

.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	HTUI0 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 (SW0=1).
 - a. Unibus Exercisor (M7885)
 - b. Bus Tester (old)
 - c. RK05
 - d. RP03
 - 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 ren (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 Exercisor (M7885)
- 1 - [carriage return] - Bus Tester 01d
- 2 - [carriage return] - RK05
- 3 - [carriage return] - RP03
- 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 - RK05

If the RK05 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 - RP03

If the RP03 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 RP03 register times out, the RP03 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 TUI0 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 Timeouts
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 (\$LPERR). (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 $A_{10}=0$, high cache is defined as that portion of cache with physical address $A_{10}=1$). 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 ($A_{10}=0$). The address BUFH corresponds to the first high cache address.

Immediately after the program is started the program identifies itself and then if $SW_0=1$ 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 hardware 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 'SSCOPE'. 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 EPROR 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 \$KT11=1, before the routine is called, memory management will be used. \$LSTAD contains the last virtual address of the last bank if memory management is used. Otherwise it contains the last absolute address of available memory. \$LSTBK 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 ASCII 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 \$TMP0 and converts it to a physical address. The physical address bits A17, A16 are stored in \$REG1 and bits A0 - A15 are stored in \$REG2.

9.11 TAG

This subroutine calculates the tag field from a page address register's contents stored in \$TMP0.

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 RP03 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 SWEEP 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 TYPFOUT 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

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

```

```

.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
;*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
;*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
;*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
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112

```

```

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      PIROVEC=240        ;PROGRAM INTERRUPT REQUEST VECTOR
139                      .SBTTL MEMORY MANAGEMENT DEFINITIONS
140
141                      ;*KT11 VECTOR ADDRESSES
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
226 172254
227 172256
228
229
230
231 172260
232 172262
233 172264
234 172266
235 172270
236 172272
237 172274
238 172276
239
240
241
242 172300
243 172302
244 172304
245 172306
246 172310
247 172312
248 172314
249 172316
250
251
252
253 172320
254 172322
255 172324
256 172326
257 172330
258 172332
259 172334
260 172336
261
262
263
264 172340
265 172342
266 172344
267 172346
268 172350
269 172352
270 172354
271 172356
272
273
274
275 172360
276 172362
277 172364
278 172366
279 172370
280 172372

SIPAR5= 172252
SIPAR6= 172254
SIPAR7= 172256

;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
SDPAR0= 172260
SDPAR1= 172262
SDPAR2= 172264
SDPAR3= 172266
SDPAR4= 172270
SDPAR5= 172272
SDPAR6= 172274
SDPAR7= 172276

;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
KIPDR0= 172300
KIPDR1= 172302
KIPDR2= 172304
KIPDR3= 172306
KIPDR4= 172310
KIPDR5= 172312
KIPDR6= 172314
KIPDR7= 172316

;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
KDPDR0= 172320
KDPDR1= 172322
KDPDR2= 172324
KDPDR3= 172326
KDPDR4= 172330
KDPDR5= 172332
KDPDR6= 172334
KDPDR7= 172336

;*KERNEL "I" PAGE ADDRESS REGISTERS
KIPAR0= 172340
KIPAR1= 172342
KIPAR2= 172344
KIPAR3= 172346
KIPAR4= 172350
KIPAR5= 172352
KIPAR6= 172354
KIPAR7= 172356

;*KERNEL "D" PAGE ADDRESS REGISTERS
KDPAR0= 172360
KDPAR1= 172362
KDPAR2= 172364
KDPAR3= 172366
KDPAR4= 172370
KDPAR5= 172372

281 172374
282 172376
283
284
285 177752
286 177746
287 000106
288 000106
289 000106
290 177744
291 177766
292 000101
293 000102
294 055016
295 060000
296 062000
297 076600
298 000100
299 000131
300 000102
301 000107
302 000106
303 000022
304 000222
305 000304
306 000226
307 000352
308 177572
309 177576
310 000114
311 177400
312 177402
313 177404
314 177406
315 177410
316 177412
317 176710
318 176712
319 176714
320 176716
321 176720
322 176722
323 176724
324 172520
325 172522
326 172524
327 172526
328 000001
329 000002
330 000004
331 000010
332 000020
333 000040
334 000015
335 000012
336

KDPA6= 172374
KDPA7= 172376

;*OTHER EQUATES
HMR=177752 ;HIT/MISS REG ADDRESS
CCR=177746 ;CACHE CONTROL REG ADDRESS
CDH=106 ;CACHE DATA HIGH ADDRESS
CDL=106 ;CACHE DATA LOW ADDRESS
CTAG=107 ;CACHE TAG ADDRESS
EREG=177744 ;MEMORY ERROR REG ADDRESS
CER=177766 ;CPU ERROR REG ADDRESS
HIADD=101 ;HIGH UNIBUS ADDRESS OF ERROR
LOADD=102 ;LOW UNIBUS ADDRESS OF ERROR
BSD=55016 ;BACKING STORE DATA ADDRESS
BUFL=60000 ;LOW ADDRESS BUFFER (A10=0)
BUFH=BUFL+2000 ;HIGH ADDRESS BUFFER (A10=1)
MED= 76600 ;MAINTENANCE INSTRUCTION
RJAM= 100 ;LOG READ ADDRESS FOR JAM REG.
RSER= 101 ;LOG READ ADDRESS FOR SERVICE REG.
RPBA= 102 ;LOG READ ADDRESS FOR PHYSICAL BUS ADDR.
RTAG= 107 ;LOG READ ADDRESS FOR CACHE TAG
RDAT= 106 ;LOG READ ADDRESS FOR CACHE DATA
RLOG= 22 ;READ ADDRESS FOR CPU INTERNAL REG "WHAMI"
WLOG= 222 ;WRITE ADDRESS FOR CPU INTERNAL REG "WHAMI"
WFLI= 304 ;WRITE ADDRESS FOR CPU INTERNAL REG "FLAG/INT"
WSW= 226 ;WRITE ADDRESS FOR CPU INTERNAL REG "SWITCH REG"
WINT= 352 ;WRITE ADDRESS FOR CPU INTERNAL REG "INIT REG"(MOD FOR D
MMR0=SR0 ;K11 STATUS REG
MMR2=SR2 ;K11 STATUS REG
PVEC=114 ;PARITY TRAP VECTOR
RKDS= 177400 ;RK05 DRIVE STATUS REG
RKER= 177402 ;RK05 ERROR REG
RKCS= 177404 ;RK05 CONTROL STATUS REG
RKWC= 177406 ;RK05 WORD COUNT REG
RKBA= 177410 ;RK05 CURRENT BUS ADDRESS REG
RKDA= 177412 ;RK05 DISK ADDRESS REG
RPDS= 176710 ;RP03 DEVICE STATUS REG
RPER= 176712 ;RP03 ERROR REG
RPCS= 176714 ;RP03 CONTROL STATUS REG
RPWC= 176716 ;RP03 WORD COUNT REG
RPBA= 176720 ;RP03 BUS ADDRESS REG
RPCA= 176722 ;RP03 CYLINDER ADDRESS REG
RPDA= 176724 ;RP03 DISK ADDRESS REG
MTS= 172520 ;TU10 STATUS REG
MTC= 172522 ;TU10 COMMAND REG
MTBRC= 172524 ;TU10 BYTE RECORD COUNTER
MTCMA= 172526 ;TU10 CURRENT MEMORY ADDRESS REG
HMR0= 1 ;HIT MISS REG BIT 0
HMR1= 2 ;HIT MISS REG BIT 1
HMR2= 4 ;HIT MISS REG BIT 2
HMR3= 10 ;HIT MISS REG BIT 3
HMR4= 20 ;HIT MISS REG BIT 4
HMR5= 40 ;HIT MISS REG BIT 5
CP= 15 ;CARRIAGE RETURN
LF= 12 ;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 DISPREG: .WORD 0      ;SOFTWARE DISPLAY REGISTER
346 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,#CCR      ;TURN CACHE OFF
355 000206 000137 001362              JMP    #START      ;JUMP TO STARTING ADDRESS OF PROGRAM.
356      .=LOC
357      .=1000
358 ;SBTTL APT PARAMETER BLOCK
359
360 ;*****
361 ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
362 ;*****
363      .SX=.      ;SAVE CURRENT LOCATION
364      .=24      ;SET POWER FAIL TO POINT TO START OF PROGRAM
365 000024 200      ;FOR APT START UP
366      .=44      ;POINT TO APT INDIRECT ADDRESS PNTR.
367 000044 $APTHDR ;POINT TO APT HEADER BLOCK
368      .=SX      ;RESET LOCATION COUNTER
369
370 ;*****
371 ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
372 ;INTERFACE SPEC.
373
374 $APTHD: .WORD 0      ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
375 $MBADR: .WORD $MAIL  ;ADDRESS OF APT MAILBOX (BITS 0-15)
376 $TSTM:  .WORD 60      ;RUN TIM OF LONGEST TEST
377 $PASTM: .WORD 60      ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
378 $UNITM: .WORD        ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
379 001012 000052      $ETEND=$MAIL/2 ;LENGTH MAILBOX-ETABLE(WORDS)
380 ;SBTTL ACT11 HOOKS
381
382 ;*****
383 ;HOOKS REQUIRED BY ACT11
384      $SVPC=.      ;SAVE PC
385      .=46
386 000046 $ENDAD      ;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
387      .=52
388 000052 .WORD 0      ;2)SET LOC.52 TO ZERO
389      .=$SVPC      ;RESTORE PC
390
391 ;*****
392

```

