137 022000 JMP #TST26 ;GO TO NEXT TEST
3295 020510 052737 000200 036034 3$: BIS #200,0#SKT11 ;KT ON FOR $SIZE
3296 020516 004737 035750 JSR PC,$SIZE ;SIZE MEMORY
3297 020522 012737 020744 000114 MOV #T13H01,0#PVEC ;SET UP PARITY ERROR HANDLER
3298 020530 013737 036322 172350 MOV 0#SLSTBK,0#KIPAR4 ;SET UP PAR4 FOR ADDRESS PATTERN A
3299
3300 ;CALC COMPLEMENT TAG PATTERN B
3301
3302 020536 013700 036322 MOV 0#SLSTBK,R0 ;GET TEST PATTERN A AND
3303 020542 005100 COM R0 ;CALC PATTERN B
3304 020544 005001 CLR R1

```

K06

MD-11-DOKKA-A 11/6X CACHE DIAGNOSTIC
DOKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 60
T25 TEST TAG ADDRESS BITS FOR HIGH HALF OF CACHE

```

3305 020546 005201      1S:  INC      R1
3306 020550 006300      ASL      R0
3307 020552 100775      BMI      1S
3308 020554 006200      2S:  ASR      R0
3309 020556 077102      SOB      R1,2S
3310 020560 042700 000037      BIC      #37,R0      ;ONLY COMPLEMENT TAG ADDR. BITS
3311
3312 020564 010037 172352      MOV      R0,#KIPARS ;SET UP PARS FOR ADDRESS PATTERN B
3313
3314 020570 012700 102000      MOV      #102000,R0 ;INIT R0 TO ADDR PATTERN A
3315 020574 012701 124000      MOV      #124000,R1 ;INIT R1 TO ADDR PATTERN B
3316 020600 005003      CLR      R3          ;INIT FLAG FOR PASS 1
3317 020602 005004      T13H02: CLR      R4          ;INIT INDICATOR FOR ERROR LOOP 1
3318 020604 012702 001000      MOV      #1000,R2   ;INIT ADDR. COUNTER
3319 020610 052737 000001 177572      BIS      #1,#MMRO   ;TURN KT ON
3320 020616 012737 000204 177746      MOV      #204,#CCR  ;TURN HALF OF CACHE ON
3321
3322 020624 005720      1S:  TST      (R0)+    ;WRITE PATTERN IN CACHE
3323 020626 077202      SOB      R2,1S      ;ALL DONE? BRANCH IF NO
3324
3325 020630 012702 001000      T13H03: MOV      #1000,R2   ;INIT. ADDR. COUNTER
3326 020634 005740      TST      -(R0)      ;READ CACHE TAG BITS
3327 020636 033727 177752 000004      BIT      @#HMR,#HMR2 ;HIT?
3328 020644 001002      BNE      2S        ;BRANCH IF YES
3329 020646 000137 021216      JMP      T13H04    ;REPORT ERROR
3330 020652 005741      2S:  TST      -(R1)    ;WRITE NEW PATTERN IN TAG
3331 020654 077211      SOB      R2,T13H03 ;HALF ADDR. TESTED? BRANCH IF NO
3332
3333 020656 005204      INC      R4          ;SET INDICATOR FOR ERROR LOOP 2
3334 020660 012702 001000      T13H05: MOV      #1000,R2   ;INIT. ADDR. COUNTER
3335 020664 005711      TST      (R1)      ;READ CACHE TAG BITS
3336 020666 033727 177752 000004      BIT      @#HMR,#HMR2 ;HIT?
3337 020674 001002      BNE      3S        ;BRANCH IF YES
3338 020676 000137 021264      JMP      T13H06    ;REPORT ERROR
3339 020702 005721      3S:  TST      (R1)+    ;UPDATE FOR NEXT ADDRESS
3340 020704 005720      TST      (R0)+    ;WRITE NEW PATTERN IN TAG
3341 020706 077212      SOB      R2,T13H05
3342
3343 020710 005703      TST      R3          ;SECOND PASS?
3344 020712 001402      BEQ      2S        ;CONTINUE TEST IF NO
3345 020714 000137 021312      JMP      T13H07    ;GO TO END OF TEST
3346 020720 052703 000001 001110      2S:  BIS      #1,R3    ;SET FLAG FOR SECOND PASS
3347 020724 012737 020732      MOV      #T13H15,#SLPERR ;INIT RETURN FOR ERROR LOOP IF ERROR OCCURS
3348 020732 012700 122000      T13H15: MOV      #122000,R0 ;INIT. R0 TO ADDR. PATTERN B
3349 020736 012701 104000      MOV      #104000,R1 ;INIT. R1 TO ADDR. PATTERN A
3350 020742 000717      BR      T13H02     ;GO TEST SECOND PASS
3351
3352 020744 052737 000014 177746      T13H01: BIS      #14,#CCR ;CACHE OFF
3353
3354 020752 010046      MOV      R0,-(SP)   ;SAVE R0 FOR MED INST
3355 020754 076600      MED      ;GET CONTENTS OF LOG REG
3356 020756 000022      .WORD   RLOG
3357 020760 052700 100001      BIS      #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
3358 020764 076600      MED      ;UNLOCK ERROR LOG
3359 020766 000222      .WORD   WLOG
3360 020770 012600      MOV      (SP)+,R0  ;RESTORE R0

```


M06

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 62
T25 TEST TAG ADDRESS BITS FOR HIGH HALF OF CACHE

3417	021212	104054				ERROR	54		;ERROR: HIGH BYTE PARITY ERROR ON TEST OF TAG ADDR. BITS
3418	021214	000436				BR	T13H07		;GO TO END OF TEST
3419									
3420	021216	052737	000014	177746	T13H04:	BIS	#14,2#CCR		;CACHE OFF
3421	021224	010037	001172			MOV	RO,\$TMPD		;GET VIRTUAL ADDRESS TESTED
3422	021230	004737	033434			JSR	PC,VIP		;SAVE ADDRESS TESTED
3423	021234	062700	000002			ADD	#2,RO		;ADJUST ADDRESS WHEN LOOP
3424	021240	004737	033634			JSR	PC,PAR		;GET PAR TESTED
3425	021244	000000				.WORD	0		;INDICATOR FOR RO
3426	021246	004737	033606		T13H13:	JSR	PC,TAG		;CALC TAG FROM PAR
3427	021252	013737	001172	001164		MOV	\$TMPD,\$REG3		;SAVE TAG
3428	021260	104055				ERROR	55		;ERROR: TEST OF TAG ADDRESS BITS FAILED
3429									;ADDR. COULD NOT BE MADE A HIT
3430	021262	000413				BR	T13H07		;GO TO NEXT TEST
3431									
3432	021264	052737	000014	177746	T13H06:	BIS	#14,2#CCR		;CACHE OFF
3433	021272	010137	001172			MOV	R1,\$TMPD		;GET VIRTUAL ADDRESS TESTED
3434	021276	004737	033434			JSR	PC,VIP		;SAVE PHYSICAL ADDRESS TESTED
3435	021302	004737	033634			JSR	PC,PAR		;GET PAR TESTED
3436	021306	000001				.WORD	1		;INDICATOR FOR R1
3437	021310	000756				BR	T13H13		;REPORT ERROR
3438									
3439	021312	005037	177572		T13H07:	CLR	2#MMRO		;KT OFF
3440	021316	012737	033142	000114		MOV	#UPERR,2#PVEC		;RESTORE UNEXPECTED PARITY ERROR HANDLER
3441	021324	052737	000014	177746		BIS	#14,2#CCR		;CACHE OFF WHEN CROSS CACHE ADDR. BOUNDARY
3442	021332	000137	022000			JMP	2#TST26		;GO TO NEXT TEST
3443									
3444									
3445		022000							;ADJUST ADDRESS SPACE FOR NEXT TEST
3446									
3447									
3448									
3449									
3450									

```

*****
*TEST 26      TEST DATA FIELD FOR LOW HALF OF CACHE
*
*   THE TEST OF THE DATA FIELD IS NOT COMPLETE UNTIL THE
*TEST OF THE DATA FIELD FOR THE OTHER HALF OF CACHE AND
*THE TEST OF THE MSB ADDRESS (A10) TO THE CACHE DATA
*FIELD ARE RUN.  A WRITE/READ PROCEDURE IS DONE WHICH
*CHECKS ALL THE DATA FIELD BITS AND DUAL ADDRESSING ON
*THEM FOR HALF OF CACHE.  ON THE FIRST PASS ONE PATTERN
*(CONTAINED IN RO) IS WRITTEN IN ALL THE DATA FIELDS.
*FOR HALF OF CACHE.  NEXT, STARTING AT THE HIGH HALF
*CACHE ADDRESS, THE LOCATION IS TESTED TO BE A HIT.  ITS
*DATA IS CHECKED AND THEN WRITTEN WITH A SECOND PATTERN
*CONTAINED IN R1.  THIS IS SEQUENTIALLY REPEATED WITH
*DECREASING ADDRESSES UNTIL THE LOW HALF CACHE ADDRESS IS
*REACHED.  AT THE LOW ADDRESS, THE SECOND PATTERN IS READ,
*TESTED TO BE A HIT AND REWRITTEN WITH THE FIRST PATTERN.
*THIS IS SEQUENTIALLY REPEATED WITH INCREASING ADDRESSES
*UNTIL THE HIGH HALF CACHE ADDRESS IS REACHED.  A SECOND
*PASS IS THEN MADE WITH THE PATTERNS REVERSED.
*
* ANY PARITY REEOR OR HIT ERROR IS REPORTED.
* RO, R1  CONTAIN THE TEST PATTERN
* R2     CONTAINS THE TEST ADDRESS
* R4     CONTAINS THE PASS NUMBER

```

3472

```

3473
3474
3475 022000 012737 000214 177746 TST26: MOV #214, @CCR ;CACHE OFF FOR SCOPE
3476 022006 000004 SCOPE
3477 022010 012737 024000 001234 MOV #TST27, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3478 022016 012737 022210 000114 MOV #T15L01, @PVEC ;SET UP PARITY ERROR HANDLER
3479 022024 012700 125252 MOV #125252, R0 ;SET UP DATA PATTERN A FOR PASS 1
3480 022030 012701 052525 MOV #52525, R1 ;SET UP DATA PATTERN B FOR PASS 1
3481 022034 012737 000210 177746 T15L05: MOV #210, @CCR ;HALF CACHE ON
3482 022042 005004 CLR R4 ;SET UP LOOP INDIC FOR ERROR LOOP 1
3483 022044 012702 060000 MOV #BUFL, R2 ;INIT STARTING TEST ADDRESS
3484 022050 012703 001000 MOV #1000, R3 ;INIT ADDRESS COUNTER
3485 022054 010022 1S: MOV R0, (R2)+ ;WRITE CACHE WITH PATTERN
3486 022056 077302 SOB R3, 1S ;LOOP TILL HALF CACHE WRITTEN
3487
3488 ;NOW READ AND WRITE PATTERN, DECREASING ADDRESS
3489
3490 022060 012703 001000 T15L21: MOV #1000, R3 ;INIT ADDRESS COUNTER
3491 022064 005742 TST -(R2) ;READ CACHE
3492 022066 033727 177752 000004 BIT @HMR, #HMR2 ;HIT?
3493 022074 001002 BNE 1S ;BRANCH IF YES
3494 022076 000137 022466 JMP T15L02 ;REPORT ERROR
3495 022102 021200 1S: CMP (R2), R0 ;IS DATA CORRECT?
3496 022104 001402 BEQ T15L17 ;BRANCH IF YES
3497 022106 000137 022510 JMP T15L03 ;REPORT ERROR
3498 022112 010112 T15L17: MOV R1, (R2) ;WRITE NEW PATTERN IN CACHE
3499 022114 077315 SOB R3, T15L21 ;LOOP TILL HALF CACHE READ & WRITTEN
3500
3501 ;NOW READ AND WRITE PATTERN, INCREASING ADDRESS
3502
3503 022116 052704 000001 BIS #1, R4 ;SET FLAG FOR ERROR LOOP 2
3504 022122 012703 001000 MOV #1000, R3 ;INIT. ADDRESS COUNTER
3505 022126 005712 T15L22: TST (R2) ;READ CACHE
3506 022130 033727 177752 000004 BIT @HMR, #HMR2 ;HIT?
3507 022136 001002 BNE 1S ;BRANCH IF YES
3508 022140 000137 022466 JMP T15L02 ;REPORT ERROR
3509 022144 021201 1S: CMP (R2), R1 ;DATA OK?
3510 022146 001402 BEQ T15L18 ;BRANCH IF YES
3511 022150 000137 022526 JMP T15L15 ;REPORT ERROR
3512 022154 010022 T15L18: MOV R0, (R2)+ ;WRITE NEW TEST PATTERN
3513 022156 077315 SOB R3, T15L22 ;LOOP TILL HALF OF CACHE READ & WRITTEN
3514
3515 022160 005700 TST R0 ;DOES R0 HAVE DATA FOR FIRST PASS?
3516 022162 100402 BMI T15L12 ;BRANCH IF YES
3517 022164 000137 022560 JMP T15L04 ;GO TO END OF TEST
3518 022170 012700 052525 T15L12: MOV #52525, R0 ;SET UP DATA PATTERN B FOR PASS 2.
3519 022174 012701 125252 MOV #125252, R1 ;SET UP DATA PATTERN A FOR PASS 2
3520 022200 012737 022170 001110 MOV #T15L12, @SLPERR ;INIT RETURN FOR ERROR LOOP IF ERROR
3521 022206 000712 BR T15L05 ;GO TEST IT
3522
3523 022210 052737 000014 177746 T15L01: BIS #14, @CCR ;CACHE OFF
3524
3525 022216 010046 MOV R0, -(SP) ;SAVE R0 FOR MED INST
3526 022220 076600 MED ;GET CONTENTS OF LOG REG
3527 022222 000022 .WORD RLOG
3528 022224 052700 100001 BIS #100001, R0 ;ENABLE ERROR LOG & LOG FIRST MODE

```

3529	022230	076600			MED			;UNLOCK ERROR LOG
3530	022232	000222			.WORD	WLOG		
3531	022234	012600			MOV	(SP)+,RO		;RESTORE RO
3532								
3533	022236	011637	001164		MOV	(SP), \$REG3		;GET PC+2 OF PARITY ERROR
3534	022242	162737	000002	001164	SUB	#2, \$REG3		;SAVE PC OF PARITY ERROR
3535	022250	022626			CMP	(SP)+, (SP)+		;RESTORE STACK
3536	022252	010046			MOV	RO, -(SP)		;SAVE RO FOR MED INST
3537	022254	076600			MED			;GET LOG INFOR FOR PHY. ADDR. A17,A16
3538	022256	000101			.WORD	RSER		
3539	022260	000300			SWAB	RO		;PUT PHY. ADDR A17, A16 IN LOW BYTE
3540	022262	042700	177776		BIC	#177776, RO		;ONLY LOOK AT A17, A16
3541	022266	010037	001160		MOV	RO, \$REG1		;SAVE ADDRESS
3542	022272	076600			MED			;GET LOG INFORMATION
3543	022274	000102			.WORD	LOADD		
3544	022276	010037	001162		MOV	RO, \$REG2		;SAVE INFORMATION
3545	022302	076600			MED			;GET LOG INFORMATION
3546	022304	000100			.WORD	RJAM		
3547	022306	010005			MOV	RO, R5	;SAVE INFORMATION	
3548	022310	012600			MOV	(SP)+, RO		;RESTORE RO
3549	022312	032705	000400		BIT	#400, R5		;ERROR IN BACKING STORE?
3550	022316	001406			BEQ	T15L06		;BRANCH IF NO
3551	022320	076600			MED			;GET LOG INFORMATION
3552	022322	055016			.WORD	BSD		
3553	022324	010037	001164		MOV	RO, \$REG3		;SAVE INFORMATION
3554	022330	104001			ERROR	1		;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
3555	022332	000512			BR	T15L04		;GO TO END OF TEST
3556								
3557	022334	010137	001166		T15L06: MOV	R1, \$REG4		;SAVE GOOD DATA
3558	022340	005704			TST	R4		;ERROR LOOP 1?
3559	022342	001002			BNE	T15L08		;BRANCH IF NO
3560	022344	010037	001166		MOV	RO, \$REG4		;SAVE GOOD DATA
3561								
3562	022350	032737	000100	177744	T15L08: BIT	#100, \$REG4		;LOW BYTE PARITY ERROR?
3563	022356	001406			BEQ	T15L13		;BRANCH IF NO
3564	022360	076600			MED			;GET LOG INFORMATION
3565	022362	000106			.WORD	CDL		
3566	022364	010037	001164		MOV	RO, \$REG3		;SAVE INFORMATION
3567	022370	104056			ERROR	56		;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD
3568	022372	000472			BR	T15L04		;GO TO END OF TEST
3569								
3570	022374	032737	000200	177744	T15L13: BIT	#200, \$REG4		;PARITY ERROR IN HIGH BYTE?
3571	022402	001406			BEQ	T15L14		;BRANCH IF NO
3572	022404	076600			MED			;GET LOG INFORMATION
3573	022406	000106			.WORD	CDH		
3574	022410	010037	001164		MOV	RO, \$REG3		;SAVE INFORMATION
3575	022414	104057			ERROR	57		;ERROR: HIGH BYTE PARITY ERROR WHEN TEST DATA FIELD
3576	022416	000460			BR	T15L04		;GO TO END OF TEST
3577								
3578	022420	010237	001166		T15L14: MOV	R2, \$REG4		;GET FAILING ADDRESS
3579	022424	012705	000013		MOV	#13, R5		;SET UP COUNTER
3580	022430	006237	001166		25: ASR	\$REG4		;PUT TAG ADDRESS BITS IN LSB 6-0
3581	022434	077503			SOB	R5, 25		;LOOP TILL DONE
3582	022436	052737	000200	001166	BIS	#200, \$REG4		;SET VALID BIT
3583	022444	076600			MED			;GET TAG LOG INFO.
3584	022446	000107			.WORD	RTAG		

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 65
T26 TEST DATA FIELD FOR LOW HALF OF CACHE

```

3585 022450 000300          SWAB      R0          ;PUT TAG IN LOW BYTE
3586 022452 042700 177400  BIC      #177400,R0    ;LOOK AT TAG ONLY
3587 022456 010037 001164  MOV     R0,$REG3      ;SAVE BAD DATA
3588 022462 104060          ERROR    60          ;ERROR: TAG PARITY ERROR WHEN TESTING DATA FIELD
3589 022464 000435          BR      T15L04       ;GO TO END OF TEST
3590
3591 022466 052737 000014 177746  T15L02: BIS     #14,#CCR    ;CACHE OFF
3592 022474 005037 001160          CLR     $REG1        ;SAVE ADDRESS
3593 022500 010237 001162          MOV     R2,$REG2     ;SAVE ADDRESS
3594 022504 104043          ERROR    43          ;ERROR: ADDRESS COULD NOT BE MADE A HIT
3595 022506 000424          BR      T15L04       ;GO TO END OF TEST
3596
3597 022510 011205          T15L03: MOV    (R2),R5   ;GET BAD DATA
3598 022512 052737 000014 177746  BIS     #14,#CCR    ;CACHE OFF
3599 022520 010037 001166          MOV     R0,$REG4    ;SAVE GOOD DATA
3600 022524 000406          BR      T15L16       ;REPORT ERROR
3601
3602 022526 011205          T15L15: MOV    (R2),R5   ;GET BAD DATA
3603 022530 052737 000014 177746  BIS     #14,#CCR    ;CACHE OFF
3604 022536 010137 001166          MOV     R1,$REG4    ;SAVE GOOD DATA
3605 022542 005037 001160          T15L16: CLR     $REG1    ;SAVE ADDRESS
3606 022546 010237 001162          MOV     R2,$REG2    ;SAVE ADDRESS
3607 022552 010537 001164          MOV     R5,$REG3    ;SAVE BAD DATA
3608 022556 104061          ERROR    61          ;ERROR: CACHE DATA LOC HELD WRONG DATA
3609
3610 022560 012737 033142 000114  T15L04: MOV    #UPERR,#PVEC ;RESTORE UNEXPECT. P.E. HANDLER
3611 022566 052737 000014 177746  BIS     #14,#CCR    ;CACHE OFF WHEN CROSS CACHE ADDR. BOUNDARY
3612 022574 000137 024000          JMP     #TST27      ;GO TO NEXT TEST
3613
3614
3615          024000          . =24000          ;ADJUST ADDRESS SPACE FOR NEXT TEST
3616
3617
3618
3619
3620
3621
3622
3623
3624
3625
3626
3627
3628
3629
3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640

```

```

*****
;TEST 27      TEST DATA FIELD FOR HIGH HALF OF CACHE
;
; THE TEST OF THE DATA FIELD IS NOT COMPLETE UNTIL THE
;TEST OF THE DATA FIELD FOR THE OTHER HALF OF CACHE AND
;THE TEST OF THE MSB ADDRESS (A10) TO THE CACHE DATA
;FIELD ARE RUN. A WRITE/READ PROCEDURE IS DONE WHICH
;CHECKS ALL THE DATA FIELD BITS AND DUAL ADDRESSING ON
;THEM FOR HALF OF CACHE. ON THE FIRST PASS ONE PATTERN
; (CONTAINED IN R0) IS WRITTEN IN ALL THE DATA FIELDS.
;FOR HALF OF CACHE. NEXT, STARTING AT THE HIGH HALF
;CACHE ADDRESS, THE LOCATION IS TESTED TO BE A HIT, ITS
;DATA IS CHECKED AND THEN WRITTEN WITH A SECOND PATTERN
;CONTAINED IN R1. THIS IS SEQUENTIALLY REPEATED WITH
;DECREASING ADDRESSES UNTIL THE LOW HALF CACHE ADDRESS IS
;REACHED. AT THE LOW ADDRESS, THE SECOND PATTERN IS READ,
;TESTED TO BE A HIT AND REWRITTEN WITH THE FIRST PATTERN.
;THIS IS SEQUENTIALLY REPEATED WITH INCREASING ADDRESSES
;UNTIL THE HIGH HALF CACHE ADDRESS IS REACHED. A SECOND
;PASS IS THEN MADE WITH THE PATTERNS REVERSED.
; ANY PARITY REEOR OR HIT ERROR IS REPORTED.
; RO, R1      CONTAIN THE TEST PATTERN

```

```

3641 ;* R2 CONTAINS THE TEST ADDRESS
3642 ;* R4 CONTAINS THE PASS NUMBER
3643
3644 ..*****
3645 024000 012737 000214 177746 TST27: MOV #214, @CCR ;CACHE OFF FOR SCOPE
3646 024006 000004 SCOPE
3647 024010 012737 024566 001234 MOV #TST30 SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3648 024016 012737 024210 000114 MOV #T15H01, @PVEC ;SET UP PARITY ERROR HANDLER
3649 024024 012700 125252 MOV #125252, R0 ;SET UP DATA PATTERN A FOR PASS 1
3650 024030 012701 052525 MOV #52525, R1 ;SET UP DATA PATTERN B FOR PASS 1
3651 024034 012737 000204 177746 T15H05: MOV #204, @CCR ;HALF CACHE ON
3652 024042 005004 CLR R4 ;SET UP LOOP INDIC FOR ERROR LOOP 1
3653 024044 012702 062000 MOV #BUFH, R2 ;INIT STARTING TEST ADDRESS
3654 024050 012703 001000 MOV #1000, R3 ;INIT ADDRESS COUNTER
3655 024054 010022 1S: MOV R0, (R2)+ ;WRITE CACHE WITH PATTERN
3656 024056 077302 SOB R3, 1S ;LOOP TILL HALF CACHE WRITTEN
3657
3658 ;NOW READ AND WRITE PATTERN, DECREASING ADDRESS
3659
3660 024060 012703 001000 T15H21: MOV #1000, R3 ;INIT ADDRESS COUNTER
3661 024064 005742 TST -(R2) ;READ CACHE
3662 024066 033727 177752 000004 BIT @HMR, #HMR2 ;HIT?
3663 024074 001002 BNE 1S ;BRANCH IF YES
3664 024076 000137 024466 JMP T15H02 ;REPORT ERROR
3665 024102 021200 1S: CMP (R2), R0 ;IS DATA CORRECT?
3666 024104 001402 BEQ T15H17 ;BRANCH IF YES
3667 024106 000137 024510 JMP T15H03 ;REPORT ERROR
3668 024112 010112 T15H17: MOV R1, (R2) ;WRITE NEW PATTERN IN CACHE
3669 024114 077315 SOB R3, T15H21 ;LOOP TILL HALF CACHE READ & WRITTEN
3670
3671 ;NOW READ AND WRITE PATTERN, INCREASING ADDRESS
3672
3673 024116 052704 000001 BIS #1, R4 ;SET FLAG FOR ERROR LOOP 2
3674 024122 012703 001000 MOV #1000, R3 ;INIT. ADDRESS COUNTER
3675 024126 005712 T15H22: TST (R2) ;READ CACHE
3676 024130 033727 177752 000004 BIT @HMR, #HMR2 ;HIT?
3677 024136 001002 BNE 1S ;BRANCH IF YES
3678 024140 000137 024466 JMP T15H02 ;REPORT ERROR
3679 024144 021201 1S: CMP (R2), R1 ;DATA OK?
3680 024146 001402 BEQ T15H18 ;BRANCH IF YES
3681 024150 000137 024526 JMP T15H15 ;REPORT ERROR
3682 024154 010022 T15H18: MOV R0, (R2)+ ;WRITE NEW TEST PATTERN
3683 024156 077315 SOB R3, T15H22 ;LOOP TILL HALF OF CACHE READ & WRITTEN
3684
3685 024160 005700 TST R0 ;DOES R0 HAVE DATA FOR FIRST PASS?
3686 024162 100402 BMI T15H12 ;BRANCH IF YES
3687 024164 000137 024560 JMP T15H04 ;GO TO END OF TEST
3688 024170 012700 052525 T15H12: MOV #52525, R0 ;SET UP DATA PATTERN B FOR PASS 2.
3689 024174 012701 125252 MOV #125252, R1 ;SET UP DATA PATTERN A FOR PASS 2.
3690 024200 012737 024170 001110 MOV #T15H12, @SLPERR ;INIT RETURN FOR ERROR LOOP IF ERROR
3691 024206 000712 BR T15H05 ;GO TEST IT
3692
3693 024210 052737 000014 177746 T15H01: BIS #14, @CCR ;CACHE OFF
3694
3695 024216 010046 MOV R0, -(SP) ;SAVE R0 FOR MED INST
3696 024220 076600 MED ;GET CONTENTS OF LOG REG

```

E07

MD-11-DGKKA-A 11/6X CACHE DIAGNOSTIC
 DGKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 67
 T27 TEST DATA FIELD FOR HIGH HALF OF CACHE

3697	024222	000022				.WORD	RLOG		
3698	024224	052700	100001			BIS	#100001,RO		;ENABLE ERROR LOG & LOG FIRST MODE
3699	024230	076600				MED			;UNLOCK ERROR LOG
3700	024232	000222				.WORD	WLOG		
3701	024234	012600				MOV	(SP)+,RO		;RESTORE RO
3702									
3703	024236	011637	001164			MOV	(SP),SREG3		;GET PC+2 OF PARITY ERROR
3704	024242	162737	000002	001164		SUB	#2,SREG3		;SAVE PC OF PARITY ERROR
3705	024250	022626				CMP	(SP)+,(SP)+		;RESTORE STACK
3706	024252	010046				MOV	RO,-(SP)		;SAVE RO FOR MED INST
3707	024254	076600				MED			;GET LOG INFOR FOR PHY. ADDR. A17,A16
3708	024256	000101				.WORD	RSER		
3709	024260	000300				SWAB	RO		;PUT PHY. ADDR A17, A16 IN LOW BYTE
3710	024262	042700	177776			BIC	#177776,RO		;ONLY LOOK AT A17, A16
3711	024266	010037	001160			MOV	RO,SREG1		;SAVE ADDRESS
3712	024272	076600				MED			;GET LOG INFORMATION
3713	024274	000102				.WORD	LOADD		
3714	024276	010037	001162			MOV	RO,SREG2		;SAVE INFORMATION
3715	024302	076600				MED			;GET LOG INFORMATION
3716	024304	000100				.WORD	RJAM		
3717	024306	010005				MOV	RO,R5 ;SAVE INFORMATION		
3718	024310	012600				MOV	(SP)+,RO		;RESTORE RO
3719	024312	032705	000400			BIT	#400,R5		;ERROR IN BACKING STORE?
3720	024316	001406				BEQ	T15H06		;BRANCH IF NO
3721	024320	076600				MED			;GET LOG INFORMATION
3722	024322	055016				.WORD	BSD		
3723	024324	010037	001164			MOV	RO,SREG3		;SAVE INFORMATION
3724	024330	104001				ERROR	1		;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
3725	024332	000512				BR	T15H04		;GO TO END OF TEST
3726									
3727	024334	010137	001166		T15H06:	MOV	R1,SREG4		;SAVE GOOD DATA
3728	024340	005704				TST	R4		;ERROR LOOP 1?
3729	024342	001002				BNE	T15H08		;BRANCH IF NO
3730	024344	010037	001166			MOV	RO,SREG4		;SAVE GOOD DATA
3731									
3732	024350	032737	000100	177744	T15H08:	BIT	#100,2#EREG		;LOW BYTE PARITY ERROR?
3733	024356	001406				BEQ	T15H13		;BRANCH IF NO
3734	024360	076600				MED			;GET LOG INFORMATION
3735	024362	000106				.WORD	CDL		
3736	024364	010037	001164			MOV	RO,SREG3		;SAVE INFORMATION
3737	024370	104056				ERROR	56		;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD
3738	024372	000472				BR	T15H04		;GO TO END OF TEST
3739									
3740	024374	032737	000200	177744	T15H13:	BIT	#200,2#EREG		;PARITY ERROR IN HIGH BYTE?
3741	024402	001406				BEQ	T15H14		;BRANCH IF NO
3742	024404	076600				MED			;GET LOG INFORMATION
3743	024406	000106				.WORD	CDH		
3744	024410	010037	001164			MOV	RO,SREG3		;SAVE INFORMATION
3745	024414	104057				ERROR	57		;ERROR: HIGH BYTE PARITY ERROR WHEN TEST DATA FIELD
3746	024416	000460				BR	T15H04		;GO TO END OF TEST
3747									
3748	024420	010237	001166		T15H14:	MOV	R2,SREG4		;GET FAILING ADDRESS
3749	024424	012705	000013			MOV	#13,R5		;SET UP COUNTER
3750	024430	006237	001166		25:	ASR	SREG4		;PUT TAG ADDRESS BITS IN LSB 6-0
3751	024434	077503				SQB	R5,25		;LOOP TILL DONE
3752	024436	052737	000200	001166		BIS	#200,SREG4		;SET VALID BIT

F07

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 68
T27 TEST DATA FIELD FOR HIGH HALF OF CACHE

```

3753 024444 076600          MED          ;GET TAG LOG INFO.
3754 024446 000107          .WORD      RTAG
3755 024450 000300          SWAB      RO          ;PUT TAG IN LOW BYTE
3756 024452 042700 177400  BIC      #177400,RO  ;LOOK AT TAG ONLY
3757 024456 010037 001164  MOV      RO,$REG3    ;SAVE BAD DATA
3758 024462 104060          ERROR     60         ;ERROR: TAG PARITY ERROR WHEN TESTING DATA FIELD
3759 024464 000435          BR        T15H04    ;GO TO END OF TEST
3760
3761 024466 052737 000014 177746  T15H02: BIS      #14,#CCR  ;CACHE OFF
3762 024474 005037 001160          CLR      $REG1      ;SAVE ADDRESS
3763 024500 010237 001162          MOV      R2,$REG2   ;SAVE ADDRESS
3764 024504 104043          ERROR     43         ;ERROR: ADDRESS COULD NOT BE MADE A HIT
3765 024506 000424          BR        T15H04    ;GO TO END OF TEST
3766
3767 024510 011205          T15H03: MOV      (R2),R5 ;GET BAD DATA
3768 024512 052737 000014 177746  BIS      #14,#CCR  ;CACHE OFF
3769 024520 010037 001166          MOV      RO,$REG4   ;SAVE GOOD DATA
3770 024524 000406          BR        T15H16    ;REPORT ERROR
3771
3772 024526 011205          T15H15: MOV      (R2),R5 ;GET BAD DATA
3773 024530 052737 000014 177746  BIS      #14,#CCR  ;CACHE OFF
3774 024536 010137 001166          MOV      R1,$REG4   ;SAVE GOOD DATA
3775 024542 005037 001160          T15H16: CLR      $REG1 ;SAVE ADDRESS
3776 024546 010237 001162          MOV      R2,$REG2   ;SAVE ADDRESS
3777 024552 010537 001164          MOV      R5,$REG3   ;SAVE BAD DATA
3778 024556 104061          ERROR     61         ;ERROR: CACHE DATA LOC HELD WRONG DATA
3779
3780 024560 012737 033142 000114  T15H04: MOV      #UPERR,#PVEC ;RESTORE UNEXPEC. PARITY ERROR HANDLER
3781
3782  ;*****
3783  ;TEST 30      TEST OF MSB ADDRESS (A10) TO VALID BIT
3784  ;
3785  ;      THIS IS THE FIRST TEST WHERE ALL OF CACHE IS TURNED
3786  ;ON.  THE TEST CHECKS FOR DUAL ADDRESSING ON THE VALID BIT FOR
3787  ;THE MSB PHYSICAL ADDRESS (A10) TO CACHE.  INITIALLY TEST ADDRESSES
3788  ;ARE CHOSEN WHICH HAVE THE CACHE ADDRESS BITS A1-A9 THE SAME
3789  ;AND A10 COMPLEMENTS.  THE ADDRESSES ARE ALSO CHOSEN TO NOT OVERLAP
3790  ;THE TEST INSTRUCTION SPACE.  THE FIRST ADDRESS IS AT THE END OF THIS
3791  ;TEST INSTRUCTION SPACE (TAD2) AND THE SECOND IS CHOSEN BY THE
3792  ;SUBROUTINE HAD TO LIE IN A 1 K BUFFER AT THE END OF THE PROGRAM.
3793  ;      THE FIRST ADDRESS IS INVALIDATED VIA WWP AND FORCING A PARITY
3794  ;TRAP.  THE SECOND IS THEN MADE VALID AND CHECKED TO BE A HIT.  THE FIRST IS
3795  ;THEN EXAMINED TO STILL BE INVALID (NOT A HIT).  ANY PARITY OR HIT
3796  ;ERROR IS REPORTED.
3797  ;*****
3798  ;*****
3799  ;TST30:  MOV  #214,#CCR ;CACHE OFF FOR SCOPE
3800  ;SCOPE
3801  ;MOV      #TST31,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3802  ;MOV      #T30LO1,#PVEC ;SET UP FOR PARITY TRAP
3803  ;JSR      PC,HAD      ;CALC CONGRUENT ADDR. IN TEST BUFFER
3804  ;.WORD   TAD2
3805  ;MOV      $TMP0,RO    ;SAVE ADDR.
3806  ;MOV      #300,#CCR  ;CACHE ON & WWP
3807  ;CLR      (RO)        ;WWP IN TEST ADDR.
3808  ;MOV      #200,#CCR  ;WWP OFF

```

```

3809 024642 005710          TST      (R0)          ;FORCE PARITY TRAP
3810 024644 000465          BR       T30L02       ;REPORT FAILURE TO TRAP
3811
3812 024646
3813
3814 024646 010046          MOV     RO,-(SP)      ;SAVE RO FOR MED INST
3815 024650 076600          MED                    ;GET CONTENTS OF LOG REG
3816 024652 000022          .WORD  RLOG
3817 024654 052700 100001      BIS     #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
3818 024660 076600          MED                    ;UNLOCK ERROR LOG
3819 024662 000222          .WORD  WLOG
3820 024664 012600          MOV     (SP)+,R0     ;RESTORE RO
3821
3822 024666 062706 000004      ADD     #4,SP        ;RESTORE STACK
3823 024672 012737 025042 000114      MOV     #T30L06,#PVEC ;SET UP PARITY ERROR HANDLER
3824 024700 023737 025242 025242      CMP     TAD2,TAD2    ;MAKE TEST ADDR A HIT
3825 024706 033727 177752 000004      BIT     @#HMR,#HMR2  ;HIT?
3826 024714 001427          BEQ     T30L03       ;REPORT ERROR IF NO
3827 024716 005710          TST     (R0)         ;CHECK OTHER LOC. STILL INVALIDATED
3828 024720 033727 177752 000004      BIT     @#HMR,#HMR2  ;MISS?
3829 024726 001011          BNE     T30L04       ;REPORT ERROR IF NO
3830
3831
3832
3833 024730 012737 000214 177746      ;RID CACHE OF BAD PARITY
3834 024736 004737 035134          MOV     #214,@#CCR   ;CACHE OFF IF ON
3835
3836
3837 024742 012737 033142 000114      JSR     PC,SWEEP     ;GO PURGE CACHE
3838
3839
3840 024752 012737 000214 177746      MOV     #UPERR,@#PVEC ;RESTORE UNEXP. PARITY ERROR HANDLER
3841 024760 005037 001160          BR       TST31       ;GO TO NEXT TEST
3842
3843 024764 010037 001162          MOV     #214,@#CCR   ;CACHE OFF
3844 024770 104121          CLR     $REG1        ;SAVE BAD ADDRESS
3845 024772 000756          MOV     R0,$REG2    ;SAVE BAD ADDRESS
3846
3847 024774 012737 000214 177746      T30L04: MOV     #214,@#CCR   ;CACHE OFF
3848 025002 005037 001160          CLR     $REG1        ;SAVE BAD ADDRESS
3849 025006 012737 025242 001162      MOV     #TAD2,$REG2  ;SAVE BAD ADDRESS
3850 025014 104043          ERROR  121          ;ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
3851 025016 000744          BR       T30L05      ;LOC. NOT INVALIDATED
3852
3853 025020 012737 000214 177746      T30L05: BR       T30L05      ;GO TO END OF TEST
3854 025026 005037 001160          MOV     #214,@#CCR   ;CACHE OFF
3855 025032 010037 001162          CLR     $REG1        ;SAVE BAD ADDRESS
3856 025036 104042          MOV     R0,$REG2    ;SAVE BAD ADDRESS
3857 025040 000733          ERROR  42          ;ERROR:NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PAR.
3858
3859 025042 012737 000214 177746      T30L03: BR       T30L05      ;GO TO END OF TEST
3860
3861 025050 010046          MOV     #214,@#CCR   ;CACHE OFF
3862 025052 076600          MOV     RO,-(SP)    ;SAVE RO FOR MED INST
3863 025054 000022          MED                    ;GET CONTENTS OF LOG REG
3864 025056 052700 100001      .WORD  RLOG
3864 025056 052700 100001      BIS     #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE

```

H07

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 70
T30 TEST OF MSB ADDRESS (A10) TO VALID BIT

```

3865 025062 076600 MED ;UNLOCK ERROR LOG
3866 025064 000222 .WORD WLOG
3867 025066 012600 MOV (SP)+,RO ;RESTORE RO
3868
3869 025070 011637 001164 MOV (SP),SREG3 ;GET PC+2 OF ERROR
3870 025074 162737 000002 001164 SUB #2,SREG3 ;SAVE PC OF ERROR
3871 025102 022626 CMP (SP)+,(SP)+ ;RESTORE STACK
3872 025104 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3873 025106 000101 .WORD RSER
3874 025110 000300 SWAB RO ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3875 025112 042700 177776 BIC #177776,RO ;ONLY LOOK AT A17, A16
3876 025116 010037 001160 MOV RO,SREG1 ;SAVE ADDRESS
3877 025122 076600 MED ;GET LOG INFORMATION
3878 025124 000102 .WORD LOADD
3879 025126 010037 001162 MOV RO,SREG2 ;SAVE INFORMATION
3880 025132 076600 MED ;GET LOG INFORMATION
3881 025134 000100 .WORD RJAM
3882 025136 032700 000400 BIT #400,RO ;ERROR IN BACKING STORE?
3883 025142 001402 BEQ 1$ ;BRANCH IF NO
3884 025144 104001 ERROR 1 ;ERROR:UNEXP. PARITY ERROR IN BACKING STORE
3885 025146 000670 BR T30L05 ;GO TO END OF TEST
3886
3887 025150 032737 000040 177744 1$: BIT #40,2$EREG ;PARITY ERROR TAG?
3888 025156 001411 BEQ 2$ ;BRANCH IF NO
3889 025160 076600 MED ;GET TAG LOG INFO.
3890 025162 000107 .WORD RTAG
3891 025164 000300 SWAB RO ;PUT TAG IN LOW BYTE
3892 025166 042700 177400 BIC #177400,RO ;LOOK AT TAG ONLY
3893 025172 010037 001164 MOV RO,SREG3 ;SAVE BAD DATA
3894 025176 104122 ERROR 122 ;ERROR:TEST OF MSB ADDR. (A10) TO VALID BIT FAILED
3895 ; PARITY ERROR TAG
3896 025200 000653 BR T30L05 ;GO TO END OF TEST
3897
3898 025202 032737 000100 177744 2$: BIT #100,2$EREG ;PARITY ERROR LOW BYTE?
3899 025210 001406 BEQ 3$ ;BRANCH IF NO
3900 025212 076600 MED ;GET LOG INFORMATION
3901 025214 000106 .WORD CDL
3902 025216 010037 001164 MOV RO,SREG3 ;SAVE INFORMATION
3903 025222 104123 ERROR 123 ;ERROR:TEST OF MSB ADDR. (A10) TO VALID BIT FAILED
3904 ; PARITY ERROR LOW BYTE
3905 025224 000641 BR T30L05 ;GO TO END OF TEST
3906
3907 025226 3$: MED ;GET LOG INFORMATION
3908 025226 076600 .WORD CDH
3909 025230 000106 MOV RO,SREG3 ;SAVE INFORMATION
3910 025232 010037 001164 ERROR 124 ;ERROR:TEST OF MSB ADDR. (A10) TO VALID BIT FAILED
3911 025236 104124 ; PARITY ERROR HIGH BYTE
3912 BR T30L05 ;GO TO END OF TEST
3913 025240 000633
3914
3915 025242 000000 TAD2: .WORD 0 ;TEST ADDRESS
3916
3917 ;*****
3918 ;*TEST 31 TEST OF MSB ADDRESS (A10) TO CACHE TAG FIELD
3919 ;*
3920 ;* THIS TEST CHECKS FOR DUAL ADDRESSING ON THE TAG

```

3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947 025244 012737 000214 177746
3948 025252 000004
3949 025254 012737 025750 001234
3950 025262 032777 010000 153644
3951 025270 001402
3952 025272 000137 025750
3953 025276 012737 025534 000114
3954 025304 052737 000200 036034
3955 025312 004737 035750
3956 025316 013700 036322
3957 025322 005100
3958 025324 005001
3959 025326 005201
3960 025330 006300
3961 025332 100775
3962 025334 006200
3963 025336 077102
3964 025340 042700 000037
3965 025344 010037 172352
3966 025350 013737 036322 172350
3967
3968
3969
3970 025356 012700 025746
3971 025362 042700 174000
3972 025366 010001
3973 025370 062700 100000
3974 025374 062701 122000
3975 025400 005005
3976 025402 052737 000001 177572