```

393 ;SBTTL COMMON TAGS
394
395 ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
396 ;*USED IN THE PROGRAM.
397
398      .=1100
399
400 $CMTAG: .WORD 0      ;START OF COMMON TAGS
401 $TSTM:  .BYTE 0      ;CONTAINS THE TEST NUMBER
402 $ERFLG: .BYTE 0      ;CONTAINS ERROR FLAG
403 $ICNT:  .WORD 0      ;CONTAINS SUBTEST ITERATION COUNT
404 $LPADR: .WORD 0      ;CONTAINS SCOPE LOOP
405 $LPERR: .WORD 0      ;CONTAINS SCOPE RETURN FOR ERRORS
406 $ERTTL: .WORD 0      ;CONTAINS TOTAL ERRORS DETECTED
407 $ITEMB: .BYTE 0      ;CONTAINS ITEM CONTROL BYTE
408 $ERMAX: .BYTE 1      ;CONTAINS MAX. ERRORS PER TEST
409 $ERRPC: .WORD 0      ;CONTAINS PC OF LAST ERROR INSTRUCTION
410 $CDADR: .WORD 0      ;CONTAINS OF 'GOOD' DATA
411 $BDADR: .WORD 0      ;CONTAINS OF 'BAD' DATA
412 $GDDAT: .WORD 0      ;CONTAINS 'GOOD' DATA
413 $BDDAT: .WORD 0      ;CONTAINS 'BAD' DATA
414      .WORD 0      ;RESERVED--NOT TO BE USED
415      .WORD 0
416      .WORD 0
417 SWR:    .WORD DSWR   ;OF SWITCH REGISTER
418 DISPLAY: .WORD DDISP ;OF DISPLAY REGISTER
419 $TKS:    177560      ;TTY KBD STATUS
420 $TKB:    177562      ;TTY KBD BUFFER
421 $TPS:    177564      ;TTY PRINTER STATUS REG.
422 $TPB:    177566      ;TTY PRINTER BUFFER REG.
423 $NULL:   .BYTE 0      ;CONTAINS NULL CHARACTER FOR FILLS
424 $FILLS:  .BYTE 2      ;CONTAINS # OF FILLER CHARACTERS REQUIRED
425 $FILLC:  .BYTE 12     ;INSERT FILL CHARS. AFTER A "LINE FEED"
426 $TPFLG: .BYTE 0      ;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
427 $REGAD: .WORD 0      ;CONTAINS THE FROM
428      ;WHICH ($REG0) WAS OBTAINED
429 $REG0:   .WORD 0      ;CONTAINS (($REGAD)+0)
430 $REG1:   .WORD 0      ;CONTAINS(($REGAD)+2)
431 $REG2:   .WORD 0      ;CONTAINS(($REGAD)+4)
432 $REG3:   .WORD 0      ;CONTAINS(($REGAD)+6)
433 $REG4:   .WORD 0      ;CONTAINS(($REGAD)+10)
434 $REG5:   .WORD 0      ;CONTAINS(($REGAD)+12)
435 $TMP0:   .WORD 0      ;USER DEFINED
436 $TMP1:   .WORD 0      ;USER DEFINED
437 $TMP2:   .WORD 0      ;USER DEFINED
438 $TMP3:   .WORD 0      ;USER DEFINED
439 $TMP4:   .WORD 0      ;USER DEFINED
440 $TMP5:   .WORD 0      ;USER DEFINED
441 $QUES:   .ASCII /?/   ;QUESTION MARK
442 $CRLF:   .ASCII <15> ;CARRIAGE RETURN
443 $LF:     .ASCII <12> ;LINE FEED
444 $REG1:   .WORD 0      ;CONTROL REG ADDR. FOR NPR DEVICE
445 $REG2:   .WORD 0      ;CONTROL REG ADDR. FOR NPR DEVICE
446 $REG3:   .WORD 0      ;CONTROL REG ADDR. FOR NPR DEVICE
447 $REG4:   .WORD 0      ;CONTROL REG ADDR. FOR NPR DEVICE
448 $REG5:   .WORD 0      ;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 000000
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 000000
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
IVECI: .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
;MAIL: ;APT MAILBOX
;MSGTY: .WORD AMSGTY ;MESSAGE TYPE CODE
;FATAL: .WORD AFATAL ;FATAL ERROR NUMBER
;TESTN: .WORD ATESTN ;TEST NUMBER
;PASS: .WORD APASS ;PASS COUNT
;DEVCT: .WORD ADEVCT ;DEVICE COUNT
;UNIT: .WORD AUNIT ;I/O UNIT NUMBER
;MSGADR: .WORD AMSGADR ;MESSAGE ADDRESS
;MSGLEN: .WORD AMSGLEN ;MESSAGE LENGTH
;ETABLE: ;APT ENVIRONMENT TABLE
;ENV: .BYTE AENV ;ENVIRONMENT BYTE
;ENVNM: .BYTE AENVNM ;ENVIRONMENT MODE BITS
;SWREG: .WORD ASWREG ;APT SWITCH REGISTER
;USWR: .WORD AUSWR ;USER SWITCHES
;CPUOP: .WORD ACPUP ;CPU TYPE,OPTIONS
;BITS 15-11=CPU TYPE
; 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
; 11/70=06,PDQ=07,Q=10
;BIT 10=REAL TIME CLOCK
;BIT 9=FLOATING POINT PROCESSOR
;BIT 8=MEMORY MANAGEMENT
;MAMS1: .BYTE AMAMS1 ;HIGH ADDRESS,M.S. BYTE
;MTYP1: .BYTE AMTYP1 ;MEM. TYPE,BLK#1
;MEM. TYPE BYTE -- (HIGH BYTE)
; 900 NSEC CORE=001
; 300 NSEC BIPOLAR=002
; 500 NSEC MOS=003
;MADR1: .WORD AMADR1 ;HIGH ADDRESS,BLK#1
; MEM. LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
;MAMS2: .BYTE AMAMS2 ;HIGH ADDRESS,M.S. BYTE
;MTYP2: .BYTE AMTYP2 ;MEM. TYPE,BLK#2
;MADR2: .WORD AMADR2 ;MEM. LAST ADDRESS,BLK#2
;MAMS3: .BYTE AMAMS3 ;HIGH ADDRESS,M.S. BYTE
;MTYP3: .BYTE AMTYP3 ;MEM. TYPE,BLK#3
;MADR3: .WORD AMADR3 ;MEM. LAST ADDRESS,BLK#3
;MAMS4: .BYTE AMAMS4 ;HIGH ADDRESS,M.S. BYTE
;MTYP4: .BYTE AMTYP4 ;MEM. TYPE,BLK#4
;MADR4: .WORD AMADR4 ;MEM. LAST ADDRESS,BLK#4
;VECT1: .WORD AVECT1 ;INTERRUPT VECTOR#1,BUS PRIORITY#1
;VECT2: .WORD AVECT2 ;INTERRUPT VECTOR#2,BUS PRIORITY#2
;BASE: .WORD ABASE ;BASE ADDRESS OF EQUIPMENT UNDER TEST
;DEVN: .WORD ADEVN ;DEVICE MAP
;CDW1: .WORD ACDW1 ;CONTROLLER DESCRIPTION WORD#1
;CDW2: .WORD ACDW2 ;CONTROLLER DESCRIPTION WORD#2
;DDW0: .WORD ADDW0 ;DEVICE DESCRIPTOR WORD#0
;DDW1: .WORD ADDW1 ;DEVICE DESCRIPTOR WORD#1

505 001326 000000
506 001330 000000
507 001332 000000
508 001334 000000
509 001336 000000
510 001340 000000
511 001342 000000
512 001344 000000
513 001346 000000
514 001350 000000
515 001352 000000
516 001354 000000
517 001356 000000
518 001360 000000
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

DDW2: .WORD ADDW2 ;DEVICE DESCRIPTOR WORD#2
DDW3: .WORD ADDW3 ;DEVICE DESCRIPTOR WORD#3
DDW4: .WORD ADDW4 ;DEVICE DESCRIPTOR WORD#4
DDW5: .WORD ADDW5 ;DEVICE DESCRIPTOR WORD#5
DDW6: .WORD ADDW6 ;DEVICE DESCRIPTOR WORD#6
DDW7: .WORD ADDW7 ;DEVICE DESCRIPTOR WORD#7
DDW8: .WORD ADDW8 ;DEVICE DESCRIPTOR WORD#8
DDW9: .WORD ADDW9 ;DEVICE DESCRIPTOR WORD#9
DDW10: .WORD ADDW10 ;DEVICE DESCRIPTOR WORD#10
DDW11: .WORD ADDW11 ;DEVICE DESCRIPTOR WORD#11
DDW12: .WORD ADDW12 ;DEVICE DESCRIPTOR WORD#12
DDW13: .WORD ADDW13 ;DEVICE DESCRIPTOR WORD#13
DDW14: .WORD ADDW14 ;DEVICE DESCRIPTOR WORD#14
DDW15: .WORD ADDW15 ;DEVICE DESCRIPTOR WORD#15

SETEND:

```
561
562
563
564
565
566
567
568
569
570
571
572 001362
573
574
575 001362 012706 001100
576 001366 005026
577 001370 022706 001134
578 001374 001374
579 001376 012706 001100
580
581 001402 012737 035152 000020
582 001410 012737 000340 000022
583 001416 012737 035412 000030
584 001424 012737 000340 000032
585 001432 012737 040306 000034
586 001440 012737 000340 000036
587 001446 012737 040364 000024
588 001454 012737 000340 000026
589 001462 013737 033054 033046
590 001470 005037 035406
591 001474 005037 035606
592 001500 112737 000001 001115
593 001506 012737 001506 001106
594 001514 012737 001514 001110
595
596
597 001522 013746 000004
598 001526 012737 001562 000004
599 001534 012737 177570 001134
600 001542 012737 177570 001136
601 001550 022777 177777 177356
602 001556 001012
603
604 001560 000403
605 001562 012716 001570 640:
606 001566 000002
607 001570 012737 000176 001134 650:
608 001576 012737 000174 001136
609 001604 012637 000004 660:
610
611 001610 005037 001244
612 001614 132737 000200 001257
613 001622 001403
614 001624 012737 001260 001134
615 001632
616 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 $SCOPE,$IOTVEC ;IOT VECTOR FOR SCOPE ROUTINE
MOV $340,$IOTVEC+2 ;LEVEL 7
MOV $ERROR,$ENTVEC ;EMT VECTOR FOR ERROR ROUTINE
MOV $340,$ENTVEC+2 ;LEVEL 7
MOV $TRAP,$TRAPVEC ;TRAP VECTOR FOR TRAP CALLS
MOV $340,$TRAPVEC+2 ;LEVEL 7
MOV $PWRDN,$PWRVEC ;POWER FAILURE VECTOR
MOV $340,$PWRVEC+2 ;LEVEL 7
MOV $ENDCT,$EOPT ;SETUP END-OF-PROGRAM COUNTER
CLR $TIMES ;INITIALIZE NUMBER OF ITERATIONS
CLR $ESCAPE ;CLEAR THE ESCAPE ON ERROR ADDRESS
MOV $1,$ERRMAX ;ALLOW ONE ERROR PER TEST
MOV $1,$LPADR ;INITIALIZE THE LOOP ADDRESS FOR SCOPE
MOV $1,$LPERR ;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 $648,$ERRVEC ;SET UP ERROR VECTOR
MOV $DSWR,$SWR ;SETUP FOR A HARDWARE SWITCH REGISTER
MOV $DDISP,$DISPLAY ;AND A HARDWARE DISPLAY REGISTER
CMP $-1,$DSWR ;TRY TO REFERENCE HARDWARE SWR
BNE 660 ;BRANCH IF NO TIMEOUT TRAP OCCURRED
;AND THE HARDWARE SWR IS NOT = -1
;BRANCH IF NO TIMEOUT
MOV $658,(SP) ;SET UP FOR TRAP RETURN
RTI
MOV $SWREG,$SWR ;POINT TO SOFTWARE SWR
MOV $DISPREG,$DISPLAY
MOV (SP)+,$ERRVEC ;RESTORE ERROR VECTOR
CLR $PASS ;CLEAR PASS COUNT
BITB $APTSIZE,$ENVM ;TEST USER SIZE UNDER APT
BEQ 670 ;YES,USE NON-APT SWITCH
MOV $DSWREG,$SWR ;NO,USE APT SWITCH REGISTER
670:
TYPE ,MSG1 ;TYPE 11/6X DIAGNOSTIC
```

```
617
618
619
620
621
622
623 001636 012700 170000
624 001642 105737 001256
625 001646 001410
626 001650 032777 000400 177256
627 001656 001074
628 001660 032777 000200 177246
629 001666 001070
630
631 001670 032777 000400 177236 20:
632 001676 001002
633 001700 000137 003056
634
635 001704 104401 040670
636 001710 104410
637 001712 012600
638 001714 020027 000005
639 001720 002002
640 001722 005700
641 001724 002003
642 001726 104401 041202
643 001732 000766
644
645 001734 000005
646 001736 012737 000214 177746
647 001744 006300
648 001746 000170 001752
649
650 001752 001764
651 001754 002156
652 001756 042232
653 001760 002412
654 001762 002540
655
656
657
658
659
660 001764 104401 041242
661 001770 104410
662 001772 012737 002006 000004
663 002000 012600
664 002002 005710
665 002004 000413
666
667 002006 012737 000006 000004 10:
668 002014 005037 000006
669 002020 022626
670 002022 104401 041312
671 002026 104401 041202
672 002032 000756

;THE FOLLOWING ROUTINE IS USED TO INTERROGATE THE PERSON
;USING THE PROGRAM WHICH NPR DEVICE HE WISHES TO USE
;
MOV $170000,R0 ;SAVE UBE ADDRESS IF RUNNING UNDER APT
TSTB $ENV ;RUNNING UNDER APT?
BEQ 20 ;BRANCH IF NO
BIT $SW00,$SWR ;ENABLE TESTS USING NPR DEVICES?
BNE UBEAPT ;BRANCH IF YES TO DEFAULT APT DEVICE:UBE
BIT $SW07,$SWR ;POWER DOWN TESTS TO BE RUN?
BNE UBEAPT ;BRANCH IF YES TO DEFAULT APT DEVICE:UBE
;
BIT $SW00,$SWR ;ENABLE TESTS USING NPR DEVICE?
BNE Q2 ;BRANCH IF YES
;
JMP START1 ;GO TO BEGINNING OF TESTS
;
Q2: TYPE ,MSG3 ;WHICH DEVICE SHOULD BE USED?
B2: RDOCT ;WAIT FOR REPLY
MOV (SP)+,R0 ;GET ANS OFF STACK
CMP R0,$5 ;WAS ANS VALID (<5)?
BGE Q1 ;BRANCH IF NO
TST R0 ;ANS VALID?
B1 ;BRANCH IF YES
Q1: TYPE ,MSG4 ;INVALID ENTRY TRY AGAIN
BR B2 ;GO WAIT FOR NEW ANS
;
B1: RESET ;INITIALIZE ALL DEVICES
MOV $214,$CCR ;CACHE OFF
ASL R0 ;ADJUST FOR WORD INDEXING
JMP $TAB(R0) ;GO ASK FURTHER QUESTIONS ON DEVICE
;
TAB: QUBEN ;POINTER TO UNIBUS EXERCISOR (NEW) QUESTIONS
QUBEO ;POINTER TO UNIBUS EXERCISOR (OLD) QUESTIONS
QRK05 ;POINTER TO RK05 QUESTIONS
GRP03 ;POINTER TO RP03 QUESTIONS
QTU10 ;POINTER TO TU10 QUESTIONS
;
;UBE NEW QUESTION AND INITIALIZE ROUTINE
;
QUBEN: TYPE ,MSG5 ;TYPE THE UBE'S DATA BUFFER ADDRESS
30: RDOCT ;WAIT FOR ANS
MOV $10,$4 ;SET UP FOR TIME OUTS
MOV (SP)+,R0 ;SEE IF DEVOCE RESPONDS
TST (R0)
BR 20 ;BRANCH IF YES
;
10: MOV $6,$4 ;RESTORE TRAP CATCHER
CLR $6 ;RESTORE TRAP CATCHER
CMP (SP)+,(SP)+ ;RESTORE STACK
TYPE ,MSG6 ;DEVICE DOES NOT RESPOND; TRAPS TO 4
40: TYPE ,MSG4 ;INVALID ENTRY, TRY AGAIN
BR 30 ;WAIT FOR ANS
```

673
674 002034 032700 007417
675 002040 001372
676 002042 022700 170000
677 002046 003367
678 002050 010001
679 002052 042700 177000
680 002056 006200
681 002060 006200
682 002062 062700 000510
683 002066 010037 001226
684 002072 010137 001224
685 002076 005721
686 002100 010137 001222
687 002104 005721
688 002106 010137 001220
689 002112 005721
690 002114 010137 001212
691 002120 005721
692 002122 010137 001216
693 002126 005721
694 002130 022121
695 002132 010137 001214
696 002136 013737 001212 001230
697 002144 012737 034046 001232
698 002152 000137 003056
699
700
701
702
703
704 002156 012737 002172 000004
705 002164 005737 170000
706 002170 000405
707 002172 022626
708 002174 104401 041312
709 002200 000137 001726
710
711 002204 012737 170006 001212
712 002212 012737 034224 001230
713 002220 012737 034174 001232
714 002226 000137 003056
715
716
717
718
719
720 002232 012737 002246 000004
721 002240 005737 177404
722 002244 000405
723
724 002246 022626
725 002250 104401 041312
726 002254 000137 001726
727
728 002260 104401 041414

20: BIT #7417,R0 ;IS ADDRESS LEGAL?
BNE 46 ;BRANCH IF NO
CMP #170000,R0 ;IS ADDRESS LEGAL?
BGT 46 ;BRANCH IF NO
UBEAPT: MOV R0,R1 ;SAVE BUFFER ADDRESS
BIC #177000,R0 ;CALCULATE DEVICE'S
R0 ;INTERRUPT VECTOR
ASR R0
ADD #510,R0 ;R0=DEVICE INT VECTOR
MOV R0,IVEC ;SAVE DEVICE INT VECTOR
MOV R1,CREG6 ;SAVE DEVICE BUFFER ADDR
TST (R1)+ ;UPDATE ADDRESS
MOV R1,CREG5 ;SAVE UBE CYCLE COUNT REG ADDR.
TST (R1)+ ;UPDATE ADDRESS
MOV R1,CREG4 ;SAVE UBE ADDRESS COUNTER ADDR.
TST (R1)+ ;UPDATE ADDRESS
MOV R1,CREG1 ;SAVE UBE CONTROL REG 1 ADDR.
TST (R1)+ ;UPDATE ADDRESS
MOV R1,CREG3 ;SAVE UBE ERROR CLEAR ADDR.
TST (R1)+ ;UPDATE ADDRESS
CMP (R1)+,(R1)+ ;UPDATE ADDRESS
MOV R1,CREG2 ;SAVE UBE CONTROL REG 2 ADDR.
MOV CREG1,EAD ;SAVE UBE ERROR ADDRESS
MOV #HUBEN,SETUP ;LOAD POINTER FOR UBE HANDLER
JMP START1 ;GO TO BEGINNING OF TEST

;/////////////////
;UBE OLD INITIALIZE ROUTINE
;/////////////////

QUBEO: MOV #18,004 ;SET UP FOR TIME OUTS
TST #0170000 ;SEE IF DATA BUFFER RESPONDS
BR 28
18: CMP (SP)+,(SP)+ ;RESTORE STACK
TYPE ,MSG6 ;DEVICE DOESN'T RESPOND
JMP Q1 ;GO CHOOSE ANOTHER DEVICE
28: MOV #170006,CREG1 ;SAVE THE GO ADDRESS
MOV #FAKE,EAD ;SETUP FAKE ADDRESS FOR ERROR TEST
MOV #HUBEO,SETUP ;LOAD PTR FOR UBE HANDLER
JMP START1 ;GO TO BEGINNING OF TEST

;/////////////////
;RK05 QUESTION AND INITIALIZE ROUTINE
;/////////////////

QRK05: MOV #18,004 ;SET UP FOR TIME OUTS
TST #0RKCS ;SEE IF RK05 STATUS REG RESPONDS
BR 28
18: CMP (SP)+,(SP)+ ;RESTORE STACK
TYPE ,MSG6 ;DEVICE DOES NOT RESPOND
JMP Q1 ;GO CHOOSE ANOTHER DEVICE
28: 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

48: RDOCT ;TYPE 0-7 <CARRIAGE RETURN>
MOV (SP)+,R0 ;WAIT FOR AMS
BGE 38 ;IS DRIVE VALID # = OR >0?
58: TYPE ,MSG4 ;BRANCH IF YES
BR 48 ;INVALID ENTRY, TRY AGAIN
;GO WAIT FOR REPLY
38: CMP #10,R0 ;IS DRIVE VALID # <7?
BLE 58 ;BRANCH IF NO
MOV #15,R1 ;PUT DRIVE #
68: ASL R0 ;IN 3 MSB OF R0
SOB R1,68 ;LOOP TILL DONE
MOV R0,00CREG2 ;SAVE DISK ADDRESS REG CONTENTS WITH SELECTED
;DRIVE AND CYLINDER ADDR, SURFACE & SECTOR=0
MOV #RKCS,CREG1 ;SAVE THE GO ADDRESS
MOV #RKCS,EAD ;SAVE THE ERROR ADDRESS
MOV #HUBEO,SETUP ;LOAD POINTER FOR RK05 HANDLER
MOV #15,00RKCS ;SET UP DRIVE #
CLR R1 ;RESET DRIVE #
R1 ;INIT COUNT
88: BIT #100,00RKDS ;DRIVE READY?
BNE 76 ;BRANCH IF YES
INC R1 ;WAIT FOR
BNE 88 ;DRIVE RDY
TYPE ,MSG13 ;DEVICE RDY BIT DOES NOT SET
JMP Q1 ;GO CHOOSE ANOTHER DEVICE
76: JMP START1 ;GO TO FIRST TEST
;/////////////////
;RP03 QUESTION AND INITIALIZE ROUTINE
;/////////////////