```

: #FIELD FOR THE MSB ADDRESS (A10) TO CACHE. THERE ARE TWO
: #PASSES. THE FIRST EXERCISES THE ADDRESS BITS IN THE TAG
: #FIELD AND THE SECOND EXERCISES THE TAG P BIT. INITIALLY
: #THE MEMORY IS SIZED TO DETERMINE THE MAXIMUM TESTABLE
: #ADDRESS. THE TAG FIELD OF THE MAX ADDR. IS USED AS THE
: #FIRST TEST VALUE AND ITS COMPLEMENT AS THE SECOND. THESE
: #TAG VALUES ARE THEN PUT INTO CACHE LOCATIONS WITH THE
: #SAME CACHE ADDRESS (A1-A9) EXCEPT FOR THEIR ADDRESS BIT
: #A10 COMPLEMENTS. THE LOCS IN CACHE ARE CHOSEN SO THAT
: #THEY DON'T OVERLAP THE TEST INSTRUCTION ADDRESS SPACE.
: #THIS IS TO PREVENT THEIR BEING SWAPT OUT WHEN THE INSTRUC-
: #TIONS ARE BEING EXECUTED. AFTER THE LOCATIONS ARE
: #WRITTEN THEY ARE EXAMINED AND CHECKED TO BE HITS.
: #FOLLOWING THIS THE SECOND PASS IS DONE FOR THE TAG P BIT.
: #TWO NEW TAG VALUES ARE CHOSEN WITH OPPOSITE P BITS. THEY ARE
: #THEN WRITTEN, READ AND TESTED FOR HITS. ANY PARITY ERRORS
: #OR HIT ERRORS ARE REPORTED.
: # KIPAR4, 5 CONTAIN THE TAG VALUES WHICH ARE STORED IN
: #CACHE.
: # R0, R1 CONTAIN THE CACHE TEST LOC THAT DON'T OVERLAP
: #THE INSTRUCTION ADDRESS SPACE
: # R5 CONTAINS THE PASS #.
: # IF THE INHIBIT TEST USING KT SWITCH (SW12) IS SET,
: #THIS TEST IS SKIPPED.

```

```

: *****
TST31: MOV #214, @CCR ;CACHE OFF FOR SCOPE
SCOPE
MOV #TST32, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
BIT #SW12, @SWR ;INHIBIT TESTS USING KT?
BEQ 3$ ;CONTINUE TEST IF NO
JMP @TST32 ;GO TO NEXT TEST
3$: MOV #T16L01, @PVEC ;SET UP PARITY ERROR HANDLER
BIS #200, @SKT11 ;USE KT FOR SSIZE
JSR PC, SSIZE ;SIZE MEMORY
MOV @LSTBK, R0 ;GET LAST ADDRESS AND
COM R0 ;CALC. ITS COMPLEMENT
CLR R1 ;KEEPING THE MSB THAT ARE 0
1$: INC R1 ;A 0
ASL R0
BMI 1$
2$: ASR R0
SOB R1, 2$
BIC #37, R0 ;ONLY COMPLEMENT TAG ADDRESS BITS
MOV R0, @KIPAR5 ;SET UP PAR5 WITH COMPLEMENT ADDRESS BITS
MOV @LSTBK, @KIPAR4 ;SET UP PAR4 WITH COMPLEMENT ADDRESS BITS

;SET UP R0, R1 TO ADDR. LOCS WHICH DON'T OVERLAP THIS TEST'S INSTRUCTION SPACE
MOV #LAST1, R0 ;GET ADDR. OF LAST IN THIS TEST
BIC #174000, R0 ;SAVE LOWER ADDR BITS A10-A0
MOV R0, R1 ;COPY ADDRESS
ADD #100000, R0 ;HAVE R0 ADDR PAR4
ADD #122000, R1 ;HAVE R1 ADDR PAR5 & HAVE A10 COMP OF R0
CLR R5 ;INDICATE PASS 1
BIS #1, @MMRO ;KT ON

```

3977	025410	012737	000200	177746	T16L05:	MOV	#200, @#CCR	;CACHE ON
3978	025416	021011				CMP	(R0), (R1)	;GET LOC IN CACHE VIA DATI
3979	025420	005710				TST	(R0)	;READ CACHE
3980	025422	033727	177752	000004		BIT	@#HMR, #HMR2	;SEE IF HIT
3981	025430	001425				BEQ	T16L02	;BRANCH IF NO TO ERROR
3982	025432	005711				TST	(R1)	;READ CACHE
3983	025434	033727	177752	000004		BIT	@#HMR, #HMR2	;HIT?
3984	025442	001412				BEQ	T16L03	;BRANCH IF NO
3985	025444	005705				TST	R5	;FIRST PASS?
3986	025446	001131				BNE	T16L04	;BRANCH IF NO TO END OF TEST
3987	025450	052705	00001			BIS	#1, R5	;SET FLAG FOR SECOND PASS
3988	025454	005037	172350			CLR	@#KIPAR4	;SET UP PAR4 TO TEST P BIT
3989	025460	012737	000040	172352		MOV	#40, @#KIPARS	;SET UP PARS TO TEST P BIT
3990	025466	000750				BR	T16L05	;TEST IT
3991								
3992	025470	052737	000014	177746	T16L03:	BIS	#14, @#CCR	;CACHE OFF
3993	025476	010137	001172			MOV	R1, \$TMP0	;GET VIRTUAL ADDRESS
3994	025502	000405				BR	T16L06	;CONVERT VIRTUAL INTO PHYSICAL ADDR
3995								
3996	025504	052737	000014	177746	T16L02:	BIS	#14, @#CCR	;CACHE OFF
3997	025512	010037	001172			MOV	R0, \$TMP0	;GET VIRTUAL ADDR.
3998	025516	004737	033434		T16L06:	JSR	PC, VIP	;CHANGE VIRTUAL ADDRESS INTO PHYSICAL
3999	025522	012737	025410	001110		MOV	@T16L05, @#SLPERR	;SETUP RETURN FOR ERROR LOOP
4000	025530	104067				ERROR	67	;ERROR: TEST OF MSB ADDRESS (A10) TO TAG FIELD FAILED
4001								;ADDRESS COULD NOT BE MADE A HIT
4002	025532	000477				BR	T16L04	;GO TO END OF TEST
4003								
4004	025534	052737	000014	177746	T16L01:	BIS	#14, @#CCR	;CACHE OFF
4005								
4006	025542	010046				MOV	R0, -(SP)	;SAVE R0 FOR MED INST
4007	025544	076600				MED		;GET CONTENTS OF LOG REG
4008	025546	000022				.WORD	RLOG	
4009	025550	052700	100001			BIS	#100001, R0	;ENABLE ERROR LOG & LOG FIRST MODE
4010	025554	076600				MED		;UNLOCK ERROR LOG
4011	025556	000222				.WORD	WLOG	
4012	025560	012600				MOV	(SP)+, R0	;RESTORE R0
4013								
4014	025562	011637	001164			MOV	(SP), \$REG3	;GET PC+2 OF ERROR
4015	025566	162737	000002	001164		SUB	#2, \$REG3	;SAVE PC OF ERROR
4016	025574	022626				CMP	(SP)+, (SP)+	;RESTORE STACK
4017	025576	076600				MED		;GET LOG INFOR FOR PHY. ADDR. A17, A16
4018	025600	000101				.WORD	RSER	
4019	025602	000300				SWAB	R0	;PUT PHY. ADDR A17, A16 IN LOW BYTE
4020	025604	042700	177776			BIC	#177776, R0	;ONLY LOOK AT A17, A16
4021	025610	010037	001160			MOV	R0, \$REG1	;SAVE ADDRESS
4022	025614	076600				MED		;GET LOG INFORMATION
4023	025616	000102				.WORD	LOADD	
4024	025620	010037	001162			MOV	R0, \$REG2	;SAVE INFORMATION
4025	025624	076600				MED		;GET LOG INFORMATION
4026	025626	000100				.WORD	RJAM	
4027	025630	032700	000400			BIT	#400, R0	;ERROR IN BACKING STORE
4028	025634	001402				BEQ	T16L07	;BRANCH IF NO
4029	025636	104001				ERROR	1	;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
4030	025640	000434				BR	T16L04	;GO TO END OF TEST
4031								
4032	025642	032737	000040	177744	T16L07:	BIT	#40, @#EREG	;ERROR IN TAG FIELD?


```

4033 025650 001411          BEQ      T16L08          ;BRANCH IF NO
4034 025652 076600          MED                      ;GET TAG LOG INFO.
4035 025654 000107          .WORD   RTAG
4036 025656 000300          SWAB    RO              ;PUT TAG IN LOW BYTE
4037 025660 042700 177400    BIC     #177400,RO      ;LOOK AT TAG ONLY
4038 025664 010037 001164    MOV     RO,$REG3       ;SAVE BAD DATA
4039 025670 104070          ERROR   70             ;ERROR: TEST OF MSB ADDR. (A10) TO ADDRESS FIELD FAILED
4040                                     TAG PARITY ERROR
4041 025672 000417          BR      T16L04          ;GO TO END OF TEST
4042
4043 025674 032737 000100 177744 T16L03: BIT     #100,$#REG     ;LOW BYTE P.E.?
4044 025702 001406          BEQ     T16L09          ;BRANCH IF NO
4045 025704 076600          MED                      ;GET LOG INFORMATION
4046 025706 000106          .WORD   CDL
4047 025710 010037 001164    MOV     RO,$REG3       ;SAVE INFORMATION
4048 025714 104071          ERROR   71             ;ERROR: TEST OF MSB ADDR. (A10) TO ADDRESS FIELD FAILED
4049                                     LOW BYTE PARITY ERROR
4050 025716 000405          BR      T16L04          ;GO TO END OF TEST
4051
4052 025720          T16L09:
4053 025720 076600          MED                      ;GET LOG INFORMATION
4054 025722 000106          .WORD   CDH
4055 025724 010037 001164    MOV     RO,$REG3       ;SAVE INFORMATION
4056 025730 104072          ERROR   72             ;ERROR: TEST OF MSB ADDR. (A10) TO TAG FIELD FAILED
4057                                     HIGH BYTE PARITY ERROR
4058
4059 025732 005037 177572          T16L04: CLR     @#MMRO    ;KT OFF
4060 025736 012737 033142 000114  MOV     @#UPERR,@#PVEC ;RESTORE PARITY ERROR HANDLER
4061 025744 000401          BR      TST32           ;GO TO NEXT TEST
4062
4063 025746 000000          LAST1: .WORD   0        ;TEST ADDRESS LOCATION
4064

```

```

4065 :*****
4066 :#TEST 32      TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD
4067 :#
4068 :# THIS TEST CHECKS FOR DUAL ADDRESSING ON THE DATA FIELD
4069 :#FOR THE MSB (A10) ADDRESS TO CACHE.  THERE ARE TWO PASSES.
4070 :#THE FIRST EXERCISES THE DATA BITS AND THE SECOND EXERCISES
4071 :#THE DATA PARITY BITS.  THE TEST DATA IS STORED IN A TABLE
4072 :#(TPAT) AT THE END OF THE TEST.  INITIALLY TEST ADDRESSES
4073 :#ARE CALCULATED WHICH DON'T OVERLAP THE TEST INSTRUCTIONS
4074 :#AND WHICH HAVE THE SAME CACHE ADDRESS (A1-A9) EXCEPT FOR
4075 :#A10.  ONE ADDRESS IS THE LAST LOC IN THIS TEST (TAD1)
4076 :#AND THE SECOND LIES IN A 1K BUFFER AT THE END OF THE
4077 :#PROGRAM.  A SUBROUTINE, MAD, GENERATES THIS SECOND ADDRESS.
4078 :#ON THE FIRST PASS DIFFERENT TEST DATA IS WRITTEN IN THE
4079 :#CONGRUENT ADDRESS AND THEN CHECKED TO BE A HIT AND TO
4080 :#BE THE CORRECT VALUE.  ON THE SECOND PASS NEW DATA IS
4081 :#CHOSEN, WHICH GENERATES OPPOSITE PARITY IN THE DATA FIELD,
4082 :#IS WRITTEN IN THE ADDRESSES AND THEN CHECKED TO BE A HIT
4083 :#AND TO BE THE CORRECT VALUE.  ANY PARITY ERRORS OR HIT
4084 :#ERRORS ARE REPORTED.
4085 :# R2 CONTAINS THE CONGRUENT ADDRESS FOR TAD1
4086

```

```

4087 :*****
4088 025750 012737 000214 177746 TST32: MOV     #214,@#CCR    ;CACHE OFF FOR SCOPE

```

4089	025756	000004				SCOPE		
4090	025760	012737	026444	001234		MOV	#TST33 SKTST	;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4091	025766	012737	026224	000114		MOV	#T17L01,#PVEC	;SET UP FOR PARITY ERRORS
4092	025774	004737	033714			JSR	PC,HAD	;CALC CONGRUENT ADDRESS IN TEST BUFFER
4093	026000	026442				.WORD	TAD1	;TEST ADDRESS
4094	026002	013702	001172			MOV	STMPD,R2	;SAVE CONGRUENT ADDRESS
4095	026006	005000				CLR	R0	;INIT TEST PATTERN ADDRESS REG
4096	026010	012737	000200	177746	T17L08:	MOV	#200,#CCR	;ALL CACHE ON
4097	026016	016037	026432	026442	T17L07:	MOV	TPAT(R0),#TAD1	;WRITE CACHE LOCS WITH
4098	026024	016012	026436			MOV	TPAT+4(R0),(R2)	;ADDRESS BIT A10 COMPLEMENTED
4099	026030	013701	026442			MOV	#TAD1,R1	;SEE IF DATA IN CACHE
4100	026034	033727	177752	000004		BIT	#HMR,#HMR2	;HIT?
4101	026042	001420				BEQ	T17L02	;BRANCH IF NO TO ERROR
4102	026044	020160	026432			CMP	R1,TPAT(R0)	;DATA CORRECT?
4103	026050	001051				BNE	T17L03	;BRANCH IF NO TO ERROR
4104	026052	011201				MOV	(R2),R1	;SEE IF NEXT DATA IN CACHE
4105	026054	033727	177752	000004		BIT	#HMR,#HMR2	;HIT?
4106	026062	001425				BEQ	T17L04	;BRANCH IF NO TO ERROR
4107	026064	020160	026436			CMP	R1,TPAT+4(R0)	;DATA OK?
4108	026070	001030				BNE	T17L05	;BRANCH IF NO TO ERROR
4109	026072	005760	026436			TST	TPAT+4(R0)	;TEST IF FIRST PASS
4110	026076	100151				BPL	T17L06	;BRANCH TO END OF TEST IF NO
4111	026100	005720				TST	(R0)+	;UPDATE ADDRESS
4112	026102	000745				BR	T17L07	;GO TEST NEW DATA
4113								
4114	026104	052737	000014	177746	T17L02:	BIS	#14,#CCR	;CACHE OFF
4115	026112	012737	026442	001162		MOV	#TAD1,\$REG2	;SAVE ADDRESS
4116	026120	012737	026010	001110	T17L09:	MOV	#T17L08,#SLPERR	;INIT. RETURN FOR ERROR LOOP
4117	026126	005037	001160			CLR	\$REG1	;SAVE ADDRESS
4118	026132	104062				ERROR	62	;ERROR: TEST OF MSB ADDRESS (A10) TO DATA FIELD FAILED
4119								;ADDRESS COULD NOT BE MADE A HIT
4120	026134	000532				BR	T17L06	;GO TO END OF TEST
4121								
4122	026136	052737	000014	177746	T17L04:	BIS	#14,#CCR	;CACHE OFF
4123	026144	010237	001162			MOV	R2,\$REG2	;SAVE ADDRESS
4124	026150	000763				BR	T17L09	;REPORT ERROR
4125								
4126	026152	052737	000014	177746	T17L05:	BIS	#14,#CCR	;CACHE OFF
4127	026160	016037	026436	001166		MOV	TPAT+4(R0),\$REG4	;SAVE GOOD DATA
4128	026166	010237	001162			MOV	R2,\$REG2	;SAVE BAD ADDRESS
4129	026172	000406				BR	T17L10	;REPORT ERROR
4130								
4131	026174	052737	000014	177746	T17L03:	BIS	#14,#CCR	;CACHE OFF
4132	026202	016037	026432	001166		MOV	TPAT(R0),\$REG4	;SAVE GOOD DATA
4133	026210	010137	001164		T17L10:	MOV	R1,\$REG3	;SAVE BAD DATA
4134	026214	005037	001160			CLR	\$REG1	;SAVE BAD ADDRESS
4135	026220	104063				ERROR	63	;ERROR: TEST OF MSB ADDR. (A10) TO DATA FIELD FAILED
4136								;ADDRESS HELD WRONG DATA
4137	026222	000477				BR	T17L06	;GO TO END OF TEST
4138								
4139	026224	052737	000014	177746	T17L01:	BIS	#14,#CCR	;CACHE OFF
4140								
4141	026232	010046				MOV	R0,-(SP)	;SAVE R0 FOR MED INST
4142	026234	076600				MED		;GET CONTENTS OF LOG REG
4143	026236	000022				.WORD	RLOG	
4144	026240	052700	100001			BIS	#100001,R0	;ENABLE ERROR LOG & LOG FIRST MODE

M07

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 75
T32 TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD

4145	026244	076600				MED			;UNLOCK ERROR LOG
4146	026246	000222				.WORD	WLOG		
4147	026250	012600				MOV	(SP)+,RO		;RESTORE RO
4148									
4149	026252	011637	001164			MOV	(SP),SREG3		;GET PC+2 OF ERROR
4150	026256	162737	000002	001164		SUB	#2,SREG3		;SAVE PC OF ERROR
4151	026264	022626				CMP	(SP)+,(SP)+		;RESTORE STACK
4152	026266	076600				MED			;GET LOG INFOR FOR PHY. ADDR. A17,A16
4153	026270	000101				.WORD	RSER		
4154	026272	000300				SWAB	RO		;PUT PHY. ADDR A17, A16 IN LOW BYTE
4155	026274	042700	177776			BIC	#177776,RO		;ONLY LOOK AT A17, A16
4156	026300	010037	001160			MOV	RO,SREG1		;SAVE ADDRESS
4157	026304	076600				MED			;GET LOG INFORMATION
4158	026306	000102				.WORD	LOADD		
4159	026310	010037	001162			MOV	RO,SREG2		;SAVE INFORMATION
4160	026314	076600				MED			;GET LOG INFORMATION
4161	026316	000100				.WORD	RJAM		
4162	026320	032700	000400			BIT	#400,RO		;ERROR IN BACKING STORE
4163	026324	001402				BEQ	T17L11		;BRANCH IF NO
4164	026326	104001				ERROR	1		;ERROR: UNEXP. PARITY ERROR IN BACKING STORE
4165	026330	000434				BR	T17L06		;GO TO END OF TEST
4166									
4167	026332	032737	000100	177744	T17L11:	BIT	#100,#EREG		;PARITY ERROR LOW BYTE?
4168	026340	001406				BEQ	T17L12		;BRANCH IF NO
4169	026342	076600				MED			;GET LOG INFORMATION
4170	026344	000106				.WORD	CDL		
4171	026346	010037	001164			MOV	RO,SREG3		;SAVE INFORMATION
4172	026352	104064				ERROR	64		;ERROR: TEST OF MSB ADDR. (A10) TO DATA FIELD FAILED
4173									;PARITY ERROR LOW BYTE
4174	026354	000422				BR	T17L06		;GO TO END OF TEST
4175									
4176	026356	032737	000200	177744	T17L12:	BIT	#200,#EREG		;PARITY ERROR HIGH BYTE?
4177	026364	001406				BEQ	T17L13		;BRANCH IF NO
4178	026366	076600				MED			;GET LOG INFORMATION
4179	026370	000106				.WORD	CDL		
4180	026372	010037	001164			MOV	RO,SREG3		;SAVE INFORMATION
4181	026376	104065				ERROR	65		;ERROR: TEST OF MSB ADDR. (A10) TO DATA FIELD FAILED
4182									;PARITY ERROR HIGH BYTE
4183	026400	000410				BR	T17L06		;GO TO END OF TEST
4184									
4185	026402				T17L13:				
4186	026402	076600				MED			;GET TAG LOG INFO.
4187	026404	000107				.WORD	RTAG		
4188	026406	000300				SWAB	RO		;PUT TAG IN LOW BYTE
4189	026410	042700	177400			BIC	#177400,RO		;LOOK AT TAG ONLY
4190	026414	010037	001164			MOV	RO,SREG3		;SAVE BAD DATA
4191	026420	104066				ERROR	66		;ERROR: TEST OF MSB ADDR. (A10) TO DATA FIELD FAILED
4192									;PARITY ERROR TAG
4193									
4194	026422	012737	033142	000114	T17L06:	MOV	#UPERR,#PVEC		;RESTORE PARITY ERROR HANDLER
4195	026430	000405				BR	TST33		;GO TO NEXT TEST
4196									
4197									
4198	026432	066666			TPAT:	.WORD	66666		;TEST DATA FOR DATA BIT TEST
4199	026434	000401				.WORD	401		;TEST DATA FOR PARITY BIT TEST
4200	026436	111111				.WORD	111111		;TEST DATA FOR DATA BIT TEST

4201 026440 001403
4202
4203 026442 000000
4204
4205
4206
4207
4208
4209
4210
4211
4212
4213
4214 026444 012737 000214 177746
4215 026452 000004
4216 026454 012737 026634 001234
4217 026462 012737 000200 177746
4218 026470 012737 026510 000004
4219 026476 012737 177777 060000
4220 026504 005037 060001
4221
4222 026510 022626
4223
4224 026512 010046
4225 026514 076600
4226 026516 000022
4227 026520 052700 100001
4228 026524 076600
4229 026526 000222
4230 026530 012600
4231
4232 026532 013700 060000
4233 026536 033727 177752 000004
4234 026544 001407
4235 026546 020027 177777
4236 026552 001016
4237 026554 012737 033352 000004
4238 026562 000424
4239
4240 026564 052737 000014 177746
4241 026572 005037 001160
4242 026576 012737 060000 001162
4243 026604 104043
4244 026606 000762
4245
4246 026610 032737 000014 177746
4247 026616 005037 001160
4248 026622 012737 060001 001162
4249 026630 104116
4250 026632 000750
4251
4252
4253
4254
4255
4256

```

.WORD 1403 ;TEST DATA FOR PARITY BIT TEST
TAD1: .WORD 0 ;TEST ADDRESS
*****
;#TEST 33 TEST CACHE IS NOT ALLOCATED DURING ODD ADDRESS TRAP
;#
;#THIS TEST FIRST PUTS DATA IN A CACHE LOC. THEN A WORD
;#INSTRUCTION TO AN ODD BYTE ADDRESS TRIES TO CLEAR THE
;#LOC AND FORCE AN ODD ADDRESS ERROR. UPON TRAPPING, THE
;#LOC IN CACHE IS LOOKED AT AND VERIFIED TO NOT HAVE CHANGED.
*****
TST33: MOV #214, @#CCR ;CACHE OFF FOR SCOPE
SCOPE
MOV #TST34, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
MOV #200, @#CCR ;CACHE ON
MOV #T27L01, @#4 ;SETUP FOR ODD ADDRESS TRAP
MOV #177777, @#BUFL ;PUT DATA IN CACHE
CLR @#BUFL+1 ;FORCE ODD ADDRESS ERROR
T27L01: CMP (SP)+, (SP)+ ;RESTORE THE STACK
MOV RO, -(SP) ;SAVE RO FOR MED INST
MED ;GET CONTENTS OF LOG REG
.WORD RLOG
BIS #100001, RO ;ENABLE ERROR LOG & LOG FIRST MODE
MED ;UNLOCK ERROR LOG
.WORD WLOG
MOV (SP)+, RO ;RESTORE RO
MOV @#BUFL, RO ;GET DATA
BIT @#HMR, @#HMR2 ;HIT?
BEQ T27L02 ;BRANCH TO ERROR IF NO
CMP RO, #177777 ;DATA UNCHANGED?
BNE T27L03 ;BRANCH IF YES TO ERROR
T27L04: MOV @UT4, @#4 ;RESTORE HANDLER FOR UNEXPECTED TRAPS TO 4
BR TST34 ;GO TO NEXT TEST
T27L02: BIS #14, @#CCR ;CACHE OFF
CLR $REG1 ;SAVE FAILING ADDRESS
MOV @#BUFL, $REG2 ;SAVE FAILING ADDRESS
ERROR 43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT
BR T27L04 ;GO TO END OF TEST
T27L03: BIT #14, @#CCR ;CACHE OFF
CLR $REG1 ;SAVE BAD ADDRESS
MOV @#BUFL+1, $REG2 ;SAVE BAD ADDRESS
ERROR 116 ;ERROR: CACHE ALLOCATED DURING ODD ADDRESS TRAP
BR T27L04 ;GO TO END OF TEST
*****
;#TEST 34 TEST CACHE NOT ALLOCATED DURING RED ZONE TRAP
;#
;# THIS TEST FIRST PUTS DATA IN A CACHE LOC WHICH COR-
;#RESPONDS TO A RED ZONE ADDRESS. A STACK OPERATION IS

```

4257
4258
4259
4260
4261
4262 026634 012737 000214 177746
4263 026642 000004
4264 026644 012737 027030 001234
4265 026652 012737 000200 177746
4266 026660 012737 026706 000004
4267 026666 005037 177774
4268 026672 005037 000336
4269 026676 012706 000336
4270 026702 012716 177777
4271
4272 026706 012706 001100
4273
4274 026712 010046
4275 026714 076600
4276 026716 000022
4277 026720 052700 100001
4278 026724 076600
4279 026726 000222
4280 026730 012600
4281
4282 026732 013700 000336
4283 026736 033727 177752 000004
4284 026744 001412
4285 026746 005700
4286 026750 001022
4287 026752 012737 033352 000004
4288 026760 005037 000000
4289 026764 005037 000002
4290 026770 000417
4291
4292 026772 052737 000014 177746
4293 027000 005037 001160
4294 027004 012737 000336 001162
4295 027012 104043
4296 027014 000756
4297
4298 027016 052737 000014 177746
4299 027024 104117
4300 027026 000751
4301
4302
4303
4304
4305
4306
4307
4308
4309
4310
4311
4312

```

; *DONE TO THIS ADDRESS WHICH WILL CHANGE THE DATA IF
; *COMPLETED. UPON TRAPPING, THE DATA IN CACHE IS LOOKED
; *AT AND VERIFIED TO NOT HAVE CHANGED.

; *****
TST34: MOV      #214, @#CCR      ; CACHE OFF FOR SCOPE
        SCOPE
        MOV      #TST35, SKTST   ; SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        MOV      #200, @#CCR     ; CACHE ON
        MOV      #T28L01, @#4    ; SET UP FOR RED ZONE TRAPS
        CLR      @#177774        ; INITIALIZE THE STACK LIMIT REG
        CLR      @#336           ; INITIALIZE TEST LOC
        MOV      #336, SP        ; PUT RED ZONE TRAP ADDRESS IN STACK PTER
        MOV      #177777, (SP)   ; FORCE RED ZONE TRAP

T28L01: MOV      #1100, SP       ; RESTORE THE STACK
        MOV      RO, -(SP)       ; SAVE RO FOR MED INST
        MED      RLOG           ; GET CONTENTS OF LOG REG
        .WORD   #100001, RO      ; ENABLE ERROR LOG & LOG FIRST MODE
        MED      WLOG           ; UNLOCK ERROR LOG
        .WORD   (SP)+, RO       ; RESTORE RO
        MOV      @#336, RO       ; GET DATA
        BIT      @#HMR, #HMR2    ; HIT?
        BEQ     T28L02          ; BRANCH IF NO
        TST     RO              ; DATA UNCHANGED?
        BNE     T28L03          ; BRANCH IF NO TO ERROR
T28L04: MOV      #UT4, @#4       ; RESTORE HANDLER FOR UNEXP. TRAPS TO 4
        CLR     @#0             ; RESTORE LOC 0
        CLR     @#2             ; RESTORE LOC 2
        BR      TST35           ; GO TO NEXT TEST

T28L02: BIS      #14, @#CCR      ; CACHE OFF
        CLR     $REG1           ; SAVE FAILING ADDR.
        MOV     #336, $REG2      ; SAVE FAILING ADDR.
        ERROR   43              ; ERROR: ADDRESS COULD NOT BE MADE A HIT
        BR      T28L04          ; GO TO END OF TEST

T28L03: BIS      #14, @#CCR      ; CACHE OFF
        ERROR   117             ; ERROR: CACHE ALLOCATED DURING RED ZONE TRAP
        BR      T28L04          ; GO TO END OF TEST

; *****
; *TEST 35 TEST CACHE NOT ALLOCATED DURING KT ABORT
; *
; * DATA IS PUT IN CACHE IN A TEST BUFFER ADDRESS. KIPAR4
; * IS SET UP TO REFERENCE THAT ADDRESS AND KIPDR4 IS SET
; * UP TO ABORT ACCESSES TO NON RESIDENT PAGE. THE KT IS
; * TURNED ON AND A MEMORY REFERENCE THROUGH KIPAR4 IS MADE
; * WHICH WOULD MODIFY THE TEST LOCATION IF COMPLETED. UPON
; * TRAPPING, THE LOCATION IS LOOKED AT AND VERIFIED TO NOT
; * HAVE CHANGED.
; * IF THE INHIBIT TEST USING KT SWITCH (SW12) IS SET,

```

```

4313
4314
4315
4316 027030 012737 000014 177746
4317 027036 000004
4318 027040 012737 027300 001234
4319 027046 032777 010000 152060
4320 027054 001111
4321 027056 052737 000200 036034
4322 027064 004737 035750
4323 027070 012737 027152 000250
4324 027076 012737 077400 172310
4325 027104 012700 060000
4326 027110 042700 160000
4327 027114 052700 100000
4328 027120 013737 172346 172350
4329 027126 012737 000200 177746
4330 027134 012737 177777 060000
4331 027142 052737 000001 177572
4332 027150 005010
4333
4334 027152 022626
4335
4336 027154 010046
4337 027156 076600
4338 027160 000022
4339 027162 052700 100001
4340 027166 076600
4341 027170 000222
4342 027172 012600
4343
4344 027174 013701 060000
4345 027200 033727 177752 000004
4346 027206 001415
4347 027210 020127 177777
4348 027214 001024
4349 027216 042737 000001 177572
4350 027224 052737 000006 172310
4351 027232 012737 000252 000250
4352 027240 000417
4353
4354 027242 052737 000014 177746
4355 027250 005037 001160
4356 027254 012737 060000 001160
4357 027262 104043
4358 027264 000754
4359
4360 027266 052737 000014 177746
4361 027274 104120
4362 027276 000747
4363
4364
4365
4366
4367
4368

```

;*THIS TEST IS SKIPPED.

```

TST35: MOV      #14, @CCR      ;CACHE OFF FOR SCOPE
        SCOPE
        MOV      @TST36, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        BIT      @SW12, @SWR   ;INHIBIT TESTS USING KT?
        BNE     TST36         ;YES, GO TO NEXT TEST
        BIS      @200, @SKT11  ;USE KT FOR $SIZE
        JSR     PC, $SIZE     ;USE $SIZE TO SET UP PAR'S AND PDR'S
        MOV      @T29L01, @250 ;SET UP FOR KT ABORTS
        MOV      @77400, @KIPDR4 ;SET UP PDR4 TO ABORT ACCESS TO NON RESIDENT PAGE
        MOV      @BUFL, @R0    ;GET TEST ADDRESS
        BIC      @160000, @R0  ;MASK ITS PAR ADDRESS
        BIS      @100000, @R0  ;HAVE IT ADDRESS PAR4
        MOV      @KIPAR3, @KIPAR4 ;INIT PAR4 TO HAVE SAME OFFSET AS PAR3 FOR THE BUFFER
        MOV      @200, @CCR    ;CACHE ON
        MOV      @177777, @BUFL ;INIT TEST ADDRESS
        BIS      @1, @MMR0    ;KT ON
        CLR     (R0)         ;FORCE KT ABORT

T29L01: CMP      (SP)+, (SP)+ ;RESTORE STACK
        MOV      @R0, -(SP)   ;SAVE R0 FOR MED INST
        MED     @RLOG         ;GET CONTENTS OF LOG REG
        .WORD   @RLOG
        BIS      @100001, @R0 ;ENABLE ERROR LOG & LOG FIRST MODE
        MED     @WLOG
        .WORD   @WLOG
        MOV      (SP)+, @R0   ;RESTORE R0

        MOV      @BUFL, @R1   ;GET ADDRESS
        BIT      @HMR, @HMR2  ;HIT?
        BEQ     T29L02       ;BRANCH IF NO
        CMP      @R1, @177777 ;DATA OK?
        BNE     T29L03       ;BRANCH IF NO
T29L04: BIC      @1, @MMR0    ;KT OFF
        BIS      @6, @KIPDR4  ;ALLOW READ OR WRITE TO PAGE
        MOV      @252, @250   ;RESTORE KT TRAP CATCHER
        BR     TST36         ;GO TO NEXT TEST

T29L02: BIS      @14, @CCR    ;CACHE OFF
        CLR     @SREG1       ;SAVE FAILING ADDRESS
        MOV      @BUFL, @SREG1 ;SAVE FAILING ADDRESS
        ERROR   43          ;ERROR: ADDRESS COULD NOT BE MADE A HIT
        BR     T29L04       ;GO TO END OF TEST

T29L03: BIS      @14, @CCR    ;CACHE OFF
        ERROR   120         ;ERROR: CACHE ALLOCATED DURING KT ABORT
        BR     T29L04       ;GO TO END OF TEST

```

```

*TEST 36      DYNAMIC TEST OF CACHE
*
* THIS TEST CREATES A GREAT DEAL OF ACTIVITY IN CACHE
*TO TRY TO FIND ANY NOISE OR TIMING PROBLEMS.  THESE

```

4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396
4397
4398
4399
4400
4401
4402
4403
4404
4405
4406
4407
4408
4409
4410
4411
4412
4413
4414
4415
4416
4417
4418
4419
4420
4421
4422
4423
4424

027300 012737 000214 177746
027306 000004
027310 012737 030260 001234
027316 012737 030036 000114
027324 012737 027710 000010
027332 012737 027340 001110
027340 012737 000200 177746

027346 012703 060000
027352 012702 002000
027356 012701 176540
027362 012700 023456
027366 060001
027370 010123
027372 000261
027374 006101
027376 006000
027400 077206

027402 012700 060000
027406 012701 060000
027412 012702 002000
027416 012703 002000
027422 005721
027424 077202
027426 022041
027430 077302

027432 012700 060000
027436 012701 060000
027442 012702 002000
027446 012703 002000
027452 005004

:#PROBLEMS WILL BE DETECTED VIA THE PARITY ERRORS, ILLEGAL
:#INSTRUCTION TRAPS OR DATA CHANGES THEY CAUSE. FIRST
:#CACHE IS LOADED WITH AN ALTERNATING DATA PATTERN (525,252).
:#THEN IT IS REFERENCED AS QUICKLY AS POSSIBLE IN OPPOSITE
:#DIRECTIONS TO CAUSE LARGE CHANGES IN THE ADDRESS LINES AND
:#RAPID CHANGES IN THE DATA LINES. THIS IS THEN REPEATED
:#WITH A DIFFERENT DATA PATTERN AND THE CACHE IS MODIFIED
:#AS THE REFERENCES OCCUR. AFTER THIS THE LOCATIONS ARE
:#CHECKED TO CONTAIN THEIR PROPER VALUES.
:# FOLLOWING THIS, THE TAG FIELD IS WRITTEN WITH A
:#CHANGING PATTERN, THEN THE CACHE IS REFERENCED AS QUICKLY
:#AS POSSIBLE IN OPPOSITE DIRECTIONS TO CAUSE LARGE CHANGES
:#IN THE ADDRESS LINES AND RAPID CHANGES IN THE TAG FIELD.
:#THIS LAST PART IS SKIPPED IF THE INHIBIT TEST USING KT
:#SWITCH (SW12) IS SET.

T36: MOV #214,2#CCR ;CACHE OFF FOR SCOPE
SCOPE
MOV #TST37,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
MOV #T18L01,2#PVEC ;SETUP FOR PARITY ERRORS
MOV #T18L02,2#10 ;SETUP FOR TRAPS TO ILLEGAL INST
MOV #T18L11,2#SLPERR ;INIT RETURN FOR ERROR LOOPS
T18L11: MOV #200,2#CCR ;CACHE ON

;GENERATE TEST DATA IN A 1K BUFFER
MOV #BUFL,R3 ;GET STARTING ADDRESS OF BUFFER
MOV #2000,R2 ;INIT REG FOR 1K COUNT
MOV #176540,R1 ;PUT RANDOM # IN REG
MOV #023456,R0 ;PUT RANDOM # IN REG
15: ADD R0,R1 ;GENERATE NEW RANDOM DATA
MOV R1,(R3)+ ;SAVE DATA
SEC ;GENERATE MORE
ROL R1 ;RANDOM DATA
ROR R0
SOB R2,15 ;LOOP TILL 1K BUFFER FULL

;LOAD CACHE WITH PATTERN AND TEST CACHE
MOV #BUFL,R0 ;SET UP TO ADDRESS BUFFER
MOV #BUFL,R1 ;ASET UP TO ADDRESS BUFFER
MOV #2000,R2 ;INIT REG FOR 1K COUNT
MOV #2000,R3 ;INIT REG FOR 1K COUNT
25: TST (R1)+ ;GET DATA IN CACHE
SOB R2,25 ;LOOP TILL 1K REFERENCED
35: CMP (R0)+,-(R1) ;REFERENCE CACHE QUICKLY AND WITH COMPLEMENT ADDR
SOB R3,35 ;LOOP TILL ALL CACHE REFERENCED

;GENERATE SECOND TEST PATTERN IN BUFFER AND TEST IT
MOV #BUFL,R0 ;SET UP TO ADDRESS BUFFER
MOV #BUFL,R1 ;SET UP TO ADDRESS BUFFER
MOV #2000,R2 ;INIT REG FOR 1K COUNT
MOV #2000,R3 ;INIT REG FOR 1K COUNT
CLR R4 ;INIT DATA