GRP03: MOV #18,004 ;SETUP FOR TIME OUT
TST #0RPCS ;SEE IF RP03 CONTROL REG RESPONDS
BR 28
18: CMP (SP)+,(SP)+ ;RESTORE STACK
TYPE ,MSG6 ;DEVICE DOES NOT RESPOND
JMP Q1 ;GO CHOOSE ANOTHER DEVICE
28: TYPE ,MSG7 ;WHICH DRIVE SHOULD BE USED?
48: RDOCT ;TYPE 0-7 <CARRIAGE RETURN>
MOV (SP)+,R0 ;WAIT FOR REPLY
BGE 38 ;GET DRIVE # FROM STACK
58: TYPE ,MSG4 ;BRANCH IF DRIVE #>OR=0
BR 48 ;INVALID ENTRY, TRY AGAIN
;GO WAIT FOR REPLY
38: CMP #10,R0 ;IS DRIVE VALID # > OR=7
BLE 58 ;BRANCH IF NO
MOV R0,CREG2 ;PUT DRIVE # IN HIGH BYTE
BIS #4,CREG2 ;SETUP CONTROL MASK WITH DRIVE # AND
CLR #RPCA ;A READ OPERATION (NPR DATO)
CLR #RCPDA ;SETUP CYLINDER ADDRESS REG FOR 0
;SETUP DISK ADDRESS RFG 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 RP03 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  #10,0#4    ;SETUP FOR TIME OUT
795 002546 005737 172522                TST   #0#MTC      ;SEE IF TU10 COMMAND REG RESPONDS
796 002552 000405                BR    2#             ;YES, BRANCH
797
798 002554 022626                18:  CMP   (SP)+,(SP)+ ;RESTORE STACK
799 002556 104401 001312          TYPE  #MSG6        ;DEVICE DOES NOT RESPOND
800 002562 000137 001726          JMP   Q1          ;GO CHOOSE ANOTHER DEVICE
801
802 002566 104401 001414          26:  TYPE  #MSG7        ;WHICH DRIVE SHOULD BE USED?
803                                     ;TYPE 0-7 <CARRIAGE RETURN>
804 002572 104410                46:  RDOCT                ;WAIT FOR REPLY
805 002574 012600                MOV   (SP)+,R0      ;GET DRIVE # FROM STACK
806 002576 002003                BGE   3#          ;BRANCH IF DRIVE # > OR = 0
807 002600 104401 001202          58:  TYPE  #MSG4        ;INVALID ENTRY TRY AGAIN
808 002604 000772                BR    4#          ;WAIT FOR REPLY
809
810 002606 022700 000010          38:  CMP   #10,R0      ;IS DRIVE VALID # < OR = 7
811 002612 003772                BLE   5#          ;BRANCH IF NO
812 002614 000300                SWAB   R0           ;PUT DRIVE # IN HIGH BYTE
813 002616 012737 010000 172522          MOV  #10000,0#MTC ;POWER CLEAR FOR POWER CLEAR
814 002624 012701 000010          MOV  #10,R1      ;SET DELAY FOR POWER CLEAR
815 002630 077101                SOB   R1,6#        ;WAIT FOR POWER CLEAR
816 002632 012737 000016 172522          MOV  #16,0#MTC ;SET UP TO REWIND
817 002640 050037 172522          BIS   R0,0#MTC ;SET UP DRIVE # IN CONTROL
818 002644 012701 000777          MOV  #777,R1     ;SET UP DELAY COUNT
819 002650 077101                SOB   R1,7#        ;DELAY FOR SELECT REMOTE
820 002652 032737 000100 172520          BIT   #100,0#MTC ;SEE IF DRIVE SELECTED
821 002660 001003                BNE   8#          ;BRANCH IF YES
822 002662 104401 001507          TYPE  #MSG10       ;DRIVE NOT SELECTED PROPERLY
823 002666 000744                BR    5#          ;SELECT ANOTHER
824
825 002670 032737 000004 172520          88:  BIT   #4,0#MTC ;WRITE PROTECT ON?
826 002676 001403                BEQ   9#          ;BRANCH IF NO
827 002700 104401 001546          TYPE  #MSG11       ;WRITE PROTECT ON
828 002704 000735                BR    5#          ;SELECT ANOTHER UNIT
829
830 002706 005237 172522          98:  INC   0#MTC      ;REWIND TAPE
831 002712 032737 000001 172520          100: BIT   #1,0#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 000000 001214          MOV  #40000,CREG2 ;SET UP CONTROL MASK WITH DENSITY=800BPI, 7 CHANNEL
837 002752 050037 001214          BIS   R0,CREG2  ;SET DRIVE # IN MASK
838
839 ;NOW WRITE MIN # OF BYTES ON TAPE (24)8
840
```

```
841 002756 013737 001214 172522      MOV  CREG2,0#MTC ;SET UP TO DO WRITE
842 002764 052737 000004 172522      BIS   #4,0#MTC  ;SET FUNCTION=WRITE
843 002772 012737 177760 172524      MOV  #20,0#MTBRC ;WRITE (20)8 BYTES
844 003000 012737 000000 172526      MOV  #BUFL,0#MTCMA ;SETUP ADDRESS FOR XFER
845 003006 005237 172522              INC   0#MTC      ;START WRITE
846 003012 012701 177777              MOV  #177777,R1 ;SET UP FOR MAX DELAY
847 003016 032737 000001 172520          128: BIT   #1,0#MTC ;UNIT DONE?
848 003024 001005                BNE   11#         ;BRANCH IF YES
849 003026 077105                SOB   R1,120      ;LOOP TILL MAX COUNT DONE
850 003030 104401 001645          TYPE  #MSG13       ;DEVICE RDY BIT DOES NOT SET
851 003034 000137 001704          JMP   Q2          ;TRY ANOTHER DEVICE
852
853 003040 005737 172522          118:  TST   0#MTC      ;ERROR BIT SET?
854 003044 100004                BPL   START1      ;BRANCH IF NO TO FIRST TEST
855 003046 104401 001614          TYPE  #MSG12       ;DEVICE ERROR BIT SET
856 003052 000137 001704          JMP   Q2          ;TRY ANOTHER DEVICE
857
858
859 003056 012737 003352 000004 START1: MOV  #UT4,0#4    ;SETUP FOR UNEXPECTED TRAPS TO VECTOR 4
860 003064 012737 003142 000114          MOV  #UPERR,0#114 ;SET UP FOR UNEXPECTED PARITY ERRORS.
861 003072 042737 000001 177572          BIC   #1,0#MMR0 ;KT OFF IF ON
862 003100 012706 001100          MOV  #STACK,8P ;INIT STACK POINTER
863
864 003104 010046                MOV  R0,-(SP)    ;SAVE R0 FOR MED INST
865 003106 076600                MED                ;GET CONTENTS OF LOG REG
866 003110 000022                .WORD  RLOG        ;ENABLE ERROR LOG & LOG FIRST MODE
867 003112 052700 100001          BIS   #100001,R0 ;UNLOCK ERROR LOG
868 003116 076600                MED                ;RESTORE R0
869 003120 000222                .WORD  (SP)+,R0
870 003122 012600                MOV
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,0#TMP0 ;GET ITS VECTOR
875 003142 062737 000002 001172          ADD   #2,0#TMP0  ;AND PUT A TRAP
876 003150 013777 001172 176050          MOV  #TMP0,0#VEEC ;CATCHER THERE
877 003156 005077 176010          CLR   0#TMP0
878
879
880 ;*****
881 ;*TEST 1 TEST PA MUX AND PHYSICAL ADDRESS DRIVERS
882 ;*
883 ;*IF THE INHIBIT TESTS USING KT SWITCH (SW12)=1, THIS
884 ;*TEST IS INHIBITED.
885 ;* THE PHYSICAL ADDRESS LINES A17,A16,A15 ARE CHECKED
886 ;*THAT THEY CAN CHANGE STATES. THE MEMORY IS FIRST SIZED
887 ;*TO SEE IF THERE IS MORE THAN 16K OF MEMORY. IF NO, THIS
888 ;*TEST IS SKIPPED. IF THERE IS MORE THAN 16K OF
889 ;*MEMORY, THE HIGH ADDRESS BITS A17, A16, A15 WILL BE TESTED
890 ;*WITH A FLOAT 1, 0 PATTERN.
891 ;* WHEN AN ADDRESS IS FOUND TO CONTAIN INCORRECT DATA
892 ;*AN ERROR MESSAGE IS TYPED. IN ADDITION, A HANDLER (NSSYN)
893 ;*FOR TRAPS TO VECTOP 4 WILL REPORT OTHER ADDRESSING ERRORS.
894
895 ;*****
896 003162 012737 000214 177746          TST1: MOV  #214,0#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    #SW12,05WR    ;INHIBIT TESTS USING KT?
900 003206 001402          BEO     A15    ;BRANCH IF NO
901 003210 000137 004164          JMP    TST2    ;YES,GO TO NEXT TEST
902 003214 012737 004072 000004 A15: MOV    #N5SYN,#04    ;SET UP FOR TRAPS TO 4 DUE TO ADDRESSING ERRORS
903
904
905          ;START CHECK OF NON PROGRAM LOCATIONS
906 003222 052737 000200 036034    BIS    #200,000KT11    ;TURN ON KT FOR $SIZE
907 003230 004737 035750          JSR    PC,$SIZE    ;SIZE MEMORY
908 003234 022737 001000 036322    CMP    #1000,0LSTBK    ;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          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    #77406,00KIPDR4 ;PAGE LENGTH=4K, EXPAND UP READ/WRITE
914 003266 012737 001000 172350    MOV    #1000,00KIPAR4 ;SET UP TO TEST ADDRESS BIT 15
915 003274 005237 177572          INC    00MMR0    ;TURN ON KT
916 003300 023737 036322 172350 A5: CMP    #LSTBK,00KIPAR4 ;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          ;SAVE CONTENTS OF ADDRESSES TESTING ON STACK AND PUT TEST DATA IN THEM
921
922 003312 011046          A3:  MOV    (R0),-(SP)    ;SAVE DATA
923 003314 010110          MOV    R1,(R0)    ;WRITE TEST DATA IN LOC
924 003316 005201          INC    R1    ;CALC NEW TEST DATA
925 003320 006337 172350          ASL    00KIPAR4    ;CALC NEXT TEST ADDRESS
926 003324 005737 172350          TST    00KIPAR4    ;AT LAST ADDRESS?
927 003330 001401          BEQ    A4    ;GO TEST DATA IF PAST LAST ADDR.
928 003332 000762          BR     A5    ;GO SEE IF ADDR. TO BE TESTED
929
930          ;SEE IF DATA AT ADDRESSES
931
932 003334 012701 077000          A4:  MOV    #77000,R1    ;INIT. TEST DATA REG
933 003340 012737 001000 172350    MOV    #1000,00KIPAR4 ;INIT PAR FOR LOWEST ADDR.
934 003346 023737 036322 172350 A8:  CMP    #LSTBK,00KIPAR4 ;LOOKED AT LAST ADDRESS?
935 003354 001401          BEQ    A6    ;BRANCH IF AT LAST
936 003356 101474          BLOS   A77    ;BRANCH IF PAST ADDRESS
937 003360 020110          A6:  CMP    R1,(R0)    ;WAS DATA IN LOC?
938 003362 001007          BNE     A1    ;BRANCH IF NO TO ERROR
939 003364 005201          INC    R1    ;CALC. TEST DATA
940 003366 006337 172350          ASL    00KIPAR4    ;CALC. NEXT TEST LOC.
941 003372 005737 172350          TST    00KIPAR4    ;AT LAST ADDR.?
942 003376 001464          BEQ    A77    ;BRANCH IF DONE WITH HIGH ADDR.
943 003400 000762          BR     A8    ;LOOK AT NEXT LOCATION
944
945 003402 011037 001164          A1:  MOV    (R0),$REG3    ;SAVE BAD DATA
946
947          ;ROUTINE TO CONVERT VIRTUAL ADDRESS IN R0 TO PHYSICAL ADDRESS IN R4,R5
948
949 003406 010002          MOV    R0,R2    ;GET VIRTUAL ADDRESS
950 003410 005003          CLR    R3    ;INIT SHIFT COUNTER
951 003412 006202          A8:  ASR    R2    ;SHIFT BLOCK NO. TO LSB 0-6
952 003414 005203          INC    R3    ;COUNT SHIFTS

```

```

953 003416 020327 000006          CMP    R3,#6    ;ALL DONE?
954 003422 001373          BNE     14    ;BRANCH IF NO
955 003424 010204          MOV    R2,R4    ;SAVE BLOCK #
956 003426 042704 177600          BIC    #177600,R4    ;CALC. BLOCK #
957 003432 006202          A8:  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     28    ;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          A8:  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     38    ;BRANCH IF NO
971 003476 005003          CLR    R3    ;INIT SHIFT COUNTER
972 003500 006304          A8:  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     48    ;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    # $STACK,SP ;RESTORE STACK IF LOOP
983 003542 104020          ERROR 20    ;ERROR: PHYSICAL ADDRESS LINE ERROR
984
985 003544 000137 004054          JMP    A17    ;GO TO NEXT TEST
986
987          ;RESTORE FLOATING '1' ADDRESSES
988
989 003550 012737 004000 172350 A77: MOV    #4000,00KIPAR4 ;INIT. KIPAR1 TO RESTORE 3 LOC
990 003556 012700 100000          MOV    #100000,R0    ;INIT R0 TO ADDRESS KIPAR4
991 003562 022737 004000 036322    CMP    #4000,00LSTBK ;WERE 3 LOC WRITTEN?
992 003570 101405          BLOS   A80    ;BRANCH IF YES
993 003572 022737 002000 036322    CMP    #2000,00LSTBK ;WERE 2 LOC WRITTEN?
994 003600 101402          BLOS   A81    ;BRANCH IF YES
995 003602 000405          BR     A82    ;RESTORE LAST LOC ONLY
996 003604 012610          A80: MOV    (SP)+,(R0)    ;
997 003606 012737 002000 172350 A81: MOV    #2000,00KIPAR4 ;SET UP KIPAR4 TO RESTORE 2 LOC
998 003614 012610          MOV    (SP)+,(R0)    ;
999 003616 012737 001000 172350 A82: MOV    #1000,00KIPAR4 ;SET UP KIPAR4 TO RESTORE LAST LOC
1000 003624 012610          MOV    (SP)+,(R0)    ;
1001
1002          ;NOW TEST ADDRESS 15,16,17 CAN FLOAT A '0'
1003
1004 003626 022737 003740 036322    CMP    #3740,$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    #3740,00KIPAR4 ;SET UP PAR4 SO A17=0 A16,A15=1 & ALL HIGH ADDR. BITS =1

```