```

4425 027454 010421          55:  MOV    R4,(R1)+      ;LOAD BUFFER WITH PATTERN
4426 027456 005204          INC    R4              ;CHANGE DATA
4427 027460 077203          SOB    R2,55          ;LOOP TILL 1K LOADED
4428
4429 027462 062041          65:  ADD    (R0)+,-(R1)    ;REFERENCE CACHE QUICKLY
4430 027464 077302          SOB    R3,65          ;LOOP TILL ALL CACHE REFERENCED
4431
4432                          ;CHECK DATA IN CACHE OR MAIN MEM CORRECT
4433
4434 027466 012701 001777        MOV    #1777,R1        ;INIT REG WITH GOOD DATA
4435 027472 012702 001000        MOV    #1000,R2        ;INIT REG FOR 1/2K COUNT
4436 027476 014003          75:  MOV    -(R0),R3        ;GET DATA
4437 027500 020103          CMP    R1,R3          ;DATA OK?
4438 027502 001140          BNE   T18L03          ;BRANCH IF NO TO ERROR
4439 027504 077204          SOB    R2,75          ;LOOP TILL 1/2K REFERRED
4440 027506 012702 001000        MOV    #1000,R2        ;INIT REG FOR 1/2K COUNT
4441 027512 012701 002776        MOV    #2776,R1        ;INIT REG WITH 'GOOD' DATA
4442 027516 014003          105: MOV    -(R0),R3        ;GET DATA
4443 027520 020103          CMP    R1,R3          ;DATA OK?
4444 027522 001130          BNE   T18L03          ;BRANCH IF NO TO ERROR
4445 027524 005301          DEC    R1              ;ADJUST GOOD DATA
4446 027526 077205          SOB    R2,105         ;LOOP TILL ALL DATA CHECKED
4447
4448                          ;NOW TEST TAG MEM
4449
4450 027530 032777 010000 151376  BIT    #SW12,@SWR      ;INHIBIT TESTS USING KT?
4451 027536 001402          BEQ   115              ;CONTINUE TEST IF NO
4452 027540 000137 030260          JMP   @TST37          ;GO TO NEXT TEST
4453 027544 052737 000200 036034  115:  BIS    #200,@#SKT11   ;KT ON FOR $SIZE
4454 027552 004737 035750          JSR   PC,$$SIZE      ;SIZE MEMORY
4455 027556 013737 027564 001110          MOV    T18L05,@#SLPERR ;INIT RETURN FOR ERROR LOOPS
4456 027564 012737 000200 177746  T18L05: MOV    #200,@#CCR      ;CACHE ON
4457 027572 012700 100000          MOV    #100000,R0     ;HAVE R0 ADDRESS PAR4
4458 027576 012701 120000          MOV    #120000,R1     ;HAVE R1 ADDR. PAR5
4459 027602 012704 172350          MOV    #KIPAR4,R4     ;PUT PAR4 ADDR IN R4
4460 027606 012705 172352          MOV    #KIPAR5,R5     ;PUT PAR5 ADDR IN R5
4461 027612 013702 036322          MOV    @#SLSTBK,R2    ;GET LAST BANK
4462 027616 010215          MOV    R2,(R5)        ;SET UP PAR5
4463 027620 010214          T18L06: MOV    R2,(R4)    ;SET UP PAR4
4464 027622 052737 000001 177572  T18L06: BIS    #1,@#MMRO   ;KT ON
4465 027630 005720          T18L07: TST    (R0)+     ;WRITE CACHE VIA DATI
4466 027632 032700 003776          BIT    #3776,R0       ;ALL CACHE WRITTEN?
4467 027636 001404          BEQ   T18L09          ;BRANCH IF YES
4468 027640 162714 000040          SUB    #40,(R4)       ;CALC NEW PAR4 TO GIVE NEW TAG PATTERN
4469 027644 100371          BPL   T18L07          ;WRITE CACHE IF TAG > OR EQUAL TO 0
4470 027646 000764          BR    T18L06          ;GO INIT PAR4 TO RESTART PATTERN
4471
4472 027650 022140          T18L09: CMP    (R1)+,-(R0)  ;REFERENCE CACHE
4473 027652 032701 003776          BIT    #3776,R1       ;ALL CACHE TESTED?
4474 027656 001002          BNE   25              ;BRANCH IF NO TO CONTINUE
4475 027660 000137 030234          JMP   T18L10          ;GO TO END OF TEST
4476 027664 162715 000040          25:  SUB    #40,(R5)       ;ADJUST PAR5 FOR NEXT TEST ADDR. REF.
4477 027670 100001          BPL   15              ;TAG > OR EQUAL 0, BRANCH IF YES
4478 027672 010215          MOV    R2,(R5)        ;NO, INIT PAR5 FOR HIGHEST TAG ADDR
4479 027674 062714 000040          15:  ADD    #40,(R4)       ;ADJUST PAR4 FOR NEXT TEST ADDR.
4480 027700 020214          CMP    R2,(R4)        ;IS PAR4 > MAX ADDRESS?

```


4481	027702	002362				BGE	T18L09		;GO TEST IT IF NO
4482	027704	005014				CLR	(R4)		;RESTART PAR4 AT LOW TEST ADDR
4483	027706	000760				BR	T18L09		;GO TEST IT
4484									
4485	027710	052737	000014	177746	T18L02:	BIS	#14,2#CCR		;CACHE OFF
4486	027716	011637	001164			MOV	(SP), \$REG3		;GET PC+2 OF TRAP
4487	027722	162737	000002	001164		SUB	#2, \$REG3		;SAVE PC OF TRAP
4488	027730	022626				CMP	(SP)+, (SP)+		;RESTORE STACK
4489	027732	076600				MED			;GET LOG INFOR FOR PHY. ADDR. A17,A16
4490	027734	000101				.WORD	RSER		
4491	027736	000300				SWAB	RO		;PUT PHY. ADDR A17, A16 IN LOW BYTE
4492	027740	042700	177776			BIC	#177776, RO		;ONLY LOOK AT A17, A16
4493	027744	010037	001160			MOV	RO, \$REG1		;SAVE ADDRESS
4494	027750	076600				MED			;GET LOG INFORMATION
4495	027752	000102				.WORD	LOADD		
4496	027754	010037	001162			MOV	RO, \$REG2		;SAVE INFORMATION
4497									
4498	027760	010046				MOV	RO, -(SP)		;SAVE RO FOR MED INST
4499	027762	076600				MED			;GET CONTENTS OF LOG REG
4500	027764	000022				.WORD	RLOG		
4501	027766	052700	100001			BIS	#100001, RO		;ENABLE ERROR LOG & LOG FIRST MODE
4502	027772	076600				MED			;UNLOCK ERROR LOG
4503	027774	000222				.WORD	WLOG		
4504	027776	012600				MOV	(SP)+, RO		;RESTORE RO
4505									
4506	030000	104074				ERROR	74		;ERROR: DYNAMIC TEST OF CACHE FAILED
4507									;TRAP TO 10 OCCURRED
4508	030002	000514				BR	T18L10		;GO TO END OF TEST
4509									
4510	030004	052737	000014	177746	T18L03:	BIS	#14,2#CCR		;CACHE OFF
4511	030012	005037	001160			CLR	\$REG1		;SAVE ADDRESS
4512	030016	010037	001162			MOV	RO, \$REG2		;SAVE ADDRESS
4513	030022	010337	001164			MOV	R3, \$REG3		;SAVE BAD DATA
4514	030026	010137	001166			MOV	R1, \$REG4		;SAVE GOOD DATA
4515	030032	104073				ERROR	73		;ERROR: DYNAMIC TEST OF CACHE FAILED
4516									;LOC HELD WRONG DATA
4517	030034	000477				BR	T18L10		;GO TO NEXT TEST
4518									
4519	030036	052737	000014	177746	T18L01:	BIS	#14,2#CCR		;CACHE OFF
4520									
4521	030044	010046				MOV	RO, -(SP)		;SAVE RO FOR MED INST
4522	030046	076600				MED			;GET CONTENTS OF LOG REG
4523	030050	000022				.WORD	RLOG		
4524	030052	052700	100001			BIS	#100001, RO		;ENABLE ERROR LOG & LOG FIRST MODE
4525	030056	076600				MED			;UNLOCK ERROR LOG
4526	030060	000222				.WORD	WLOG		
4527	030062	012600				MOV	(SP)+, RO		;RESTORE RO
4528									
4529	030064	011637	001164			MOV	(SP), \$REG3		;GET PC+2 OF TRAP
4530	030070	162737	000002	001164		SUB	#2, \$REG3		;SAVE PC OF TRAP
4531	030076	022626				CMP	(SP)+, (SP)+		;ADJUST STACK
4532	030100	076600				MED			;GET LOG INFOR FOR PHY. ADDR. A17,A16
4533	030102	000101				.WORD	RSER		
4534	030104	000300				SWAB	RO		;PUT PHY. ADDR A17, A16 IN LOW BYTE
4535	030106	042700	177776			BIC	#177776, RO		;ONLY LOOK AT A17, A16
4536	030112	010037	001160			MOV	RO, \$REG1		;SAVE ADDRESS

```

4537 030116 076600 MED ;GET LOG INFORMATION
4538 030120 000102 .WORD LOADD
4539 030122 010037 001162 MOV RO,$REG2 ;SAVE INFORMATION
4540 030126 076600 MED ;GET LOG INFORMATION
4541 030130 000100 .WORD RJAM
4542 030132 032700 000400 BIT #400,RO ;ERROR IN BACKING STORE?
4543 030136 001402 BEQ T18L12 ;BRANCH IF NO
4544 030140 104001 ERROR 1 ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
4545 030142 000434 BR T18L10 ;GO TO NEXT TEST
4546
4547 030144 032737 000100 177744 T18L12: BIT #100,2#REG ;LOW BYTE PE?
4548 030152 001406 BEQ T18L13 ;BRANCH IF NO
4549 030154 076600 MED ;GET LOG INFORMATION
4550 030156 000106 .WORD CDL
4551 030160 010037 001164 MOV RO,$REG3 ;SAVE INFORMATION
4552 030164 104075 ERROR 75 ;ERROR: DYNAMIC TEST OF CACHE FAILED
4553 ; LOW BYTE PARITY ERROR
4554 030166 000422 BR T18L10 ;GO TO END OF TEST
4555
4556 030170 032737 000200 177744 T18L13: BIT #200,2#REG ;HIGH BYTE PE?
4557 030176 001406 BEQ T18L14 ;BRANCH IF NO
4558 030200 076600 MED ;GET LOG INFORMATION
4559 030202 000106 .WORD CDH
4560 030204 010037 001164 MOV RO,$REG3 ;SAVE INFORMATION
4561 030210 104076 ERROR 76 ;ERROR: DYNAMIC TEST OF CACHE FAILED
4562 ; HIGH BYTE PARITY ERROR
4563 030212 000410 BR T18L10 ;GO TO END OF TEST
4564
4565 030214 T18L14:
4566 030214 076600 MED ;GET TAG LOG INFO.
4567 030216 000107 .WORD RTAG
4568 030220 000300 SWAB RO ;PUT TAG IN LOW BYTE
4569 030222 042700 177400 BIC #177400,RO ;LOOK AT TAG ONLY
4570 030226 010037 001164 MOV RO,$REG3 ;SAVE BAD DATA
4571 030232 104077 ERROR 77 ;ERROR: DYNAMIC TEST OF CACHE FAILED
4572 ; TAG PARITY ERROR
4573
4574 030234 005037 177572 T18L10: CLR 2#MMRO ;KT OFF
4575 030240 012737 000012 000010 MOV #12,2#10 ;RESTORE TRAP CATCHER
4576 030246 005037 000012 CLR 2#12 ;RESTORE TRAP CATCHER
4577 030252 012737 033142 000114 MOV #UPERR,2#PVEC ;RESTORE HANDLER FOR PARITY ERRORS
4578
4579
4580 ;*****
4581 ;*TEST 37 TEST RETRIES TO BACKING STORE DONE ON CACHE PARITY TRAP
4582 ;*
4583 ;* THE JAMUPP ON CACHE PARITY ERROR BIT IS CLEARED AND
4584 ;*THE CACHE CONTROL REG IS TESTED TO CONTAIN THE CORRECT
4585 ;*VALUE. A CACHE LOC IS THEN WRITTEN WITH WRONG PARITY
4586 ;*AND A TRAP IS FORCED. THE LOC IS THEN REFERENCED TO SEE
4587 ;*IF IT STILL IS IN CACHE (RETRY DONE).
4588
4589 ;*****
4590 030260 012737 000214 177746 TST37: MOV #214,2#CCR ;CACHE OFF FOR SCOPE
4591 030266 000004 SCOPE
4592 030270 012737 030524 001234 MOV #TST40,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR

```

H08

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 83
T37 TEST RETRIES TO BACKING STORE DONE ON CACHE PARITY TRAP

```

4593 030276 012737 030350 000114      MOV      #15, @PVEC      ;SET UP FOR PARITY TRAP
4594 030304 042737 000200 177746      BIC      #200, @CCR      ;ENABLE RETRIES
4595 030312 032737 000200 177746      BIT      #200, @CCR      ;WAS BIT CLEARED?
4596 030320 001045                BNE      25              ;BRANCH IF NO TO ERROR
4597 030322 012737 000100 177746      MOV      #100, @CCR     ;CACHE ON, WRITE WRONG PARITY, DO RETRIES
4598 030330 005037 060000                CLR      @BUFL          ;WRITE WRONG PARITY
4599 030334 012737 000000 177746      MOV      #0, @CCR      ;WMP OFF
4600 030342 005737 060000                TST      @BUFL          ;FORCE TRAP
4601 030346 000445                BR       35              ;REPORT ERROR IF NO TRAP
4602
4603 030350 062706 000004      15:      ADD      #4, SP          ;RESTORE THE STACK
4604
4605 030354 010046                MOV      RO, -(SP)      ;SAVE RO FOR MED INST
4606 030356 076600                MED                      ;GET CONTENTS OF LOG REG
4607 030360 000022                .WORD   RLOG
4608 030362 052700 100001      BIS      #100001, RO    ;ENABLE ERROR LOG & LOG FIRST MODE
4609 030366 076600                MED                      ;UNLOCK ERROR LOG
4610 030370 000222                .WORD   WLOG
4611 030372 012600                MOV      (SP)+, RO     ;RESTORE RO
4612
4613 030374 005737 060000                TST      @BUFL          ;SEE IF DATA IN CACHE
4614 030400 033727 177752 000004      BIT      @HMR, @HMR2   ;HIT?
4615 030406 001036                BNE      T23L01        ;GO TO END OF TEST IF YES
4616 030410 012737 000214 177746      MOV      #214, @CCR    ;CACHE OFF
4617
4618                ;RID CACHE OF BAD PARITY
4619 030416 012737 000214 177746      MOV      #214, @CCR    ;CACHE OFF IF ON
4620 030424 004737 035134      JSR      PC, SWEEP     ;GO PURGE CACHE
4621
4622
4623 030430 104110                ERROR   110            ;ERROR: RETRY TO BACKING STORE NOT DONE ON CACHE PARITY
4624 030432 000424                BR      T23L01        ;GO TO END OF TEST
4625
4626 030434 013737 177746 001160 25:      MOV      @CCR, $REG1   ;SAVE BAD DATA
4627 030442 012737 000214 177746      MOV      #214, @CCR    ;CACHE OFF
4628 030450 012737 000014 001162      MOV      #14, $REG2    ;SAVE GOOD DATA
4629 030456 104026                ERROR   26            ;ERROR: CACHE CONTROL REG HELD WRONG DATA
4630 030460 000411                BR      T23L01        ;GO TO END OF TEST
4631
4632 030462 012737 000214 177746 35:      MOV      #214, @CCR    ;CACHE OFF
4633 030470 005037 001160                CLR      $REG1         ;SAVE ADDR. OF TESTED LOC
4634 030474 012737 060000 001162      MOV      @BUFL, $REG2  ;SAVE ADDR. OF TESTED LOC
4635 030502 104042                ERROR   42            ;ERROR: NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PARIT
4636
4637 030504                T23L01:
4638
4639                ;RID CACHE OF BAD PARITY
4640 030504 012737 000214 177746      MOV      #214, @CCR    ;CACHE OFF IF ON
4641 030512 004737 035134      JSR      PC, SWEEP     ;GO PURGE CACHE
4642
4643
4644 030516 012737 033142 000114      MOV      #UPERR, @PVEC ;RESTORE PARITY ERROR HANDLER
4645
4646                ;*****
4647                ;*TEST 40      TEST DATO TO I/O LOC NOT WRITTEN IN CACHE AND I/O
4648                ;*

```

4649
4650
4651
4652
4653
4654
4655
4656
4657
4658 030524 012737 000214 177746
4659 030532 000004
4660 030534 012737 030674 001234
4661 030542 012737 000200 177746
4662 030550 012737 030610 001172
4663 030556 042737 174000 001172
4664 030564 023727 001172 002326
4665 030572 002404
4666 030574 023727 001172 002340
4667 030602 003422
4668 030604 005737 002340
4669 030610 005037 172340
4670 030614 005737 002340
4671 030620 033727 177752 000004
4672 030626 001022
4673 030630 012737 000003 001160
4674 030636 012737 172340 001162
4675 030644 104025
4676 030646 000412
4677
4678 030650 005737 002340
4679 030654 005037 172340
4680 030660 005737 002340
4681 030664 033727 177752 000004
4682 030672 001756
4683
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697 030674 012737 000214 177746
4698 030702 000004
4699 030704 012737 031132 001234
4700 030712 012737 000200 177746
4701 030720 012702 060000
4702 030724 012701 002000
4703
4704

```

;* THE TEST INSTRUCTION ADDRESSES ARE FIRST EXAMINED TO
;* DETERMINE IF THEY OVERLAP THE TEST LOCATION ADDRESS IN
;* CACHE. IF THEY DO, THE TEST IS RUN IN A NON OVERLAPPING
;* ADDRESS SPACE. A LOC IS PUT IN CACHE WHICH HAS THE SAME
;* 11 LEAST SIGNIFICANT ADDRESS BITS AS THE MEMORY MANAGEMENT
;* REG KIPARO. A DATO IS THEN DONE TO KIPARO AND THE LOC
;* IS CHECKED TO STILL BE IN CACHE.

```

```

TST40: MOV #214, @CCR ;CACHE OFF FOR SCOPE
        SCOPE
        MOV #TST41, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        MOV #200, @CCR ;TURN ON ALL OF CACHE
        MOV #T19L01, @STMP0 ;SAVE ADDRESS OF TEST INSTRUCTION
        BIC #174000, @STMP0 ;LOOK AT ITS CACHE ADDRESS
        CMP @STMP0, #2326 ;INSTRUCTION AT TEST LOC?
        BLT T19L02 ;BRANCH IF NO
        CMP @STMP0, #2340 ;INSTRUCTION AT TEST LOC?
        BLE T19L03 ;BRANCH IF YES
T19L02: TST @#2340 ;PUT TEST LOC IN CACHE
T19L01: CLR @#KIPARO ;DO DATO TO I/O
        TST @#2340 ;DATA STILL IN CACHE
        BIT @#HMR, #HMR2 ;WAS IT A HIT?
        BNE TST41 ;GO TO NEXT TEST IF YES
T19L04: MOV #3, SREG1 ;SAVE PHYSICAL ADDRESS HIGH
        MOV #172340, SREG2 ;SAVE PHYSICAL ADDRESS LOW
        ERROR 25 ;ERROR: DATO TO I/O ADDRESS WRITTEN IN CACHE
        BR TST41 ;GO TO NEXT TEST

T19L03: TST @#2340 ;PUT TEST LOC IN CACHE
        CLR @#KIPARO ;DO DATO TO I/O
        TST @#2340 ;DATA STILL IN CACHE?
        BIT @#HMR, #HMR2 ;STILL A HIT?
        BEQ T19L04 ;BRANCH TO ERROR IF NO

```

TEST 41 TEST CONSOLE INITIATED SWEEP INVALIDATES ALL CACHE

```

;* A LOC IS PUT IN CACHE, CHECKED TO BE A HIT AND THEN
;* A CONSOLE SWEEP IS INITIATED. THE LOC IS AGAIN REF-
;* ERENCED TO SEE IF IT WAS INVALIDATED (NOT A HIT). THIS
;* IS DONE FOR ALL OF CACHE. BEFORE THE CONSOLE SWEEP IS
;* STARTED, THE TEST LOC IS VERIFIED TO NOT OVERLAP THE
;* PROGRAM INSTRUCTION ADDRESSES IN CACHE. IF THEY DO, THE
;* TEST IS RUN OUT OF A DIFFERENT ADDRESS SPACE.
;* RD CONTAINS THE ADDRESS UNDER TEST.

```

```

TST41: MOV #214, @CCR ;CACHE OFF FOR SCOPE
        SCOPE
        MOV #TST42, SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        MOV #200, @CCR ;CACHE ON
        MOV #BUF1, R2 ;INIT REG FOR TEST ADDRESS
        MOV #2000, R1 ;INIT LOOP COUNT

```

;DOES THE TEST ADDR OVERLAP THE SAME ADDR SPACE IN CACHE AS THE PROGRAM INSTRUCT

JOB

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 85
T41 TEST CONSOLE INITIATED SWEEP INVALIDATES ALL CACHE

```

4705
4706 030730 010237 001172 T25L04: MOV R2,STMP0 ;GET TEST ADDR.
4707 030734 012737 031006 001174 MOV #15,STMP1 ;GET PROGRAM TEST INSTRUCTION ADDR.
4708 030742 042737 174000 011172 BIC #174000,STMP0 ;CALC ADDRESSES CORRESP. CACHE ADDR
4709 030750 042737 174000 001174 BIC #174000,STMP1 ;CALC ADDRESSES CORRESP. CACHE ADDR
4710 030756 023737 001174 001172 CMP STMP1,STMP0 ;DO THE CACHE ADDRESSES OVERLAP?
4711 030764 101010 BHI 15 ;BRANCH IF NO
4712 030766 062737 000012 001174 ADD #12,STMP1 ;CALC LAST PROG. TEST INSTRUCTION ADDR
4713 030774 023737 001172 001174 CMP STMP0,STMP1 ;DO THE CACHE ADDRESSES OVERLAP?
4714 031002 101415 BLOS T25L01 ;BRANCH IF YES
4715
4716 031004 005012 CLR (R2) ;PUT THE DATA IN CACHE
4717 031006 IS: MOV #200,R0 ;SET BIT IN R0 FOR CONSOLE CACHE SWEEP
4718 031006 012700 000200 MED ;CONSOLE CACHE SWEEP
4719 031012 076600 .WORD WINIT
4720 031014 000352 TST (R2) ;SEE IF LOC STILL IN CACHE
4721 031016 005712 BIT @#HMR,#HMR2 ;HIT?
4722 031020 033727 177752 000004 BNE T25L02 ;BRANCH TO ERROR IF YES
4723 031026 001016 T25L03: TST (R2)+ ;UPDATE ADDRESS
4724 031030 005722 SOB R1,T25L04 ;BRANCH IF ALL CACHE NOT TESTED
4725 031032 077142 BR T25L04 ;GO TO NEXT TEST
4726 031034 000436
4727
4728 031036 005012 T25L01: CLR (R2) ;PUT DATA IN CACHE
4729 031040 012700 000200 MOV #200,R0 ;SET BIT IN R0 FOR CONSOLE CACHE SWEEP
4730 031044 076600 MED ;CONSOLE CACHE SWEEP
4731 031046 000352 .WORD WINIT
4732 031050 005712 TST (R2) ;SEE IF LOC STILL IN CACHE
4733 031052 033727 177752 000004 BIT @#HMR,#HMR2 ;HIT?
4734 031060 001001 BNE T25L02 ;BRANCH TO ERROR IF YES
4735 031062 000762 BR T25L03 ;LOOK AT NEXT ADDRESS
4736
4737 031064 052737 000014 177746 T25L02: BIS #14,@#CCR ;CACHE OFF
4738 031072 005037 001160 CLR $REG1 ;SAVE FAILING ADDRESS
4739 031076 010237 001162 MOV R2,$REG2 ;SAVE FAILING ADDRESS
4740 031102 012737 030730 001110 MOV #T25L04,@#SLPERR ;INIT RETURN FOR ERROR LOOP
4741 031110 104113 ERROR 113 ;ERROR: ADDR. NOT INVALIDATED BY CONSOLE SWEEP
4742 031112 123727 001103 000003 CMPB @#SERFLG,#3 ;MORE THAN 3 ERRORS?
4743 031120 101004 BHI T25L02 ;GO TO NEXT TEST IF YES
4744 031122 012737 000200 177746 MOV #200,@#CCR ;CACHE ON
4745 031130 000737 BR T25L03 ;CONTINUE TEST
4746
4747
4748 ;*****
4749 ;*TEST 42 TEST POWER UP INVALIDATES CACHE AND CLEARS CACHE CONTROL REG
4750 ;*
4751 ;* THIS TEST IS ONLY RUN IF SMD7=1. THIS IS BECAUSE
4752 ;*OPERATOR INTERVENTION IS NEEDED TO POWER DOWN AND THEN
4753 ;*UP THE MACHINE WHEN THE MESSAGE IS TYPED ON THE TTY.
4754 ;*IF RUNNING UNDER APT AND SMD7=1, THE PROGRAM ASSUMES
4755 ;*THAT A UNIBUS EXERCISOR (M7855) IS AVAILABLE
4756 ;*TO POWER DOWN AND THEN UP THE MACHINE.
4757 ;*AFTER THE MACHINE HAS DONE THIS, THE CACHE CONTROL REG
4758 ;*IS EXAMINED TO HAVE BEEN PROPERLY INITIALIZED BY POWER
4759 ;*UP. AFTER THIS ALL CACHE IS REFERENCED. THERE IS A
4760 ;*VERY HIGH PROBABILITY THAT CACHE WILL HAVE PARITY ERRORS
;*IF THE POWER UP FAILED TO SWEEP CACHE. ANY CACHE PARITY

```

K08

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 86
T42 TEST POWER UP INVALIDATES CACHE AND CLEARS CACHE CONTROL REG

4761
4762
4763
4764
4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
4802
4803
4804
4805
4806
4807
4808
4809
4810
4811
4812
4813
4814
4815
4816

031132 012737 000214 177746
031140 000004
031142 012737 031524 001234
031150 032777 000200 147756
031156 001562
031160 012737 031412 000114
031166 012737 031226 000024
031174 012737 000314 177746
031202 105737 001256
031206 001404
031210 012777 000020 147776
031216 000777
031220 104401 040625
031224 000777
031226 012737 031246 000024
031234 022626
031236 017737 147672 001172
031244 000777
031246 012706 001100
031252 105737 001256
031256 001402
031260 005077 147730
031264 013700 001172
031270 076600
031272 000226
031274 012701 177000
031300 012700 177400
031304 063737 060000 060000
031312 005200
031314 001373
031316 005201
031320 001367
031322 013737 177746 001160
031330 001401
031332 104101
031334
031334 010046
031336 076600
031340 000022

```

; *ERROR THEREFORE IS REPORTED AS THE POWER UP FAILING TO
; *INVALIDATE CACHE. IT SHOULD BE POINTED OUT THAT
; *THE SWEEP MECHANISM IS CHECKED IN THE PREVIOUS TEST. THIS
; *TEST VERIFIES THAT THE MECHANISM CAN BE INITIATED BY
; *THE POWER UP SEQUENCE.
; *
; *NOTE: IF MACHINE HAS VOLATILE MEMORY, THE SWITCH
; * SETTINGS WILL HAVE TO BE RESTORED AFTER THIS TEST

; *****
TST42: MOV      #214, @CCR      ; CACHE OFF FOR SCOPE
        SCOPE
        MOV      @TST43, SKTST ; SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        BIT      @SW07, @SWR   ; RUN THIS TEST?
        BEQ      TST43        ; BRANCH TO NEXT TEST IF NO
        MOV      @T20L01, @PVEC ; SET UP FOR PARITY ERRORS
        MOV      @T20L02, @PWRVEC ; SET UP FOR POWER DOWN
        MOV      @314, @CCR    ; SET ALL BITS IN CCR
        TSTB     @SENV        ; RUNNING UNDER APT?
        BEQ      IS          ; BRANCH IF NO
        MOV      @20, @CREG2  ; SET UP UBE TO POWER FAIL
        BR       .           ; WAIT FOR POWER FAIL

IS:     TYPE     ,MSG2        ; POWER DOWN AND THEN UP
        BR       .           ; WAIT FOR POWER DOWN

T20L02: MOV      @T20L03, @PWRVEC ; SET UP FOR POWER UP
        CMP      (SP)+, (SP)+ ; RESTORE STACK
        MOV      @SWR, $TMPD   ; SAVE (SWR)
        BR       .           ; WAIT FOR POWER UP

T20L03: MOV      @STACK, SP    ; RESTORE STACK
        TSTB     @SENV        ; RUNNING UNDER APT?
        BEQ      IS          ; BRANCH IF NO
        CLR      @CREG2       ; STOP UBE POWER FAIL

IS:     MOV      $TMPD, RO     ; GET (SWR)
        MED      .WORD      WSW ; RESTORE SWR
        MOV      #177000, R1   ; INIT DELAY
        MOV      @177400, RO   ; INIT DELAY COUNTER FOR TTY
        ADD      @BUFL, @BUFL ; DELAY
        INC      RO
        BNE     2$           ; WAIT FOR TTY
        INC      R1
        BNE     3$           ; CONTINUE DELAY
        MOV      @CCR, $REG1   ; SEE IF CCR INITIALIZED
        BEQ     T20L04        ; BRANCH IF CCR CLEARED
        ERROR    101         ; ERROR: CACHE CONTROL REG NOT INIT BY POWER FAIL

T20L04:
        MOV      RO, -(SP)    ; SAVE RO FOR MED INST
        MED      .WORD      RLOG ; GET CONTENTS OF LOG REG

```

```

4817 031342 052700 100001      BIS      #100001,RO      ;ENABLE ERROR LOG & LOG FIRST MODE
4818 031346 076600              MED              ;UNLOCK ERROR LOG
4819 031350 000222      .WORD    WLOG
4820 031352 012600      MOV      (SP)+,RO      ;RESTORE RO
4821
4822 031354 012737 000200 177746      MOV      #200,#CCR      ;JAMUPP ON PARITY ERRORS
4823 031362 012701 002000              MOV      #2000,R1      ;INIT LOOP COUNT
4824 031366 005000              CLR      RO            ;INIT ADDRESS
4825 031370 005720      1S:      TST      (RO)+      ;REFERENCE ALL CACHE LOC
4826 031372 077102              SOB      R1,1S        ;LOOP TILL DONE
4827 031374 012737 033142 000114 T20L06: MOV      #UPERR,#PVEC ;RESTORE PARITY ERROR HANDLER
4828 031402 012737 040364 000024      MOV      #SPWRDN,#PWRVEC ;RESTORE POWER FAIL HANDLER
4829 031410 000445              BR       TST43        ;GO TO NEXT TEST
4830
4831 031412 052737 000014 177746 T20L01: BIS      #14,#CCR      ;CACHE OFF TO STOP FURTHER PARITY ERRORS
4832
4833 031420 010046              MOV      RO,-(SP)     ;SAVE RO FOR MED INST
4834 031422 076600              MED              ;GET CONTENTS OF LOG REG
4835 031424 000022      .WORD    RLOG
4836 031426 052700 100001      BIS      #100001,RO   ;ENABLE ERROR LOG & LOG FIRST MODE
4837 031432 076600              MED              ;UNLOCK ERROR LOG
4838 031434 000222      .WORD    WLOG
4839 031436 012600      MOV      (SP)+,RO   ;RESTORE RO
4840
4841 031440 011637 001164              MOV      (SP),SREG3   ;GET PC+2 OF ERROR
4842 031444 162737 000002 001164      SUB      #2,SREG3    ;SAVE PC OF ERROR
4843 031452 022626      CMP      (SP)+,(SP)+ ;RESTORE STACK
4844 031454 076600              MED              ;GET LOG INFORMATION
4845 031456 000100      .WORD    RJAM
4846 031460 032700 000400      BIT      #400,RO     ;ERROR IN BACKING STORE?
4847 031464 001415      BEQ      T20L05     ;BRANCH IF NO
4848 031466 076600              MED              ;GET LOG INFOR FOR PHY. ADDR. A17,A16
4849 031470 000101      .WORD    RSER
4850 031472 000300      SWAB    RO          ;PUT PHY. ADDR A17, A16 IN LOW BYTE
4851 031474 042700 177776      BIC      #177776,RO  ;ONLY LOOK AT A17, A16
4852 031500 010037 001160      MOV      RO,SREG1    ;SAVE ADDRESS
4853 031504 076600              MED              ;GET LOG INFORMATION
4854 031506 000102      .WORD    LOADD
4855 031510 010037 001162      MOV      RO,SREG2    ;SAVE INFORMATION
4856 031514 104001      ERROR   1           ;UNEXPECTED PARITY ERROR IN BACKING STORE
4857 031516 000726      BR       T20L06     ;GO TO NEXT TEST
4858
4859 031520 104102 T20L05: ERROR   102   ;ERROR: POWER UP FAILED TO INVALIDATE CACHE
4860 031522 000724      BR       T20L06     ;GO TO NEXT TEST

```

;;*****

```

4861
4862
4863
4864
4865
4866
4867
4868
4869
4870
4871
4872
;THE FOLLOWING TESTS ARE RUN ONLY IF SW08=1 AT THE
;BEGINNING OF THE PROGRAM. AND THE NPR DEVICE WAS SELECTED.
;THESE TESTS USE DATA SUPPLIED BY THE USER WHEN THE PROGRAM
;IS STARTED TO SETUP VARIOUS CONTROL REGISTERS TO RUN THE
;NPR DEVICE. CREG1 ALWAYS CONTAINS THE DEVICES GO ADDRESS,
;EAD CONTAINS THE DEVICES ERROR REG ADDRESS, IVEC CONTAINS
;THE DEVICE'S INTERRUPT VECTOR (IF USED), SETUP CONTAINS THE

```

M08

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 88
T42 TEST POWER UP INVALIDATES CACHE AND CLEARS CACHE CONTROL REG

: ADDRESS OF THE DEVICE'S HANDLER AND THE REMAINING CREG2-5
: CONTAIN VARIOUS CONTROL INFORMATION NEEDED FOR THE PARTICULAR
: DEVICE USED. THE SETUP HANDLERS ARE USED TO INITIALIZE THE
: DEVICE TO DO NPR DATOS TO THE STARTING ADDRESS FOLLOWING
: THE SUBROUTINE CALL.

::*****

::*****
: TEST 43 TEST NPR DATO INVALIDATES CACHE FOR PHYSICAL ADDRESS BITS A1-A10

* THIS TEST IS ONLY RUN IF SW08=1. THE PREVIOUSLY SEL-
* ECTED DEVICE IS SETUP TO DO NPR DATO'S TO ADDRESSES IN
* A 1K BUFFER AREA. ON THE FIRST PASS A '1' IS FLOATED
* THROUGH THE ADDRESS LINES A1-A10. AT EACH BIT POSITION,
* THE LOC IS PUT IN CACHE AND THEN A NPR DATO IS DONE TO
* THAT LOC. A MINIMUM TIME IS THEN WAITED TO ALLOW THE
* SLOWEST DEVICE SELECTABLE TO FINISH ITS TRANSFERS. THE
* LOC IS THEN CHECKED TO BE A MISS.
* FOR THE SECOND PASS, A '0' IS FLOATED THROUGH ADDRESS
* BITS A1-A10 AND THE SAME PROCEDURE IS REPEATED. BEFORE
* THE DEVICE'S GO BIT IS SET, THE TRANSFER ADDRESS IS CHECKED
* TO SEE IF IT OVERLAPS THE INSTRUCTION ADDRESS IN CACHE.
* IF IT DOES, THE INSTRUCTIONS ARE EXECUTED OUT OF A NON
* OVERLAPPING ADDRESS SPACE.
* R0 CONTAINS THE DEVICES GO ADDRESS
* R1 CONTAINS THE PASS INDICATOR
* R2 IS THE DELAY COUNTER
* R3 CONTAINS THE TRANSFER ADDRESS
* R4 USED TO CALCULATE NEXT TRANSFER ADDRESS

::*****

```
TST43: MOV #214,2#CCR ;CACHE OFF FOR SCOPE
        SCOPE
        MOV #TST44,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
        BIT #SW08,2#SWR ;NPR DEVICE AVAILABLE?
        BNE 1$ ;BRANCH IF YES
        JMP 2#TST44 ;NO GO TO NEXT TEST
1$: MOV #200,2#CCR ;CACHE ON
    JSR PC,VEC ;SEE IF USE NEW USED AND SETUP INTERRUPT VECTOR
    MOV CREG1,R0 ;GET DEVICE'S GO ADDRESS
    CLR R1 ;CLEAR FLAG FOR PASS 1 (FLOAT 1 PATTERN)
    MOV #2,R4 ;INIT REG FOR ADDR, CALC.
    MOV #BUFL+2,2#ADD1L ;INIT ADDRESS LOWER FOR TEST
    CLR 2#ADD1H ;INIT ADDRESS HIGHER FOR TEST
T21L09: JSR PC,2#SETUP ;SETUP DEVICE TO DO NPR DATO TO FOLLOWING ADDRESS
ADD1L: .WORD 0 ;TEST ADDRESS LOWER 16 BITS
ADD1H: .WORD 0 ;TEST ADDRESS UPPER 2 BITS
        CLR R2 ;INIT R2 FOR TIME DELAY COUNT
```

: FIND OUT IF THE TEST INSTRUCTION ADDRESS IN CACHE
: OVERLAP THE XFER ADDRESS IN CACHE. IF THEY DO, USE THE
: TEST INSTRUCTIONS AT NON OVERLAPPING ADDRESS. THIS IS TO
: ENSURE THAT A MISS IS DUE TO A INVALIDATE RATHER THAN
: THE TEST INSTRUCTION SWAPPING OUT OF CACHE THE XFER LOCATION.

4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891
4892
4893
4894
4895
4896
4897
4898
4899
4900
4901
4902
4903
4904
4905
4906 031524 012737 000214 177746
4907 031532 000004
4908 031534 012737 032240 001234
4909 031542 032777 000400 147364
4910 031550 001002
4911 031552 000137 032240
4912 031556 012737 000200 177746 1\$:
4913 031564 004737 034000
4914 031570 013700 001212
4915 031574 005001
4916 031576 012704 000002
4917 031602 012737 060002 031620
4918 031610 005037 031622
4919 031614 004777 147412 T21L09:
4920 031620 000000 ADD1L: .WORD 0
4921 031622 000000 ADD1H: .WORD 0
4922 031624 005002 CLR R2
4923
4924
4925
4926
4927
4928

N08

MD-11-DOKKA-A 11/6X CACHE DIAGNOSTIC
DOKKAA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 89
T43 TEST NPR DATO INVALIDATES CACHE FOR PHYSICAL ADDRESS BITS A1-A10

4929										
4930	031626	013737	031620	001172		MOV	ADDIL,STMP0		;GET XFER ADDRESS	
4931	031634	042737	174000	001172		BIC	#174000,STMP0		;CALC ITS CACHE ADDRESS	
4932	031642	012737	031712	001174		MOV	#T21L01,STMP1		;GET TEST INST ADDRESS	
4933	031650	042737	174000	001174		BIC	#174000,STMP1		;CALC ITS CACHE ADDRESS	
4934	031656	023737	001172	001174		CMP	STMP0,STMP1		;DOES XFER ADDRESS OVERLAP TEST INST?	
4935	031664	002407				BLT	T21L02		;BRANCH IF NO	
4936	031666	062737	000022	001174		ADD	#22,STMP1		;GET ADDRESS OF LAST OVERLAPPING TEST INST.	
4937	031674	023737	001172	001174		CMP	STMP0,STMP1		;DOES XFER ADDRESS STILL OVERLAP TEST INST?	
4938	031702	003503				BLE	T21L03		;BRANCH IF YES TO TEST INST. AT DIFFERENT CACHE ADDRESS	
4939	031704	013703	031620		T21L02:	MOV	ADDIL,R3		;GET XFER ADDRESS	
4940	031710	021313				CMP	(R3),(R3)		;MAKE XFER ADDRESS A HIT	
4941	031712	033727	177752	000004	T21L01:	BIT	#HMR,#HMR2		;MAKE SURE ITS IN CACHE	
4942	031720	001514				BEQ	T21L04		;BRANCH IF NO TO ERROR	
4943	031722	005210				INC	(R0)		;SET DEVICES GO BIT TO START DATA XFERS	
4944	031724	005202			1S:	INC	R2		;DELAY TILL THE SLOWEST DEVICE	
4945	031726	001376				BNE	1S		;HAS FINISHED ITS XFERS	
4946	031730	005713				TST	(R3)		;SEE IF NPR DATO HAS INVALIDATED THE XFER ADDRESS IN CAC	
4947	031732	033727	177752	000004		BIT	#HMR,#HMR2		;LOC NOW A MISS? (CACHE INVALIDATED?)	
4948	031740	001117				BNE	T21L05		;GO REPORT ERROR IF LOC A HIT	
4949	031742	005777	147262		T21L11:	TST	#EAD		;SEE IF DEVICE HAD AN ERROR	
4950	031746	100514				BMI	T21L05		;REPORT DEVICE ERROR IF YES	
4951	031750	005701				TST	R1		;PASS 1?	
4952	031752	001024				BNE	T21L07		;BRANCH IF NO	
4953	031754	032704	002000			BIT	#2000,R4		;LAST FLOAT 1 PATTERN USED?	
4954	031760	001007				BNE	T21L08		;BRANCH IF YES	
4955	031762	006304				ASL	R4		;GENERATE NEXT FLOAT 1 PATTERN	
4956	031764	010437	031620			MOV	R4,ADDIL		;SAVE ITS LOWER BITS	
4957	031770	052737	060000	031620		BIS	#BUFL,ADDIL		;SET ITS HIGH BITS SO ITS IN TEST BUFFER	
4958	031776	000706				BR	T21L09		;GO TEST IT	
4959										
4960	032000	052701	000001		T21L08:	BIS	#1,R1		;SET FLAG FOR PASS 2 TO INDICATE FLOAT 0 PATTERN	
4961	032004	012704	001776			MOV	#1776,R4		;INIT REG FOR TEST ADDR. CALC.	
4962	032010	010437	031620			MOV	R4,ADDIL		;SAVE LOWER TEST ADDR.	
4963	032014	052737	060000	031620		BIS	#BUFL,ADDIL		;MAKE SURE ADDR. IN TEST AREA	
4964	032022	000674				BR	T21L09		;GO TEST IT	
4965										
4966	032024	022704	003776		T21L07:	CMP	#3776,R4		;AT LAST FLOAT 0 PATTERN?	
4967	032030	001413				BEQ	T21L10		;BRANCH IF YES TO END OF TEST	
4968	032032	006204				ASR	R4		;GENERATE NEW TEST ADDR.	
4969	032034	052704	002000			BIS	#2000,R4		;MAKE IT A FLOAT 0 PATTERN	
4970	032040	042704	000001			BIC	#1,R4		;MAKE IT A WORD ADDR.	
4971	032044	010437	031620			MOV	R4,ADDIL		;SAVE LOWER TEST ADDR.	
4972	032050	052737	060000	031620		BIS	#BUFL,ADDIL		;MAKE SURE ADDRESS IS IN TEST BUFFER	
4973	032056	000656				BR	T21L09		;GO TEST IT	
4974										
4975	032060	022737	034046	001232	T21L10:	CMP	#HUBEN,SETUP		;NEW UNIBUS EXERCISOR USED?	
4976	032066	001064				BNE	TST44		;BRANCH TO NEXT TEST IF NO	
4977	032070	013737	001226	001172		MOV	IVFC,STMP0		;GET IBE INTERRUPT VECTOR	
4978	032076	062737	000002	001172		ADD	#2,STMP0		;AND RESTORE	
4979	032104	005077	147062			CLR	#STMP0		;THE TRAP CATCHER	
4980	032110	000453				BR	TST44		;GO TO NEXT TEST	
4981										
4982										
4983										
4984	032112	013703	031620		T21L03:	MOV	ADDIL,R3		;GET XFER ADDRESS	

;TEST INST TO TEST XFER ADDRESSES WHICH OVERLAP TEST CODE

```

4985 032116 021313          CMP      (R3), (R3)      ;MAKE ADDRESS A HIT
4986 032120 033727 177752 000004 BIT      @#HMR, #HMR2    ;MAKE SURE ITS IN CACHE
4987 032126 001411          BEQ      T21L04         ;BRANCH IF NO TO ERROR
4988 032130 005210          INC      (R0)          ;SET DEVICES GO BIT TO START DATA XFER
4989 032132 005202          IS:    INC      R2      ;DELAY TILL THE SLOWEST DEVICE
4990 032134 001376          BNE     1$            ;HAS FINISHED ITS XFERS.
4991 032136 005713          TST     (R3)          ;SEE IF NPR DATO HAS INVALIDATED XFER ADDR. IN CACHE
4992 032140 033727 177752 000004 BIT      @#HMR, #HMR2    ;IS LOC NOW A MISS? (CACHE INVALIDATED?)
4993 032146 001014          BNE     T21L05        ;GO REPORT ERROR IF LOC A HIT
4994 032150 000674          BR      T21L11        ;CHECK FOR DEVICE ERROR
4995
4996 032152 012737 031614 001110 T21L04: MOV     @T21L09, @#SLPERR ;SET UP RETURN FOR ERROR LOOP
4997 032160 013737 031622 001160      MOV     ADD1H, $REG1    ;SAVE 'BAD' ADDRESS
4998 032166 013737 031620 001162      MOV     ADD1L, $REG2   ;SAVE 'BAD' ADDRESS
4999 032174 104043          ERROR  43             ;ERROR: ADDRESS COULD NOT BE MADE A HIT
5000 032176 000730          BR      T21L10        ;GO TO END OF TEST
5001
5002 032200 012737 031614 001110 T21L05: MOV     @T21L09, @#SLPERR ;SET UP RETURN FOR ERROR LOOP
5003 032206 013737 031622 001160      MOV     ADD1H, $REG1    ;SAVE BAD ADDRESS
5004 032214 013737 031620 001162      MOV     ADD1L, $REG2   ;SAVE BAD ADDRESS
5005 032222 005777 147002          TST     @EAD           ;DID DEVICE HAVE ERROR?
5006 032226 100002          BPL     1$            ;BRANCH IF NO
5007 032230 104103          ERROR  103           ;ERROR: DEVICE ERROR BIT SET WHEN DOING DATO TO ADDRESS
5008 032232 000712          BR      T21L10        ;GO TO END OF TEST
5009 032234 104104          IS:    ERROR  104     ;ERROR: CACHE LOC NOT INVALIDATED BY NPR DATO TO ADDR.
5010 032236 000710          BR      T21L10        ;GO TO END OF TEST.
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037
5038
5039 032240 012737 000214 177746 †ST44: MOV     @214, @#CCR    ;CACHE OFF FOR SCOPE
5040 032246 000004          SCOPE

```

```

*****
;TEST 44      TEST NPR DATO INVALIDATES CACHE FOR PHYSICAL ADDRESS BITS A17-A11
;
; THIS TEST IS RUN ONLY IF SW08=1 AND THE INHIBIT TESTS
; USING KT (SW12)=0. THE PREVIOUSLY SELECTED DEVICE IS
; SETUP TO DO NPR DATOS TO LOCATIONS LYING OUTSIDE THE
; PROGRAM AND MONITOR ADDRESS SPACE. (THE MONITOR, IF IT
; EXISTS, LIES IN THE LAST 1.5K OF MEMORY).
; MEMORY IS FIRST SIZED TO DETERMINE THE MAXIMUM TESTABLE
; ADDRESS. A VIRTUAL ADDRESS IS GENERATED AND STORED IN KIPAR4.
; THEN ITS PHYSICAL ADDRESS IS CALCULATED FOR THE DEVICES
; NPR TRANSFER. THE ADDRESS IS MADE A HIT IN CACHE AND THEN
; AN NPR DATO IS DONE TO IT. A MINIMUM TIME IS THEN WAITED
; TO ALLOW THE SLOWEST SELECTABLE DEVICE TO FINISH ITS
; TRANSFERS. THE LOCATION IS THEN CHECKED TO BE A MISS
; (INVALIDATED). A NEW TAG VALUE IS THEN GENERATED AND
; THE PROCEDURE REPEATS TO THE MAXIMUM ALLOWABLE ADDRESS.
; BEFORE THE DEVICE'S GO BIT IS SET, THE ADDRESS IS
; CHECKED TO SEE IF IT OVERLAPS THE INSTRUCTION ADDRESSES
; IN CACHE. IF IT DOES, THE INSTRUCTIONS ARE EXECUTED
; OUT OF NON OVERLAPPING ADDRESSES.
;
; R0 CONTAINS THE DEVICE'S GO ADDRESS
; R2 IS THE DELAY COUNTER
; R3 IS USED TO GENERATE THE TEST ADDRESS

```

```

*****
†ST44: MOV     @214, @#CCR    ;CACHE OFF FOR SCOPE
SCOPE

```

```

5041 032250 012737 033020 001234      MOV      #SEOP,SKTST
5042 032256 032777 000400 146650      BIT      #SM08,SWR      ;NPR DEVICE AVAILABLE?
5043 032264 001002                BNE     1$              ;BRANCH IF YES
5044 032266 000137 033020                JMP     @#SEOP
5045 032272 032777 010000 146634 1$:      BIT      #SW12,SWR      ;INHIBIT TESTS USING KT?
5046 032300 001402                BEQ     2$              ;BRANCH IF NO
5047 032302 000137 033020                JMP     @#SEOP
5048 032306 012737 000200 177746 2$:      MOV     #200,@#CCR      ;CACHE ON
5049 032314 052737 000200 036034      BIS     #200,@#SKT11   ;USE KT FOR $SIZE
5050 032322 004737 035750                JSR     PC,$$SIZE      ;SIZE MEM
5051 032326 162737 000040 036322      SUB     #40,$LSTBK     ;REDUCE TESTABLE MEM BY 1K SO DON'T KILL MONITOR IF EXIS
5052 032334 013700 001212                MOV     CREG1,RO       ;GET THE GO ADDRESS OF THE DEVICE
5053 032340 005237 177572                INC     @#HMR          ;TURN KT ON
5054
5055                ;CALC TEST ADDR
5056
5057 032344 012703 000020                MOV     #20,R3         ;INIT ADDR REG
5058 032350 006303                T22L08: ASL     R3             ;CALC NEXT ADDR
5059 032352 010337 172350                MOV     R3,@#KIPAR4   ;SETUP PAR WITH TEST ADDR
5060 032356 023727 172350 001000      CMP     @#KIPAR4,#1000 ;PAST INSTRUCTION SPACE?
5061 032364 002003                BGE     1$             ;BRANCH IF YES
5062 032366 052737 000600 172350      BIS     #600,@#KIPAR4 ;MAKE SURE TEST ADDR. LIES OUTSIDE OF TEST CODE
5063 032374 023737 172350 036322 1$:      CMP     @#KIPAR4,$LSTBK ;IS TEST ADDRESS IN MONITOR ADDRESS SPACE?
5064 032402 003164                BGT     T22L02        ;BRANCH IF YES TO END OF TEST
5065 032404 004737 034000                JSR     PC,VEC         ;SET UP UBE NEW INT. VECT IF IT IS USED
5066
5067                ;CALC THE PHYSICAL ADDRESS FROM THE VIRTUAL TEST ADDRESS
5068
5069 032410 012737 101776 001172      MOV     #101776,$TMP0  ;GET VIRTUAL ADDRESS
5070 032416 004737 033434                JSR     PC,VIP         ;CALC ITS PHYSICAL ADDRESS
5071 032422 013737 001160 032444      MOV     $REG1,ADD2H    ;SAVE PHYSICAL TEST ADDRESS
5072 032430 013737 001162 032442      MOV     $REG2,ADD2L    ;SAVE PHYSICAL TEST ADDRESS
5073 032436 004777 146570                T22L11: JSR     PC,@SETUP     ;SETUP NPR DEVICE TO DO DATO TO FOLLOWING ADDRESS
5074 032442 000000                ADD2L: .WORD 0         ;TEST ADDRESS LOWER 16 BITS
5075 032444 000000                ADD2H: .WORD 0         ;TEST ADDRESS UPPER 2 BITS
5076 032446 005002                CLR     R2             ;INIT REG FOR TIME DELAY
5077
5078                ;FIND OUT IF THE TEST INSTRUCTION ADDRESS IN CACHE
5079                ;OVERLAP THE XFER ADDRESS IN CACHE. IF THEY DO, USE
5080                ;TEST INSTRUCTIONS AT NON OVERLAPPING ADDRESS. THIS IS TO
5081                ;ENSURE THAT A MISS IS DUE TO A INVALIDATE RATHER THAN
5082                ;THE TEST INSTRUCTION SWAPPING OUT OF CACHE THE XFER LOCATION.
5083
5084 032450 013737 032442 001172      MOV     ADD2L,$TMP0    ;GET XFER ADDRESS
5085 032456 042737 174000 001172      BIC     #174000,$TMP0  ;CALC ITS CACHE ADDRESS
5086 032464 012737 032534 001174      MOV     #T22L01,$TMP1 ;GET TEST INST ADDRESS
5087 032472 042737 174000 001174      BIC     #174000,$TMP1  ;CALC ITS CACHE ADDRESS
5088 032500 023737 001172 001174      CMP     $TMP0,$TMP1    ;DOES XFER ADDRESS OVERLAP TEST INST?
5089 032506 002407                BLT     T22L03        ;BRANCH IF NO
5090 032510 062737 000024 001174      ADD     #24,$TMP1     ;CALC ADDR OF LAST OVERLAPPING TEST INST.
5091 032516 023737 001172 001174      CMP     $TMP0,$TMP1    ;DOES XFER ADDR STILL OVERLAP TEST INST?
5092 032524 003440                BLE     T22L04        ;BRANCH IF YES
5093
5094 032526 023737 101776 101776 T22L03: CMP     @#101776,@#101776 ;MAKE ADDR A HIT
5095 032534 033727 177752 000004 T22L01: BIT     @#HMR,#HMR2    ;MAKE SURE ITS IN CACHE
5096 032542 001452                BEQ     T22L05        ;BRANCH IF NO TO ERROR

```

```

5097 032544 005210          INC      (R0)          ;SET DEVICE'S GO BIT TO DO DATA XFERS
5098 032546 005202          IS:      INC      R2          ;DELAY TILL THE SLOWEST DEVICE
5099 032550 001376          BNE     1$          ;HAS FINISHED ITS XFERS
5100 032552 005737 101776      TST     @#101776     ;SEE IF NPR DATO HAS INVALIDATED THE XFER ADDR IN CACHE
5101 032556 033727 177752 000004 BIT     @#HMR, #HMR2 ;LOC NOW A MISS? (CACHE INVALIDATED?)
5102 032564 001054          BNE     T22L06     ;GO REPORT ERROR IF LOC A HIT
5103 032566 005777 146436      T22L10: TST     @EAD        ;SEE IF DEVICE HAS AN ERROR
5104 032572 100451          BMI     T22L06     ;REPORT DEVICE ERROR IF YES
5105 032574 023727 172350 004000 CMP     @#KIPAR4, #4000 ;TESTED LAST ADDRESS?
5106 032602 001464          BEQ     T22L02     ;BRANCH TO END OF TEST IF YES
5107 032604 022737 034174 001232 CMP     #HUBEO, SETUP ;WAS THE OLD UBE USED?
5108 032612 001256          BNE     T22L08     ;NO, GO CALC NEXT TEST ADDR
5109 032614 023727 172350 001000 CMP     @#KIPAR4, #1000 ;AT LAST TESTABLE ADDRESS FOR OLD UBE?
5110 032622 002652          BLT     T22L08     ;BRANCH IF NO
5111 032624 000453          BR      T22L02     ;GO TO END OF TEST
5112
5113 032626 023737 101776 101776 T22L04: CMP     @#101776, @#101776 ;MAKE XFER ADDRESS A HIT
5114 032634 033727 177752 000004 BIT     @#HMR, #HMR2 ;MAKE SURE ITS IN CACHE
5115 032642 001412          BEQ     T22L05     ;BRANCH TO ERROR IF NO
5116 032644 005210          INC     (R0)       ;SET DEVICES TO BIT TO DO XFERS.
5117 032646 005202          IS:      INC     R2          ;DELAY TILL THE SLOWEST DEVICE
5118 032650 001376          BNE     1$          ;HAS FINISHED
5119 032652 005737 101776      TST     @#101776     ;SEE IF NPR DATO HAS INVALID THE XFER ADDR. IN CACHE
5120 032656 033727 177752 000004 BIT     @#HMR, #HMR2 ;LOC NOW A MISS? (CACHE INVALIDATED?)
5121 032664 001014          BNE     T22L06     ;GO REPORT ERROR IF LOC A HIT
5122 032666 000737          BR      T22L10     ;GO SEE IF DEVICE HAD AN ERROR
5123
5124 032670 012737 032436 001110 T22L05: MOV     @T22L11, @#SLPERR ;INIT RETURN FOR ERROR LOOP
5125 032676 013737 032444 001160 MOV     ADD2H, $REG1 ;SAVE BAD ADDRESS
5126 032704 013737 032442 001162 MOV     ADD2L, $REG2 ;SAVE BAD ADDRESS
5127 032712 104043          ERROR  43          ;ERROR: ADDRESS COULD NOT BE MADE A HIT
5128 032714 000417          BR      T22L02     ;GO TO END OF TEST
5129
5130 032716 012737 032436 001110 T22L06: MOV     @T22L11, @#SLPERR ;SETUP RETURN FOR ERROR LOOPS
5131 032724 013737 032444 001160 MOV     ADD2H, $REG1 ;SAVE BAD ADDRESS
5132 032732 013737 032442 001162 MOV     ADD2L, $REG2 ;SAVE BAD ADDRESS
5133 032740 005777 146264      TST     @EAD        ;DID DEVICE HAVE AN ERROR?
5134 032744 100002          BPL     1$          ;BRANCH IF NO
5135 032746 104103          ERROR  103         ;ERROR: DEVICE ERROR BIT SET WHEN DOING DATO TO ADDR.
5136 032750 000401          BR      T22L02     ;GO TO END OF TEST
5137
5138 032752 104104          IS:      ERROR  104         ;ERROR: CACHE LOC NOT INVALID BY NPR DATO TO ADDR.
5139 032754 042737 000001 177572 T22L02: BIC     @1, @MMRO    ;KT OFF
5140 032762 023727 001232 034046 CMP     SETUP, #HUBEN ;WAS THE NEW UBE USED?
5141 032770 001013          BNE     SEOP        ;IF NO, GO TO END-OF-PASS
5142 032772 013737 001226 001172 MOV     IVEC, $TMP0 ;GET UBE INTERRUPT VECTOR
5143 033000 062737 000002 001172 ADD     @2, $TMP0   ;AND RESTORE
5144 033006 013777 001172 146212 MOV     $TMP0, @IVEC
5145 033014 005077 146152          CLR     @$TMP0     ;THE TRAP CATCHER
5146
5147
5148
5149
5150          .SBTTL  END OF PASS ROUTINE
5151          ;*****
5152          ;*INCREMENT THE PASS NUMBER ($PASS)

```

```

5153
5154
5155
5156
5157 033020
5158 033020 000004
5159 033022 005037 001102
5160 033026 005037 035406
5161 033032 005237 001244
5162 033036 042737 100000 001244
5163 033044 005327
5164 033046 000001
5165 033050 003022
5166 033052 012737
5167 033054 000001
5168 033056 033046
5169 033060 104401 033125
5170 033064 013746 001244
5171 033070 104405
5172 033072 104401 033122
5173 033076 013700 000042
5174 033102 001405
5175 033104 000005
5176 033106 004710
5177 033110 000240
5178 033112 000240
5179 033114 000240
5180 033116
5181 033116 000137
5182 033120 003056
5183 033122 377 377 000
5184 033125 015 042412 042116
5185 033132 050040 051501 020123
5186 033140 000043
5187
5188
5189
5190
5191
5192 033142 012737 000214 177746
5193 033150 011637 001166
5194 033154 162737 000002 001166
5195 033162 022626
5196 033164 076600
5197 033166 000101
5198 033170 000300
5199 033172 042700 177776
5200 033176 010037 001160
5201 033202 076600
5202 033204 000102
5203 033206 010037 001162
5204 033212 076600
5205 033214 000100
5206 033216 032700 000400
5207 033222 001016
5208 033224 032737 000040 177744

```

```

;#TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
;#IF THERES A MONITOR GO TO IT
;#IF THERE ISN'T JUMP TO START1

```

```

SEOP:
SCOPE
CLR $STSTM ;; ZERO THE TEST NUMBER
CLR $STIMS ;; ZERO THE NUMBER OF ITERATIONS
INC $PASS ;; INCREMENT THE PASS NUMBER
BIC #100000,$PASS ;; DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;; LOOP?
SEOPCT: .WORD 1
BGT $DOAGN ;; YES
MOV (PC)+,2(PC)+ ;; RESTORE COUNTER
SENDCT: .WORD 1
TYPE $SENDMG ;; TYPE "END PASS #"
MOV $PASS,-(SP) ;; SAVE $PASS FOR TYPEOUT
TYPDS ;; GO TYPE--DECIMAL ASCII WITH SIGN
TYPE $SENULL ;; TYPE A NULL CHARACTER
SGET42: MOV #42,RO ;; GET MONITOR ADDRESS
BEQ $DOAGN ;; BRANCH IF NO MONITOR
RESET ;; CLEAR THE WORLD
SENDAD: JSR PC,(RO) ;; GO TO MONITOR
NOP ;; SAVE ROOM
NOP ;; FOR
NOP ;; ACT11
$DOAGN: JMP 2(PC)+ ;; RETURN
$RTNAD: .WORD START1
$SENULL: .BYTE -1,-1,0 ;; NULL CHARACTER STRING
$SENDMG: .ASCIZ <15><12>/END PASS #/

```

```

;////////////////////////////////////
;SUBROUTINE TO REPORT AN UNEXPECTED PARITY ERRORS
;////////////////////////////////////

```

```

UPERR: MOV #214,2$CCR ;; TURN OFF CACHE TO PREVENT OTHER ERRORS
MOV (SP),$REG4 ;; SAVE PC+2 WHERE PARITY ERROR OCCURRED
SUB #2,$REG4 ;; CALC. PC WHERE PARITY ERROR OCCURRED
CMP (SP)+,(SP)+ ;; RESTORE STACK
MED ;; GET LOG INFOR FOR PHY. ADDR. A17,A16
.WORD RSER
SWAB RO ;; PUT PHY. ADDR A17, A16 IN LOW BYTE
BIC #177776,RO ;; ONLY LOOK AT A17, A16
MOV RO,$REG1 ;; SAVE ADDRESS
MED ;; GET LOG INFORMATION
.WORD LOADD
MOV RO,$REG2 ;; SAVE INFORMATION
MED ;; GET LOG INFORMATION
.WORD RJAM
BIT #400,RO ;; WAS ERROR IN BACKING STORE?
BNE UPI ;; BRANCH IF YES
BIT #40,2$EREG ;; WAS ERROR IN CACHE TAG FIELD?

```

```

5209 033232 001017          BNE      UP2          ;BRANCH IF YES
5210 033234 032737 000100 177744 BIT      @100,@#EREG ;WAS ERROR IN CACHE LOW BYTE?
5211 033242 001024          BNE      UP3          ;BRANCH IF YES
5212 033244 076600          MED          ;GET LOG INFORMATION
5213 033246 000106          .WORD     CDH
5214 033250 010037 001164 MOV      RO,$REG3    ;SAVE INFORMATION
5215 033254 104004          ERROR     4          ;ERROR: UNEXPECTED PARITY ERROR IN CACHE DATA HIGH
5216 033256 000423          BR        UPR          ;RETURN
5217
5218 033260 013737 001166 001164 UP1:  MOV     $REG4,$REG3 ;SAVE PC OF TRAP
5219 033266 104001          ERROR     1          ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
5220 033270 000416          BR        UPR          ;RETURN
5221
5222          UP2:
5223 033272 076600          MED          ;GET TAG LOG INFO.
5224 033274 000107          .WORD     RTAG
5225 033276 000300          SWAB      RO          ;PUT TAG IN LOW BYTE
5226 033300 042700 177400 BIC      #177400,RO  ;LOOK AT TAG ONLY
5227 033304 010037 001164 MOV      RO,$REG3    ;SAVE CACHE TAG DATA
5228 033310 104002          ERROR     2          ;ERROR: UNEXPECTED PARITY ERROR IN CACHE TAG
5229 033312 000405          BR        UPR          ;RETURN
5230
5231          UP3:
5232 033314 076600          MED          ;GET LOG INFORMATION
5233 033316 000106          .WORD     CDL
5234 033320 010037 001164 MOV      RO,$REG3    ;SAVE INFORMATION
5235 033324 104003          ERROR     3          ;ERROR: UNEXPECTED PARITY ERROR IN CACHE DATA LOW
5236
5237          UPR:
5238 033326
5239 033326 010046          MOV      RO,-(SP)   ;SAVE RO FOR MED INST
5240 033330 076600          MED          ;GET CONTENTS OF LOG REG
5241 033332 000022          .WORD     RLOG
5242 033334 052700 100001 BIS      #100001,RO  ;ENABLE ERROR LOG & LOG FIRST MODE
5243 033340 076600          MED          ;UNLOCK ERROR LOG
5244 033342 000222          .WORD     WLOG
5245 033344 012600          MOV      (SP)+,RO   ;RESTORE RO
5246
5247 033346 000177 145662 JMP      @SKTST      ;START SUBTEST FOLLOWING ONE WHERE ERROR OCCURRED
5248
5249          ;////////////////////
5250          ;ROUTINE TO REPORT UNEXPECTED TRAPS TO VECTOR 4
5251          ;////////////////////
5251 033352 012737 000214 177746 UT4:  MOV      @214,@#CCR  ;TURN OFF CACHE
5252 033360 011637 001162          MOV      (SP),$REG2 ;SAVE FAILING PC
5253 033364 013737 177766 001160 MOV      @#CER,$REG1 ;GET CPU ERROR REG (CER)
5254 033372 032777 001000 145534 BIT      @SW09,@SWR  ;LOOP ON ERROR?
5255 033400 001401          BEQ      1$          ;BRANCH IF NO
5256 033402 022626          CMP      (SP)+,(SP)+ ;RESTORE STACK
5257 033404 104016          1$:  ERROR     16        ;ERROR: UNEXPECTED TRAP TO VECTOR 4
5258
5259 033406 010046          MOV      RO,-(SP)   ;SAVE RO FOR MED INST
5260 033410 076600          MED          ;GET CONTENTS OF LOG REG
5261 033412 000022          .WORD     RLOG
5262 033414 052700 100001 BIS      #100001,RO  ;ENABLE ERROR LOG & LOG FIRST MODE
5263 033420 076600          MED          ;UNLOCK ERROR LOG
5264 033422 000222          .WORD     WLOG

```

```

5265 033424 012600          MOV      (SP)+,R0          ;RESTORE R0
5266
5267 033426 000000          HALT
5268 033430 000137 000200      JMP      #200              ;RESTART TEST IF CONTINUE
5269
5270 ;////////////////////
5271 ;SUBROUTINE TO CONVERT VIRTUAL ADDRESS IN $TMPD TO A PHYSICAL ADDRESS IN $REG2, $REG1
5272 ;////////////////////
5273 033434 010146          VIP:  MOV      R1,-(SP)          ;SAVE R1 ON STACK
5274 033436 010246          MOV      R2,-(SP)          ;SAVE R2 ON STACK
5275 033440 013701 001172      MOV      $TMPD,R1          ;GET VIRTUAL ADDRESS
5276 033444 005002          CLR      R2                ;INT SHIFT COUNTER
5277 033446 006201          1$:  ASR      R1                ;SHIFT BLOCK # TO LSB 0-6
5278 033450 005202          INC      R2                ;COUNT SHIFTS
5279 033452 020227 000006      CMP      R2,#6            ;ALL DONE?
5280 033456 001373          BNE      1$                ;BRANCH IF NO
5281 033460 010137 001162      MOV      R1,$REG2          ;SAVE BLOCK #
5282 033464 042737 177600 001162  BIC      #177600,$REG2     ;MASK BLOCK #
5283 033472 006201          2$:  ASR      R1                ;SHIFT ACTIVE PAGE FIELD TO LSB 1-3
5284 033474 005202          INC      R2                ;COUNT SHIFTS
5285 033476 020227 000014      CMP      R2,#14           ;ALL DONE?
5286 033502 001373          BNE      2$                ;BRANCH IF NO
5287 033504 042701 177761          BIC      #177761,R1        ;CALC. APFX2
5288 033510 062701 172340      ADD      #KIPAR0,R1        ;CALC ADDRESS OF PAR REFERENCING
5289 033514 011101          MOV      (R1),R1           ;GET (PAR)
5290 033516 060137 001162      ADD      R1,$REG2          ;CALC. PHYSICAL BLOCK #
5291 033522 013737 001162 001160  MOV      $REG2,$REG1       ;SAVE PHYSICAL ADDRESS BITS 17,16
5292 033530 005002          CLR      R2                ;INI. SHIFT COUNT
5293 033532 006237 001160          3$:  ASR      $REG1            ;SHIFT ADDRESS BITS 17,16 TO LSB 1,0
5294 033536 005202          INC      R2                ;COUNT SHIFTS
5295 033540 020227 000012      CMP      R2,#12           ;DONE?
5296 033544 001372          BNE      3$                ;BRANCH IF NO
5297 033546 005002          CLR      R2                ;INIT. SHIFT COUNT
5298 033550 006337 001162          4$:  ASL      $REG2            ;SHIFT MSB OF ADDRESS TO BIT 16
5299 033554 005202          INC      R2                ;COUNT SHIFTS
5300 033556 020227 000006      CMP      R2,#6            ;ALL DONE?
5301 033562 001372          BNE      4$                ;BRANCH IF NO
5302 033564 013701 001172      MOV      $TMPD,R1          ;GET VIRTUAL ADDRESS
5303 033570 042701 177700          BIC      #177700,R1        ;MASK OFF BLOCK COUNT
5304 033574 060137 001162      ADD      R1,$REG2          ;HAVE $REG2 CONTAIN PHYSICAL ADDRESS 0-15
5305 033600 012602          MOV      (SP)+,R2          ;RESTORE R2
5306 033602 012601          MOV      (SP)+,R1          ;RESTORE R1
5307 033604 000207          RTS      PC                ;RETURN
5308
5309 ;////////////////////
5310 ;SUBROUTINE TO CALC. TAG FIELD FROM A PAR IN LOC $TMPD
5311 ;////////////////////
5312
5313 033606 010146          TAG:  MOV      R1,-(SP)          ;SAVE R1 ON STACK
5314 033610 012701 000005          MOV      #5,R1            ;INIT R1 TO COUNT 5 SHIFTS
5315 033614 006237 001172          1$:  ASR      $TMPD            ;CALC TAG CONTENTS
5316 033620 077103          SOB      R1,1$            ;ALL DONE?
5317 033622 052737 000200 001172  BIS      #200,$TMPD        ;SET VALID BIT
5318 033630 012601          MOV      (SP)+,R1          ;RESTORE R1
5319 033632 000207          RTS      PC                ;RETURN
5320

```

```

5321 ;////////////////////////////////////
5322 ; SUBROUTINE TO FIND PAR FROM A VIRTUAL ADDRESS IN R0 OR R1 AND
5323 ; PUT ITS CONTENTS IN $TMP0
5324 ;////////////////////////////////////
5325 PAR: MOV R0,$TMP0 ;GET VIRTUAL ADDRESS
5326 TST @($SP) ;WAS R0 USED?
5327 BEQ $S ;BRANCH IF YES
5328 MOV R1,$TMP0 ;GET VIRTUAL ADDRESS
5329 $S: ADD @2,$($SP) ;ADJUST PC
5330 MOV @14,$R5 ;INIT COUNT
5331 $S: ASR $TMP0 ;SHIFT ADDRESS TO GET ACTIVE PAGE FIELD
5332 SOB $R5,$S ;APF IN LSB 1-3? BRANCH IF NO
5333 BIC @177761,$TMP0 ;MASK APF X 2
5334 ADD @KIPAR,$TMP0 ;PUT PAR ADDRESS IN $TMP0
5335 MOV @($TMP0),$TMP0 ;GET CONTENTS OF PAR
5336 RTS PC ;RETURN
5337 ;////////////////////////////////////
5338 ; SUBROUTINE TO GENERATE A TEST BUFFER ADDRESS 512(10) LOCATIONS FROM GIVEN
5339 ; ADDRESS FOLLOWING ITS CALL
5340 ;////////////////////////////////////
5341 $HAD: MOV @($SP),$TMP0 ;GET ADDRESS TO BE USED
5342 ADD @2,$($SP) ;ADJUST PC
5343 ADD @2000,$TMP0 ;CALC. ADDR WITH ADDRESS BIT A10 COMPLEMENTED
5344 BIC @174000,$TMP0 ;MASK A15-A11
5345 BIT @2000,$TMP0 ;BIT 10 SET?
5346 BNE $S ;BRANCH IF YES
5347 $S: ADD @BUFL,$TMP0 ;CALC TEST BUFFER ADDR.
5348 BR $S
5349
5350 $S: BIC @2000,$TMP0 ;ADJUST ADDRESS BIT A10
5351 ADD @BUFH,$TMP0 ;CALC TEST BUFFER ADR.
5352 $S: RTS PC ;RETURN
5353 ;////////////////////////////////////
5354 ; SUBROUTINE TO SEE IF A NEW UNIBUS EXER. IS USED AND TO SETUP AN
5355 ; RTI IN ITS INTERRUPT VECTOR
5356 ;////////////////////////////////////
5357
5358 $VEC: CMP SETUP,$HUBEN ;NEW UBE USED?
5359 BNE $S ;BRANCH IF NO
5360 MOV IVEC,$TMP0 ;GET ITS INTERRUPT VECTOR
5361 ADD @2,$TMP0
5362 MOV $TMP0,@IVEC ;PUT ON RTI
5363 MOV @RTI,$TMP0 ;IN ITS INTERRUPT AREA
5364 $S: CLR @PSW ;LOWER PRIORITY LEVEL FOR INTERRUPTS
5365 RTS PC ;RETURN
5366
5367 ;////////////////////////////////////
5368 ; SUBROUTINE TO SETUP THE NEW UNIBUS EXERCISOR TO DO ONE NPR DATO
5369 ; TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL
5370 ;////////////////////////////////////
5371
5372 $HUBEN: CLR $TMP5 ;INIT COUNTER TO WAIT FOR RDY BIT
5373 $S: TSTB @CREG1 ;READY BIT SET?
5374 BMI $S ;BRANCH IF YES
5375 INC $TMP5 ;WAIT FOR RDY TO SET
5376 BNE $S ;BRANCH IF HAVEN'T WAITED MAX TIME

```



```

5377 034066 032777 020000 145040      BIT      #SW13, @SWR      ;INHIBIT TYPEOUTS?
5378 034074 001004                      BNE      3$           ;BRANCH IF YES
5379 034076 104401 041645                      TYPE     ,MSG13       ;DEVICE RDY BIT DOES NOT SET
5380 034102 104401 041705                      TYPE     ,MSG14       ;FURTHER NPR DEVICE TESTS ABORTED
5381 034106 005726                      TST      (SP)+        ;RESTORE STACK FROM SUBROUTINE CALL
5382 034110 042737 000001 177572 3$:    BIC      #1, @MMRD    ;KT OFF IF ON
5383 034116 000137 033020                      JMP      SEOP         ;GO TO END OF PROGRAM
5384 034122 005077 145070 1$:    CLR      @CREG3      ;CLEAR ANY ERROR BITS SET
5385 034126 012777 003040 145056      MOV      #3040, @CREG1 ;HAVE UBE DO INPR DATO DATA XFER
5386 034134 005077 145054                      CLR      @CREG2      ;HAVE UBE DO INPR DATO DATA XFER
5387 034140 012777 177777 145054      MOV      #177777, @CREG5 ;CYCLE COUNT=1 XFER
5388 034146 017677 000000 145044      MOV      @ (SP), @CREG4 ;GET ADDRESS FOR XFER
5389 034154 062716 000002                      ADD      #2, (SP)     ;GET HIGH ADDRESS BITS A17, A16
5390 034160 057677 000000 145026      BIS      @ (SP), @CREG2 ;PUT ADDRESS BITS IN CONTROL REG
5391 034166 062716 000002                      ADD      #2, (SP)     ;ADJUST PC FOR RETURN
5392 034172 000207                      RTS      PC           ;RETURN
5393
5394 ;////////////////////////////////////
5395 ;SUBROUTINE TO SETUP THE OLD UNIBUS EXERCISOR TO DO 1 NPR DATO
5396 ;TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL.
5397 ;////////////////////////////////////
5398
5399 034174 012737 050200 170006 HUBEO: MOV      #50200, @#170006 ;HAVE UBE DO 1 NPR DATO AND RELEASE BUS
5400 034202 012737 000002 170004      MOV      #2, @#170004 ;SET BYTE COUNT FOR 1 WORD XFER
5401 034210 017637 000000 170002      MOV      @ (SP), @#170002 ;SET UP XFER ADDRESS
5402 034216 062716 000004                      ADD      #4, (SP)     ;ADJUST PC FOR RETURN
5403 034222 000207                      RTS      PC           ;RETURN
5404 034224 000000      FAKE: .WORD 0 ;FAKE ERROR REG. MSB=0 FOR NO ERRORS
5405
5406 ;////////////////////////////////////
5407 ;SUBROUTINE TO SETUP AN RKDS FOR 1 NPR DATO
5408 ;TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL
5409 ;////////////////////////////////////
5410
5411 034226 005037 001204      HRKDS: CLR      $TMP5      ;INIT COUNTER TO WAIT FOR RDY BIT
5412 034232 105737 177404 2$:    TSTB    @RKCS        ;IS CONTROLLER RDY?
5413 034236 100421                      BMI      1$           ;BRANCH IF YES
5414 034240 005237 001204                      INC      $TMP5       ;WAIT FOR RDY TO SET
5415 034244 001372                      BNE      2$           ;BRANCH IF HAVEN'T WAITED MAX TIME
5416 034246 042737 000001 177572 5$:    BIC      #1, @MMRD    ;KT OFF IF ON
5417 034254 032777 020000 144652      BIT      #SW13, @SWR ;INHIBIT TYPEOUTS?
5418 034262 001004                      BNE      9$           ;BRANCH IF YES
5419 034264 104401 041645                      TYPE     ,MSG13       ;DEVICE RDY BIT DOES NOT SET
5420 034270 104401 041705                      TYPE     ,MSG14       ;FURTHER NPR TESTS ABORTED
5421 034274 005726                      TST      (SP)+        ;RESTORE STACK FROM SUBROUTINE CALL
5422 034276 000137 033020 9$:    JMP      SEOP         ;GO TO END OF PROGRAM
5423
5424 034302 005037 001204 1$:    CLR      $TMP5      ;INIT COUNTER TO WAIT FOR RDY BIT
5425 034306 032737 000100 177400 4$:    BIT      #100, @RKDS ;IS DRIVE RDY?
5426 034314 001004                      BNE      3$           ;BRANCH IF YES
5427 034316 005237 001204                      INC      $TMP5       ;WAIT FOR RDY TO SET
5428 034322 001371                      BNE      4$           ;BRANCH IF HAVEN'T WAITED MAX TIME
5429 034324 000750                      BR       5$           ;REPORT DEVICE NOT READY
5430
5431 034326 012737 000001 177404 3$:    MOV      #1, @RKCS   ;RESET CONTROLLER
5432 034334 005037 001204                      CLR      $TMP5      ;INIT COUNTER TO WAIT FOR RDY BIT

```

```

5433 034340 105737 177404      7S:  TSTB   @RKCS      :CONTROLLER RDY?
5434 034344 100404              BMI     6S          :BRANCH IF YES
5435 034346 005237 001204      INC     $TMP5      :WAIT FOR RDY TO SET
5436 034352 001372              BNE     7S          :BRANCH IF HAVEN'T WAITED MAX TIME
5437 034354 000734              BR      5S          :REPORT DEVICE NOT RDY
5438
5439 034356 012737 177777 177406 6S:  MOV     @-1,@RKWC   :SET WORD COUNT FOR 1 XFER
5440 034364 013737 001214 177412      MOV     @CREG2,@RKDA :SET UP DISK ADDRESS REG
5441 034372 012737 000004 177404      MOV     #4,@RKCS     :SET UP DISK TO DO DATO
5442 034400 017637 000000 177410      MOV     @($P),@RKBA  :SET UP XFER ADDRESS
5443 034406 062716 000002              ADD     #2,$P        :LOOK AT HIGH ADDRESS BITS
5444 034412 017637 000000 001204      MOV     @($P),$TMP5  :GET HIGH ADDRESS BITS
5445 034420 062716 000002              ADD     #2,$P        :ADJUST PC FOR RETURN
5446 034424 005037 001202              CLR     $TMP4       :INIT COUNT FOR SHIFT
5447 034430 006337 001204      8S:  ASL     $TMP5       :SHIFT ADDRESS BITS TO RKCS ADDR. BIT'S POSITION
5448 034434 005237 001202              INC     $TMP4       :COUNT SHIFTS
5449 034440 023727 001202 000004      CMP     $TMP4,#4    :ALL DONE?
5450 034446 001370              BNE     8S          :BRANCH IF NO
5451 034450 053737 001204 177404      BIS     $TMP5,@RKCS :SET UP THE EXTENDED MEMORY BITS
5452 034456 000207              RTS     PC          :RETURN
5453
5454 ;////////////////////////////////////
5455 ;SUBROUTINE TO SETUP AN RPO3 TO DO 1 NPR DATO
5456 ;TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL
5457 ;////////////////////////////////////
5458
5459 034460 005737 176714      HRPO3: TST     @RPCS     :ANY ERRORS?
5460 034464 001416              BEQ     1S          :BRANCH IF NO
5461 034466 042737 000001 177572      BIC     #1,@MMRO    :KT OFF IF ON
5462 034474 032777 020000 144432      BIT     @SW13,@SWR  :INHIBIT TYPEOUTS?
5463 034502 001004              BNE     2S          :BRANCH IF YES
5464 034504 104401 041614      TYPE   ,MSG12      :DEVICE ERROR BIT SET
5465 034510 104401 041705      TYPE   ,MSG14      :FURTHER NPR TESTS ABORTED
5466 034514 005726              TST     ($P)+      :RESTORE STACK FROM SUBROUTINE CALL
5467 034516 000137 033020      JMP     $EOP       :GO TO END OF PROG
5468
5469 034522 005037 001204      1S:  CLR     $TMP5     :INIT COUNTER TO WAIT FOR RDY BIT
5470 034526 105737 176714      4S:  TSTB   @RPCS     :CONTROLLER RDY?
5471 034532 100421              BMI     3S          :BRANCH IF YES
5472 034534 005237 001204      INC     $TMP5      :WAIT FOR RDY TO SET
5473 034540 001372              BNE     4S          :BRANCH IF HAVEN'T WAITED MAX TIME
5474 034542 042737 000001 177572      BIC     #1,@MMRO    :KT OFF IF ON
5475 034550 032777 020000 144356      BIT     @SW13,@SWR  :INHIBIT TYPEOUTS?
5476 034556 001004              BNE     5S          :BRANCH IF YES
5477 034560 104401 041645      TYPE   ,MSG13      :DEVICE RDY BIT DID NOT SET
5478 034564 104401 041705      TYPE   ,MSG14      :FURTHER NPR DEVICE TEST ABORTED
5479 034570 005726              TST     ($P)+      :RESTORE STACK FROM SUBROUTINE CALL
5480 034572 000137 033020      JMP     $EOP       :GO TO END OF PROG
5481
5482 034576 005037 001204      3S:  CLR     $TMP5     :INIT COUNTER TO WAIT FOR RDY BIT
5483 034602 005737 176710      7S:  TST     @RPDS     :IS DEVICE RDY?
5484 034606 100404              BMI     6S          :BRANCH IF YES
5485 034610 005237 001204      INC     $TMP5      :WAIT FOR RDY TO SET
5486 034614 001372              BNE     7S          :BRANCH IF HAVEN'T WAITED MAX TIME
5487 034616 000751              BR      8S          :REPORT RDY DID NOT SET
5488

```

```

5489 034620 012737 177776 176716 6S:  MOV      #2, @#RPMC      ;SET UP TO DO MIN # OF XFERS(2)
5490 034626 013737 001214 176714      MOV      CREG2, @#RPCS    ;START TO SETUP CONTROLLER FOR NPR DATO
5491 034634 017637 000000 176720      MOV      @2(SP), @#RPBA   ;SETUP XFER ADDRESS
5492 034642 062716 000002      ADD      @2, (SP)        ;LOOK AT HIGH XFER ADDRESS
5493 034646 017637 000000 001202      MOV      @2(SP), STMP4    ;GET HIGH XFER ADDR.
5494 034654 062716 000002      ADD      @2, (SP)        ;ADJUST PC FOR RETURN
5495 034660 005037 001204      CLR      STMP5           ;INIT SHIFT COUNTER
5496 034664 006337 001202 9S:  ASL      STMP4           ;SHIFT ADDR. BITS TO COINCIDE WITH RPCS EXTENDED ADDR. B
5497 034670 005237 001204      INC      STMP5           ;COUNT SHIFTS
5498 034674 022737 000004 001204      CMP      #4, STMP5      ;FINISHED?
5499 034702 001370      BNE      9S             ;BRANCH IF NO
5500 034704 053737 001202 176714      BIS      STMP4, @#RPCS   ;SETUP THE EXTENDED MEM ADDR.
5501 034712 000207      RTS      PC              ;RETURN

```

```

;////////////////////////////////////
;SUBROUTINE TO SETUP A TUIO TO DO NPR DATO XFERS
;TO THE STARTING ADDRESS FOLLOWING THE SUBROUTINE CALL
;////////////////////////////////////

```