```

1009 003654 011046      MOV      (R0),-(SP)      ;SAVE DATA ON STACK
1010 003656 010110      MOV      R1,(R0)       ;LOAD TEST ADDRESS WITH DATA
1011 003660 005201      INC      R1       ;CHANGE DATA
1012 003662 022737 005740 036322  CMP      # 5740,$LSTBK  ;ENOUGH MEM TO TEST A16?
1013 003670 003006      BGT      A10      ;BRANCH IF NO
1014 003672 012737 005740 172350  MOV      # 5740,$KIPAR4 ;HAVE A17,A16,A15=101
1015 003700 011046      MOV      (R0),-(SP)     ;SAVE DATA
1016 003702 010110      MOV      R1,(R0)       ;LOAD TEST DATA
1017 003704 005201      INC      R1       ;CHANGE DATA
1018
1019 003706 022737 006740 036322 A10:  CMP      # 6740,$LSTBK  ;ENOUGH MEM TO TEST A15?
1020 003714 003005      BGT      A12      ;BRANCH IF NO
1021 003716 012737 006740 172350  MOV      # 6740,$KIPAR4 ;
1022 003724 011046      MOV      (R0),-(SP)     ;SAVE DATA
1023 003726 010110      MOV      R1,(R0)       ;LOAD TEST DATA
1024
1025      ;SEE IF DATA WRITTEN PROPERLY
1026
1027 003730 012737 003740 172350 A12:  MOV      # 3740,$KIPAR4 ;SET UP ADDRESS
1028 003736 012701 177000      MOV      #177000,R1
1029 003742 020110      CMP      R1,(R0)       ;DATA OK?
1030 003744 001402      BEQ      A11      ;BRANCH IF YES
1031 003746 000137 003402      JMP      A1       ;REPORT ERROR
1032
1033 003752 022737 005740 036322 A11:  CMP      # 5740,$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,$KIPAR4 ;SETUP ADDRESS
1037 003772 020110      CMP      R1,(R0)       ;DATA OK?
1038 003774 001402      BEQ      A13      ;BRANCH YES
1039 003776 000137 003402      JMP      A1       ;REPORT ERROR
1040
1041 004002 022737 006740 036322 A13:  CMP      # 6740,$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,$KIPAR4 ;SETUP ADDRESS
1045 004022 020110      CMP      R1,(R0)       ;DATA OK?
1046 004024 001402      BEQ      A86      ;BRANCH YES
1047 004026 000137 003102      JMP      A1       ;REPORT ERROR
1048
1049      ;RESTORE DATA
1050
1051 004032 012610      MOV      (SP)+,(R0)     ;RESTORE 3 LOCS
1052 004034 012737 005740 172350  MOV      # 5740,$KIPAR4 ;
1053 004042 012610      MOV      (SP)+,(R0)     ;RESTORE 2 LOCS
1054 004044 012737 003740 172350  MOV      # 3740,$KIPAR4 ;
1055 004052 012610      MOV      (SP)+,(R0)     ;RESTORE 1 LOC
1056
1057      ;EXIT TEST
1058
1059 004054 042737 000001 177572 A17:  BIC      #1,$MMNR0      ;TURN OFF KT IF ON
1060 004062 012737 033352 000004      MOV      #UT4,$#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      NSSYN: MOV      R0,-(SP)
1066 004074 076600      10:      MED
1067 004076 000101      .WORD   HIADD
1068 004100 010037 001160      MOV      R0,$#REG1
1069 004104 076600      MED
1070 004106 000102      .WORD   LOADD
1071 004110 010037 001162      MOV      R0,$#REG2
1072 004114 012600      MOV      (SP)+,R0
1073 004116 022626      CMP      (SP)+,(SP)+
1074 004120 013737 177744 001164      MOV      $#REG,$#REG3
1075 004126 104021      EPROR   21      ;ERROR: TRAP TO VECTOR 4 WHEN TESTING PHYSICAL ADDR. LI
1076 004130 012737 033352 000004      MOV      #UT4,$#4      ;RESTORE HANDLER FOR UNEXPECT. TRAPS
1077 004136 042737 000001 177572      BIC      #1,$MMNR0      ;TURN OFF KT
1078
1079 004144 010046      MOV      R0,-(SP)      ;SAVE R0 FOR MED INST
1080 004146 076600      MED      ;GET CONTENTS OF LOG REG
1081 004150 000022      .WORD   RLOG
1082 004152 052700 100001      BIS      #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
1083 004156 076600      MED      ;UNLOCK ERROR LOG
1084 004160 000222      .WORD   WLOG
1085 004162 012600      MOV      (SP)+,R0      ;RESTORE R0
1086
1087
1088
1089      ;*****
1090      ;*TEST 2      TEST CACHE CAN BE TURNED OFF AND HIT REG CLEARED
1091      ;*
1092      ;* THE CACHE IS TURNED OFF AND THE CACHE CONTROL REG
1093      ;* IS CHECKED TO CONTAIN ALL 1'S FOR ALL SETTABLE BITS
1094      ;* EXCEPT BIT6 (NWP), NEXT THE HIT REG (HMR) IS TESTED TO BE ALL 0'S. AFTER THIS,
1095      ;* A LOW CACHE ADDRESS AND THEN A HIGH ADDRESS ARE TRIED TO
1096      ;* BE MADE HITS AND THEN THE HMR IS CHECKED TO BE ALL 0'S.
1097      ;* (LOW CACHE ADDRESS HAS PHYSICAL ADDRESS BIT 10=0).
1098      ;*
1099      ;* IF THIS TEST REPORTS A FATAL ERROR, ALL FOLLOWING TESTS
1100      ;* ARE ABORTED
1101
1102      ;*****
1103 004164 012737 000214 177746      TST2:  MOV      #214,$#CCR      ;CACHE OFF FOR SCOPE
1104 004172 000004      SCOPE
1105 004174 012737 000214 177746      MOV      #214,$#CCR      ;SET UP DATA
1106 004202 012737 004320 001234      MOV      #TST3,$KTST    ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1107 004210 013737 177746 001160      MOV      #CCR,$#REG1    ;GET (CCR)
1108 004216 022737 000214 001160      CMP      #214,$#REG1    ;WERE BITS SET IN CCR?
1109 004224 001406      BEQ      T01L01        ;BRANCH IF YES
1110 004226 012737 000214 001162      MOV      #214,$#REG2    ;SAVE GOOD DATA
1111 004234 104005      ERROR   5          ;FATAL ERROR: CACHE CONTROL REG HELD WRONG DATA
1112 004236 000137 033020      JMP      SEOP          ;ABORT TEST
1113
1114 004242 013737 177752 001160      T01L01: MOV      #HMR,$#REG1 ;SEE IF HIT MISS REG HAS ALL MISSES
1115 004250 001405      BEQ      T01L02        ;BRANCH IF YES
1116 004252 005037 001162      T01L03: CLR      #REG2    ;SAVE GOOD DATA
1117 004256 104006      ERROR   6          ;FATAL ERROR:HIT/MISS REG HELD WRONG DATA
1118 004260 000137 033020      JMP      SEOP          ;ABORT TEST
1119
1120 004264 012700 060000      T01L02: MOV      #BUFL,R0    ;INITIALIZE R0 TO LOW ADDRESS

```

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
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
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
1154
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
1169
1170 004462 030327 000004
1171 004466 001006
1172 004470 010337 001160
1173 004474 012737 000004 001162
1174 004502 000764
1175
1176 004504 030427 000010

CMP (R0),(R0) ;TRY TO MAKE LOC A HIT
MOV 00HMR,\$REG1 ;SEE IF MISS ON LOW ADDRESS SPACE
BNE T01L03 ;BRANCH IF GOT FALSE HIT
MOV 00BUFH,R0 ;SET R0=TO HIGH ADDRESS SPACE
CMP (R0),(R0) ;TRY TO MAKE HIGH ADDRESS A HIT
MOV 00HMR,\$REG1 ;SEE IF MISS AT HIGH ADDRESS
BNE T01L03 ;BRANCH IF GET FALSE HIT

;*****
;*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.
;*****
TST3: MOV \$214,\$CCR ;CACHE OFF FOR SCOPE
SCOPE
MOV #TST4,\$KTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
MOV #204,\$CCR ;TURN ON HIGH ADDRESSES OF CACHE
MOV 00CCR,R0 ;GET (CCR)
CMP #204,R0 ;WAS CACHE TURNED ON?
BEQ T02L01 ;BRANCH IF YES
BIC #14,\$CCR ;TURN CACHE OFF
MOV R0,\$REG1 ;SAVE BAD DATA
MOV #210,\$REG2 ;SAVE GOOD DATA
18: ERROR 5 ;FATAL ERROR: CACHE CONTROL REG HELD WRONG DATA
JMP \$EOP ;ABORT TEST

T02L01: MOV 00HMR,R1 ;SAVE HIT/MISS ADDRESS
MOV 00BUFH,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 R2,00HMR1 ;DID WE GET A HIT AND WAS IT SHIFTED?
BNE T02L02 ;BRANCH IF YES
MOV R2,\$REG1 ;SAVE BAD DATA
MOV #2,\$REG2 ;SAVE GOOD DATA
T02L06: ERROR 13 ;FATAL ERROR:HIT/MISS REG HELD WRONG DATA
JMP \$EOP ;ABORT TEST

T02L02: BIT R3,00HMR2 ;WAS DATA SHIFTED?
BNE T02L03 ;BRANCH IF YES
MOV R3,\$REG1 ;SAVE BAD DATA
MOV #4,\$REG2 ;SAVE GOOD DATA
BR T02L06 ;REPORT ERROR

T02L03: BIT R4,00HMR3 ;WAS DATA SHIFTED?

1177 004510 001006
1178 004512 010437 001160
1179 004516 012737 000010 001162
1180 004524 000753
1181
1182 004526 030527 000020
1183 004532 001006
1184 004534 010537 001160
1185 004540 012737 000020 001162
1186 004546 000742
1187
1188 004560 012737 000204 177746
1189 004556 021010
1190 004560 021010
1191 004562 000240
1192 004564 011102

1193 004566 030227 000040
1194 004572 001011
1195 004574 052737 000014 177746
1196 004602 010237 001160
1197 004606 012737 000054 001162
1198 004614 000717
1199
1200
1201
1202
1203
1204
1205
1206
1207 004616 012737 000214 177746
1208 004624 000004
1209 004626 012737 004712 001234
1210 004634 012737 000204 177746
1211 004642 012700 062000
1212 004646 021010
1213 004650 052737 000014 177746
1214 004656 005710
1215 004660 033727 177752 000004
1216 004666 001411
1217 004670 013737 177746 001160
1218 004676 012737 000000 001162
1219 004704 010037 001164
1220 004710 104012
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232

BNE T02L04 ;BRANCH IF YES
MOV R4,\$REG1 ;SAVE BAD DATA
MOV #10,\$REG2 ;SAVE GOOD DATA
BR T02L06 ;REPORT ERROR

T02L04: BIT R5,00HMR4 ;WAS DATA SHIFTED?
BNE T02L05 ;BRANCH IF YES
MOV R5,\$REG1 ;SAVE BAD DATA
MOV #20,\$REG2 ;SAVE GOOD DATA
BR T02L06 ;REPORT ERROR

T02L05: MOV #204,\$CCR ;TURN HALF CACHE ON
CMP (R0),(R0) ;MAKE ADDRESS A HIT
CMP (R0),(R0) ;SHIFT HIT 3 TIMES
NOP ;SHIFT HIT FOURTH TIME
MOV (R1),R2 ;SHIFT HIT FIFTH TIME AND SAVE

BIT R2,00HMR5 ;WAS DATA SHIFTED?
BNE TST4 ;BRANCH IF YES TO NEXT TEST
BIS #14,\$CCR ;TURN CACHE OFF
MOV R2,\$REG1 ;SAVE BAD DATA
MOV #54,\$REG2 ;SAVE GOOD DATA
BR T02L06 ;REPORT ERROR

;*****
;*TEST 4 TEST FORCE MISS ON HIGH ADDRESS
;*
;* A LOCATION IS PUT IN CACHE. CACHE IS THEN TURNED OFF
;*AND THE LOCATION IS CHECKED TO BE A MISS.
;*****
TST4: MOV #214,\$CCR ;TURN OFF CACHE FOR SCOPE
SCOPE
MOV #TST5,\$KTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
MOV #204,\$CCR ;TURN ON HIGH ADDRESS OF CACHE
MOV 00BUFH,R0 ;INITIALIZE R0=HIGH ADDRESS
CMP (R0),(R0) ;MAKE LOC A HIT
BIS #14,\$CCR ;TURN OFF CACHE
TST (R0) ;SEE IF LOC STILL A HIT
BIT 00HMR,00HMR2 ;WAS IT A MISS?
BEQ TST5 ;BRANCH IF YES
MOV 00CCR,\$REG1 ;SAVE (CCR)
MOV #0,\$REG2 ;SAVE PHYSICAL ADDRESS HIGH
MOV R0,\$REG3 ;SAVE PHYSICAL ADDRESS LOW
18: ERROR 12 ;ERROR:FORCE MISS BIT FAILED TO CAUSE MISS.

;*****
;*TEST 5 TEST CACHE TRACKS WHEN CACHE IS OFF
;*
;* A LOC IS MADE A HIT IN CACHE. CACHE IS THEN TURNED OFF
;*AND A SECOND LOC IS REFERENCED WHICH HAS AN OVERLAPPING
;*CACHE ADDRESS WITH THE FIRST ONE. CACHE IS TURNED ON
;*AND THE SECOND LOC IS TESTED TO BE A HIT (IMPLYING
;*CACHE HAS TRACKED).
;*****

```

1233 004712 012737 000214 177746 TST5: MOV #214,#CCR ;CACHE OFF FOR SCOPE
1234 004720 000004 SCOPE
1235 004722 012737 005044 001234 MOV #TST6,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1236 004730 012737 000204 177746 MOV #204,#CCR ;HALF CACHE ON
1237 004736 023737 002000 002000 CMP #2000,#2000 ;PUT DATA IN CACHE
1238 004744 033727 177752 000004 BIT #HMR,#HMR2 ;DATA IN CACHE?
1239 004752 001423 BEQ 18 ;BRANCH IF NO TO ERROR
1240 004754 052737 000014 177746 BIS #14,#CCR ;CACHE OFF
1241 004762 005737 002000 TST #8BUFH ;REFERENCE LOC NOT IN CACHE AND SEE IF TRACK
1242 004766 012737 000204 177746 MOV #204,#CCR ;HALF CACHE ON
1243 004774 005737 002000 TST #8BUFH ;SEE IF CACHE TRACKED
1244 005000 033727 177752 000004 BIT #HMR,#HMR2 ;HIT?
1245 005006 001016 BNE TST6 ;YES, GO TO NEXT TEST
1246
1247 005010 052737 000014 177746 BIS #14,#CCR ;CACHE OFF
1248 005016 101107 ERROR 107 ;ERROR: CACHE DID NOT TRACK WHEN FORCE MISS ON
1249 005020 000411 BR TST6 ;GO TO NEXT TEST
1250
1251 005022 052737 000014 177746 18: BIS #14,#CCR ;CACHE OFF
1252 005030 005037 001160 CLR #REG1 ;SAVE BAD ADDR.
1253 005034 012737 002000 001162 MOV #2000,#REG2 ;SAVE BAD ADDR.
1254 005042 104043 ERROR 43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT
1255
1256 ;*****
1257 ;*TEST 6 TEST DATOB OPERATION
1258 ;*
1259 ;* A DATOB IS DONE TO AN ADDRESS NOT IN CACHE AND THEN
1260 ;*THE LOC IS REFERENCED TO SEE THAT CACHE WAS NOT ALLOCATED.
1261 ;*NEXT A DATOB IS DONE TO AN ODD LOC IN CACHE AND THE
1262 ;*CORRECT BYTE IS CHECKED TO BE MODIFIED. THIS IS RE-
1263 ;*PEATED FOR AN EVEN ADDRESS.
1264
1265 ;*****
1266 005044 012737 000214 177746 TST6: MOV #214,#CCR ;TURN OFF CACHE FOR SCOPE
1267 005052 000004 SCOPE
1268 005054 012737 005364 001234 MOV #TST7,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1269 005062 012737 000204 177746 MOV #204,#CCR ;TURN ON CACHE HIGH ADDRESS
1270 005070 005737 002000 TST #2000 ;MAKE LOC BUFH IN NEXT INST. A MISS
1271 005074 112737 000377 002000 MOV #377,#8BUFH ;DO DATOB TO NON-HIT LOC TO SEE IT DOESN'T GET CACHED
1272 005102 005737 002000 TST #8BUFH ;SEE IF DATA PUT IN CACHE
1273 005106 033727 177752 000004 BIT #HMR,#HMR2 ;WAS DATA A HIT?
1274 005114 001413 BEQ T04L01 ;BRANCH IF NO
1275 005116 052737 000014 177746 BIS #14,#CCR ;TURN OFF CACHE
1276 005124 012737 000000 001160 MOV #0,#REG1 ;SAVE PHYSICAL ADDRESS HIGH
1277 005132 012737 002000 001162 MOV #8BUFH,#REG2 ;SAVE NO HIT PHYSICAL ADDRESS LOW
1278 005140 104007 18: ERROR 7 ;ERROR:DATA CACHED ON DATOB TO NO 'HIT' ADD.
1279 005142 000510 BR TST7 ;GO TO NEXT TEST
1280
1281 005144 005037 002000 T04L01: CLR #8BUFH ;INITIALIZE LOC BUFH
1282 005150 112737 177777 002001 MOV #177777,#8BUFH+1 ;DO DATOB TO A HIT LOC
1283 005156 005737 002000 TST #8BUFH ;SEE IF DATA PUT IN CACHE
1284 005162 033727 177752 000004 BIT #HMR,#HMR2 ;WAS DATA A HIT?
1285 005170 001013 BNE T04L02 ;BRANCH IF YES
1286 005172 052737 000014 177746 BIS #14,#CCR ;TURN OFF CACHE
1287 005200 012737 000000 001160 MOV #0,#REG1 ;SAVE PHYSICAL ADDRESS HIGH
1288 005206 012737 002000 001162 MOV #8BUFH,#REG2 ;SAVE PHYSICAL ADDRESS LOW

```