```

5508 034714 052737 010000 172522 HTU10: BIS      #10000, @#MTC    ;POWER CLEAR THE UNIT
5509 034722 000240      NOP                     ;WAIT FOR POWER CLEAR
5510 034724 000240      NOP
5511 034726 012737 177777 172524      MOV      #1, @#MTBRC     ;PREPARE TO BACKSPACE ONE RECORD
5512 034734 013737 001214 172522      MOV      CREG2, @#MTC   ;GET CONTROL MASK
5513 034742 052737 000012 172522      BIS      #12, @#MTC     ;SET UP BACKSPACE COMMAND
5514 034750 005237 172522      INC      @#MTC          ;BACKSPACE
5515 034754 005037 001204      CLR      STMP5         ;INIT COUNTER TO WAIT FOR RDY
5516 034760 032737 000001 172520 2S:  BIT      #1, @#MTC      ;UNIT DONE?
5517 034766 001021      BNE      1S            ;BRANCH IF YES
5518 034770 005237 001204      INC      STMP5         ;WAIT TILL UNIT DONE
5519 034774 001371      BNE      2S            ;BRANCH IF HAVEN'T WAITED MAX TIME
5520 034776 005726      TST      (SP)+         ;RESTORE STACK FROM SUBROUTINE CALL
5521 035000 042737 000001 177572      BIC      #1, @#MMRO     ;KT OFF IF ON
5522 035006 032777 020000 144120      BIT      #SW13, @#SWR   ;INHIBIT TYPEOUTS?
5523 035014 001004      BNE      3S            ;BRANCH IF YES
5524 035016 104401 041645      TYPE    'MSG13         ;RDY BIT DID NOT SET
5525 035022 104401 041705      TYPE    'MSG14         ;ABORT ALL TESTS USING NPR DEVICE
5526 035026 000137 033020 3S:  JMP      $EOP          ;GO TO END OF PROGRAM
5527
5528 035032 013737 001214 172522 1S:  MOV      CREG2, @#MTC   ;GET CONTROL MASK
5529 035040 052737 000002 172522      BIS      #2, @#MTC     ;SETUP CONTROL TO DO READ
5530 035046 012737 177760 172524      MOV      #20, @#MTBRC   ;PREPARE TO READ MIN # OF BYTES [20(B)]
5531 035054 017637 000000 172526      MOV      @2(SP), @#MTCMA ;SETUP XFER ADDRESS
5532 035062 062716 000002      ADD      @2, (SP)      ;LOOK AT HIGH ADDRESS
5533 035066 017637 000000 001202      MOV      @2(SP), STMP4  ;GET HIGH ADDRESS
5534 035074 005037 001204      CLR      STMP5         ;INIT SHIFT COUNTER
5535 035100 006337 001202 4S:  ASL      STMP4         ;SHIFT ADDR. BITS TO COINCIDE WITH MTC ADDR BITS
5536 035104 005237 001204      INC      STMP5         ;COUNT SHIFTS
5537 035110 022737 000004 001204      CMP      #4, STMP5     ;DONE?
5538 035116 001370      BNE      4S            ;BRANCH IF NO
5539 035120 053737 001202 172522      BIS      STMP4, @#MTC   ;SETUP HIGH ADDRESS BITS
5540 035126 062716 000002      ADD      @2, (SP)      ;ADJUST PC FOR RETURN
5541 035132 000207      RTS      PC              ;RETURN

```

```

;////////////////////////////////////
;SUBROUTINE TO RID CACHE OF BAD PARITY BY
;OVERWRITING IT WHEN CACHE IS OFF
;////////////////////////////////////

```

5542
5543
5544

```

5545
5546 035134 012705 060000
5547 035140 011525
5548 035142 020527 064000
5549 035146 001374
5550 035150 000207
5551
5552
5553
5554
5555
5556
5557
5558
5559
5560
5561
5562
5563
5564
5565 035152
5566 035152 032777 040000 143754
5567 035160 001104
5568
5569 035162 000416
5570
5571 035164 013746 000004
5572 035170 012737 035210 000004
5573 035176 005737 177060
5574 035202 012637 000004
5575 035206 000453
5576 035210 022626
5577 035212 012637 000004
5578 035216 000413
5579 035220
5580 035220 105737 001103
5581 035224 001421
5582 035226 123737 001115 001103
5583 035234 101015
5584 035236 032777 001000 143670
5585 035244 001404
5586 035246 013737 001110 001106
5587 035254 000446
5588 035256 105037 001103
5589 035262 005037 035406
5590 035266 000415
5591 035270 032777 004000 143636
5592 035276 001011
5593 035300 005737 001244
5594 035304 001406
5595 035306 005237 001104
5596 035312 023737 035406 001104
5597 035320 002024
5598 035322 012737 000001 001104
5599 035330 013737 035410 035406
5600 035336 105237 001102

```

```

////////////////////////////////////
SWEEP:  MOV    #BUFL,RS           ;GET STARTING ADDRESS
64$:   MOV    (RS),(RS)+         ;WRITE ALL CACHE WITH GOOD PARITY
      CMP    RS,#BUFL+4000      ;ALL CACHE WRITTEN?
      BNE    64$                ;BRANCH IF NO
      RTS   PC                  ;RETURN

.SBTTL  SCOPE HANDLER ROUTINE

:*****
:THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
:AND LOAD THE TEST NUMBER(STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
:AND LOAD THE ERROR FLAG (SERFLG) 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
:CALL
:*          SCOPE                ;;SCOPE=IOT

$SCOPE:
1$:   BIT    #BIT14,$SWR         ;;LOOP ON PRESENT TEST?
      BNE    $OVER              ;;YES IF SW14=1
:*****START OF CODE FOR THE XOR TESTER*****
$XTSTR: BR    6$                ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                           THIS INSTRUCTION TO A "NOP" (NOP=240)
                                           SAVE THE CONTENTS OF THE ERROR VECTOR
                                           SET FOR TIMEOUT
                                           TIME OUT ON XOR?
                                           RESTORE THE ERROR VECTOR
                                           GO TO THE NEXT TEST
5$:   CMP    (SP)+,(SP)+         ;CLEAR THE STACK AFTER A TIME OUT
      MOV    (SP)+,$ERRVEC      ;RESTORE THE ERROR VECTOR
      BR    7$                 ;LOOP ON THE PRESENT TEST
6$:; *****END OF CODE FOR THE XOR TESTER*****
2$:   TSTB   SERFLG             ;HAS AN ERROR OCCURRED?
      BEQ    3$                 ;BR IF NO
      CMPB   $ERMAX,SERFLG     ;MAX. ERRORS FOR THIS TEST OCCURRED?
      BHI    3$                 ;BR IF NO
      BIT    #BIT09,$SWR       ;LOOP ON ERROR?
      BEQ    4$                 ;BR IF NO
7$:   MOV    $LPERR,$LPADR      ;SET LOOP ADDRESS TO LAST SCOPE
      BR    $OVER
4$:   CLRB   SERFLG             ;ZERO THE ERROR FLAG
      CLR    $TIMES            ;CLEAR THE NUMBER OF ITERATIONS TO MAKE
      BR    1$                 ;ESCAPE TO THE NEXT TEST
3$:   BIT    #BIT11,$SWR       ;INHIBIT ITERATIONS?
      BNE    1$                 ;BR IF YES
      TST    $PASS             ;IF FIRST PASS OF PROGRAM
      BEQ    1$                 ;INHIBIT ITERATIONS
      INC    $SICNT            ;INCREMENT ITERATION COUNT
      CMP    $TIMES,$SICNT     ;CHECK THE NUMBER OF ITERATIONS MADE
      BGE    $OVER            ;BR IF MORE ITERATION REQUIRED
      MOV    #1,$SICNT         ;REINITIALIZE THE ITERATION COUNTER
      MOV    $MXCNT,$TIMES     ;SET NUMBER OF ITERATIONS TO DO
      INCB  $STSTNM           ;COUNT TEST NUMBERS
$SVLAD: INCB

```

```

5601 035342 113737 001102 001242      MOVB  STSTNM,STSTN      ;; SET TEST NUMBER IN APT MAILBOX
5602 035350 011637 001106              MOV   (SP),SLPADR      ;; SAVE SCOPE LOOP ADDRESS
5603 035354 011637 001110              MOV   (SP),SLPERR      ;; SAVE ERROR LOOP ADDRESS
5604 035360 005037 035606              CLR   SESCOAPE        ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
5605 035364 112737 000001 001115      MOVB  #1,SERMAX        ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
5606 035372 013777 001102 143536  SOVER: MOV  STSTNM,DISP   ;; DISPLAY TEST NUMBER
5607 035400 013716 001106              MOV   SLPADR,(SP)     ;; FUDGE RETURN ADDRESS
5608 035404 000002              RTI                    ;; FIXES PS
5609 035406 000000              $TIMES: 0             ;; NUMBER OF ITERATIONS TO PERFORM
5610 035410 000005              $MXCNT: 5             ;; MAX. NUMBER OF ITERATIONS
5611              .SBTTL  ERROR HANDLER ROUTINE
5612
5613              ;; *****
5614              ;; THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
5615              ;; SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
5616              ;; AND GO TO SERRTYP ON ERROR
5617              ;; THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
5618              ;; $SW15=1      HALT ON ERROR
5619              ;; $SW13=1      INHIBIT ERROR TYPEOUTS
5620              ;; $SW10=1     BELL ON ERROR
5621              ;; $SW09=1     LOOP ON ERROR
5622              ;; $CALL
5623              ;; *      ERROR  N      ;; ERROR=EMT AND N=ERROR ITEM NUMBER
5624
5625 035412      SERROR:
5626 035412 105237 001103      7$:  INCB  SERFLG      ;; SET THE ERROR FLAG
5627 035416 001775              BEQ   7$             ;; DON'T LET THE FLAG GO TO ZERO
5628 035420 013777 001102 143510      MOV   STSTNM,DISP   ;; DISPLAY TEST NUMBER AND ERROR FLAG
5629 035426 032777 002000 143500      BIT   #BIT10,$SWR   ;; BELL ON ERROR?
5630 035434 001402              BEQ   1$             ;; NO - SKIP
5631 035436 104401 035610      TYPE  $BELL         ;; RING BELL
5632 035442 005237 001112      1$:  INC   $ERTTL     ;; COUNT THE NUMBER OF ERRORS
5633 035446 011637 001116      MOV   (SP),SERAPC   ;; GET ADDRESS OF ERROR INSTRUCTION
5634 035452 162737 000002 001116      SUB   #2,SERAPC
5635 035460 117737 143432 001114      MOVB  $SERAPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
5636 035466 032777 020000 143440      BIT   #BIT13,$SWR   ;; SKIP TYPEOUT IF SET
5637 035474 001004              BNE   20$           ;; SKIP TYPEOUTS
5638 035476 004737 035614      JSR   PC,SERRTYP    ;; GO TO USER ERROR ROUTINE
5639 035502 104401 001207      TYPE  ,SCLRF
5640 035506      20$:
5641 035506 122737 000001 001256      CMPB  #APTENV,$ENV  ;; RUNNING IN APT MODE
5642 035514 001007              BNE   2$             ;; NO, SKIP APT ERROR REPORT
5643 035516 113737 001114 035530      MOVB  $ITEMB,21$    ;; SET ITEM NUMBER AS ERROR NUMBER
5644 035524 004737 037050      JSR   PC,$ATY4      ;; REPORT FATAL ERROR TO APT
5645 035530 000              21$: .BYTE  0
5646 035531 000              .BYTE  0
5647 035532 000777      22$: BR    22$           ;; APT ERROR LOOP
5648 035534 005777 143374      2$:  TST   $SWR        ;; HALT ON ERROR
5649 035540 100001              BPL   3$             ;; SKIP IF CONTINUE
5650 035542 000000              HALT                    ;; HALT ON ERROR!
5651 035544 032777 001000 143362      3$:  BIT   #BIT09,$SWR  ;; LOOP ON ERROR SWITCH SET?
5652 035552 001402              BEQ   4$             ;; BR IF NO
5653 035554 013716 001110      MOV   SLPERR,(SP)   ;; FUDGE RETURN FOR LOOPING
5654 035560 005737 035606      4$:  TST   SESCOAPE    ;; CHECK FOR AN ESCAPE ADDRESS
5655 035564 001402              BEQ   5$             ;; BR IF NONE
5656 035566 013716 035606      MOV   SESCOAPE,(SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE

```

```

5657 035572
5658 035572 022737 033106 000042
5659 035600 001001
5660 035602 000000
5661 035604
5662 035604 000002
5663 035606 000000
5664 035610 177607 000377
5665
5666
5667
5668
5669
5670
5671
5672 035614
5673 035614 104401 001207
5674 035620 010046
5675 035622 005000
5676 035624 153700 001114
5677 035630 001004
5678
5679 035632 013746 001116
5680
5681 035636 104402
5682 035640 000426
5683 035642 005300
5684 035644 006300
5685 035646 006300
5686 035650 006300
5687 035652 062700 055074
5688 035656 012037 035666
5689 035662 001404
5690 035664 104401
5691 035666 000000
5692 035670 104401 001207
5693 035674 012037 035704
5694 035700 001404
5695 035702 104401
5696 035704 000000
5697 035706 104401 001207
5698 035712 011000
5699 035714 001004
5700 035716 012600
5701 035720 104401 001207
5702 035724 000207
5703 035726
5704 035726 013046
5705 035730 104402
5706 035732 005710
5707 035734 001770
5708 035736 104401 035744
5709 035742 000771
5710 035744 020040 000
5711 035750
5712

```

```

5S:  CMP      #SENDAD,2#42    ;;ACT-11 AUTO-ACCEPT?
      BNE     6S              ;;BRANCH IF NO
      HALT                    ;;YES

6S:  RTI                      ;;RETURN
$ESCAPE: .WORD 0              ;;ESCAPE ON ERROR ADDRESS
$BELL:   .ASCIIZ <207><377><377> ;;ASCII CODE FOR BELL
.SBTTL  ERROR MESSAGE TIMEOUT ROUTINE

;*****
;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

$ERRTYP:
      TYPE    ,SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
      MOV     RO,-(SP)        ;; SAVE RO
      CLR     RO              ;; PICKUP THE ITEM INDEX
      BISB   2#$ITEMB,RO
      BNE    1S              ;; IF ITEM NUMBER IS ZERO, JUST
                              ;; TYPE THE PC OF THE ERROR
      MOV     $ERRPC,-(SP)    ;; SAVE $ERRPC FOR TIMEOUT
                              ;; ERROR ADDRESS
                              ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
                              ;; GET OUT
                              ;; ADJUST THE INDEX SO THAT IT WILL
                              ;; WORK FOR THE ERROR TABLE

1S:  DEC     RO
      ASL    RO
      ASL    RO
      ASL    RO
      ADD    # $ERRTB,RO     ;; FORM TABLE POINTER
      MOV    (RO)+,2$       ;; PICKUP "ERROR MESSAGE" POINTER
      BEQ    3S              ;; SKIP TIMEOUT IF NO POINTER
      TYPE   "ERROR MESSAGE" ;; TYPE THE "ERROR MESSAGE"
                              ;; "ERROR MESSAGE" POINTER GOES HERE
2S:  .WORD 0
      TYPE   ,SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
3S:  MOV     (RO)+,4$       ;; PICKUP "DATA HEADER" POINTER
      BEQ    5S              ;; SKIP TIMEOUT IF 0
      TYPE   "DATA HEADER"   ;; TYPE THE "DATA HEADER"
                              ;; "DATA HEADER" POINTER GOES HERE
4S:  .WORD 0
      TYPE   ,SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
5S:  MOV     (RO),RO        ;; PICKUP "DATA TABLE" POINTER
      BNE    7S              ;; GO TYPE THE DATA
6S:  MOV     (SP)+,RO       ;; RESTORE RO
      TYPE   ,SCRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
7S:  RTS     PC              ;; RETURN

      MOV    2(RO)+,-(SP)    ;; SAVE 2(RO)+ FOR TIMEOUT
      TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
      TST   (RO)            ;; IS THERE ANOTHER NUMBER?
      BEQ   6S              ;; BR IF NO
      TYPE  ,8$             ;; TYPE TWO(2) SPACES
      BR    7S              ;; LOOP
8S:  .ASCIIZ / /           ;; TWO(2) SPACES
      .SBTTL

```

ROUTINE TO SIZE MEMORY

```

5713
5714
5715
5716
5717
5718
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729 035750 010046
5730 035752 010146
5731 035754 010246
5732 035756 010346
5733 035760 013746 000004
5734 035764 013746 000006
5735 035770 010600
5736
5737 035772 013746 000034
5738 035776 012737 036006 000034
5739 036004 104400
5740 036006 016637 000002 000006
5741 036014 012716 036022
5742 036020 000002
5743 036022 012637 000034
5744 036026 012701 003776
5745 036032 105727
5746 036034 000200
5747 036036 100062
5748 036040 012737 036176 000004
5749 036046 005737 177572
5750 036052 052737 100000 036034
5751 036060 005046
5752 036062 012702 172340
5753 036066 012703 000010
5754 036072 012762 077406 177740
5755 036100 011622
5756 036102 062716 000200
5757 036106 077307
5758 036110 012742 177600
5759 036114 005042
5760 036116 012737 036134 000004
5761 036124 012737 000020 172516
5762 036132 000401
5763 036134 022626
5764 036136 005237 177572
5765 036142 012737 036166 000004
5766 036150 005737 143776
5767 036154 062712 000040
5768 036160 023712 172356

```

```

*****
*CALL:
*      JSR          PC,$SIZE
*      RETURN
*SLSTAD WILL CONTAIN:
*      WITH KT11--LAST VIRTUAL ADDRESS OF THE LAST BANK
*      WITHOUT KT11 --LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
*SLSTBK WILL CONTAIN THE LAST BANK AS A SAF
*
*SKT11 IS THE MEMORY MANAGEMENT KEY
*BIT07 = 0 DON'T USE MEMORY MANAGEMENT
*      MUST BE SET UP BEFORE THE CALL
*BIT15 =0 DON'T HAVE MEMORY MANAGEMENT OPTION
*      DETERMINED BY ROUTINE
*
$SIZE:  MOV      RO,-(SP)          ;SAVE RO ON THE STACK
        MOV      R1,-(SP)          ;SAVE R1 ON THE STACK
        MOV      R2,-(SP)          ;SAVE R2 ON THE STACK
        MOV      R3,-(SP)          ;SAVE R3 ON THE STACK
        MOV      @ERRVEC,-(SP)     ;SAVE PRESENT ERROR VECTOR PS & PC
        MOV      @ERRVEC+2,-(SP)
        MOV      SP,RO             ;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
        MOV      @TRAPVEC,-(SP)    ;SAVE CURRENT TRAP VECTOR
        MOV      @64S,@TRAPVEC     ;SETUP NEW TRAP VECTOR
        TRAP
        MOV      2(SP),@ERRVEC+2   ;SAVE PSW IN @ERRVEC+2
        MOV      @65S,(SP)         ;REPLACE OLD PC WITH NEW
        RTI                         ;RESTORE PSW
64S:   MOV      (SP)+,@TRAPVEC     ;RESTORE OLD TRAP VECTOR
        MOV      @3776,R1          ;SETUP ADDRESS
        TSTB    (PC)+              ;USE MEMORY MANAGEMENT?
SKT11: .WORD    200                ;SET TO USE MEMORY MANAGEMENT
        BPL     SCORE              ;BR IF NO
        MOV     @SKTNEX,@ERRVEC    ;SET FOR TIMEOUT
        TST     @SR0               ;KT11 ARE YOU THERE?
        BIS     @10000,SKT11       ;YES--SET KT11 KEY
        CLR     -(SP)              ;INITIALIZE FOR "PAR" LOADING
        MOV     @KIPAR,R2           ;ADDRESS OF FIRST "PAR"
        MOV     @ID8,R3            ;LOAD EIGHT "PAR.'S" AND EIGHT "PDR.'S"
1S:    MOV     @77406,-40(R2)      ;PDR = 4K, UP, READ/WRITE
        MOV     (SP),(R2)+         ;LOAD "PAR"
        ADD     @200,(SP)          ;UPDATE FOR NEXT "PAR"
        SOB    R3,1S              ;LOOP UNTIL ALL EIGHT ARE LOADED
        MOV     @177600,-(R2)      ;SETUP KIPAR7 FOR I/O
        CLR     -(R2)              ;SETUP KIPAR6 FOR TESTING
        MOV     @2S,@ERRVEC        ;CATCH TIMEOUT IF NO SR3
        MOV     @20,@SR3           ;ENABLE 22 BIT MODE
        BR     3S                  ;THIS PDP-11 HAS A SR3 REGISTER
2S:    CMP     (SP)+,(SP)+         ;CLEAN OFF THE STACK--NO SR3
3S:    INC     @SR0                ;TURN ON MEMORY MANAGEMENT
4S:    MOV     @SKTOUT,@ERRVEC    ;SET FOR TIME OUT
        TST     @143776            ;TRAP ON NON-EX-MEM
        ADD     @40,(R2)           ;MAKE A 1K STEP
        CMP     @KIPAR7,(R2)      ;LAST ONE?

```

```

5769 036164 101371
5770 036166 011202
5771 036170 005037 177572
5772 036174 000421
5773 036176 042737 100000 036034
5774 036204 012737 036234 000004
5775 036212 005002
5776 036214 062701 004000
5777 036220 062702 000040
5778 036224 005711
5779 036226 022701 177776
5780 036232 001370
5781 036234 162701 004000
5782 036240 162702 000040
5783 036244 010006
5784 036246 012637 000006
5785 036252 012637 000004
5786 036256 010137 036320
5787 036262 010237 036322
5788 036266 012603
5789 036270 012602
5790 036272 012601
5791 036274 012600
5792
5793 036276 010046
5794 036300 076600
5795 036302 000022
5796 036304 052700 100001
5797 036310 076600
5798 036312 000222
5799 036314 012600
5800
5801 036316 000207
5802 036320 000000
5803 036322 000000
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816 036324
5817 036324 010046
5818 036326 010146
5819 036330 010246
5820 036332 010346
5821 036334 010546
5822 036336 012746 020200
5823 036342 016605 000020
5824 036346 100004

SKTOUT: BHI 4$ ;;NO--TRY IT
MOV (R2),R2 ;;GET LAST BANK+1
CLR @#SR0 ;;TURN OFF MEMORY MANAGEMENT
BR $SIZE
SKTNEX: BIC @100000,SKT11 ;;KT11 NON-EXISTENT
SCORE: MOV @#SCROUT,@#ERRVEC ;;SET FOR TIMEOUT
CLR R2 ;;SET UP BANK
IS: ADD @4000,R1 ;;INCREMENT BY 1K
ADD @40,R2 ;;1K STEP
TST (R1) ;;TRAP ON TIME OUT
CMP @177776,R1 ;;LAST ONE
BNE IS ;;NO--TRY AGAIN
SCROUT: SUB @4000,R1
SSIZE: SUB @40,R2 ;;DROP BACK
MOV RO,SP ;;RESTORE THE STACK
MOV (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
MOV (SP)+,@#ERRVEC
MOV R1,$LSTAD ;;LAST ADDRESS
MOV R2,$LSTBK ;;LAST BANK
MOV (SP)+,R3 ;;RESTORE R3
MOV (SP)+,R2 ;;RESTORE R2
MOV (SP)+,R1 ;;RESTORE R1
MOV (SP)+,RO ;;RESTORE RO
MOV RO,-(SP) ;;SAVE RO FOR MED INST
MED ;;GET CONTENTS OF LOG REG
.WORD RLOG
BIS @100001,RO ;;ENABLE ERROR LOG & LOG FIRST MODE
MED ;;UNLOCK ERROR LOG
.WORD WLOG
MOV (SP)+,RO ;;RESTORE RO

RTS PC
SLSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
SLSTBK: .WORD 0 ;;CONTAINS THE LAST BANK
.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

STYPDS: MOV RO,-(SP) ;;PUSH RO ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV @20200,-(SP) ;;SET BLANK SWITCH AND SIGN
MOV 20(SP),R5 ;;GET THE INPUT NUMBER
BPL IS ;;BR IF INPUT IS POS.

```



```

5825 036350 005405          NEG      R5          ;; MAKE THE BINARY NUMBER POS.
5826 036352 112766 000055 000001 1S:  MOVB   #'-,1(SP)    ;; MAKE THE ASCII NUMBER NEG.
5827 036360 005000          CLR     R0          ;; ZERO THE CONSTANTS INDEX
5828 036362 012703 036540          MOV     #SDBLK,R3   ;; SETUP THE OUTPUT POINTER
5829 036366 112723 000040          MOVB   #' ,(R3)+    ;; SET THE FIRST CHARACTER TO A BLANK
5830 036372 005002          CLR     R2          ;; CLEAR THE BCD NUMBER
5831 036374 016001 036530          MOV     $DTBL(R0),R1 ;; GET THE CONSTANT
5832 036400 160105          SUB     R1,R5       ;; FORM THIS BCD DIGIT
5833 036402 002402          BLT    4$          ;; BR IF DONE
5834 036404 005202          INC     R2          ;; INCREASE THE BCD DIGIT BY 1
5835 036406 000774          BR     3$          ;;
5836 036410 060105          4S:  ADD     R1,R5       ;; ADD BACK THE CONSTANT
5837 036412 005702          TST    R2          ;; CHECK IF BCD DIGIT=0
5838 036414 001002          BNE    5$          ;; FALL THROUGH IF 0
5839 036416 105716          TSTB   (SP)        ;; STILL DOING LEADING 0'S?
5840 036420 100407          BMI    7$          ;; BR IF YES
5841 036422 106316          5S:  ASLB   (SP)        ;; MSD?
5842 036424 103003          BCC    6$          ;; BR IF NO
5843 036426 116663 000001 177777 6S:  MOVB   1(SP),-1(R3) ;; YES--SET THE SIGN
5844 036434 052702 000060          BIS    #'0,R2      ;; MAKE THE BCD DIGIT ASCII
5845 036440 052702 000040          7S:  BIS    #' ,R2      ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
5846 036444 110223          MOVB   R2,(R3)+    ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
5847 036446 005720          TST   (R0)+        ;; JUST INCREMENTING
5848 036450 020027 000010          CMP    R0,#10     ;; CHECK THE TABLE INDEX
5849 036454 002746          BLT    2$          ;; GO DO THE NEXT DIGIT
5850 036456 003002          BGT    8$          ;; GO TO EXIT
5851 036460 010502          MOV    R5,R2      ;; GET THE LSD
5852 036462 000764          BR     6$          ;; GO CHANGE TO ASCII
5853 036464 105726          8S:  TSTB   (SP)+      ;; WAS THE LSD THE FIRST NON-ZERO?
5854 036466 100003          BPL    9$          ;; BR IF NO
5855 036470 116663 177777 177776 9S:  MOVB   -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
5856 036476 105013          CLRB   (R3)        ;; SET THE TERMINATOR
5857 036500 012605          MOV    (SP)+,R5    ;; POP STACK INTO R5
5858 036502 012603          MOV    (SP)+,R3    ;; POP STACK INTO R3
5859 036504 012602          MOV    (SP)+,R2    ;; POP STACK INTO R2
5860 036506 012601          MOV    (SP)+,R1    ;; POP STACK INTO R1
5861 036510 012600          MOV    (SP)+,R0    ;; POP STACK INTO R0
5862 036512 104401 036540          TYPE   $SDBLK      ;; NOW TYPE THE NUMBER
5863 036516 016666 000002 000004          MOV    2(SP),4(SP) ;; ADJUST THE STACK
5864 036524 012616          MOV    (SP)+,(SP) ;;
5865 036526 000002          RTI                                ;; RETURN TO USER
5866 036530 023420          SOTBL: 10000.
5867 036532 001750          1000.
5868 036534 000144          100.
5869 036536 000012          10.
5870 036540 000004          SDBLK: .BLKW 4
5871          .SBTTL TYPE ROUTINE
5872
5873          ;; *****
5874          ;; *ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5875          ;; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5876          ;; *NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5877          ;; *NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5878          ;; *NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5879          ;; *
5880          ;; *CALL:

```

```

5881
5882
5883
5884
5885
5886
5887
5888 036550 105737 001153 $TYPE: TST@ $TPFLG
5889 036554 100002 BPL 1$
5890 036556 000000 HALT
5891 036560 000430 BR 3$
5892 036562 010046 1$: MOV RO, -(SP)
5893 036564 017600 000002 MOV @2(SP), RO
5894 036570 122737 000001 001256 CMPB @APTENV, SENV
5895 036576 001011 BNE 62$
5896 036600 132737 000100 001257 BITB @APTPOOL, SENVM
5897 036606 001405 BEQ 62$
5898 036610 010037 036620 MOV RO, 61$
5899 036614 004737 037040 JSR PC, $ATY3
5900 036620 000000 61$: .WORD 0
5901 036622 132737 000040 001257 62$: BITB @APTCSUP, SENVM
5902 036630 001003 BNE 60$
5903 036632 112046 2$: MOVB (RO)+, -(SP)
5904 036634 001005 BNE 4$
5905 036636 005726 TST (SP)+
5906 036640 012600 60$: MOV (SP)+, RO
5907 036642 062716 000002 3$: ADD #2, (SP)
5908 036646 000002 RTI
5909 036650 122716 000011 4$: CMPB @HT, (SP)
5910 036654 001430 BEQ 8$
5911 036656 122716 000200 CMPB @CRLF, (SP)
5912 036662 001006 BNE 5$
5913 036664 005726 TST (SP)+
5914 036666 104401 TYPE
5915 036670 001207 $CRLF
5916 036672 105037 037026 CLRB $CHARCNT
5917 036676 000755 BR 2$
5918 036700 004737 036762 5$: JSR PC, $TYPEC
5919 036704 123726 001152 6$: CMPB $FILLC, (SP)+
5920 036710 001350 BNE 2$
5921 036712 013746 001150 MOV $NULL, -(SP)
5922
5923 036716 105366 000001 7$: DECB 1(SP)
5924 036722 002770 BLT 6$
5925 036724 004737 036762 JSR PC, $TYPEC
5926 036730 105337 037026 DECB $CHARCNT
5927 036734 000770 BR 7$
5928
5929
5930 ;HORIZONTAL TAB PROCESSOR
5931 036736 112716 000040 8$: MOVB #' (SP)
5932 036742 004737 036762 9$: JSR PC, $TYPEC
5933 036746 132737 000007 037026 BITB #7, $CHARCNT
5934 036754 001372 BNE 9$
5935 036756 005726 TST (SP)+
5936 036760 000724 BR 2$

```

```

;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
;; IS THERE A TERMINAL?
;; BR IF YES
;; HALT HERE IF NO TERMINAL
;; LEAVE
;; SAVE RO
;; GET ADDRESS OF ASCIZ STRING
;; RUNNING IN APT MODE
;; NO, GO CHECK FOR APT CONSOLE
;; SPOOL MESSAGE TO APT
;; NO, GO CHECK FOR CONSOLE
;; SETUP MESSAGE ADDRESS FOR APT
;; SPOOL MESSAGE TO APT
;; MESSAGE ADDRESS
;; APT CONSOLE SUPPRESSED
;; YES, SKIP TYPE OUT
;; PUSH CHARACTER TO BE TYPED ONTO STACK
;; BR IF IT ISN'T THE TERMINATOR
;; IF TERMINATOR POP IT OFF THE STACK
;; RESTORE RO
;; ADJUST RETURN PC
;; RETURN
;; BRANCH IF <HT>
;; BRANCH IF NOT <CRLF>
;; POP <CR><LF> EQUIV
;; TYPE A CR AND LF
;; CLEAR CHARACTER COUNT
;; GET NEXT CHARACTER
;; GO TYPE THIS CHARACTER
;; IS IT TIME FOR FILLER CHARS.?
;; IF NO GO GET NEXT CHAR.
;; GET # OF FILLER CHARS. NEEDED
;; AND THE NULL CHAR.
;; DOES A NULL NEED TO BE TYPED?
;; BR IF NO--GO POP THE NULL OFF OF STACK
;; GO TYPE A NULL
;; DO NOT COUNT AS A COUNT
;; LOOP

```

5937	036762	105777	142156		\$TYPEC: TSTB	\$STPS		::WAIT UNTIL PRINTER IS READY
5938	036766	100375			BPL	\$TYPEC		
5939	036770	116677	000002	142150	MOVB	2(SP), \$STPB		::LOAD CHAR TO BE TYPED INTO DATA REG.
5940	036776	122766	000015	000002	CMPB	#CR, 2(SP)		::IS CHARACTER A CARRIAGE RETURN?
5941	037004	001003			BNE	1\$::BRANCH IF NO
5942	037006	105037	037026		CLRB	\$CHARCNT		::YES--CLEAR CHARACTER COUNT
5943	037012	000406			BR	\$TYPEX		::EXIT
5944	037014	122766	000012	000002	1\$: CMPB	#LF, 2(SP)		::IS CHARACTER A LINE FEED?
5945	037022	001402			BEQ	\$TYPEX		::BRANCH IF YES
5946	037024	105227			INCB	(PC)+		::COUNT THE CHARACTER
5947	037026	000000			\$CHARCNT: .WORD	0		::CHARACTER COUNT STORAGE
5948	037030	000207			\$TYPEX: RTS	PC		
5949								
5950					.SBTTL APT COMMUNICATIONS ROUTINE			
5951								
5952					:*****			
5953	037032	112737	000001	037276	\$ATY1: MOVB	#1, \$FFLG		::TO REPORT FATAL ERROR
5954	037040	112737	000001	037274	\$ATY3: MOVB	#1, \$MFLG		::TO TYPE A MESSAGE
5955	037046	000403			BR	\$ATYC		
5956	037050	112737	000001	037276	\$ATY4: MOVB	#1, \$FFLG		::TO ONLY REPORT FATAL ERROR
5957	037056				\$ATYC:			
5958	037056	010046			MOV	RO, -(SP)		::PUSH RO ON STACK
5959	037060	010146			MOV	RI, -(SP)		::PUSH RI ON STACK
5960	037062	105737	037274		TSTB	\$MFLG		::SHOULD TYPE A MESSAGE?
5961	037066	001450			BEQ	5\$::IF NOT: BR
5962	037070	122737	000001	001256	CMPB	#APTENV, \$ENV		::OPERATING UNDER APT?
5963	037076	001031			BNE	3\$::IF NOT: BR
5964	037100	132737	000100	001257	BITB	#APTPOOL, \$ENVM		::SHOULD SPOOL MESSAGES?
5965	037106	001425			BEQ	3\$::IF NOT: BR
5966	037110	017600	000004		MOV	24(SP), RO		::GET MESSAGE ADDR.
5967	037114	062766	000002	000004	ADD	#2, 4(SP)		::BUMP RETURN ADDR.
5968	037122	005737	001236		1\$: TST	\$MSGTYPE		::SEE IF DONE W/ LAST XMISSION?
5969	037126	001375			BNE	1\$::IF NOT: WAIT
5970	037130	010037	001252		MOV	RO, \$MSGAD		::PUT ADDR IN MAILBOX
5971	037134	105720			2\$: TSTB	(RO)+		::FIND END OF MESSAGE
5972	037136	001376			BNE	2\$		
5973	037140	163700	001252		SUB	\$MSGAD, RO		::SUB START OF MESSAGE
5974	037144	006200			ASR	RO		::GET MESSAGE LNTH IN WORDS
5975	037146	010037	001254		MOV	RO, \$MSGGLT		::PUT LENGTH IN MAILBOX
5976	037152	012737	000004	001236	MOV	#4, \$MSGTYPE		::TELL APT TO TAKE MSG.
5977	037160	000413			BR	5\$		
5978	037162	017637	000004	037206	3\$: MOV	24(SP), 4\$::PUT MSG ADDR IN JSR LINKAGE
5979	037170	062766	000002	000004	ADD	#2, 4(SP)		::BUMP RETURN ADDRESS
5980	037176	013746	177776		MOV	177776, -(SP)		::PUSH 177776 ON STACK
5981	037202	004737	036550		JSR	PC, \$TYPE		::CALL TYPE MACRO
5982	037206	000000			4\$: .WORD	0		
5983	037210				5\$:			
5984	037210	105737	037276		10\$: TSTB	\$FFLG		::SHOULD REPORT FATAL ERROR?
5985	037214	001416			BEQ	12\$::IF NOT: BR
5986	037216	005737	001256		TST	\$ENV		::RUNNING UNDER APT?
5987	037222	001413			BEQ	12\$::IF NOT: BR
5988	037224	005737	001236		11\$: TST	\$MSGTYPE		::FINISHED LAST MESSAGE?
5989	037230	001375			BNE	11\$::IF NOT: WAIT
5990	037232	017637	000004	001240	MOV	24(SP), \$FATAL		::GET ERROR #
5991	037240	062766	000002	000004	ADD	#2, 4(SP)		::BUMP RETURN ADDR.
5992	037246	005237	001236		INC	\$MSGTYPE		::TELL APT TO TAKE ERROR

```

5993 037252 105037 037276
5994 037256 105037 037275
5995 037262 105037 037274
5996 037266 012601
5997 037270 012600
5998 037272 000207
5999 037274 000
6000 037275 000
6001 037276 000
6002 037300
6003 000200
6004 000001
6005 000100
6006 000040
6007
6008
6009
6010
6011
6012
6013
6014
6015
6016
6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032 037300 017646 000000
6033 037304 116637 000001 037523
6034 037312 112637 037525
6035 037316 062716 000002
6036 037322 000406
6037 037324 112737 000001 037523
6038 037332 112737 000006 037525
6039 037340 112737 000005 037522
6040 037346 010346
6041 037350 010446
6042 037352 010546
6043 037354 113704 037525
6044 037360 005404
6045 037362 062704 000006
6046 037366 110437 037524
6047 037372 113704 037523
6048 037376 016605 000012
    
```

```

125:  CLRB  SFFLG  ;; CLEAR FATAL FLAG
      CLRB  SLFLG  ;; CLEAR LOG FLAG
      CLRB  SMFLG  ;; CLEAR MESSAGE FLAG
      MOV   (SP)+,R1  ;; POP STACK INTO R1
      MOV   (SP)+,R0  ;; POP STACK INTO R0
      RTS   PC        ;; RETURN
SMFLG: .BYTE 0      ;; MESSG. FLAG
SLFLG: .BYTE 0      ;; LOG FLAG
SFFLG: .BYTE 0      ;; FATAL FLAG
      .EVEN
    
```

```

APTSIZE=200
APTENV=001
APTSPool=100
APTCSUP=040
.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.
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOS   N              ;; CALL FOR TYPEOUT
*   .BYTE  M              ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;; M=1 OR 0
*                               ;; 1=TYPE LEADING ZEROS
*                               ;; 0=SUPPRESS LEADING ZEROS
*
*STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPON   N              ;; CALL FOR TYPEOUT
*
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOC   N              ;; CALL FOR TYPEOUT
*
STYPOS: MOV     2(SP),-(SP)  ;; PICKUP THE MODE
        MOV     1(SP),SOFILL ;; LOAD ZERO FILL SWITCH
        MOV     (SP)+,SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
        ADD     #2,(SP)     ;; ADJUST RETURN ADDRESS
        BR     STYPON
STYPOC: MOV     #1,SOFILL   ;; SET THE ZERO FILL SWITCH
        MOV     #6,SOMODE+1 ;; SET FOR SIX(6) DIGITS
STYPON: MOV     #5,SOCNT    ;; SET THE ITERATION COUNT
        MOV     R3,-(SP)    ;; SAVE R3
        MOV     R4,-(SP)    ;; SAVE R4
        MOV     R5,-(SP)    ;; SAVE R5
        MOV     SOMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
        NEG     R4
        ADD     #6,R4       ;; SUBTRACT IT FOR MAX. ALLOWED
        MOV     R4,SOMODE   ;; SAVE IT FOR USE
        MOV     SOFILL,R4   ;; GET THE ZERO FILL SWITCH
        MOV     12(SP),R5   ;; PICKUP THE INPUT NUMBER
    
```

H10

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 109
BINARY TO OCTAL (ASCII) AND TYPE

```
6049 037402 005003
6050 037404 006105
6051 037406 000404
6052 037410 006105
6053 037412 006105
6054 037414 006105
6055 037416 010503
6056 037420 006103
6057 037422 105337 037524
6058 037426 100016
6059 037430 042703 177770
6060 037434 001002
6061 037436 005704
6062 037440 001403
6063 037442 005204
6064 037444 052703 000060
6065 037450 052703 000040
6066 037454 110337 037520
6067 037460 104401 037520
6068 037464 105337 037522
6069 037470 003347
6070 037472 002402
6071 037474 005204
6072 037476 000744
6073 037500 012605
6074 037502 012604
6075 037504 012603
6076 037506 016666 000002 000004
6077 037514 012616
6078 037516 000002
6079 037520 000
6080 037521 000
6081 037522 000
6082 037523 000
6083 037524 000000
6084
6085
6086
6087
6088
6089
6090
6091
6092
6093
6094
6095
6096
6097
6098
6099
6100 037526 011646
6101 037530 016666 000004 000002
6102 037536 105777 141376
6103 037542 100375
6104 037544 117766 141372 000004
```

```
CLR R3
1$: ROL R5
BR 3$
2$: ROL R5
ROL R5
ROL R5
MOV R5,R3
3$: ROL R3
DECB $OMODE
BPL 7$
BIC #177770,R3
4$: BNE 4$
TST R4
BEQ 5$
INC R4
5$: BIS #'0,R3
BIS #' ,R3
MOVB R3,$$
TYPE 8$
7$: DECB $OCNT
BGT 2$
BLT 6$
INC R4
BR 2$
6$: MOV (SP)+,R5
MOV (SP)+,R4
MOV (SP)+,R3
MOV 2(SP),4(SP)
MOV (SP)+,(SP)
RTI
8$: .BYTE 0
.BYTE 0
$OCNT: .BYTE 0
$OFILL: .BYTE 0
$OMODE: .WORD 0
.SBTTL TTY INPUT ROUTINE

;*****
;ENABL LSB
;DSABL LSB

;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;CALL:
; * R0CHR ; INPUT A SINGLE CHARACTER FROM THE TTY
; * RETURN HERE ; CHARACTER IS ON THE STACK
; * ; WITH PARITY BIT STRIPPED OFF
;