```

1289 005214 104010 18: ERROR 10 ;ERROR:DATA NOT CACHED ON DATOB TO A 'HIT' LOC.
1290 005216 000462 BR TST7 ;GO TO NEXT TEST
1291
1292 005220 072737 177400 002000 T04L02: CMP #177400,#8BUFH ;WAS DATA WRITTEN CORRECTLY?
1293 005226 001424 BEQ T04L03 ;BRANCH IF YES
1294 005230 013700 002000 MOV #8BUFH,R0 ;GET BAD DATA
1295 005234 052737 000014 177746 BIS #14,#CCR ;TURN OFF CACHE
1296 005242 012737 000000 001160 MOV #0,#REG1 ;SAVE PHYSICAL ADDRESS HIGH
1297 005250 012737 002000 001162 MOV #8BUFH,#REG2 ;SAVE PHYSICAL ADDRESS LOW
1298 005256 010037 001164 MOV R0,#REG3 ;SAVE BAD DATA
1299 005262 012737 177400 001166 MOV #177400,#REG4 ;SAVE GOOD DATA
1300 005270 104011 18: ERROR 11 ;ERROR:CACHE DID NOT CONTAIN PROPER DATA ON DATOB
1301 005272 042737 000010 177746 BIC #10,#CCR ;TURN CACHE ON
1302
1303 005300 005037 002000 T04L03: CLR #8BUFH ;INITIALIZE LOCATION
1304 005304 112737 000377 002000 MOV #377,#8BUFH ;DO DATOB TO EVEN ADDRESS
1305 005312 022737 000377 002000 CMP #377,#8BUFH ;WAS DATA WRITTEN CORRECTLY?
1306 005320 001421 BEQ TST7 ;BRANCH IF YES TO NEXT TEST
1307 005322 013700 002000 MOV #8BUFH,R0 ;GET BAD DATA
1308 005326 052737 000014 177746 BIS #14,#CCR ;TURN CACHE OFF
1309 005334 012737 000000 001160 MOV #0,#REG1 ;SAVE PHYSICAL ADDRESS HIGH
1310 005342 012737 002000 001162 MOV #8BUFH,#REG2 ;SAVE PHYSICAL ADDRESS LOW
1311 005350 010037 001164 MOV R0,#REG3 ;SAVE BAD DATA
1312 005354 012737 000377 001166 MOV #377,#REG4 ;SAVE GOOD DATA
1313 005362 104011 18: 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,#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,#CCR ;HALF CACHE ON
1328 005410 023737 002000 002000 CMP #2000,#2000 ;PUT LOC IN CACHE TO MAKE NEXT REF A MISS
1329 005416 033727 177752 000004 BIT #HMR,#HMR2 ;HIT?
1330 005424 001422 BEQ T05L01 ;BRANCH TO ERROR IF NO
1331 005426 005037 002000 CLR #8BUFH ;DO DATO TO A MISS ADDRESS
1332 005432 005737 002000 TST #8BUFH ;LOC IN CACHE?
1333 005436 033727 177752 000004 BIT #HMR,#HMR2 ;HIT?
1334 005444 001023 BNE T05L02 ;YES, GO TO END OF TEST
1335 005446 052737 000014 177746 BIS #14,#CCR ;CACHE OFF
1336 005454 005037 001160 CLR #REG1 ;SAVE FAILING ADDRESS
1337 005460 012737 002000 001162 MOV #8BUFH,#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,#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 #14,00CCR ;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
1353 ;*****
1354 ;*TEST 10 TEST CAN GET HIT AND FORCE MISS ON LOW CACHE ADDRESS
1355 ;*
1356 ;* THIS IS THE FIRST TEST WHERE LOW CACHE IS TURNED
1357 ;*ON. THE CACHE CONTROL REG IS FIRST LOADED AND CHECKED
1358 ;*TO CONTAIN THE PROPER VALUE. THEN ONE LOC IN LOW
1359 ;*CACHE IS MADE A HIT. THE HIT IS CHECKED FOR AND THEN
1360 ;*CACHE IS TURNED OFF AND THE LOC IS RETESTED TO NOW BE
1361 ;*A MISS.
1362 ;*
1363 ;*IF THIS TEST REPORTS A FATAL ERROR, ALL FOLLOWING TESTS
1364 ;*ARE ABORTED.
1365
1366 ;*****
1367 006000 012737 000214 177746 TST10: MOV #214,00CCR ;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 #210,00CCR ;TURN ON LOW CACHE
1371 006024 013700 177746 MOV #0CCR,R0 ;GET (CCR)
1372 006030 022700 000210 CMP #210,R0 ;(CCR) OK?
1373 006034 001413 BEQ 28 ;BRANCH IF YES
1374 006036 052737 000014 177746 BIS #14,00CCR ;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 #EOP ;ABORT PROGRAM
1379
1380 006064 012700 060000 28: 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 46 ;BRANCH IF YES
1384 006102 052737 000014 177746 BIS #14,00CCR ;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 48: BIS #14,00CCR ;CACHE OFF
1391 006134 005710 TST (R0) ;SET 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 #0CCR,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 ;*****
1401 ;*TEST 11 TEST OF TAG ADDRESS COMPARATOR
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
```

```
1401
1402
1403 ;*
1404 ;* THIS TEST USES ONE LOC IN CACHE AND LOADS IT WITH
1405 ;*VARIOUS TAG ADDRESSES. A GROUP OF MEMORY REFERENCES
1406 ;*ARE MADE FOR EACH TAG ADDRESS AND IT IS DETERMINED
1407 ;*WHETHER EACH REFERENCE WILL BE A HIT OR A MISS. THE
1408 ;*LOW ADDRESS COMPARATOR FOR BITS A11-A14 IS TESTED FIRST.
1409 ;*A TAG ADDRESS IS LOADED AND THEN ALL POSSIBLE COMBINATIONS
1410 ;*OF MEMORY ADDRESSES TO THAT CHIP ARE MADE. ALL TAG
1411 ;*COMBINATIONS ARE TRIED IN THIS MANNER FOR THESE LOW
1412 ;*ADDRESSES. THE HIGH ADDRESS COMPARATOR FOR BITS A15-
1413 ;*A17 IS HELD CONSTANT DURING THIS TIME. THE SAME PRO-
1414 ;*CEDURE IS REPEATED FOR THIS HIGH ADDRESS COMPARATOR
1415 ;*WHILE THE LOW ONE IS HELD CONSTANT. THE COMPARATOR
1416 ;*TEST IS LIMITED TO THE MEMORY SIZE AVAILABLE.
1417 ;* KIPAR4 CONTAINS THE TAG ADDRESS BEING TESTED. KIPARS
1418 ;*CONTAINS THE MEMORY REFERENCE ADDRESS BEING MADE. IF
1419 ;*INHIBIT TESTS USING KT SWITCH IS SET (SW12), THIS TEST
1420 ;*IS INHIBITED.
1421
1422 ;*****
1423 006166 012737 000214 177746 TST11: MOV #214,00CCR ;TURN OFF CACHE FOR SCOPE
1424 006174 000004 SCOPE
1425 006176 012737 006626 001234 MOV #TST12,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
1426 006204 032777 010000 172722 BIT #SW12,0SWR ;INHIBIT TESTS USING KT?
1427 006210 001402 BEQ 28 ;BRANCH IF NO
1428 006214 000137 006626 JMP #TST12 ;GO TO NEXT TEST
1429 006220 052737 000200 036034 28: BIS #200,00KT11 ;TURN ON KT FOR MEM SIZING
1430 006226 004737 035750 JSR PC,SIZE ;SIZE MEM
1431 006232 012700 172350 MOV #KIPAR4,R0 ;SET UP TO
1432 006236 012701 172310 MOV #KIPDR4,R1 ;INIT KIPDR4, 5 & KIPAR4, 5
1433 006242 005020 16: CLR (R0)+ ;FOR TESTING
1434 006244 012721 077406 MOV #77406,(R1)+ ;PAGE LENGTH=4K, EXPAND UP, READ/WRITE
1435 006250 020127 172314 CMP R1,#KIPDR6 ;KT SET UP?
1436 006254 001372 BNE 18 ;BRANCH IF NO
1437 006256 000403 BR T06L12 ;GO TO START OF TEST
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
```