SRDCHR: MOV (SP)-,(SP)
MOV 4(SP),2(SP)
1$: TSTB 2$TKS
BPL 1$
MOV 2$TKB,4(SP)
```

;; CLEAR THE OUTPUT WORD
;; ROTATE MSB INTO "C"
;; GO DO MSB
;; FORM THIS DIGIT
;; GET LSB OF THIS DIGIT
;; TYPE THIS DIGIT?
;; BR IF NO
;; GET RID OF JUNK
;; TEST FOR 0
;; SUPPRESS THIS 0?
;; BR IF YES
;; DON'T SUPPRESS ANYMORE 0'S
;; MAKE THIS DIGIT ASCII
;; MAKE ASCII IF NOT ALREADY
;; SAVE FOR TYPING
;; GO TYPE THIS DIGIT
;; COUNT BY 1
;; BR IF MORE TO DO
;; BR IF DONE
;; INSURE LAST DIGIT ISN'T A BLANK
;; GO DO THE LAST DIGIT
;; RESTORE R5
;; RESTORE R4
;; RESTORE R3
;; SET THE STACK FOR RETURNING
;; RETURN
;; STORAGE FOR ASCII DIGIT
;; TERMINATOR FOR TYPE ROUTINE
;; OCTAL DIGIT COUNTER
;; ZERO FILL SWITCH
;; NUMBER OF DIGITS TO TYPE

```

6105 037552 042766 177600 000004      BIC      #1C(177), 4(SP)  ;; GET RID OF JUNK IF ANY
6106 037550 026627 000004 000023      CMP      4(SP), #23      ;; IS IT A CONTROL-S?
6107 037556 001013      BNE      3$              ;; BRANCH IF NO
6108 037570 105777 141344      2$: TSTB    2$TKS          ;; WAIT FOR A CHARACTER
6109 037574 100375      BPL      2$              ;; LOOP UNTIL ITS THERE
6110 037576 117746 141340      MOVB    2$TKB, -(SP)     ;; GET CHARACTER
6111 037602 042716 177600      BIC      #1C177, (SP)    ;; MAKE IT 7-BIT ASCII
6112 037606 022627 000021      CMP      (SP)+, #21      ;; IS IT A CONTROL-Q?
6113 037612 001366      BNE      2$              ;; IF NOT DISCARD IT
6114 037614 000750      BR      1$              ;; YES, RESUME
6115 037616 026627 000004 000140      3$: CMP      4(SP), #140    ;; IS IT UPPER CASE?
6116 037624 002407      BLT      4$              ;; BRANCH IF YES
6117 037626 026627 000004 000175      CMP      4(SP), #175     ;; IS IT A SPECIAL CHAR?
6118 037634 003003      BGT      4$              ;; BRANCH IF YES
6119 037636 042766 000040 000004      BIC      #40, 4(SP)      ;; MAKE IT UPPER CASE
6120 037644 000002      4$: RTI                    ;; GO BACK TO USER
6121                                     ;; *****
6122                                     ;; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
6123                                     ;; *CALL:
6124                                     ;; *      RDLIN
6125                                     ;; *      RETURN HERE
6126                                     ;; *
6127                                     ;; INPUT A STRING FROM THE TTY
6128 037646 010346      $RDLIN: MOV      R3, -(SP)  ;; SAVE R3
6129 037650 005046      CLR      -(SP)          ;; CLEAR THE RUBOUT KEY
6130 037652 012703 040102      1$: MOV      #TTYIN, R3   ;; GET ADDRESS
6131 037656 022703 040112      2$: CMP      #TTYIN+8., R3 ;; BUFFER FULL?
6132 037662 101456      BLOS     4$              ;; BR IF YES
6133 037664 104406      RDCHR    ;; GO READ ONE CHARACTER FROM THE TTY
6134 037666 112613      MOVB    (SP)+, (R3)     ;; GET CHARACTER
6135 037670 122713 000177      10$: CMPB    #177, (R3)    ;; IS IT A RUBOUT
6136 037674 001022      BNE      5$              ;; BR IF NO
6137 037676 005716      TST      (SP)           ;; IS THIS THE FIRST RUBOUT?
6138 037700 001007      BNE      6$              ;; BR IF NO
6139 037702 112737 000134 040100      MOVB    #' \, 9$        ;; TYPE A BACK SLASH
6140 037710 104401 040100      TYPE     9$
6141 037714 012716 177777      MOV      #-1, (SP)      ;; SET THE RUBOUT KEY
6142 037720 005303      6$: DEC      R3          ;; BACKUP BY ONE
6143 037722 020327 040102      CMP      R3, #TTYIN     ;; STACK EMPTY?
6144 037726 103434      BLO      4$              ;; BR IF YES
6145 037730 111337 040100      MOVB    (R3), 9$        ;; SETUP TO TYPEOUT THE DELETED CHAR.
6146 037734 104401 040100      TYPE     9$
6147 037740 000746      BR      2$              ;; GO TYPE
6148 037742 005716      5$: TST      (SP)           ;; GO READ ANOTHER CHAR.
6149 037744 001406      BEQ      7$              ;; RUBOUT KEY SET?
6150 037746 112737 000134 040100      MOVB    #' \, 9$        ;; BR IF NO
6151 037754 104401 040100      TYPE     9$
6152 037760 005016      CLR      (SP)           ;; TYPE A BACK SLASH
6153 037762 122713 000025      7$: CMPB    #25, (R3)    ;; CLEAR THE RUBOUT KEY
6154 037766 001003      BNE      8$              ;; IS CHARACTER A CTRL U?
6155 037770 104401 040112      TYPE     %CNTLU         ;; BR IF NO
6156 037774 000726      BR      1$              ;; TYPE A CONTROL "U"
6157 037776 122713 000022      8$: CMPB    #22, (R3)    ;; GO START OVER
6158 040002 001011      BNE      3$              ;; IS CHARACTER A "↑R"?
6159 040004 105013      CLRB    (R3)           ;; BRANCH IF NO
6160 040006 104401 001207      TYPE     , %CRLF        ;; CLEAR THE CHARACTER
;; TYPE A "CR" & "LF"

```

6161	040012	104401	040102			TYPE	\$TTYIN	;;	TYPE THE INPUT STRING	
6162	040016	000717				BR	2\$;;	GO PICKUP ANOTHER CHAFTER	
6163	040020	104401	001206		4\$:	TYPE	\$QUES	;;	TYPE A '?'	
6164	040024	000712				BR	1\$;;	CLEAR THE BUFFER AND LOOP	
6165	040026	111337	040100		3\$:	MOVB	(R3),9\$;;	ECHO THE CHARACTER	
6166	040032	104401	040100			TYPE	9\$			
6167	040036	122723	000015			CMPB	#15,(R3)+	;;	CHECK FOR RETURN	
6168	040042	001305				BNE	2\$;;	LOOP IF NOT RETURN	
6169	040044	105063	177777			CLRB	-1(R3)	;;	CLEAR RETURN (THE 15)	
6170	040050	104401	001210			TYPE	\$LF	;;	TYPE A LINE FEED	
6171	040054	005726				TST	(SP)+	;;	CLEAN RUBOUT KEY FROM THE STACK	
6172	040056	012603				MOV	(SP)+,R3	;;	RESTORE R3	
6173	040060	011646				MOV	(SP)-,(SP)	;;	ADJUST THE STACK AND PUT ADDRESS OF THE	
6174	040062	016666	000004	000002		MOV	4(SP),2(SP)	;;	FIRST ASCII CHARACTER ON IT	
6175	040070	012766	040102	000004		MOV	#TTYIN,4(SP)			
6176	040076	000002				RTI		;;	RETURN	
6177	040100	000			9\$:	.BYTE	0	;;	STORAGE FOR ASCII CHAR. TO TYPE	
6178	040101	000				.BYTE	0	;;	TERMINATOR	
6179	040102	000010				\$TTYIN:	.BLKB	8.	;;	RESERVE 8 BYTES FOR TTY INPUT
6180	040112	052536	005015	000		\$CNTLU:	.ASCIZ	/↑U/<15><12>	;;	CONTROL "U"
6181	040117	136	006507	000012		\$CNTLG:	.ASCIZ	/↑G/<15><12>	;;	CONTROL "G"
6182	040124	005015	053523	020122		\$MSWR:	.ASCIZ	<15><12>/SWR = /		
6183	040132	020075	000							
6184	040135	040	047040	053505		\$MNEW:	.ASCIZ	/ NEW = /		
6185	040142	036440	000040							
6186						.SBTTL	READ AN OCTAL NUMBER FROM THE TTY			
6187										
6188										
6189										
6190										
6191										
6192										
6193										
6194										
6195										
6196										
6197										
6198										
6199										
6200	040146	011646				\$RDOCT:	MOV	(SP)-,(SP)	;;	PROVIDE SPACE FOR THE
6201	040150	016666	000004	000002			MOV	4(SP),2(SP)	;;	INPUT NUMBER
6202	040156	010046					MOV	R0,-(SP)	;;	PUSH R0 ON STACK
6203	040160	010146					MOV	R1,-(SP)	;;	PUSH R1 ON STACK
6204	040162	010246					MOV	R2,-(SP)	;;	PUSH R2 ON STACK
6205	040164	104407			1\$:	RDLIN		;;	READ AN ASCIZ LINE	
6206	040166	012600					MOV	(SP)+,R0	;;	GET ADDRESS OF 1ST CHARACTER
6207	040170	010037	040274				MOV	R0,5\$;;	AND SAVE IT
6208	040174	005001					CLR	R1	;;	CLEAR DATA WORD
6209	040176	005002					CLR	R2		
6210	040200	112046			2\$:	MOVB	(R0)+,-(SP)	;;	PICKUP THIS CHARACTER	
6211	040202	001420					BEQ	3\$;;	IF ZERO GET OUT
6212	040204	122716	000060				CMPB	#'0,(SP)	;;	MAKE SURE THIS CHARACTER
6213	040210	003026					BGT	4\$;;	IS AN OCTAL DIGIT
6214	040212	122716	000067				CMPB	#'7,(SP)		
6215	040216	002423					BLT	4\$		
6216	040220	006301					ASL	R1	;;	*2

K10

MD-11-DOKKA-A 11/6X CACHE DIAGNOSTIC
DOKKAA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 112
READ AN OCTAL NUMBER FROM THE TTY

```

6217 040222 006102          ROL    R2
6218 040224 006301          ASL    R1          ;;#4
6219 040226 006102          ROL    R2
6220 040230 006301          ASL    R1          ;;#8
6221 040232 006102          ROL    R2
6222 040234 042716 177770  BIC    #1C7 (SP)  ;; STRIP THE ASCII JUNK
6223 040240 062601          ADD    (SP)+,R1  ;; ADD IN THIS DIGIT
6224 040242 000756          BR     2$        ;; LOOP
6225 040244 005726          3$:   TST    (SP)+  ;; CLEAN TERMINATOR FROM STACK
6226 040246 010166 000012    MOV    R1,12(SP) ;; SAVE THE RESULT
6227 040252 010237 040304    MOV    R2,$HIOCT
6228 040256 012602          MOV    (SP)+,R2  ;; POP STACK INTO R2
6229 040260 012601          MOV    (SP)+,R1  ;; POP STACK INTO R1
6230 040262 012600          MOV    (SP)+,R0  ;; POP STACK INTO R0
6231 040264 000002          RTI
6232 040266 005726          4$:   TST    (SP)+  ;; CLEAN PARTIAL FROM STACK
6233 040270 105010          CLRB   (R0)      ;; SET A TERMINATOR
6234 040272 104401          TYPE  ;; TYPE UP THRU THE BAD CHAR.
6235 040274 000000          5$:   .WORD  0
6236 040276 104401 001206    TYPE  $QUES     ;; "?" "CR" & "LF"
6237 040302 000730          BR     1$        ;; TRY AGAIN
6238 040304 000000          $HIOCT: .WORD  0 ;; HIGH ORDER BITS GO HERE
6239
6240
6241
6242
6243
6244
6245
6246
6247 040306 010046          $TRAP: MOV    R0,-(SP) ;; SAVE R0
6248 040310 016600 000002    MOV    2(SP),R0  ;; GET TRAP ADDRESS
6249 040314 005740          TST    -(R0)     ;; BACKUP BY 2
6250 040316 111000          MOVB   (R0),R0  ;; GET RIGHT BYTE OF TRAP
6251 040320 006300          ASL    R0        ;; POSITION FOR INDEXING
6252 040322 016000 040342    MOV    $TRPAD(R0),R0 ;; INDEX TO TABLE
6253 040326 000200          RTS    R0        ;; GO TO ROUTINE
6254
6255
6256
6257
6258 040330 011646          ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
6259 040332 016666 000004 000002 $TRAP2: MOV    (SP),-(SP) ;; MOVE THE PC DOWN
6260 040340 000002          MOV    4(SP),2(SP) ;; MOVE THE PSW DOWN
6261
6262
6263          .SBTTL TRAP TABLE
6264
6265          ;#THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
6266          ;#BY THE "TRAP" INSTRUCTION.
6267
6268          ;
6269          ; ROUTINE
6270          ; -----
6271          $TRPAD: .WORD  $TRAP2
6272          $TYPE   ;;CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
                   $TYPOC  ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
                   $TYPOS  ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)

```


6273	040352	037340			STYPON	::CALL=TYPON	TRAP+4(104404)	TYPE OCTAL NUMBER (AS PER LAST CALL)
6274	040354	036324			STYPDS	::CALL=TYPDS	TRAP+5(104405)	TYPE DECIMAL NUMBER (WITH SIGN)
6275								
6276								
6277	040356	037526			SRDCHR	::CALL=RDCHR	TRAP+6(104406)	TTY TYPEIN CHARACTER ROUTINE
6278	040360	037646			SRDLIN	::CALL=RDLIN	TRAP+7(104407)	TTY TYPEIN STRING ROUTINE
6279	040362	040146			SRDOCT	::CALL=RDOCT	TRAP+10(104410)	READ AN OCTAL NUMBER FROM TTY
6280								
6281								
6282								
6283								

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

6284	040364	012737	040524	000024	\$PWRDN:	MOV	\$SILLUP, @PWRVEC	::SET FOR FAST UP
6285	040372	012737	000340	000026		MOV	\$340, @PWRVEC+2	::PRIO:7
6286	040400	010046				MOV	R0, -(SP)	::PUSH R0 ON STACK
6287	040402	010146				MOV	R1, -(SP)	::PUSH R1 ON STACK
6288	040404	010246				MOV	R2, -(SP)	::PUSH R2 ON STACK
6289	040406	010346				MOV	R3, -(SP)	::PUSH R3 ON STACK
6290	040410	010446				MOV	R4, -(SP)	::PUSH R4 ON STACK
6291	040412	010546				MOV	R5, -(SP)	::PUSH R5 ON STACK
6292	040414	017746	140514			MOV	@SWR, -(SP)	::PUSH @SWR ON STACK
6293	040420	010637	040530			MOV	SP, \$SAVR6	::SAVE SP
6294	040424	012737	040436	000024		MOV	\$PWRUP, @PWRVEC	::SET UP VECTOR
6295	040432	000000				HALT		
6296	040434	000776				BR	.-2	::HANG UP

:POWER UP ROUTINE

6300	040436	012737	040524	000024	\$PWRUP:	MOV	\$SILLUP, @PWRVEC	::SET FOR FAST DOWN
6301	040444	013706	040530			MOV	\$SAVR6, SP	::GET SP
6302	040450	005037	040530			CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
6303	040454	005237	040530		15:	INC	\$SAVR6	::WAIT FOR THE INC
6304	040460	001375				BNE	15	::OF WORD
6305	040462	012677	140446			MOV	(SP)+, @SWR	::POP STACK INTO @SWR
6306	040466	012605				MOV	(SP)+, R5	::POP STACK INTO R5
6307	040470	012604				MOV	(SP)+, R4	::POP STACK INTO R4
6308	040472	012603				MOV	(SP)+, R3	::POP STACK INTO R3
6309	040474	012602				MOV	(SP)+, R2	::POP STACK INTO R2
6310	040476	012601				MOV	(SP)+, R1	::POP STACK INTO R1
6311	040500	012600				MOV	(SP)+, R0	::POP STACK INTO R0
6312	040502	012737	040364	000024		MOV	\$PWRDN, @PWRVEC	::SET UP THE POWER DOWN VECTOR
6313	040510	012737	000340	000026		MOV	\$340, @PWRVEC+2	::PRIO:7
6314	040516	104401				TYPE		::REPORT THE POWER FAILURE
6315	040520	040532			SPWRMG:	.WORD	SPOWER	::POWER FAIL MESSAGE POINTER
6316	040522	000002				RTI		
6317	040524	000000			\$SILLUP:	HALT		::THE POWER UP SEQUENCE WAS STARTED
6318	040526	000776				BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
6319	040530	000000			\$SAVR6:	0		::PUT THE SP HERE
6320	040532	005015	047520	042527	SPOWER:	.ASCIZ	<15><12>"POWER"	
6321	040540	000122						

.EVEN

6327	040542	005015	005015	040515	MSG1:	.ASCIZ	<15><12><15><12>*MAINDEC-11-DQKAR-1 11/6X CACHE DIAGNOSTIC*	<15><12><15><1
6328	040550	047111	042504	026503				

6329	040556	030461	042055	045521
6330	040554	040501	030455	020040
6331	040572	030461	033057	020130
6332	040600	040503	044103	020105
6333	040606	044504	043501	047516
6334	040614	052123	041511	005015
6335	040622	005015	000	
6336	040625	015	050012	053517
6337	040632	051105	046440	041501
6338	040640	044510	042516	042040
6339	040646	053517	020116	047101
6340	040654	020104	044124	047105
6341	040662	052440	006520	000012
6342	040670	005015	054524	042520
6343	040676	053440	044510	044103
6344	040704	042040	053105	041511
6345	040712	020105	044123	052517
6346	040720	042114	041040	020105
6347	040726	051525	042105	005015
6348	040734	030012	055440	040503
6349	040742	051122	040511	042507
6350	040750	051040	052105	051125
6351	040756	056516	052455	044516
6352	040764	052502	020123	054105
6353	040772	051105	044503	047523
6354	041000	020122	046450	034067
6355	041006	032465	006451	012
6356	041013	061	055440	040503
6357	041020	051122	040511	042507
6358	041026	051040	052105	051125
6359	041034	056516	041055	051525
6360	041042	052040	051505	042524
6361	041050	020122	047450	042114
6362	041056	006451	012	
6363	041061	062	055440	040503
6364	041066	051122	040511	042507
6365	041074	051040	052105	051125
6366	041102	056516	051055	030113
6367	041110	006465	012	
6368	041113	063	055440	040503
6369	041120	051122	040511	042507
6370	041126	051040	052105	051125
6371	041134	056516	051055	030120
6372	041142	006463	012	
6373	041145	064	055440	040503
6374	041152	051122	040511	042507
6375	041160	051040	052105	051125
6376	041166	056516	052055	030525
6377	041174	006460	006412	000012
6378	041202	005015	020077	044440
6379	041210	053116	046101	042111
6380	041216	042440	052116	054522
6381	041224	020054	051124	020131
6382	041232	043501	044501	006516
6383	041240	000012		
6384	041242	005015	052040	050131

MSG2: .ASCIZ <15><12>*POWER MACHINE DOWN AND THEN UP*<15><12>

MSG3: .ASCII<CR><LF>*TYPE WHICH DEVICE SHOULD BE USED*<CR><LF>

.ASCII<LF>*0 [CARRIAGE RETURN]-UNIBUS EXERCISOR (M7855)*<CR><LF>

.ASCII*1 [CARRIAGE RETURN]-BUS TESTER (OLD)*<CR><LF>

.ASCII*2 [CARRIAGE RETURN]-RK05*<CR><LF>

.ASCII*3 [CARRIAGE RETURN]-RP03*<CR><LF>

.ASCIZ*4 [CARRIAGE RETURN]-TU10*<CR><LF><CR><LF>

MSG4: .ASCIZ<CR><LF>*? INVALID ENTRY, TRY AGAIN*<CR><LF>

MSG5: .ASCIZ<CR><LF>* TYPE THE UBE'S DATA BUFFER ADDRESS*<CR><LF>

6385	041250	020105	044124	020105	
6386	041256	041125	023505	020123	
6387	041264	040504	040524	041040	
6388	041272	043125	042506	020122	
6389	041300	042101	051104	051505	
6390	041306	006523	000012		
6391	041312	005015	042040	053105	MSG6: .ASCII<CR><LF>* DEVICE DOES NOT RESPOND*<CR><LF>
6392	041320	041511	020105	047504	
6393	041326	051505	047040	052117	
6394	041334	051040	051505	047520	
6395	041342	042116	005015		
6396	041346	020040	020040	020040	.ASCIZ* REFERENCE TO IT TRAPS TO 4*<CR><LF>
6397	041354	020040	051040	043105	
6398	041362	051105	047105	042503	
6399	041370	052040	020117	052111	
6400	041376	052040	040522	051520	
6401	041404	052040	020117	006464	
6402	041412	000012			
6403	041414	005015	044127	041511	MSG7: .ASCII<CR><LF>*WHICH DRIVE SHOULD BE USED?*<CR><LF>
6404	041422	020110	051104	053111	
6405	041430	020105	044123	052517	
6406	041436	042114	041040	020105	
6407	041444	051525	042105	006477	
6408	041452	012			
6409	041453	124	050131	020105	.ASCIZ*TYPE 0-7<CARRIAGE RETURN>*<CR><LF>
6410	041460	026460	036067	040503	
6411	041466	051122	040511	042507	
6412	041474	051040	052105	051125	
6413	041502	037116	005015	000	
6414	041507	015	052412	044516	MSG10: .ASCIZ<CR><LF>*UNIT NOT SELECTED PROPERLY*<CR><LF>
6415	041514	020124	047516	020124	
6416	041522	042523	042514	052103	
6417	041530	042105	050040	047522	
6418	041536	042520	046122	006531	
6419	041544	000012			
6420	041546	005015	047125	052111	MSG11: .ASCIZ<CR><LF>*UNIT WRITE LOCK ON, SHOULD BE OFF*<CR><LF>
6421	041554	053440	044522	042524	
6422	041562	046040	041517	020113	
6423	041570	047117	020054	044123	
6424	041576	052517	042114	041040	
6425	041604	020105	043117	006506	
6426	041612	000012			
6427	041614	005015	042504	044526	MSG12: .ASCIZ<CR><LF>*DEVICE ERROR BIT SET*<CR><LF>
6428	041622	042503	042440	051122	
6429	041630	051117	041040	052111	
6430	041636	051440	052105	005015	
6431	041644	000			
6432	041645	015	042012	053105	MSG13: .ASCIZ<CR><LF>*DEVICE RDY BIT DOES NOT SET*<CR><LF>
6433	041652	041511	020105	042122	
6434	041660	020131	044502	020124	
6435	041666	047504	051505	047040	
6436	041674	052117	051440	052105	
6437	041702	005015	000		
6438	041705	015	043012	051125	MSG14: .ASCIZ<CR><LF>*FURTHER NPR DEVICE TESTS ABORTED*<CR><LF>
6439	041712	044124	051105	047040	
6440	041720	051120	042040	053105	

6441	041726	041511	020105	042524	
6442	041734	052123	020123	041101	
6443	041742	051117	042524	006504	
6444	041750	000012			
6445					
6446	041752	051105	047522	035122	EM1: .ASCIZ*ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE*
6447	041760	052440	042516	050130	
6448	041766	041505	042524	020104	
6449	041774	040520	044522	054524	
6450	042002	042440	051122	051117	
6451	042010	044440	020116	040502	
6452	042016	045503	047111	020107	
6453	042024	052123	051117	000105	
6454	042032	051105	047522	035122	EM2: .ASCIZ*ERROR: UNEXPECTED PARITY ERROR IN CACHE TAG*
6455	042040	052440	042516	050130	
6456	042046	041505	042524	020104	
6457	042054	040520	044522	054524	
6458	042062	042440	051122	051117	
6459	042070	044440	020116	040503	
6460	042076	044103	020105	040524	
6461	042104	000107			
6462	042106	051105	047522	035122	EM3: .ASCIZ*ERROR: UNEXPECTED PARITY ERROR IN CACHE DATA LOW*
6463	042114	052440	042516	050130	
6464	042122	041505	042524	020104	
6465	042130	040520	044522	054524	
6466	042136	042440	051122	051117	
6467	042144	044440	020116	040503	
6468	042152	044103	020105	040504	
6469	042160	040524	046040	053517	
6470	042166	000			
6471	042167	105	051122	051117	EM4: .ASCIZ*ERROR: UNEXPECTED PARITY ERROR IN CACHE DATA HIGH*
6472	042174	020072	047125	054105	
6473	042202	042520	052103	042105	
6474	042210	050040	051101	052111	
6475	042216	020131	051105	047522	
6476	042224	020122	047111	041440	
6477	042232	041501	042510	042040	
6478	042240	052101	020101	044510	
6479	042246	044107	000		
6480	042251	106	052101	046101	EM5: .ASCIZ*FATAL ERROR: CACHE CONTROL REG HELD WRONG DATA*
6481	042256	042440	051122	051117	
6482	042264	020072	040503	044103	
6483	042272	020105	047503	052116	
6484	042300	047522	020114	042522	
6485	042306	020107	042510	042114	
6486	042314	053440	047522	043516	
6487	042322	042040	052101	000101	
6488	042330	040506	040524	020114	EM6: .ASCIZ*FATAL ERROR: HIT/MISS REG HELD WRONG DATA*
6489	042336	051105	047522	035122	
6490	042344	044040	052111	046457	
6491	042352	051511	020123	042522	
6492	042360	020107	042510	042114	
6493	042366	053440	047522	043516	
6494	042374	042040	052101	000101	
6495	042402	051105	047522	035122	EM7: .ASCIZ*ERROR: DATA CACHED ON DATOB TO NO 'HIT' ADDR.*
6496	042410	042040	052101	020101	

6497	042416	040503	044103	042105	
6498	042424	047440	020116	040504	
6499	042432	047524	020102	047524	
6500	042440	047040	020117	044047	
6501	042446	052111	020047	042101	
6502	042454	051104	000056		
6503	042460	051105	047522	035122	EM10: .ASCIZ*ERROR: DATA NOT CACHED ON DATOB TO A 'HIT' ADDR.*
6504	042466	042040	052101	020101	
6505	042474	047516	020124	040503	
6506	042502	044103	042105	047440	
6507	042510	020116	040504	047524	
6508	042516	020102	047524	040440	
6509	042524	023440	044510	023524	
6510	042532	040440	042104	027122	
6511	042540	000			
6512	042541	105	051122	051117	EM11: .ASCIZ*ERROR: CACHE DID NOT CONTAIN PROPER DATA ON DATOB*
6513	042546	020072	040503	044103	
6514	042554	020105	044504	020104	
6515	042562	047516	020124	047503	
6516	042570	052116	044501	020116	
6517	042576	051120	050117	051105	
6518	042604	042040	052101	020101	
6519	042612	047117	042040	052101	
6520	042620	041117	000		
6521	042623	105	051122	051117	EM12: .ASCIZ*ERROR: FORCE MISS BIT FAILED TO CAUSE MISS*
6522	042630	020072	047506	041522	
6523	042636	020105	044515	051523	
6524	042644	041040	052111	043040	
6525	042652	044501	042514	020104	
6526	042660	047524	041440	052501	
6527	042666	042523	046440	051511	
6528	042674	000123			
6529	042676	051105	047522	035122	EM14: .ASCIZ*ERROR: ADDRESS COULD NOT BE MADE A 'HIT' AFTER DATO TO IT*
6530	042704	040440	042104	042522	
6531	042712	051523	041440	052517	
6532	042720	042114	047040	052117	
6533	042726	041040	020105	040515	
6534	042734	042504	040440	023440	
6535	042742	044510	023524	040440	
6536	042750	052106	051105	042040	
6537	042756	052101	020117	047524	
6538	042764	044440	000124		
6539	042770	051105	047522	035122	EM16: .ASCIZ*ERROR: UNEXPECTED TRAP TO VECTOR 4*
6540	042776	052440	042516	050130	
6541	043004	041505	042524	020104	
6542	043012	051124	050101	052040	
6543	043020	020117	042526	052103	
6544	043026	051117	032040	000	
6545	043033	105	051122	051117	EM17: .ASCIZ*ERROR: FORCE MISS DID NOT PREVENT CACHE TRACKING*
6546	043040	020072	047506	041522	
6547	043046	020105	044515	051523	
6548	043054	042040	042111	047040	
6549	043062	052117	050040	042522	
6550	043070	042526	052116	041440	
6551	043076	041501	042510	052040	
6552	043104	040522	045503	047111	

6553	043112	000107					
6554	043114	051105	047522	035122	EM20:	.ASCIZ*ERROR: PHYSICAL ADDRESS LINES ERROR*(15)<12>*	ADDRESS HELD WRONG D
6555	043122	050040	054510	044523			
6556	043130	040503	020114	042101			
6557	043136	051104	051505	020123			
6558	043144	044514	042516	020123			
6559	043152	051105	047522	006522			
6560	043160	020012	020040	020040			
6561	043166	020040	040440	042104			
6562	043174	042522	051523	044040			
6563	043202	046105	020104	051127			
6564	043210	047117	020107	040504			
6565	043216	040524	000				
6566	043221	105	051122	051117	EM21:	.ASCIZ*ERROR: TRAP TO VECTOR 4 WHEN TESTING PHYSICAL ADDRESS LINES*	
6567	043226	020072	051124	050101			
6568	043234	052040	020117	042526			
6569	043242	052103	051117	032040			
6570	043250	020040	044127	047105			
6571	043256	052040	051505	044524			
6572	043264	043516	050040	054510			
6573	043272	044523	040503	020114			
6574	043300	042101	051104	051505			
6575	043306	020123	044514	042516			
6576	043314	000123					
6577	043316	051105	047522	035122	EM22:	.ASCIZ*ERROR:TEST OF ADDRESS COMPARATOR FAILED TO BE A MISS WHEN*	
6578	043324	042524	052123	047440			
6579	043332	020106	042101	051104			
6580	043340	051505	020123	047503			
6581	043346	050115	051101	052101			
6582	043354	051117	043040	044501			
6583	043362	042514	020104	047524			
6584	043370	041040	020105	020101			
6585	043376	044515	051523	053440			
6586	043404	042510	000116				
6587	043410	051105	047522	035122	EM23:	.ASCIZ*ERROR:TEST OF ADDRESS COMPARATOR FAILED TO BE A HIT WHEN*	
6588	043416	042524	052123	047440			
6589	043424	020106	042101	051104			
6590	043432	051505	020123	047503			
6591	043440	050115	051101	052101			
6592	043446	051117	043040	044501			
6593	043454	042514	020104	047524			
6594	043462	041040	020105	020101			
6595	043470	044510	020124	044127			
6596	043476	047105	000				
6597	043501	105	051122	051117	EM24:	.ASCIZ*ERROR:FORCE MISS DID NOT INHIBIT PARITY ERRORS*	
6598	043506	043072	051117	042503			
6599	043514	046440	051511	020123			
6600	043522	044504	020104	047516			
6601	043530	020124	047111	044510			
6602	043536	044502	020124	040520			
6603	043544	044522	054524	042440			
6604	043552	051122	051117	000123			
6605	043560	051105	047522	035122	EM25:	.ASCIZ*ERROR:DATO TO I/O ADDRESS WRITTEN IN CACHE*	
6606	043566	040504	047524	052040			
6607	043574	020117	027511	020117			
6608	043602	042101	051104	051505			

6609	043610	020123	051127	052111
6610	043616	042524	020116	047111
6611	043624	041440	041501	042510
6612	043632	000		
6613	043633	105	051122	051117
6614	043640	041472	041501	042510
6615	043646	041440	047117	051124
6616	043654	046117	051040	043505
6617	043662	044040	046105	020104
6618	043670	051127	047117	020107
6619	043676	040504	040524	000
6620	043703	105	051122	051117
6621	043710	052072	051505	020124
6622	043716	043117	052040	043501
6623	043724	050040	051101	052111
6624	043732	020131	042507	042516
6625	043740	040522	047524	027522
6626	043746	044103	041505	042513
6627	043754	020122	040506	046111
6628	043762	042105	005015	
6629	043766	020040	020040	020040
6630	043774	044504	020104	047516
6631	044002	020124	042507	020124
6632	044010	040520	044522	054524
6633	044016	052040	040522	020120
6634	044024	051106	046517	052040
6635	044032	043501	043040	042511
6636	044040	042114	053440	042510
6637	044046	020116	051127	052117
6638	044054	020105	051127	047117
6639	044062	020107	040520	044522
6640	044070	054524	000	
6641	044073	105	051122	051117
6642	044100	052072	051505	020124
6643	044106	043117	052040	043501
6644	044114	050040	051101	052111
6645	044122	020131	042507	042516
6646	044130	040522	047524	027522
6647	044136	044103	041505	042513
6648	044144	020122	040506	046111
6649	044152	042105	005015	
6650	044156	020040	020040	020040
6651	044164	040524	020107	044506
6652	044172	046105	020104	042510
6653	044200	042114	053440	047522
6654	044206	043516	042040	052101
6655	044214	020101	047117	050040
6656	044222	051101	052111	020131
6657	044230	051124	050101	000
6658	044235	105	051122	051117
6659	044242	052072	051505	020124
6660	044250	043117	052040	043501
6661	044256	050040	051101	052111
6662	044264	020131	042507	042516
6663	044272	040522	047524	027522
6664	044300	044103	041505	042513

EM26: .ASCIZ*ERROR:CACHE CONTROL REG HELD WRONG DATA*

EM27: .ASCII*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*(15)<12>

.ASCIZ* DID NOT GET PARITY TRAP FROM TAG FIELD WHEN WROTE WRONG PARITY*

EM31: .ASCII*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*(15)<12>

.ASCIZ* TAG FIELD HELD WRONG DATA ON PARITY TRAP*

EM32: .ASCII*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*(15)<12>

6665	044306	020122	040506	046111	
6666	044314	042105	005015		
6667	044320	020040	020040	020040	.ASCIZ* PARITY ERROR IN HIGH DATA BYTE*
6668	044326	040520	044522	054524	
6669	044334	042440	051122	051117	
6670	044342	044440	020116	044510	
6671	044350	044107	042040	052101	
6672	044356	020101	054502	042524	
6673	044364	000			
6674	044365	105	051122	051117	EM33: .ASCII*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*(15)<12>
6675	044372	052072	051505	020124	
6676	044400	043117	052040	043501	
6677	044406	050040	051101	052111	
6678	044414	020131	042507	042516	
6679	044422	040522	047524	027522	
6680	044430	044103	041505	042513	
6681	044436	020122	040506	046111	
6682	044444	042105	005015		
6683	044450	020040	020040	020040	.ASCIZ* PARITY ERROR IN LOW DATA BYTE*
6684	044456	040520	044522	054524	
6685	044464	042440	051122	051117	
6686	044472	044440	020116	047514	
6687	044500	020127	040504	040524	
6688	044506	041040	052131	000105	
6689	044514	051105	047522	035122	EM34: .ASCII*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*(15)<12>
6690	044522	042524	052123	047440	
6691	044530	020106	040524	020107	
6692	044536	040520	044522	054524	
6693	044544	043440	047105	051105	
6694	044552	052101	051117	041457	
6695	044560	042510	045503	051105	
6696	044566	043040	044501	042514	
6697	044574	006504	012		
6698	044577	040	020040	020040	.ASCIZ* PARITY ERROR IN TAG FIELD*
6699	044604	050040	051101	052111	
6700	044612	020131	051105	047522	
6701	044620	020122	047111	052040	
6702	044626	043501	043040	042511	
6703	044634	042114	000		
6704	044637	105	051122	051117	EM35: .ASCII*ERROR:TEST OF DATA PARITY GENERATOR/CHECKER FAILED*(15)<12>
6705	044644	052072	051505	020124	
6706	044652	043117	042040	052101	
6707	044660	020101	040520	044522	
6708	044666	054524	043440	047105	
6709	044674	051105	052101	051117	
6710	044702	041457	042510	045503	
6711	044710	051105	043040	044501	
6712	044716	042514	006504	012	
6713	044723	040	020040	020040	.ASCIZ* NO PARITY TRAP WHEN WROTE WRONG PARITY*
6714	044730	047040	020117	040520	
6715	044736	044522	054524	052040	
6716	044744	040522	020120	044127	
6717	044752	047105	053440	047522	
6718	044760	042524	053440	047522	
6719	044766	043516	050040	051101	
6720	044774	052111	000131		

6721	045000	051105	047522	035122
6722	045006	042524	052123	047440
6723	045014	020106	040504	040524
6724	045022	050040	051101	052111
6725	045030	020131	042507	042516
6726	045036	040522	047524	027522
6727	045044	044103	041505	042513
6728	045052	020122	040506	046111
6729	045060	042105	005015	
6730	045064	020040	020040	020040
6731	045072	047516	050040	051101
6732	045100	052111	020131	051124
6733	045106	050101	043040	047522
6734	045114	020115	047514	020127
6735	045122	054502	042524	053440
6736	045130	042510	020116	051127
6737	045136	052117	020105	051127
6738	045144	047117	020107	040520
6739	045152	044522	054524	000
6740	045157	105	051122	051117
6741	045164	052072	051505	020124
6742	045172	043117	042040	052101
6743	045200	020101	040520	044522
6744	045206	054524	043440	047105
6745	045214	051105	052101	051117
6746	045222	041457	042510	045503
6747	045230	051105	043040	044501
6748	045236	042514	006504	012
6749	045243	040	020040	020040
6750	045250	047040	020117	040520
6751	045256	044522	054524	052040
6752	045264	040522	020120	051106
6753	045272	046517	044040	043511
6754	045300	020110	054502	042524
6755	045306	053440	042510	020116
6756	045314	051127	052117	020105
6757	045322	051127	047117	020107
6758	045330	040520	044522	054524
6759	045336	000		
6760	045337	105	051122	051117
6761	045344	052072	051505	020124
6762	045352	043117	042040	052101
6763	045360	020101	040520	044522
6764	045366	054524	043440	047105
6765	045374	051105	052101	051117
6766	045402	041457	042510	045503
6767	045410	051105	043040	044501
6768	045416	042514	006504	012
6769	045423	040	020040	020040
6770	045430	050040	051101	052111
6771	045436	020131	051105	047522
6772	045444	020122	047111	046040
6773	045452	053517	041040	052131
6774	045460	000105		
6775	045462	051105	047522	035122
6776	045470	042524	052123	047440

EM36: .ASCII*ERROR:TEST OF DATA PARITY GENERATOR/CHECKER FAILED*<15><12>

.ASCIZ* NO PARITY TRAP FROM LOW BYTE WHEN WROTE WRONG PARITY*

EM37: .ASCII*ERROR:TEST OF DATA PARITY GENERATOR/CHECKER FAILED*<15><12>

.ASCIZ* NO PARITY TRAP FROM HIGH BYTE WHEN WROTE WRONG PARITY*

EM40: .ASCII*ERROR:TEST OF DATA PARITY GENERATOR/CHECKER FAILED*<15><12>

.ASCIZ* PARITY ERROR IN LOW BYTE*

EM41: .ASCII*ERROR:TEST OF DATA PARITY GENERATOR/CHECKER FAILED*<15><12>

6777	045476	020106	040504	040524
6778	045504	050040	051101	052111
6779	045512	020131	042507	042516
6780	045520	040522	047524	027522
6781	045526	044103	041505	042513
6782	045534	020122	040506	046111
6783	045542	042105	005015	
6784	045546	020040	020040	020040
6785	045554	040520	044522	054524
6786	045562	042440	051122	051117
6787	045570	044440	020116	044510
6788	045576	044107	041040	052131
6789	045604	000105		
6790	045606	051105	047522	035122
6791	045614	047516	050040	051101
6792	045622	052111	020131	051124
6793	045630	050101	043040	047522
6794	045636	020115	047514	020103
6795	045644	051127	052111	042524
6796	045652	020116	044527	044124
6797	045660	053440	047522	043516
6798	045666	050040	051101	052111
6799	045674	000131		
6800	045676	051105	047522	035122
6801	045704	040440	042104	042522
6802	045712	051523	041440	052517
6803	045720	042114	047040	052117
6804	045726	041040	020105	040515
6805	045734	042504	040440	044040
6806	045742	052111	000	
6807	045745	105	051122	051117
6808	045752	020072	042101	051104
6809	045760	051505	020123	047516
6810	045766	020124	047111	040526
6811	045774	044514	040504	042524
6812	046002	020104	054502	050040
6813	046010	051101	052111	020131
6814	046016	051124	050101	000
6815	046023	105	051122	051117
6816	046030	020072	040524	020107
6817	046036	040520	044522	054524
6818	046044	042440	051122	051117
6819	046052	053440	042510	020116
6820	046060	042524	052123	047111
6821	046066	020107	040524	020107
6822	046074	020120	044502	000124
6823	046102	051105	047522	035122
6824	046110	046040	053517	041040
6825	046116	052131	020105	040520
6826	046124	044522	054524	042440
6827	046132	051122	051117	053440
6828	046140	042510	020116	042524
6829	046146	052123	047111	020107
6830	046154	040524	020107	040520
6831	046162	044522	054524	041040
6832	046170	052111	000	

.ASCIZ* PARITY ERROR IN HIGH BYTE*

EM42: .ASCIZ*ERROR:NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PARITY*

EM43: .ASCIZ*ERROR: ADDRESS COULD NOT BE MADE A HIT*

EM44: .ASCIZ*ERROR: ADDRESS NOT INVALIDATED BY PARITY TRAP*

EM45: .ASCIZ*ERROR: TAG PARITY ERROR WHEN TESTING TAG P BIT*

EM46: .ASCIZ*ERROR: LOW BYTE PARITY ERROR WHEN TESTING TAG PARITY BIT*

6833	046173	105	051122	051117
6834	046200	020072	044510	044107
6835	046206	041040	052131	020105
6836	046214	040520	044522	054524
6837	046222	042440	051122	051117
6838	046230	053440	042510	020116
6839	046236	042524	052123	047111
6840	046244	020107	040524	020107
6841	046252	020120	044502	000124
6842	046260	051105	047522	035122
6843	046266	044040	043511	020110
6844	046274	054502	042524	050040
6845	046302	051101	052111	020131
6846	046310	051105	047522	020122
6847	046316	044127	047105	052040
6848	046324	051505	044524	043516
6849	046332	042040	052101	020101
6850	046340	020120	044502	000124
6851	046346	051105	047522	035122
6852	046354	046040	053517	041040
6853	046362	052131	020105	040520
6854	046370	044522	054524	042440
6855	046376	051122	051117	053440
6856	046404	042510	020116	042524
6857	046412	052123	047111	020107
6858	046420	040504	040524	050040
6859	046426	041040	052111	000
6860	046433	105	051122	051117
6861	046440	020072	040524	020107
6862	046446	040520	044522	054524
6863	046454	042440	051122	051117
6864	046462	053440	042510	020116
6865	046470	042524	052123	047111
6866	046476	020107	040524	020107
6867	046504	042101	051104	051505
6868	046512	020123	044502	051524
6869	046520	000		
6870	046521	105	051122	051117
6871	046526	020072	047514	020127
6872	046534	054502	042524	050040
6873	046542	051101	052111	020131
6874	046550	051105	047522	020122
6875	046556	044127	047105	052040
6876	046564	051505	044524	043516
6877	046572	052040	043501	040440
6878	046600	042104	042522	051523
6879	046606	041040	052111	000123
6880	046614	051105	047522	035122
6881	046622	044040	043511	020110
6882	046630	054502	042524	050040
6883	046636	051101	052111	020131
6884	046644	051105	047522	020122
6885	046652	044127	047105	052040
6886	046660	051505	044524	043516
6887	046666	052040	043501	040440
6888	046674	042104	042522	051523

EM47: .ASCIZ*ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG P BIT*

EM50: .ASCIZ*ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA P BIT*

EM51: .ASCIZ*ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BIT*

EM52: .ASCIZ*ERROR: TAG PARITY ERROR WHEN TESTING TAG ADDRESS BITS*

EM53: .ASCIZ*ERROR: LOW BYTE PARITY ERROR WHEN TESTING TAG ADDRESS BITS*

EM54: .ASCIZ*ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG ADDRESS BITS*

6889	046702	041040	052111	000123
6890	046710	051105	047522	035122
6891	046716	052040	051505	020124
6892	046724	043117	052040	043501
6893	046732	040440	042104	042522
6894	046740	051523	041040	052111
6895	046746	020123	040506	046111
6896	046754	042105	005015	
6897	046760	020040	020040	020040
6898	046766	040440	042104	042522
6899	046774	051523	041440	052517
6900	047002	042114	047040	052117
6901	047010	041040	020105	040515
6902	047016	042504	040440	044040
6903	047024	052111	000	
6904	047027	105	051122	051117
6905	047034	020072	047514	020127
6906	047042	054502	042524	050040
6907	047050	051101	052111	020131
6908	047056	051105	047522	020122
6909	047064	044127	047105	052040
6910	047072	051505	044524	043516
6911	047100	042040	052101	020101
6912	047106	044506	046105	000104
6913	047114	051105	047522	035122
6914	047122	044040	043511	020110
6915	047130	054502	042524	050040
6916	047136	051101	052111	020131
6917	047144	051105	047522	020122
6918	047152	044127	047105	052040
6919	047160	051505	044524	043516
6920	047166	042040	052101	020101
6921	047174	044506	046105	000104
6922	047202	051105	047522	035122
6923	047210	052040	043501	050040
6924	047216	051101	052111	020131
6925	047224	051105	047522	020122
6926	047232	044127	047105	052040
6927	047240	051505	044524	043516
6928	047246	042040	052101	020101
6929	047254	044506	046105	000104
6930	047262	051105	047522	035122
6931	047270	041440	041501	042510
6932	047276	042040	052101	020101
6933	047304	047514	020103	042510
6934	047312	042114	053440	047522
6935	047320	043516	042040	052101
6936	047326	000101		
6937	047330	051105	047522	035122
6938	047336	042524	052123	047440
6939	047344	020106	051515	020102
6940	047352	042101	051104	051505
6941	047360	020123	040450	030061
6942	047366	020051	047524	041440
6943	047374	041501	042510	042040
6944	047402	052101	020101	044506

EM55: .ASCII*ERROR: TEST OF TAG ADDRESS BITS FAILED*(15)<(12)

.ASCIZ* ADDRESS COULD NOT BE MADE A HIT*

EM56: .ASCIZ*ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD*

EM57: .ASCIZ*ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA FIELD*

EM60: .ASCIZ*ERROR: TAG PARITY ERROR WHEN TESTING DATA FIELD*

EM61: .ASCIZ*ERROR: CACHE DATA LOC HELD WRONG DATA*

EM62: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD FAILED*(15)<(12)

6945	047410	046105	020104	040506
6946	047416	046111	042105	005015
6947	047424	020040	020040	020040
6948	047432	042101	051104	051505
6949	047440	020123	047503	046125
6950	047446	020104	047516	020124
6951	047454	042502	046440	042101
6952	047462	020105	044510	000124
6953	047470	051105	047522	035122
6954	047476	042524	052123	047440
6955	047504	020106	051515	020102
6956	047512	042101	051104	051505
6957	047520	020123	040450	030061
6958	047526	020051	047524	041440
6959	047534	041501	042510	042040
6960	047542	052101	020101	044506
6961	047550	046105	020104	040506
6962	047556	046111	042105	005015
6963	047564	020040	020040	020040
6964	047572	042101	051104	051505
6965	047600	020123	042510	042114
6966	047606	053440	047522	043516
6967	047614	042040	052101	000101
6968	047622	051105	047522	035122
6969	047630	042524	052123	047440
6970	047636	020106	051515	020102
6971	047644	042101	051104	051505
6972	047652	020123	040450	030061
6973	047660	020051	047524	041440
6974	047666	041501	042510	042040
6975	047674	052101	020101	044506
6976	047702	046105	020104	040506
6977	047710	046111	042105	005015
6978	047716	020040	020040	020040
6979	047724	040520	044522	054524
6980	047732	042440	051122	051117
6981	047740	046040	053517	041040
6982	047746	052131	000105	
6983	047752	051105	047522	035122
6984	047760	042524	052123	047440
6985	047766	020106	051515	020102
6986	047774	042101	051104	051505
6987	050002	020123	040450	030061
6988	050010	020051	047524	041440
6989	050016	041501	042510	042040
6990	050024	052101	020101	044506
6991	050032	046105	020104	040506
6992	050040	046111	042105	005015
6993	050046	020040	020040	020040
6994	050054	040520	044522	054524
6995	050062	042440	051122	051117
6996	050070	044040	043511	020110
6997	050076	054502	042524	000
6998	050103	105	051122	051117
6999	050110	052072	051505	020124
7000	050116	043117	046440	041123

.ASCIZ* ADDRESS COULD NOT BE MADE HIT*

EM63: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD FAILED*(15)<12>

.ASCIZ* ADDRESS HELD WRONG DATA*

EM64: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD FAILED*(15)<12>

.ASCIZ* PARITY ERROR LOW BYTE*

EM65: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD FAILED*(15)<12>

.ASCIZ* PARITY ERROR HIGH BYTE*

EM66: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD FAILED*(15)<12>

7001	050124	040440	042104	042522
7002	050132	051523	024040	030501
7003	050140	024460	052040	020117
7004	050146	040503	044103	020105
7005	050154	040504	040524	043040
7006	050162	042511	042114	043040
7007	050170	044501	042514	006504
7008	050176	012		
7009	050177	040	020040	020040
7010	050204	050040	051101	052111
7011	050212	020131	051105	047522
7012	050220	020122	040524	000107
7013	050226	051105	047522	035122
7014	050234	042524	052123	047440
7015	050242	020106	051515	020102
7016	050250	042101	051104	051505
7017	050256	020123	040450	030061
7018	050264	020051	047524	041440
7019	050272	041501	042510	040440
7020	050300	042104	042522	051523
7021	050306	043040	042511	042114
7022	050314	043040	044501	042514
7023	050322	006504	012	
7024	050325	040	020040	020040
7025	050332	040440	042104	042522
7026	050340	051523	041440	052517
7027	050346	042114	047040	052117
7028	050354	041040	020105	040515
7029	050362	042504	040440	044040
7030	050370	052111	000	
7031	050373	105	051122	051117
7032	050400	052072	051505	020124
7033	050406	043117	046440	041123
7034	050414	040440	042104	042522
7035	050422	051523	024040	030501
7036	050430	024460	052040	020117
7037	050436	040503	044103	020105
7038	050444	042101	051104	051505
7039	050452	020123	044506	046105
7040	050460	020104	040506	046111
7041	050466	042105	005015	
7042	050472	020040	020040	020040
7043	050500	040524	020107	040520
7044	050506	044522	054524	042440
7045	050514	051122	051117	000
7046	050521	105	051122	051117
7047	050526	052072	051505	020124
7048	050534	043117	046440	041123
7049	050542	040440	042104	042522
7050	050550	051523	024040	030501
7051	050556	024460	052040	020117
7052	050564	040503	044103	020105
7053	050572	042101	051104	051505
7054	050600	020123	044506	046105
7055	050606	020104	040506	046111
7056	050614	042105	005015	

.ASCIZ* PARITY ERROR TAG*

EM67: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE ADDRESS FIELD FAILED*(15)<12>

.ASCIZ* ADDRESS COULD NOT BE MADE A HIT*

EM70: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE ADDRESS FIELD FAILED*(15)<12>

.ASCIZ* TAG PARITY ERROR*

EM71: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE ADDRESS FIELD FAILED*(15)<12>

7057	050620	020040	020040	020040	.ASCIZ*	LOW BYTE PARITY ERROR*
7058	050626	047514	020127	054502		
7059	050634	042524	050040	051101		
7060	050642	052111	020131	051105		
7061	050650	047522	000122			
7062	050654	051105	047522	035122	EM72:	.ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO CACHE ADDRESS FIELD FAILED*(15)<12>
7063	050662	042524	052123	047440		
7064	050670	020106	051515	020102		
7065	050676	042101	051104	051505		
7066	050704	020123	040450	030061		
7067	050712	020051	047524	041440		
7068	050720	041501	042510	040440		
7069	050726	042104	042522	051523		
7070	050734	043040	042511	042114		
7071	050742	043040	044501	042514		
7072	050750	006504	012			
7073	050753	040	020040	020040	.ASCIZ*	HIGH BYTE PARITY ERROR*
7074	050760	044040	043511	020110		
7075	050766	054502	042524	050040		
7076	050774	051101	052111	020131		
7077	051002	051105	047522	000122		
7078	051010	051105	047522	035122	EM73:	.ASCII*ERROR:DYNAMIC TEST OF CACHE FAILED*(15)<12>
7079	051016	054504	040516	044515		
7080	051024	020103	042524	052123		
7081	051032	047440	020106	040503		
7082	051040	044103	020105	040506		
7083	051046	046111	042105	005015		
7084	051054	020040	020040	020040	.ASCIZ*	LOC HELD WRONG DATA*
7085	051062	047514	020103	042510		
7086	051070	042114	053440	047522		
7087	051076	043516	042040	052101		
7088	051104	000101				
7089	051106	051105	047522	035122	EM74:	.ASCII*ERROR:DYNAMIC TEST OF CACHE FAILED*(15)<12>
7090	051114	054504	040516	044515		
7091	051122	020103	042524	052123		
7092	051130	047440	020106	040503		
7093	051136	044103	020105	040506		
7094	051144	046111	042105	005015		
7095	051152	020040	020040	020040	.ASCIZ*	TRAP TO 10 OCCURRED*
7096	051160	051124	050101	052040		
7097	051166	020117	030061	047440		
7098	051174	041503	051125	042522		
7099	051202	000104				
7100	051204	051105	047522	035122	EM75:	.ASCII*ERROR:DYNAMIC TEST OF CACHE FAILED*(15)<12>
7101	051212	054504	040516	044515		
7102	051220	020103	042524	052123		
7103	051226	047440	020106	040503		
7104	051234	044103	020105	040506		
7105	051242	046111	042105	005015		
7106	051250	020040	020040	020040	.ASCIZ*	LOW BYTE PARITY ERROR*
7107	051256	047514	020127	054502		
7108	051264	042524	050040	051101		
7109	051272	052111	020131	051105		
7110	051300	047522	000122			
7111	051304	051105	047522	035122	EM76:	.ASCII*ERROR:DYNAMIC TEST OF CACHE FAILED*(15)<12>
7112	051312	054504	040516	044515		

7113	051320	020103	042524	052123
7114	051326	047440	020106	040503
7115	051334	044103	020105	040506
7116	051342	046111	042105	005015
7117	051350	020040	020040	020040
7118	051356	044510	044107	041040
7119	051364	052131	020105	040520
7120	051372	044522	054524	042440
7121	051400	051122	051117	000
7122	051405	105	051122	051117
7123	051412	042072	047131	046501
7124	051420	041511	052040	051505
7125	051426	020124	043117	041440
7126	051434	041501	042510	043040
7127	051442	044501	042514	006504
7128	051450	012		
7129	051451	040	020040	020040
7130	051456	052040	043501	050040
7131	051464	051101	052111	020131
7132	051472	051105	047522	000122
7133	051500	051105	047522	035122
7134	051506	040503	044103	020105
7135	051514	047503	052116	047522
7136	051522	020114	042522	020107
7137	051530	047516	020124	047111
7138	051536	052111	040511	044514
7139	051544	042532	020104	054502
7140	051552	050040	053517	051105
7141	051560	043040	044501	000114
7142	051566	051105	047522	035122
7143	051574	047520	042527	020122
7144	051602	050125	043040	044501
7145	051610	042514	020104	047524
7146	051616	044440	053116	046101
7147	051624	042111	052101	020105
7148	051632	040503	044103	000105
7149	051640	051105	047522	035122
7150	051646	042504	044526	042503
7151	051654	042440	051122	051117
7152	051662	041040	052111	051440
7153	051670	052105	053440	042510
7154	051676	020116	047504	047111
7155	051704	020107	050116	026122
7156	051712	042040	052101	020117
7157	051720	047524	040440	042104
7158	051726	042522	051523	000
7159	051733	105	051122	051117
7160	051740	041472	041501	042510
7161	051746	046040	041517	052101
7162	051754	047511	020116	047516
7163	051762	020124	047111	040526
7164	051770	044514	040504	042524
7165	051776	020104	054502	047040
7166	052004	051120	020054	040504
7167	052012	047524	052040	020117
7168	052020	042101	051104	051505

.ASCIZ* HIGH BYTE PARITY ERROR*

EM77: .ASCII*ERROR:DYNAMIC TEST OF CACHE FAILED*(15)<12>

.ASCIZ* TAG PARITY ERROR*

EM101: .ASCIZ*ERROR:CACHE CONTROL REG NOT INITIALIZED BY POWER FAIL*

EM102: .ASCIZ*ERROR:POWER UP FAILED TO INVALIDATE CACHE*

EM103: .ASCIZ*ERROR:DEVICE ERROR BIT SET WHEN DOING NPR, DATO TO ADDRESS*

EM104: .ASCIZ*ERROR:CACHE LOCATION NOT INVALIDATED BY NPR, DATO TO ADDRESS*

7169	052026	000123			
7170	052030	051105	047522	035122	EM105: .ASCII*ERROR:DID NOT GET PARITY TRAP WHEN DID NPR, DATO TO ADDRESS*(CR)<LF>
7171	052036	044504	020104	047516	
7172	052044	020124	042507	020124	
7173	052052	040520	044522	054524	
7174	052060	052040	040522	020120	
7175	052066	044127	047105	042040	
7176	052074	042111	047040	051120	
7177	052102	020054	040504	047524	
7178	052110	052040	020117	042101	
7179	052116	051104	051505	006523	
7180	052124	012			
7181	052125	040	020040	020040	.ASCIZ* WRITTEN WITH WRONG PARITY*
7182	052132	053440	044522	052124	
7183	052140	047105	053440	052111	
7184	052146	020110	051127	047117	
7185	052154	020107	040520	044522	
7186	052162	054524	000		
7187	052165	105	051122	051117	EM107: .ASCIZ*ERROR:CACHE DID NOT TRACK WHEN FORCE MISS ON*
7188	052172	041472	041501	042510	
7189	052200	042040	042111	047040	
7190	052206	052117	052040	040522	
7191	052214	045503	053440	042510	
7192	052222	020116	047506	041522	
7193	052230	020105	044515	051523	
7194	052236	047440	000116		
7195	052242	051105	047522	035122	EM110: .ASCIZ*ERROR:RETRY TO BACKING STORE NOT DONE ON CACHE PARITY TRAP*
7196	052250	042522	051124	020131	
7197	052256	047524	041040	041501	
7198	052264	044513	043516	051440	
7199	052272	047524	042522	047040	
7200	052300	052117	042040	047117	
7201	052306	020105	047117	041440	
7202	052314	041501	042510	050040	
7203	052322	051101	052111	020131	
7204	052330	051124	050101	000	
7205	052335	105	051122	051117	EM111: .ASCII*ERROR:TEST OF VALID BIT FAILED*(CR)<LF>
7206	052342	052072	051505	020124	
7207	052350	043117	053040	046101	
7208	052356	042111	041040	052111	
7209	052364	043040	044501	042514	
7210	052372	006504	012		
7211	052375	040	020040	020040	.ASCIZ* LOC COULD NOT BE MADE A HIT*
7212	052402	046040	041517	041440	
7213	052410	052517	042114	047040	
7214	052416	052117	041040	020105	
7215	052424	040515	042504	040440	
7216	052432	044040	052111	000	
7217	052437	105	051122	051117	EM112: .ASCII*ERROR:TEST OF VALID BIT FAILED*(CR)<LF>
7218	052444	052072	051505	020124	
7219	052452	043117	053040	046101	
7220	052460	042111	041040	052111	
7221	052466	043040	044501	042514	
7222	052474	006504	012		
7223	052477	040	020040	020040	.ASCIZ* LOC NOT INVALIDATED BY PARITY TRAP*
7224	052504	046040	041517	047040	

7225	052512	052117	044440	053116
7226	052520	046101	042111	052101
7227	052526	042105	041040	020131
7228	052534	040520	044522	054524
7229	052542	052040	040522	000120
7230	052550	051105	047522	035122
7231	052556	042101	051104	051505
7232	052564	020123	047516	020124
7233	052572	047111	040526	044514
7234	052600	040504	042524	020104
7235	052606	054502	041440	047117
7236	052614	047523	042514	051440
7237	052622	042527	050105	005015
7238	052630	000		
7239	052631	105	051122	051117
7240	052636	046072	041517	053440
7241	052644	044522	052124	047105
7242	052652	053440	052111	020110
7243	052660	051127	047117	020107
7244	052666	040520	044522	054524
7245	052674	047040	052117	044440
7246	052702	053116	046101	042111
7247	052710	052101	042105	053040
7248	052716	040511	047040	051120
7249	052724	042040	052101	000117
7250	052732	051105	047522	035122
7251	052740	040520	044522	054524
7252	052746	052040	040522	020120
7253	052754	044127	046111	020105
7254	052762	042524	052123	047111
7255	052770	020107	047514	020103
7256	052776	051127	052111	042524
7257	053004	020116	044527	044124
7258	053012	053440	047522	043516
7259	053020	050040	051101	052111
7260	053026	006531	012	
7261	053031	040	020040	020040
7262	053036	040440	042116	044440
7263	053044	053116	046101	042111
7264	053052	052101	047111	020107
7265	053060	052111	053040	040511
7266	053066	047040	051120	042040
7267	053074	052101	000117	
7268	053100	051105	047522	035122
7269	053106	040503	044103	020105
7270	053114	046101	047514	040503
7271	053122	042524	020104	052504
7272	053130	044522	043516	047440
7273	053136	042104	040440	042104
7274	053144	042522	051523	052040
7275	053152	040522	000120	
7276	053156	051105	047522	035122
7277	053164	040503	044103	020105
7278	053172	046101	047514	040503
7279	053200	042524	020104	052504
7280	053206	044522	043516	051040

EM113: .ASCIZ*ERROR:ADDRESS NOT INVALIDATED BY CONSOLE SWEEP*<CR><LF>

EM114: .ASCIZ*ERROR:LOC WRITTEN WITH WRONG PARITY NOT INVALIDATED VIA NPR DATO*

EM115: .ASCII*ERROR:PARITY TRAP WHILE TESTING LOC WRITTEN WITH WRONG PARITY*<CR><LF>

.ASCIZ* AND INVALIDATING IT VIA NPR DATO*

EM116: .ASCIZ*ERROR:CACHE ALLOCATED DURING ODD ADDRESS TRAP*

EM117: .ASCIZ*ERROR:CACHE ALLOCATED DURING RED ZONE TRAP*

7281	053214	042105	055040	047117
7282	053222	020105	051124	050101
7283	053230	000		
7284	053231	105	051122	051117
7285	053236	041472	041501	042510
7286	053244	040440	046114	041517
7287	053252	052101	042105	042040
7288	053260	051125	047111	020107
7289	053266	052113	040440	047502
7290	053274	052122	000	
7291	053277	105	051122	051117
7292	053304	052072	051505	020124
7293	053312	043117	046440	041123
7294	053320	040440	042104	042522
7295	053326	051523	024040	030501
7296	053334	024460	052040	020117
7297	053342	040526	044514	020104
7298	053350	044502	020124	040506
7299	053356	046111	042105	005015
7300	053364	020040	020040	020040
7301	053372	047514	020103	047516
7302	053400	020124	047111	040526
7303	053406	044514	040504	042524
7304	053414	000104		
7305	053416	051105	047522	035122
7306	053424	042524	052123	047440
7307	053432	020106	051515	020102
7308	053440	042101	051104	051505
7309	053446	020123	040450	030061
7310	053454	020051	047524	053040
7311	053462	046101	042111	041040
7312	053470	052111	043040	044501
7313	053476	042514	006504	012
7314	053503	011	020040	020040
7315	053510	050040	051101	052111
7316	053516	020131	051105	047522
7317	053524	020122	040524	000107
7318	053532	051105	047522	035122
7319	053540	042524	052123	047440
7320	053546	020106	051515	020102
7321	053554	042101	051104	051505
7322	053562	020123	040450	030061
7323	053570	020051	047524	053040
7324	053576	046101	042111	041040
7325	053604	052111	043040	044501
7326	053612	042514	006504	012
7327	053617	011	020040	020040
7328	053624	050040	051101	052111
7329	053632	020131	051105	047522
7330	053640	020122	047514	020127
7331	053646	054502	042524	000
7332	053653	105	051122	051117
7333	053660	052072	051505	020124
7334	053666	043117	046440	041123
7335	053674	040440	042104	042522
7336	053702	051523	024040	030501

EM120: .ASCIZ*ERROR:CACHE ALLOCATED DURING KT ABORT*

EM121: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED*<CR><LF>

.ASCIZ* LOC NOT INVALIDATED*

EM122: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED*<CR><LF>

.ASCIZ* PARITY ERROR TAG*

EM123: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED*<CR><LF>

.ASCIZ* PARITY ERROR LOW BYTE*

EM124: .ASCII*ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED*<CR><LF>

7337	053710	024460	052040	020117	
7338	053716	040526	044514	020104	
7339	053724	044502	020124	040506	
7340	053732	046111	042105	005015	
7341	053740	020011	020040	020040	.ASCIZ* PARITY ERROR HIGH BYTE*
7342	053746	040520	044522	054524	
7343	053754	042440	051122	051117	
7344	053762	044040	043511	020110	
7345	053770	054502	042524	000	
7346					
7347	053775	120	036503	020040	DH1: .ASCIZ*PC= / P ADDH/ P ADDL/ PC OF PE*
7348	054002	027440	050040	040440	
7349	054010	042104	027510	050040	
7350	054016	040440	042104	027514	
7351	054024	050040	020103	043117	
7352	054032	050040	000105		
7353	054036	041520	020075	020040	DH2: .ASCIZ*PC= / P ADDH/ P ADDL/ DATA/ PC OF PE*
7354	054044	020057	020120	042101	
7355	054052	044104	020057	020120	
7356	054060	042101	046104	020057	
7357	054066	040504	040524	020057	
7358	054074	041520	047440	020106	
7359	054102	042520	000		
7360	054105	120	036503	020040	DH5: .ASCIZ*PC= / DATA IS/DATA SHOULD BE*
7361	054112	027440	042040	052101	
7362	054120	020101	051511	042057	
7363	054126	052101	020101	044123	
7364	054134	052517	042114	041040	
7365	054142	000105			
7366	054144	041520	020075	020040	DH6: .ASCIZ*PC= / DATA IS/DATA EXPECTED SET (0= DON'T CARE)*
7367	054152	020057	040504	040524	
7368	054160	044440	027523	040504	
7369	054166	040524	042440	050130	
7370	054174	041505	042524	020104	
7371	054202	042523	020124	030050	
7372	054210	020075	047504	023516	
7373	054216	020124	040503	042522	
7374	054224	000051			
7375	054226	041520	020075	020040	DH7: .ASCIZ*PC= / P ADDH/ P ADDL*
7376	054234	020057	050040	040440	
7377	054242	042104	027510	050040	
7378	054250	040440	042104	000114	
7379	054256	041520	020075	020040	DH11: .ASCIZ*PC= / P ADDH/ P ADDL/ DATA IS/ DATA SHOULD BE*
7380	054264	020057	020120	042101	
7381	054272	044104	020057	020120	
7382	054300	042101	046104	020057	
7383	054306	040504	040524	044440	
7384	054314	027523	042040	052101	
7385	054322	020101	044123	052517	
7386	054330	042114	041040	000105	
7387	054336	041520	020075	020040	DH12: .ASCIZ*PC= / (CCR) / P ADDH/ P ADDL*
7388	054344	020057	041450	051103	
7389	054352	020051	027440	050040	
7390	054360	040440	042104	027510	
7391	054366	050040	040440	042104	
7392	054374	000114			

7393	054376	041520	020075	020040	DH16:	.ASCIZ*PC=	/(CER)/PC WHEN TRAPPED*
7394	054404	024057	042503	024522			
7395	054412	050057	020103	044127			
7396	054420	047105	052040	040522			
7397	054426	050120	042105	000			
7398	054433	120	036503	020040	DH21:	.ASCIZ*PC=	/ P ADDH/ P ADDL/ (EREG)*
7399	054440	027440	050040	040440			
7400	054446	042104	027510	050040			
7401	054454	040440	042104	027514			
7402	054462	024040	051105	043505			
7403	054470	000051					
7404	054472	041520	020075	020040	DH22:	.ASCIZ*PC=	/ P ADDH/ P ADDL/ TAG FIELD=*
7405	054500	020057	020120	042101			
7406	054506	044104	020057	020120			
7407	054514	042101	046104	020057			
7408	054522	040524	020107	044506			
7409	054530	046105	036504	000			
7410	054535	120	036503	020040	DH27:	.ASCIZ*PC=	/ P ADDH/ P ADDL/ TAG SHOULD=*
7411	054542	027440	050040	040440			
7412	054550	042104	027510	050040			
7413	054556	040440	042104	027514			
7414	054564	052040	043501	051440			
7415	054572	047510	046125	036504			
7416	054600	000					
7417	054601	120	036503	020040	DH30:	.ASCIZ*PC=	/ P ADDH/ P ADDL/ (TAG)/ (TAG) SHOULD BE*
7418	054606	027440	050040	040440			
7419	054614	042104	027510	050040			
7420	054622	040440	042104	027514			
7421	054630	024040	040524	024507			
7422	054636	020057	052050	043501			
7423	054644	020051	044123	052517			
7424	054652	042114	041040	000105			
7425	054660	041520	020075	020040	DH35:	.ASCIZ*PC=	/ P ADDH/ P ADDL/ DATA SHOULD=*
7426	054666	020057	020120	042101			
7427	054674	044104	020057	020120			
7428	054702	042101	046104	020057			
7429	054710	040504	040524	051440			
7430	054716	047510	046125	036504			
7431	054724	000					
7432	054725	120	036503	020040	DH45:	.ASCIZ*PC=	/ P ADDH/ P ADDL/ DATA=*
7433	054732	027440	050040	040440			
7434	054740	042104	027510	050040			
7435	054746	040440	042104	027514			
7436	054754	042040	052101	036501			
7437	054762	000					
7438	054763	120	036503	020040	DH100:	.ASCIZ*PC=	/DATA=*
7439	054770	027440	040504	040524			
7440	054776	000075					
7441	055000	041520	000075		DH107:	.ASCIZ*PC=*	
7442						.EVEN	
7443	055004	001116	001160	001162	DT1:	.WORD	SERRPC, SREG1, SREG2, SREG3, SREG4, 0
7444	055012	001164	001166	000000			
7445	055020	001116	001160	001162	DT5:	.WORD	SERRPC, SREG1, SREG2, 0
7446	055026	000000					
7447	055030	001116	001160	001162	DT12:	.WORD	SERRPC, SREG1, SREG2, SREG3, 0
7448	055036	001164	000000				

```

7449 055042 001116 001160 000000
7450 055050 001116 001160 001162
7451 055056 001166 000000
7452 055062 001116 001160 000000
7453 055070 001116 000000
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472 055074
7473
7474
7475 055074 041752
7476 055076 053775
7477 055100 055030
7478 055102 000000
7479
7480 055104 042032
7481 055106 054036
7482 055110 055004
7483 055112 000000
7484
7485 055114 042106
7486 055116 054036
7487 055120 055004
7488 055122 000000
7489
7490 055124 042167
7491 055126 054036
7492 055130 055004
7493 055132 000000
7494
7495 055134 042251
7496 055136 054105
7497 055140 055020
7498 055142 000000
7499
7500 055144 042330
7501 055146 054105
7502 055150 055020
7503 055152 000000
7504
    
```

```

DT16: .WORD $ERRPC,$REG1,0
DT35: .WORD $ERRPC,$REG1,$REG2,$REG4,0
DT100: .WORD $ERRPC,$REG1,0
DT107: .WORD $ERRPC,0
    
```

```

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

.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 SITEMB, THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
    
```

```

;#NOTE1: IF SITEMB 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
    
```

\$ERRTB:

```

;ITEM 1
EM1          ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
DH1          ;PC=/P ADDH /P ADDL /PC OF PE
DT12        ;$ERRPC,$REG1,$REG2,$REG3
0

;ITEM 2
EM2          ;ERROR:UNEXPECTED PARITY ERROR IN CACHE TAG
DH2          ;PC= /P ADDH /P ADDL /DATA /PC OF PE
DT1         ;$ERRPC,$REG1,$REG2,$REG3,$REG4
0

;ITEM 3
EM3          ;ERROR:UNEXPECTED PARITY ERROR IN CACHE DATA LOW
DH2          ;PC= /P ADDH /P ADDL /DATA /PC OF PE
DT1         ;$ERRPC,$REG1,$REG2,$REG3,$REG4
0

;ITEM 4
EM4          ;ERROR:UNEXPECTED PARITY ERROR IN CACHE DATA HIGH
DH2          ;PC= /P ADDH /P ADDL /DATA /PC OF PE
DT1         ;$ERRPC,$REG1,$REG2,$REG3,$REG4
0

;ITEM 5
EM5          ;FATAL ERROR: CACHE CONTROL REG HELD WRONG DATA
DH5          ;PC=/DATA IS /DATA SHOULD BE
DT5         ;$ERRPC,$REG1,$REG2
0

;ITEM 6
EM6          ;FATAL ERROR: HIT MISS REG HELD WRONG DATA
DH5          ;PC=/DATA IS/DATA SHOULD BE
DT5         ;$ERRPC,$REG1,$REG2
0

;ITEM 7
    
```

7505	055154	042402	EM7	; ERROR: DATA CACHED ON DATOB TO NO 'HIT' ADDR..
7506	055156	054226	DH7	; PC=/P ADDH/P ADDL
7507	055160	055020	DT5	; SERRPC, \$REG1, \$REG2
7508	055162	000000	0	
7509			; ITEM 10	
7510	055164	042460	EM10	; ERROR: DATA NOT CACHED ON DATOB TO A HIT LOC.
7511	055166	054226	DH7	; PC=/P ADDH/P ADDL
7512	055170	055020	DT5	; SERRPC, \$REG1, \$REG2
7513	055172	000000	0	
7514			; ITEM 11	
7515	055174	042541	EM11	; ERROR: CACHE DID NOT CONTAIN PROPER DATA ON DATOB
7516	055176	054256	DH11	; PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7517	055200	055004	DT1	; SERRPC, \$REG1, \$REG2, \$REG3, \$REG4
7518	055202	000000	0	
7519			; ITEM 12	
7520	055204	042623	EM12	; ERROR: FORCE MISS BIT FAILED TO CAUSE MISS
7521	055206	054336	DH12	; PC=/(CCR)/P ADDH/P ADDL
7522	055210	055030	DT12	; SERRPC, \$REG1, \$REG2, \$REG3
7523	055212	000000	0	
7524			; ITEM 13	
7525	055214	042330	EM6	; FATAL ERROR: HIT MISS REG HELD WRONG DATA
7526	055216	054144	DH6	; PC=/DATA IS/DATA EXPECTED SET (0= DON'T CARE)
7527	055220	055020	DT5	; SERRPC, \$REG1, \$REG2
7528	055222	000000	0	
7529			; ITEM 14	
7530	055224	042676	EM14	; ERROR: ADDRESS COULD NOT BE MADE A HIT AFTER DATO TO IT
7531	055226	054226	DH7	; PC=/P ADDH/P ADDL
7532	055230	055020	DT5	; SERRPC, \$REG1, \$REG2
7533	055232	000000	0	
7534			; ITEM 15	
7535	055234	000000	0	
7536	055236	000000	0	
7537	055240	000000	0	
7538	055242	000000	0	
7539			; ITEM 16	
7540	055244	042770	EM16	; ERROR: UNEXPECTED TRAP TO VECTOR 4
7541	055246	054376	DH16	; PC=/(CER)/PC WHEN TRAPPED
7542	055250	055020	DT5	; SERRPC, \$REG1, \$REG2
7543	055252	000000	0	
7544			; ITEM 17	
7545	055254	043033	EM17	; ERROR: FORCE MISS DID NOT PREVENT CACHE TRACKING
7546	055256	054226	DH7	; PC=/P ADDH/P ADDL
7547	055260	055020	DT5	; SERRPC, \$REG1, \$REG2
7548	055262	000000	0	
7549			; ITEM 20	
7550	055264	043114	EM20	; ERROR: PHYSICAL ADDRESS LINES ERROR
7551				; ADDR. HELD WRONG DATA
7552	055266	054256	DH11	; PC=/P ADDH/P ADDL/DATA IS/ DATA SHOULD BE
7553	055270	055004	DT1	; SERRPC, \$REG1, \$REG2, \$REG3
7554	055272	000000	0	
7555			; ITEM 21	
7556	055274	043221	EM21	; ERROR: TRAP TO VECTOR 4 WHEN TESTING P.A. LINES
7557	055276	054433	DH21	; PC=/P ADDH/P ADDL/(EREG)
7558	055300	055030	DT12	; SERRPC, \$REG1, \$REG2, \$REG3
7559	055302	000000	0	
7560			; ITEM 22	

7561	055304	043316	EM22	;ERROR: TEST OF ADDR. COMPARATOR FAILED TO BE A MISS
7562	055306	054472	DH22	;PC=/P ADDH/P ADDL/TAG FIELD=
7563	055310	055030	DT12	;SERRPC, \$REG1, \$REG2, \$REG3
7564	055312	000000	0	
7565			; ITEM 23	
7566	055314	043410	EM23	;ERROR: TEST OF ADDR. COMPARATOR FAILED TO BE A HIT
7567	055316	054472	DH22	;PC=/P ADDH/P ADDL/TAG FIELD=
7568	055320	055030	DT12	;SERRPC, \$REG1, \$REG2, \$REG3
7569	055322	000000	0	
7570			; ITEM 24	
7571	055324	043501	EM24	;ERROR: FORCE MISS DID NOT INHIBIT PARITY ERRORS
7572	055326	000000	0	
7573	055330	000000	0	
7574	055332	000000	0	
7575			; ITEM 25	
7576	055334	043560	EM25	;ERROR: DATO TO I/O ADDRESS WRITTEN IN CACHE
7577	055336	054226	DH7	;PC=/P ADDH/P ADDL
7578	055340	055020	DT5	;SERRPC, \$REG1, \$REG2
7579	055342	000000	0	
7580			; ITEM 26	
7581	055344	043633	EM26	;ERROR: CACHE CONTROL REG HOLD WRONG DATA
7582	055346	054105	DH5	;PC=/DATA IS /DATA SHOULD BE
7583	055350	055020	DT5	;SERRPC, \$REG1, \$REG2
7584	055352	000000	0	
7585			; ITEM 27	
7586	055354	043703	EM27	;ERROR: TEST OF TAG PARITY GEN/CKER FAILED
7587				; NO TAG PARITY TRAP WHEN WWP
7588	055356	054535	DH27	;PC=/P ADDH/P ADDL/(TAG) SHOULD BE
7589	055360	055030	DT12	;SERRPC, \$REG1, \$REG2, \$REG3
7590	055362	000000	0	
7591			; ITEM 30	
7592	055364	043703	EM27	;ERROR: TEST OF TAG PARITY GEN/CKER FAILED
7593				; NO TAG PARITY TRAP WHEN WWP
7594	055366	054601	DH30	;PC=/P ADDH/P ADDL/(TAG)/(TAG) SHOULD BE
7595	055370	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3, \$REG4
7596	055372	000000	0	
7597			; ITEM 31	
7598	055374	044073	EM31	;ERROR: TEST OF TAG PARITY GEN/CKER FAILED
7599				(TAG) BAD ON PTRAP
7600	055376	054601	DH30	;PC=/P ADDH/P ADDL/(TAG)/(TAG) SHOULD BE
7601	055400	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7602	055402	000000	0	
7603			; ITEM 32	
7604	055404	044235	EM32	;ERROR: TEST OF TAG PARITY GEN/CKER FAILED
7605				PARITY ERROR HIGH DATA BYTE
7606	055406	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/ DATA SHOULD BE
7607	055410	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7608	055412	000000	0	
7609			; ITEM 33	
7610	055414	044365	EM33	;ERROR: TEST OF TAG PARITY GEN/CKER FAILED
7611				PARITY ERROR LOW DATA BYTE
7612	055416	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7613	055420	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7614	055422	000000	0	
7615			; ITEM 34	
7616	055424	044514	EM34	;ERROR: TEST OF TAG PARITY GEN/CKER FAILED

7673	055546	054725	DH45	;PC=/P ADDH/P ADDL/DATA=
7674	055550	055030	DT12	;SERRPC, \$REG1, \$REG2, \$REG3
7675	055552	000000	0	
7676			;ITEM 47	
7677	055554	046173	EM47	;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG P BIT
7678	055556	054725	DH45	;PC=/P ADDH/P ADDL/DATA=
7679	055560	055030	DT12	;SERRPC, \$REG1, \$REG2, \$REG3
7680	055562	000000	0	
7681			;ITEM 50	
7682	055564	046260	EM50	;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA P BIT
7683	055566	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7684	055570	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7685	055572	000000	0	
7686			;ITEM 51	
7687	055574	046346	EM51	;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BIT
7688	055576	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7689	055600	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7690	055602	000000	0	
7691			;ITEM 52	
7692	055604	046433	EM52	;ERROR: TAG PARITY ERROR WHEN TESTING TAG ADDR. BITS
7693	055606	054601	DH30	;PC=/P ADDH/P ADDL/(TAG)/(TAG) SHOULD BE
7694	055610	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7695	055612	000000	0	
7696			;ITEM 53	
7697	055614	046521	EM53	;ERROR: LOW BYTE PAR. ERROR WHEN TESTING TAG ADDR. BITS
7698	055616	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7699	055620	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7700	055622	000000	0	
7701			;ITEM 54	
7702	055624	046614	EM54	;ERROR: HIGH BYTE PAR. ERROR WHEN TESTING TAG ADDR. BITS
7703	055626	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7704	055630	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7705	055632	000000	0	
7706			;ITEM 55	
7707	055634	046710	EM55	;ERROR: TEST OF TAG ADDR. BITS FAILED
7708				;ADDR. COULD NOT BE MADE A HIT
7709	055636	054535	DH27	;PC=/P ADDH/P ADDL/(TAG) SHOULD=
7710	055640	055030	DT12	;SERRPC, \$REG1, \$REG2, \$REG3
7711	055642	000000	0	
7712			;ITEM 56	
7713	055644	047027	EM56	;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD
7714	055646	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7715	055650	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7716	055652	000000	0	
7717			;ITEM 57	
7718	055654	047114	EM57	;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA FIELD
7719	055656	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7720	055660	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7721	055662	000000	0	
7722			;ITEM 60	
7723	055664	047202	EM60	;ERROR: TAG PARITY ERROR WHEN TESTING DATA FIELD
7724	055666	054601	DH30	;PC=/P ADDH/P ADDL/(TAG)/(TAG) SHOULD BE
7725	055670	055004	DT1	;SERRPC, \$REG1, \$REG2, \$REG3
7726	055672	000000	0	
7727			;ITEM 61	
7728	055674	047262	EM61	;ERROR: CACHE DATA LOC HELD WRONG DATA

7729	055676	054256	DH11	:PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7730	055700	055004	DT1	:SERRPC, SREG1, SREG2, SREG3
7731	055702	000000	0	
7732			; ITEM 62	
7733	055704	047330	EM62	:ERROR: TEST OF MSB ADDRESS (A10) TO DATA FIELD FAILED
7734				: ADDRESS COULD NOT BE MADE A HIT
7735	055706	054226	DH7	:PC=/P ADDH/P ADDL
7736	055710	055020	DT5	:SERRPC, SREG1, SREG2
7737	055712	000000	0	
7738			; ITEM 63	
7739	055714	047470	EM63	:ERROR: TEST OF MSB ADDRESS (A10) TO DATA FIELD FAILED
7740				: ADDRESS HELD WRONG DATA
7741	055716	054256	DH11	:PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7742	055720	055004	DT1	:SERRPC, SREG1, SREG2, SREG3
7743	055722	000000	0	
7744			; ITEM 64	
7745	055724	047622	EM64	:ERROR: TEST OF MSB ADDRESS (A10) TO DATA FIELD FAILED
7746				: PARITY ERROR LOW BYTE
7747	055726	054725	DH45	:PC=/P ADDH/P ADDL/DATA=
7748	055730	055030	DT12	:SERRPC, SREG1, SREG2, SREG3
7749	055732	000000	0	
7750			; ITEM 65	
7751	055734	047752	EM65	:ERROR: TEST OF MSB ADDRESS (A10) TO DATA FIELD FAILED
7752				: PARITY ERROR HIGH BYTE
7753	055736	054725	DH45	:PC=/P ADDH/P ADDL/DATA=
7754	055740	055030	DT12	:SERRPC, SREG1, SREG2, SREG3
7755	055742	000000	0	
7756			; ITEM 66	
7757	055744	050103	EM66	:ERROR: TEST OF MSB ADDRESS (A10) TO DATA FIELD FAILED
7758				: PARITY ERROR TAG
7759	055746	054725	DH45	:PC=/P ADDH/P ADDL/DATA=
7760	055750	055030	DT12	:SERRPC, SREG1, SREG2, SREG3
7761	055752	000000	0	
7762			; ITEM 67	
7763	055754	050226	EM67	:ERROR: TEST OF MSB ADDRESS (A10) TO TAG FIELD FAILED
7764				: ADDRESS COULD NOT BE MADE A HIT
7765	055756	054226	DH7	:PC=/P ADDH/P ADDL
7766	055760	055020	DT5	:SERRPC, SREG1, SREG2
7767	055762	000000	0	
7768			; ITEM 70	
7769	055764	050373	EM70	:ERROR: TEST OF MSB ADDRESS (A10) TO TAG FIELD FAILED
7770				: TAG PARITY ERROR
7771	055766	054472	DH22	:PC=/P ADDH/P ADDL/TAG FIELD=
7772	055770	055030	DT12	:SERRPC, SREG1, SREG2, SREG3
7773	055772	000000	0	
7774			; ITEM 71	
7775	055774	050521	EM71	:ERROR: TEST OF MSB ADDRESS (A10) TO TAG FIELD FAILED
7776				: LOW BYTE PARITY ERROR
7777	055776	054725	DH45	:PC=/P ADDH/P ADDL/DATA=
7778	056000	055030	DT12	:SERRPC, SREG1, SREG2, SREG3
7779	056002	000000	0	
7780			; ITEM 72	
7781	056004	050654	EM72	:ERROR: TEST OF MSB ADDRESS (A10) TO TAG FIELD FAILED
7782				: HIGH BYTE PARITY ERROR
7783	056006	054725	DH45	:PC=/P ADDH/P ADDL/DATA=
7784	056010	055030	DT12	:SERRPC, SREG1, SREG2, SREG3

7785	056012	000000	0		
7786			; ITEM 73	EM73	; ERROR: DYNAMIC TEST OF CACHE FAILED
7787	056014	051010			: LOC HELD WRONG DATA
7788				DH11	: PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7789	056016	054256		DT1	: SERRPC, SREG1, SREG2, SREG3
7790	056020	055004		0	
7791	056022	000000	; ITEM 74	EM74	; ERROR: DYNAMIC TEST OF CACHE FAILED
7792					: TRAP TO 10 OCCURRED
7793	056024	051106		DH1	: PC=/P ADDH/P ADDL/PC OF PE
7794				DT1	: SERRPC, SREG1, SREG2, SREG3
7795	056026	053775		0	
7796	056030	055004	; ITEM 75	EM75	; ERROR: DYNAMIC TEST OF CACHE FAILED
7797	056032	000000			: LOW BYTE PARITY ERROR
7798				DH45	: PC=/P ADDH/P ADDL/DATA=
7799	056034	051204		DT12	: SERRPC, SREG1, SREG2, SREG3
7800				0	
7801	056036	054725	; ITEM 76	EM76	; ERROR: DYNAMIC TEST OF CACHE FAILED
7802	056040	055030			: HIGH BYTE PARITY ERROR
7803	056042	000000		DH45	: PC=/P ADDH/P ADDL/DATA=
7804				DT12	: SERRPC, SREG1, SREG2, SREG3
7805	056044	051304		0	
7806			; ITEM 77	EM77	; ERROR: DYNAMIC TEST OF CACHE FAILED
7807	056046	054725			: TAG PARITY ERROR
7808	056050	055030		DH22	: PC=/P ADDH/P ADDL/TAG FIELD=
7809	056052	000000		DT12	: SERRPC, SREG1, SREG2, SREG3
7810				0	
7811	056054	051405	; ITEM 100	0	
7812				0	
7813	056056	054472		0	
7814	056060	055030		0	
7815	056062	000000	; ITEM 101	EM101	; ERROR: CACHE CONTROL REG NOT INITIALIZED BY POWER FAIL
7816				DH100	: PC=/DATA=
7817	056064	000000		DT100	: SERRPC, SREG1
7818	056066	000000		0	
7819	056070	000000	; ITEM 102	EM102	; ERROR: POWER UP FAILED TO INVALIDATE CACHE
7820	056072	000000		DH107	: PC=
7821				DT107	: SERRPC
7822	056074	051500		0	
7823	056076	054763	; ITEM 103	EM103	; ERROR: DEVICE ERROR BIT SET WHEN DOING NPR, DATO TO ADDR
7824	056100	055062		DH7	: PC=/P ADDH/P ADDL
7825	056102	000000		DT5	: SERRPC, SREG1, SREG2
7826				0	
7827	056104	051566	; ITEM 104	EM104	; ERROR: CACHE LOC NOT INVALIDATED BY NPR, DATO
7828	056106	055000		DH7	: PC=/P ADDH/P ADDL
7829	056110	055070		DT5	: SERRPC, SREG1, SREG2
7830	056112	000000		0	
7831					
7832	056114	051640			
7833	056116	054226			
7834	056120	055020			
7835	056122	000000			
7836					
7837	056124	051733			
7838	056126	054226			
7839	056130	055020			
7840	056132	000000			

7841			;ITEM 105		
7842	056134	052030	EM105		;ERROR: DID NOT GET PARITY TRAP WHEN DID NPR
7843					DATA TO ADDR. WRITTEN WITH WRONG PARITY
7844	056136	054226	DH7		;PC=/P ADDH/P ADDL
7845	056140	055020	DT5		;SERRPC,\$REG1,\$REG2
7846	056142	000000	0		
7847			;ITEM 106		
7848	056144	000000	0		
7849	056146	000000	0		
7850	056150	000000	0		
7851	056152	000000	0		
7852			;ITEM 107		
7853	056154	052165	EM107		;ERROR: CACHE DID NOT TRACK WHEN FORCE MISS ON
7854	056156	055000	DH107		;PC=
7855	056160	055070	DT107		;SERRPC
7856	056162	000000	0		
7857			;ITEM 110		
7858	056164	052242	EM110		;ERROR: RETRY TO BACKING STORE NOT DONE ON CACHE PARITY
7859	056166	055000	DH107		;PC=
7860	056170	055070	DT107		;SERRPC
7861	056172	000000	0		
7862			;ITEM 111		
7863	056174	052335	EM111		;ERROR: TEST OF VALID BIT FAILED
7864					LOC COULD NOT BE MADE A HIT
7865	056176	054226	DH7		;PC=/P ADDH/P ADDL
7866	056200	055020	DT5		;SERRPC,\$REG1,\$REG2
7867	056202	000000	0		
7868			;ITEM 112		
7869	056204	052437	EM112		;ERROR: TEST OF VALID BIT FAILED
7870					LOC NOT INVALIDATED BY P TRAP
7871	056206	054226	DH7		;PC=/P ADDH/P ADDL
7872	056210	055020	DT5		;SERRPC,\$REG1,\$REG2
7873	056212	000000	0		
7874			;ITEM 113		
7875	056214	052550	EM113		;ERROR: ADDR. NOT INVALIDATED BY CONSOLE SWEEP
7876	056216	054226	DH7		;PC=/P ADDH/P ADDL
7877	056220	055020	DT5		;SERRPC,\$REG1,\$REG2
7878	056222	000000	0		
7879			;ITEM 114		
7880	056224	052631	EM114		;ERROR: LOC WRITTEN WITH WRONG PARITY NOT
7881					INVALIDATED VIA NPR DATO
7882	056226	054226	DH7		;PC=/P ADDH/P ADDL
7883	056230	055020	DT5		;SERRPC,\$REG1,\$REG2
7884	056232	000000	0		
7885			;ITEM 115		
7886	056234	052732	EM115		;ERROR: PARITY TRAP WHILE TESTING LOC
7887					WRITTEN WITH WRONG PARITY AND
7888					INVALIDATING VIA NPR DATO
7889	056236	054226	DH7		;PC=/P ADDH/P ADDL
7890	056240	055020	DT5		;SERRPC,\$REG1,\$REG2
7891	056242	000000	0		
7892			;ITEM 116		
7893	056244	053100	EM116		;ERROR: CACHE ALLOCATED DURING ODD ADDRESS TRAP
7894	056246	054226	DH7		;PC=/P ADDH/P ADDL
7895	056250	055020	DT5		;SERRPC,\$REG1,\$REG2
7896	056252	000000	0		

```

7897
7898 056254 053156
7899 056256 055000
7900 056260 055070
7901 056262 000000
7902
7903 056264 053231
7904 056266 055000
7905 056270 055070
7906 056272 000000
7907
7908 056274 053277
7909
7910 056276 054226
7911 056300 055020
7912 056302 000000
7913
7914 056304 053416
7915
7916 056306 054725
7917 056310 055030
7918 056312 000000
7919
7920 056314 053532
7921
7922 056316 054725
7923 056320 055030
7924 056322 000000
7925
7926 056324 053653
7927
7928 056326 054725
7929 056330 055030
7930 056332 000000
7931
7932
7933
7934 000001

```

```

;ITEM 117
EM117
DH107
DT107
0
;ITEM 120
EM120
DH107
DT107
0
;ITEM 121
EM121
DH7
DT5
0
;ITEM 122
EM122
DH45
DT12
0
;ITEM 123
EM123
DH45
DT12
0
;ITEM 124
EM124
DH45
DT12
0
; *****
; TEST BUFFER
; *****
.END

```

```

;ERROR: CACHE ALLOCATED DURING RED ZONE TRAP
;PC=
;SERRPC

;ERROR: CACHE ALLOCATED DURING KT ABORT
;PC=
;SERRPC

;ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
;LOC NOT INVALIDATED
;PC=/P ADDH/P ADDL
;SERRPC,$REG1,$REG2

;ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
;PARITY ERROR TAG
;PC=/P ADDH/P ADDL/DATA=
;SERRPC,$REG1,$REG2,$REG3

;ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
;PARITY ERROR LOW BYTE
;PC=/P ADDH/P ADDL/DATA=
;SERRPC,$REG1,$REG2,$REG3

;ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
;PARITY ERROR HIGH BYTE
;PC=/P ADDH/P ADDL/DATA=
;SERRPC,$REG1,$REG2,$REG3

```


EM32	044235	6658#	7604											
EM33	044365	6674#	7610											
EM34	044514	6689#	7616											
EM35	044637	6704#	7622											
EM36	045000	6721#	7628											
EM37	045157	6740#	7634											
EM4	042167	6471#	7490											
EM40	045337	6760#	7640											
EM41	045462	6775#	7646											
EM42	045606	6790#	7652											
EM43	045676	6800#	7657											
EM44	045745	6807#	7662											
EM45	046023	6815#	7667											
EM46	046102	6823#	7672											
EM47	046173	6833#	7677											
EM5	042251	6480#	7495											
EM50	046260	6842#	7682											
EM51	046346	6851#	7687											
EM52	046433	6860#	7692											
EM53	046521	6870#	7697											
EM54	046614	6880#	7702											
EM55	046710	6890#	7707											
EM56	047027	6904#	7713											
EM57	047114	6913#	7718											
EM6	042330	6488#	7500	7525										
EM60	047202	6922#	7723											
EM61	047262	6930#	7728											
EM62	047330	6937#	7733											
EM63	047470	6953#	7739											
EM64	047622	6968#	7745											
EM65	047752	6983#	7751											
EM66	050103	6998#	7757											
EM67	050226	7013#	7763											
EM7	042402	6495#	7505											
EM70	050373	7031#	7769											
EM71	050521	7046#	7775											
EM72	050654	7062#	7781											
EM73	051010	7078#	7787											
EM74	051106	7089#	7793											
EM75	051204	7100#	7799											
EM76	051304	7111#	7805											
EM77	051405	7122#	7811											
EREG =	177744	290#	1074	1673	1767	1782	1793	1864	1874	1912	1923	1965	1972	2301
		2311	2436	2454	2462	2771	2781	2942	2960	3072	3084	3211	3229	3237
		3387	3405	3562	3570	3732	3740	3887	3898	4032	4043	4167	4176	4547
		4556	5208	5210										
ERRVEC=	000004	127#	597	598#	609*	5571	5572*	5574*	5577*	5733	5734	5740*	5748*	5760*
		5765#	5774#	5784#	5785#									
FAKE	034224	712	5404#											
GNS =	***** U	344	6270	6271	6272	6273	6274	6277	6278	6279				
HAD	033714	3803	4092	5341#										
HIADD =	000101	292#	1067											
HMR =	177752	285#	1114	1122	1126	1155	1215	1238	1244	1273	1284	1329	1333	1382
		1392	1473	1482	2036	2050	2077	2114	2133	2246	2253	2383	2395	2517
		2526	2553	2590	2604	2716	2723	2882	2891	3169	3181	3327	3336	3492
		3506	3662	3676	3825	3828	3980	3983	4100	4105	4233	4283	4345	4614

PSM = 177776	42#	5364*												
PVEC = 000114	310#	1515*	1558*	1593*	1626*	1732*	1833*	1887*	1939*	2043*	2075*	2086*	2111*	
	2126#	2168*	2230*	2336*	2367*	2408*	2523*	2551*	2562*	2587*	2602*	2639*	2700*	
	2806#	2852*	2996*	3015*	3116*	3153*	3254*	3297*	3440*	3478*	3610*	3648*	3780*	
	3802#	3823*	3837*	3953*	4060*	4091*	4194*	4389*	4577*	4593*	4644*	4777*	4827*	
	133#	587*	588*	4778*	4788*	4828*	6284*	6285*	6294*	6300*	6312*	6313*		
PURVEC= 000024	652	720#												
QRK05 002232	653	762#												
QRPD3 002412	654	794#												
QTU10 002540	650	660#												
QUBEN 001764	651	704#												
QUBEO 002156	639	642#	709	726	754	768	800							
Q1 001726	632	635#	851	856										
Q2 001704	302#	1962												
RDAT = 000106	6133	6277#												
RDCHR = 104406	6205	6278#												
RDLIN = 104407	636	661	730	772	804	6279#								
RDOCT = 104410	128#													
RESVEC= 000010	298#	2291	2761	2930	3063	3375	3546	3716	3881	4026	4161	4541	4845	
RJAM = 000100	5205													
RKBA = 177410	315#	5442*												
RKCS = 177404	313#	721	743	744	747*	5412	5431*	5433	5441*	5451*				
RKDA = 177412	316#	746*	5440*											
RKDS = 177400	311#	749	5425											
RKER = 177402	312#													
RKWC = 177406	314#	5439*												
RLOG = 000022	303#	866	1081	1551	1587	1666	1752	1855	1905	1953	2064	2100	2152	
	2276	2422	2540	2576	2623	2746	2911	3044	3197	3356	3527	3697	3816	
	3863	4008	4143	4226	4276	4338	4500	4523	4607	4816	4835	5241	5261	
	5795													
RPBA = 176720	300#	321#	5491*											
RPCA = 176722	322#	783*												
RPCS = 176714	319#	763	785	786	5459	5470	5490*	5500*						
RPDA = 176724	323#	784*												
RPDS = 176710	317#	5483												
RPER = 176712	318#													
RPWC = 176716	320#	5489*												
RSER = 000101	299#	1679	1710	1760	2283	2429	2753	2922	3055	3204	3367	3538	3708	
	3873	4018	4153	4490	4533	4849	5197							
	301#	1684	1700	1770	2304	2441	2774	2953	3216	3398	3584	3754	3890	
	4035	4187	4567	5224										
SDPAR0= 172260	231#													
SDPAR1= 172262	232#													
SDPAR2= 172264	233#													
SDPAR3= 172266	234#													
SDPAR4= 172270	235#													
SDPAR5= 172272	236#													
SDPAR6= 172274	237#													
SDPAR7= 172276	238#													
SDPOR0= 172220	209#													
SDPOR1= 172222	210#													
SDPOR2= 172224	211#													
SDPOR3= 172226	212#													
SDPOR4= 172230	213#													
SDPOR5= 172232	214#													
SDPOR6= 172234	215#													

SWS	= 000040	91#																	
SW6	= 000100	90#																	
SW7	= 000200	89#																	
SW8	= 000400	88#																	
SW9	= 001000	87#																	
TAB	001752	648	650#																
TAD1	026442	4093	4097*	4099	4115	4203#													
TAD2	025242	3804	3824	3849	3915#														
TAG	033606	1648	1689	1698	1775	2950	2981	3395	3426	5313#									
TBITVE=	000014	129#																	
TKVEC =	000060	136#																	
TPAT	026432	4097	4098	4102	4107	4109	4127	4132	4198#										
TPVEC =	000064	137#																	
TRAPVE=	000034	135#	585*	586*	5737	5738*	5743*												
TRTVEC=	000014	130#																	
TST1	003162	896#																	
TST10	006000	1326	1347	1367#															
TST11	006166	1369	1388	1393	1421#														
TST12	006626	1423	1426	1479	1488	1512#													
TST13	007140	1514	1594	1618#															
TST14	010230	1620	1623	1830#															
TST15	012000	1832	1986	2026#															
TST16	012734	2028	2170	2227#															
TST17	013406	2229	2364#																
TST2	004164	898	901	1061	1103#														
TST20	014100	2366	2410	2507#															
TST21	015000	2509	2640	2697#															
TST22	016000	2699	2808	2844#															
TST23	016646	2846	2849	3012#															
TST24	020000	3014	3118	3150#															
TST25	020456	3152	3289#																
TST26	022000	3291	3294	3442	3475#														
TST27	024000	3477	3612	3645#															
TST3	004320	1106	1142#																
TST30	024566	3647	3799#																
TST31	025244	3801	3838	3947#															
TST32	025750	3949	3952	4061	4088#														
TST33	026444	4090	4195	4214#															
TST34	026634	4216	4238	4262#															
TST35	027030	4264	4290	4316#															
TST36	027300	4318	4320	4352	4386#														
TST37	030260	4388	4452	4590#															
TST4	004616	1144	1194	1207#															
TST40	030524	4592	4658#																
TST41	030674	4660	4672	4676	4697#														
TST42	031132	4699	4726	4743	4772#														
TST43	031524	4774	4776	4829	4906#														
TST44	032240	4908	4911	4976	4980	5039#													
TST5	004712	1209	1216	1233#															
TST6	005044	1235	1245	1249	1266#														
TST7	005364	1268	1279	1290	1306	1324#													
TYPDS =	104405	5171	6274#																
TYPE =	104401	616	635	642	660	670	671	708	725	728	733	753	767	770					
		775	799	802	807	822	827	850	855	4785	5169	5172	5379	5380					
		5419	5420	5464	5465	5477	5478	5524	5525	5631	5639	5673	5690	5692					
		5695	5697	5701	5708	5862	5914	6067	6140	6146	6151	6155	6160	6161					

M13

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 154
CROSS REFERENCE TABLE -- USER SYMBOLS

	6163	6166	6170	6234	6236	6270*	6314
TYPOC = 104402	5681	5705	6271*				
TYPON = 104404	6273*						
TYPOS = 104403	6272*						
T01L01 004242	1109	1114*					
T01L02 004264	1115	1120*					
T01L03 004252	1116*	1123	1127				
T02L01 004404	1148	1155*					
T02L02 004462	1164	1170*					
T02L03 004504	1171	1176*					
T02L04 004526	1177	1182*					
T02L05 004550	1183	1188*					
T02L06 004454	1167*	1174	1180	1186	1198		
T04L01 005144	1274	1281*					
T04L02 005220	1285	1292*					
T04L03 005300	1293	1303*					
T05L01 005472	1330	1341*					
T05L02 005514	1334	1339	1346*				
T06L01 006260	1437*	1494	1496				
T06L02 006366	1440	1454	1457*				
T06L03 006276	1438	1441*	1491				
T06L04 006620	1448	1450	1498*				
T06L05 006316	1442	1446*					
T06L06 006324	1444	1447*					
T06L07 006360	1452	1456*					
T06L08 006440	1469*	1476	1485				
T06L09 006522	1471	1481*					
T06L10 006566	1474	1483	1490*				
T06L11 006610	1492	1495*					
T06L12 006266	1435	1439*					
T07L01 007354	1626	1656*					
T07L02 007650	1632	1725*					
T07L03 007254	1635*						
T07L04 007232	1630*	1650	1691	1715	1723		
T07L05 010214	1654	1695	1720	1735	1780	1791	1802
T07L06 007520	1674	1697*					1808*
T07L07 007636	1704	1719	1722*				
T07L08 007734	1732	1742*					
T07L09 007674	1734*	1740	1777	1788	1799	1805	
T07L10 010076	1768	1782*					
T07L11 010136	1783	1793*					
T07L12 010176	1794	1804*					
T08L01 010350	1833	1851*					
T08L02 011052	1849	1872	1881	1899	1921	1932	1945
T08L03 010432	1863	1874*					1970
T08L04 010460	1865	1883*					1978*
T08L05 010466	1875	1887*					
T08L06 010262	1836*	1885					
T08L07 010546	1887	1901*					
T08L09 010630	1913	1923*					
T08L10 010504	1890*	1918	1929	1937			
T08L11 010662	1924	1934*					
T08L12 010676	1935	1939*					
T08L13 010740	1939	1949*					
T08L14 010706	1941*	1947	1967	1974			
T08L15 011032	1966	1972*					

T09L16	010724	1944#	1973					
T09L01	006770	1515	1541#					
T09L02	006722	1519	1526#					
T09L03	007072	1558	1573#					
T09L04	007056	1564#						
T09L06	007110	1539	1571	1583#				
T11H01	015172	2700	2736#					
T11H02	015040	2704#	2734					
T11H03	015100	2712	2715#	2719				
T11H04	015400	2717	2796#					
T11H05	015420	2724	2800#					
T11H06	015444	2729	2768	2779	2787	2794	2806#	
T11H07	015304	2763	2770#					
T11H08	015340	2772	2781#					
T11H09	015364	2782	2789#					
T11H10	015436	2799	2803#					
T11H11	015122	2722#	2726					
T11H12	015150	2731#	2732					
T11L01	013126	2230	2266#					
T11L02	012774	2234#	2264					
T11L03	013034	2242	2245#	2249				
T11L04	013334	2247	2326#					
T11L05	013354	2254	2330#					
T11L06	013400	2259	2298	2309	2317	2324	2336#	
T11L07	013240	2293	2300#					
T11L08	013274	2302	2311#					
T11L09	013320	2312	2319#					
T11L10	013372	2329	2333#					
T11L11	013056	2252#	2256					
T11L12	013104	2261#	2262					
T12H01	020200	3153	3193#					
T12H02	020030	3156#	3191					
T12H06	020130	3177	3179#					
T12H07	020430	3170	3182	3249#				
T12H08	020450	3187	3222	3235	3243	3254#		
T12H09	020322	3212	3224#					
T12H11	020342	3227	3229#					
T12H12	020366	3230	3237#					
T12H13	020160	3188#	3190					
T12H14	020412	3238	3245#					
T12L01	013646	2367	2418#					
T12L02	013436	2370#	2405					
T12L06	013536	2391	2393#					
T12L07	013624	2384	2396	2412#				
T12L08	013606	2401	2408#	2416	2447	2460	2468	2473
T12L09	013770	2437	2449#					
T12L11	014010	2452	2454#					
T12L12	014034	2455	2462#					
T12L13	013566	2402#	2404					
T12L14	014060	2463	2470#					
T13H01	020744	3297	3352#					
T13H02	020602	3317#	3350					
T13H03	020634	3326#	3331					
T13H04	021216	3329	3420#					
T13H05	020664	3335#	3341					
T13H06	021264	3338	3432#					

T13407	021312	3345	3380	3403	3411	3418	3430	3439#
T13408	021056	3378	3382#					
T13409	021072	3384	3387#					
T13410	021156	3388	3405#					
T13411	021122	3392	3395#					
T13412	021202	3406	3413#					
T13413	021246	3426#	3437					
T13415	020732	3347	3348#					
T13L01	016266	2852	2907#					
T13L02	016124	2872#	2905					
T13L03	016156	2881#	2886					
T13L04	016540	2884	2975#					
T13L05	016206	2890#	2896					
T13L06	016606	2893	2987#					
T13L07	016634	2900	2935	2958	2966	2973	2985	2995#
T13L08	016400	2933	2937#					
T13L09	016414	2939	2942#					
T13L10	016500	2943	2960#					
T13L11	016444	2947	2950#					
T13L12	016524	2961	2968#					
T13L13	016570	2981#	2992					
T13L15	016254	2902	2903#					
T14L01	016760	3015	3040#					
T14L02	016704	3019#	3071	3112				
T14L03	017242	3022	3106#					
T14L04	016746	3024	3034#					
T14L05	016734	3026	3030#					
T14L06	016712	3020#	3028	3032	3038			
T14L07	017304	3035	3068	3080	3116#			
T14L08	017072	3065	3070#					
T14L09	017150	3073	3084#					
T14L10	016720	3023#	3082					
T14L11	017176	3085	3093#					
T14L12	017130	3079#	3091	3104	3114			
T15401	024210	3648	3693#					
T15402	024466	3664	3678	3761#				
T15403	024510	3667	3767#					
T15404	024560	3687	3725	3738	3746	3759	3765	3780#
T15405	024034	3651#	3691					
T15406	024334	3720	3727#					
T15408	024350	3729	3732#					
T15412	024170	3686	3688#	3690				
T15413	024374	3733	3740#					
T15414	024420	3741	3748#					
T15415	024526	3681	3772#					
T15416	024542	3770	3775#					
T15417	024112	3666	3668#					
T15418	024154	3680	3682#					
T15421	024064	3661#	3669					
T15422	024126	3675#	3683					
T15L01	022210	3478	3523#					
T15L02	022466	3494	3508	3591#				
T15L03	022510	3497	3597#					
T15L04	022560	3517	3555	3568	3576	3589	3595	3610#
T15L05	022034	3481#	3521					
T15L06	022334	3550	3557#					

T15L08	022350	3559	3562#						
T15L12	022170	3516	3518#	3520					
T15L13	022374	3563	3570#						
T15L14	022420	3571	3578#						
T15L15	022526	3511	3602#						
T15L16	022542	3600	3605#						
T15L17	022112	3496	3498#						
T15L18	022154	3510	3512#						
T15L21	022064	3491#	3499						
T15L22	022126	3505#	3513						
T16L01	025534	3953	4004#						
T16L02	025504	3981	3996#						
T16L03	025470	3984	3992#						
T16L04	025732	3986	4002	4030	4041	4050	4059#		
T16L05	025410	3977#	3990	3999					
T16L06	025516	3994	3998#						
T16L07	025642	4028	4032#						
T16L08	025674	4033	4043#						
T16L09	025720	4044	4052#						
T17L01	026224	4091	4139#						
T17L02	026104	4101	4114#						
T17L03	026174	4103	4131#						
T17L04	026136	4106	4122#						
T17L05	026152	4108	4126#						
T17L06	026422	4110	4120	4137	4165	4174	4183	4194#	
T17L07	026016	4097#	4112						
T17L08	026010	4096#	4116						
T17L09	026120	4116#	4124						
T17L10	026210	4129	4133#						
T17L11	026332	4163	4167#						
T17L12	026356	4168	4176#						
T17L13	026402	4177	4185#						
T18L01	030036	4389	4519#						
T18L02	027710	4390	4485#						
T18L03	030004	4438	4444	4510#					
T18L05	027564	4455	4456#						
T18L06	027620	4463#	4470						
T18L07	027630	4465#	4469						
T18L09	027650	4467	4472#	4481	4483				
T18L10	030234	4475	4508	4517	4545	4554	4563	4574#	
T18L11	027340	4391	4392#						
T18L12	030144	4543	4547#						
T18L13	030170	4548	4556#						
T18L14	030214	4557	4565#						
T19L01	030610	4662	4669#						
T19L02	030604	4665	4668#						
T19L03	030650	4667	4678#						
T19L04	030630	4673#	4682						
T20L01	031412	4777	4831#						
T20L02	031226	4778	4788#						
T20L03	031246	4788	4793#						
T20L04	031334	4809	4812#						
T20L05	031520	4847	4859#						
T20L06	031374	4827#	4857	4860					
T21L01	031712	4932	4941#						
T21L02	031704	4935	4939#						

T21L03	032112	4938	4984#				
T21L04	032152	4942	4987	4996#			
T21L05	032200	4948	4950	4993	5002#		
T21L07	032024	4952	4966#				
T21L08	032000	4954	4960#				
T21L09	031614	4919#	4958	4964	4973	4996	5002
T21L10	032060	4967	4975#	5000	5008	5010	
T21L11	031742	4949#	4994				
T22L01	032534	5086	5095#				
T22L02	032754	5064	5106	5111	5128	5136	5139#
T22L03	032526	5089	5094#				
T22L04	032626	5092	5113#				
T22L05	032670	5096	5115	5124#			
T22L06	032716	5102	5104	5121	5130#		
T22L08	032350	5058#	5108	5110			
T22L10	032566	5103#	5122				
T22L11	032436	5073#	5124	5130			
T23L01	030504	4615	4624	4630	4637#		
T24H01	014670	2519	2642#				
T24H02	014242	2523	2536#				
T24H03	014712	2528	2648#				
T24H04	014734	2534	2655#				
T24H05	014200	2525#	2549				
T24H06	014756	2555	2594	2661#			
T24H07	014404	2562	2572#				
T24H08	014346	2564#	2569	2585			
T24H09	014504	2591	2596#				
T24H10	014334	2562#	2593	2607	2615		
T24H12	014450	2588#	2598				
T24H13	014552	2605	2610#				
T24H14	014720	2608	2649#				
T24H15	014742	2570	2616	2656#			
T24H16	014610	2602	2618#				
T24H17	014522	2603#	2630				
T24H18	014646	2632#	2646	2653	2659	2666	
T24H19	014156	2518	2520#				
T24H20	014140	2516#	2521				
T24H21	014172	2524#					
T24H22	014216	2527	2530#				
T24L01	012624	2038	2172#				
T24L02	012154	2043	2060#				
T24L03	012646	2052	2178#				
T24L04	012670	2058	2185#				
T24L05	012076	2045#	2073				
T24L06	012712	2079	2118	2191#			
T24L07	012316	2086	2096#				
T24L08	012260	2088#	2093	2109			
T24L09	012416	2115	2120#				
T24L10	012246	2086#	2117	2136	2144		
T24L12	012362	2112#	2122				
T24L13	012500	2128	2130	2134	2139#		
T24L14	012654	2137	2179#				
T24L15	012676	2094	2145	2186#			
T24L16	012536	2126	2147#				
T24L17	012434	2127#	2159				
T24L18	012574	2161#	2176	2183	2189	2196	

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 162
CROSS REFERENCE TABLE -- USER SYMBOLS

SMAMS2	001272	488#															
SMAMS3	001276	491#															
SMAMS4	001302	494#															
SMADR	001002	375#															
SMFLG	037274	5954#	5960	5995*	5999#												
SMHEM	040135	6184#															
SMGAD	001252	466#	5970*	5973													
SMGLG	001254	467#	5975*														
SMSTY	001236	460#	5968	5976*	5988	5992*											
SMR	040124	6182#															
SMTYP1	001267	481#															
SMTYP2	001273	489#															
SMTYP3	001277	492#															
SMTYP4	001303	495#															
SMXCNT	035410	5599	5610#														
SMULL	001150	423#	5921	5950													
SMWTST=	000001	880#	882	1089#	1091	1129#	1131	1200#	1202	1223#	1225	1256#	1258	1315#			
		1317	1353#	1355	1399#	1401	1500#	1502	1597#	1599	1811#	1813	1993#	1995			
		2198#	2200	2338#	2340	2477#	2479	2668#	2670	2813#	2815	2998#	3000	3124#			
		3126	3258#	3260	3449#	3451	3619#	3621	3782#	3784	3917#	3919	4065#	4067			
		4205#	4207	4252#	4254	4302#	4304	4364#	4366	4580#	4582	4646#	4648	4684#			
		4686	4747#	4749	4882#	4884	5012#	5014									
SOCNT	037522	6039#	6068#	6081#													
SOMODE	037524	6034#	6038#	6043	6046*	6057*	6083#										
SOVER	035372	5567	5587	5597	5606#												
SPASS	001244	463#	611#	5161*	5162*	5170	5183	5593	5611								
SPASTH	001006	377#															
SPOWER	040532	6315	6320#														
SPWRD	040364	587	4828	6284#	6312												
SPWRNG	040520	6315#															
SPWRUP	040436	6294	6300#														
SQUES	001206	441#	5665	5950	6163	6180	6236	6239									
SROCHR	037526	6100#	6277														
SRODEC=	***** U	6280															
SROLIN	037646	6128#	6278														
SRODOCT	040146	6200#	6279														
SROSZ =	000010	6121#															
SREGAD	001154	427#															
SREGO	001156	429#															
SREG1	001160	430#	980#	1068#	1107*	1108	1114#	1122#	1126*	1150#	1165#	1172#	1178*	1184*			
		1196#	1217#	1252#	1276*	1287*	1296#	1309#	1336*	1342#	1375#	1385#	1394*	1521*			
		1536#	1682#	1713#	1763*	1837*	2173#	2179#	2186*	2192#	2286*	2333#	2414*	2432*			
		2643#	2649#	2656#	2662*	2756*	2803#	2925#	3058*	3108#	3207#	3251#	3370#	3541*			
		3592#	3605#	3711#	3762*	3775*	3841#	3848*	3854*	3876#	4021#	4117*	4134*	4156*			
		4241#	4247#	4293#	4355*	4356*	4493#	4511#	4536*	4626*	4633*	4673*	4738*	4808*			
		4852#	4997#	5003#	5071	5125*	5131*	5200*	5253*	5291#	5293#	7443	7445	7447			
		7449	7450	7452													
SREG2	001162	431#	979#	1071#	1110#	1116#	1151#	1166#	1173#	1179#	1185#	1197#	1218#	1253#			
		1277#	1288#	1297#	1310#	1337#	1343#	1376#	1386#	1395#	1522#	1537#	1677#	1708#			
		1766#	1838#	1888#	2174#	2180#	2187#	2193#	2289*	2328*	2331#	2413#	2435*	2644*			
		2650#	2657#	2663#	2759#	2798#	2801#	2928*	3061*	3109*	3210#	3250#	3373*	3544*			
		3593#	3606#	3714#	3763*	3776*	3842#	3849*	3855*	3879*	4024#	4115*	4123*	4128*			
		4159#	4242#	4248#	4294*	4496*	4512#	4539*	4628*	4634*	4674*	4739*	4855*	4998*			
		5004#	5072	5126#	5132*	5203*	5252*	5281*	5282*	5290#	5291	5298#	5304*	7443			
		7445	7447	7450													
SREG3	001164	432#	945#	1074#	1219*	1298*	1311*	1396*	1466*	1649*	1687*	1705*	1773*	1786*			

		1797*	1804*	1869*	1878*	1917*	1928*	1963*	2294*	2307*	2315*	2322*	2444*	2458*
		2466*	2470*	2471*	2764*	2777*	2785*	2792*	2917*	2918*	2956*	2964*	2971*	2982*
		3050*	3051*	3076*	3088*	3096*	3110*	3219*	3233*	3241*	3245*	3246*	3362*	3363*
		3401*	3409*	3416*	3427*	3533*	3534*	3553*	3566*	3574*	3587*	3607*	3703*	3704*
		3723*	3736*	3744*	3757*	3777*	3869*	3870*	3893*	3902*	3910*	4014*	4015*	4038*
		4047*	4055*	4133*	4149*	4150*	4171*	4180*	4190*	4486*	4487*	4513*	4529*	4530*
		4551*	4560*	4570*	4841*	4842*	5214*	5218*	5227*	5234*	7443	7447		
SREG4	001166	433#	981*	1299*	1312*	1690*	1714*	1776*	1787*	1798*	1836*	1890*	1960*	1964
		2295*	2438*	2439*	2450*	2453*	2765*	2937*	2940*	2951*	3070*	3098*	3100*	3102*
		3111*	3213*	3214*	3225*	3228*	3382*	3385*	3396*	3557*	3560*	3578*	3580*	3582*
		3599*	3604*	3727*	3730*	3748*	3750*	3752*	3769*	3774*	4127*	4132*	4514*	5193*
		5194*	5218	7443	7450									
		434#												
SREG5	001170	5182#												
SRTNAD	033120	6280												
SR2A =	*****	6280												
SSAVRE =	*****	6280												
SSAVR6	040530	6293*	6301	6302*	6303*	6319*								
SSCOPE	035152	581	5565#											
SSETUP =	000037	337#	580	581	583	585	587	589	590	591	593	5159	5566	5626
		5651	5658	6089	6186									
SSIZE	035750	907	1428	1625	2851	3296	3955	4322	4454	5050	5729#			
SSIZEX	036240	5772	5782#											
SSTUP =	177777	337#												
SSVLAD	035336	5575	5600#											
SSVPC =	001014	384#	389											
SSMR =	167000	2#	12	17	18	19	20	21	22	23	24	590	591	593
		594	897	1104	1143	1208	1234	1267	1325	1368	1422	1513	1619	1831
		2027	2228	2365	2508	2698	2845	3013	3151	3290	3476	3646	3800	3948
		4089	4215	4263	4317	4387	4591	4659	4698	4773	4907	5040	5154	5160
		5175	5181	5183	5558	5559	5560	5561	5562	5566	5578	5580	5581	5582
		5589	5590	5591	5603	5606	5609	5617	5618	5619	5620	5621	5629	5636
		5648	5651	5663	5664	6316								
		471#	614											
SSMREG	001260	5562												
SSMRK =	000000	462#	5601*											
STESTN	001242	590*	5160*	5589*	5596	5599*	5609*							
STIMES	035406	420#	6087	6104	6110									
STKB	001142	419#	6087	6102	6108									
STKS	001140	435#	874*	875*	876	877*	1458*	1645*	1647*	1649	1688*	1690	1697*	1703
STMP0	001172	1714	1774*	1776	2951	2976*	2982	2988*	3396	3421*	3427	3433*	3805	3993*
		3997*	4094	4662*	4663*	4664	4666	4706*	4708*	4710	4713	4790*	4798	4930*
		4931*	4934	4937	4977*	4978*	4979*	5069*	5084*	5085*	5088	5091	5142*	5143*
		5144	5145*	5275	5302	5315*	5317*	5325*	5328*	5331*	5333*	5334*	5335*	5341*
		5343*	5344*	5345	5347*	5350*	5351*	5360*	5361*	5362	5363*			
STMP1	001174	436#	4707*	4709*	4710	4712*	4713	4932*	4933*	4934	4936*	4937	5086*	5087*
		5088	5090*	5091										
		437#												
STMP2	001176	438#												
STMP3	001200	439#	5446*	5448*	5449	5493*	5496*	5500	5533*	5535*	5539			
STMP4	001202	440#	5372*	5375*	5411*	5414*	5424*	5427*	5432*	5435*	5444*	5447*	5451	5469*
STMP5	001204	5472*	5482*	5485*	5495*	5497*	5498	5515*	5518*	5534*	5536*	5537		
		12#	880	897#	898	1061	1089	1104#	1106	1129	1143#	1144	1194	1200
STN =	000045	1208#	1209	1216	1223	1234#	1235	1245	1249	1256	1267#	1268	1279	1290
		1306	1315	1325#	1326	1347	1353	1368#	1369	1388	1393	1399	1422#	1423
		1426	1479	1488	1500	1513#	1514	1594	1597	1619#	1620	1623	1811	1831#
		1832	1986	1993	2027#	2028	2170	2198	2228#	2229	2338	2365#	2366	2410

M14

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC
DQKKA.P11 07-FEB-77 11:01

MACY11 27(1006) 09-FEB-77 15:33 PAGE 168
CROSS REFERENCE TABLE -- MACRO NAMES

SSSKIP	1398	1061	1194	1216	1245	1249	1279	1290	1306	1347	1388	1393	1479	1488	1594
	2410	2640	2808	3838	4061	4195	4238	4290	4320	4352	4672	4676	4726	4743	4776
	4829	4976	4980												
.EQUAT	28	29													
.HEADE	28														
.KT11	28	139													
.SETUP	28	337													
.SMRHI	28	13													
.SMRLO	258														
.SACT1	28	380													
.SAPT8	455														
.SAPTH	28	358													
.SAPTY	28	5950													
.SCATC	28	338													
.SCHTA	28														
.SEOP	28	5149													
.SERRO	28	5611													
.SERRT	28	5665													
.SPOWE	28	6280													
.SRODC	28	6186													
.SREAD	28	6084													
.SSCOP	28	5552													
.STRAP	28	6239													
.STYPD	28	5804													
.STYPE	28	5871													
.STYPO	28	6007													

. ABS. 056334 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DSKZ:DQKKA, DSKZ:DQKKA/SOL/CRF=DSKZ:DQKKA.P11
RUN-TIME: 25 22 2 SECONDS
RUN-TIME RATIO: 487/51=9.5
CORE USED: 34K (68 PAGES)