```
1457 006366 052737 000014 177746 T06L02: BIS #14,0#CCR ;TURN CACHE OFF
1458 006374 012737 120000 001172 MOV #120000,0#TMP0 ;START CALC. OF PHYSICAL
1459 006402 004737 033434 JSR PC,VIP ;ADDRESS REFERENCING AND
1460 006406 013700 172350 MOV #KIPAR4,R0 ;START CALC OF TAG ADDRESS TESTING
1461 006412 005001 CLR R1 ;GET TAG FIELD TO 7 LSB R0
1462 006414 006200 ASR R0 ;GET TAG FIELD TO 7 LSB R0
1463 006416 005201 INC R1 ;GET TAG FIELD TO 7 LSB R0
1464 006420 020127 000005 CMP R1,#5 ;GET TAG FIELD TO 7 LSB R0
1465 006424 001373 BNE 18 ;GET TAG FIELD TO 7 LSB R0
1466 006426 010037 MOV R0,$REG3 ;SAVE TAG IN CASE OF ERROR
1467
1468 006432 052737 000001 177572 T06L08: BIS #1,0#MMR0 ;TURN ON KT
1469 006440 012737 000210 177746 MOV #210,0#CCR ;TURN ON HALF OF CACHE ON
1470 006446 023737 172350 172352 CMP #KIPAR4,0#KIPAR5 ;WILL REFERENCE BE A HIT
1471 006454 001422 BEQ T06L09 ;BRANCH IF YES
1472 006456 023737 100000 120000 CMP #100000,0#120000 ;LOAD ADDRESS IN TAG FIELD & THEN REFERENCE IT
1473 006464 033727 177752 000004 BIT 0#MMR,0#MMR2 ;WAS REFERENCE A MISS?
1474 006472 001435 BEQ T06L10 ;BRANCH IF YES
1475 006474 052737 000014 177746 BIS #14,0#CCR ;TURN OFF CACHE
1476 006502 012737 006440 001110 MOV #T06L08,0#0LPERR ;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 #1,0#MMR0 ;TURN OFF KT
1479 006520 000442 BR TST12 ;GO TO NEXT TEST
1480
1481 006522 023737 100000 120000 T06L09: CMP #100000,0#120000 ;LOAD ADDRESS IN TAG FIELD & THEN REFERENCE IT
1482 006530 033727 177752 000004 BIT 0#MMR,0#MMR2 ;WAS REF. A HIT?
1483 006536 001013 BNE T06L10 ;BRANCH IF YES
1484 006540 052737 177746 BIS #14,0#CCR ;TURN OFF CACHE FOR ERROR REPORT
1485 006546 012737 006440 001110 MOV #T06L08,0#0LPERR ;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 #1,0#MMR0 ;TURN OFF KT
1488 006564 000420 BR TST12 ;GO TO NEXT TEST
1489
1490 006566 023727 172352 000740 T06L10: CMP 0#KIPAR5,#740 ;REFERENCED ADDRESSES OF LOWER ADDR. COMP.?
1491 006574 001640 BEQ T06L03 ;BRANCH IF YES
1492 006576 022404 BLT T06L11 ;BRANCH IF PAR5 STILL REF. LOW ADDR. COMP.
1493 006600 062737 001000 172352 ADD #1000,0#KIPAR5 ;ADDRESS NEXT LOC FOR HIGH ADDR. COMPARTOR
1494 006606 000624 BR T06L01 ;SEE IF DONE
1495 006610 062737 000040 172352 T06L11: ADD #40,0#KIPAR5 ;ADDRESS NEXT LOC FOR LOW ADDR. COMP.
1496 006616 000620 BR T06L01 ;SEE IF DONE
1497
1498 006620 042737 000001 177572 T06L04: BIC #1,0#MMR0 ;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 REWRITTEN 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 TST12: MOV #214,0#CCR ;TURN OFF CACHE
```

```
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 #T09L01,0#PVEC ;SETUP PARITY TRAP HANDLER
1516 006652 012737 000310 177746 MOV #310,0#CCR ;TURN ON HALF OF CACHE & WWP
1517 006660 013700 177746 MOV 0#CCR,R0
1518 006664 020027 000310 CMP R0,#310 ;WERE BITS SET IN CCR?
1519 006670 001414 BEQ T09L02 ;BRANCH IF YES
1520 006672 012737 000014 177746 MOV #14,0#CCR ;TURN CACHE OFF
1521 006700 010037 001160 MOV R0,$REG1 ;SAVE BAD DATA
1522 006704 012737 000310 001162 MOV #310,$REG2 ;SAVE GOOD DATA
1523 006712 104026 ERROR 26 ;ERROR! CACHE CONTROL REG HELD WRONG DATA
1524 006714 012737 000310 177746 MOV #310,0#CCR ;TURN ON HALF OF CACHE & WWP
1525
1526 006722 005037 060000 T09L02: CLR 0#BUFL ;WRITE WRONG PARITY IN 1 LOC
1527 006726 012737 000210 177746 MOV #210,0#CCR ;WWP OFF
1528 006734 005737 060000 TST 0#BUFL ;SEE IF GET PARITY TRAP
1529
1530
1531 ;RID CACHE OF BAD PARITY
1532 006740 012737 000214 177746 MOV #214,0#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 #214,0#CCR ;CACHE OFF IF ON
1545 006776 004737 035134 JSR PC,SWEEP ;GO PURGE CACHE
1546
1547
1548
1549 007002 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
1550 007004 076600 MED ;GET CONTENTS OF LOG REG
1551 007006 000022 .WORD RLOG
1552 007010 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
1553 007014 076600 MED .WORD WLOG ;UNLOCK ERROR LOG
1554 007016 000222 .WORD
1555 007020 012600 MOV (SP)+,R0 ;RESTORE R0
1556
1557 007022 022626 CMP (SP)+,(SP)+ ;RESTORE STACK
1558 007024 012737 007072 000114 MOV #T09L03,0#PVEC ;SET UP PARITY TRAP HANDLER
1559 007032 012737 000310 177746 MOV #310,0#CCR ;TURN HALF OF CACHE ON & WWP
1560 007040 005037 060000 CLR 0#BUFL ;WRITE WRONG PARITY IN ONE LOC
1561 007044 012737 000214 177746 MOV #214,0#CCR ;CACHE OFF
1562 007052 005737 060000 TST 0#BUFL ;SEE IF SEE GET PARITY TRAP
1563
1564 007056 T09L04:
1565
1566 ;RID CACHE OF BAD PARITY
1567 007056 012737 000214 177746 MOV #214,0#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,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 R0,-(SP) ;SAVE R0 FOR MED INST
1586 007112 076600 MED ;GET CONTENTS OF LOG REG
1587 007114 000022 .WORD RLOG
1588 007116 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
1589 007122 076600 MED ;UNLOCK ERROR LOG
1590 007124 000222 .WORD WLOG
1591 007126 012600 MOV (SP)+,R0 ;RESTORE R0
1592
1593 007130 012737 033142 000114 MOV #UPERR,PPVEC ;RESTORE HANDLER FOR UNEXPECTED PARITY ERRORS
1594 007136 000400 BR TST13 ;GO TO NEXT TEST
1595
1596
1597 ;*****
1598 ;*TEST 13 TEST OF TAG PARITY GENERATOR/CHECKER
1599 ;*
1600 ;* THIS TEST INITIALLY SIZES MEMORY TO DETERMINE THE
1601 ;*MAXIMUM TESTABLE ADDRESS. KIPAR4 IS SETUP TO WRITE ALL
1602 ;*TAG COMBINATIONS UP TO THE MAX ADDRESS INTO ONE CACHE
1603 ;*LOCATION. FIRST, THE LOCATION IS WRITTEN WITH WRONG
1604 ;*PARITY FOR ALL THE TAG COMBINATIONS AND A PARITY TRAP
1605 ;*IS FORCED AND TESTED FOR. AFTER EACH TRAP, THE PROGRAM
1606 ;*CHECKS THAT THE TRAP WAS FROM THE TAG FIELD AND THAT
1607 ;*THE TAG CONTENTS (FROM ERROR LOG) WAS WHAT WAS WRITTEN.
1608 ;*THIS LATTER CHECK IS DONE PRIMARILY TO ENSURE THAT THE
1609 ;*TRAP WAS BECAUSE WRONG PARITY WAS WRITTEN AND NOT DUE
1610 ;*TO A FAILING LOCATION.
1611 ;* SECOND, THE LOCATION IS WRITTEN WITH GOOD PARITY FOR
1612 ;*ALL TAG COMBINATIONS. THE LOC IS REFERENCED AND ANY
1613 ;*PARITY ERROR IS DETECTED AND REPORTED.
1614 ;* IF INHIBIT TESTS USING KT SWITCH (SW12) IS SET,
1615 ;*THIS TEST IS INHIBITED.
1616
1617 ;*****
1618 007140 012737 000214 177746 TST13: MOV #214,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,SWNR ;INHIBIT TEST USING KT11?
1622 007164 001402 BEQ 18 ;BRANCH IF NO
1623 007166 000137 010230 JMP #TST14 ;GO TO NEXT TEST
1624 007172 052737 000200 036034 18: BIS #200,SKT11 ;TURN ON KT FOR $SIZE
```

```
1625 007200 004737 035750 JSR PC,$SIZE ;SIZE MEMORY
1626 007204 012737 007354 000114 MOV #T07L01,PPVEC ;SET UP TO HANDLE PARITY TRAPS
1627 007212 012737 077406 172310 MOV #77406,SKIPDR4 ;PAGE LENGTH=4K, EXPAND UP, READ/WRITE
1628 007220 005037 172350 CLR #KIPAR4 ;INIT PAR
1629 007224 052737 000001 177572 BIS #1,MMNR0 ;TURN KT ON
1630 007232 023737 172350 036322 T07L04: CMP #KIPAR4,MLSTBK ;TESTED ALL POSSIBLE ADDRESSES?
1631 007240 003402 BLE 18 ;BRANCH IF NO TO CONTINUE
1632 007242 000137 007650 JMP T07L02 ;TEST GOOD PARITY GEN.
1633 007246 012737 000310 177746 18: MOV #310,CCR ;TURN HALF OF CACHE ON & WWP
1634
1635 007254 013737 100000 100000 T07L03: MOV #100000,MLSTBK ;WRITE WRONG PARITY IN LOC
1636 007262 012737 000210 177746 MOV #210,CCR ;WWP 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,STMP0 ;GET ADDRESS JUST TESTED
1646 007314 004737 033434 JSR PC,VIP ;CALC ITS PHYSICAL ADDRESS
1647 007320 013737 172350 001172 MOV #KIPAR4,STMP0 ;GET PAR FOR TAG CALC.
1648 007326 004737 033606 JSR PC,TAG ;CALC WHAT TAG CONTENTS SHOULD BE
1649 007332 013737 001172 001164 MOV #STMP0,AREG3 ;SAVE (TAG) SHOULD BE
1650 007340 012737 007232 001110 MOV #T07L04,MLPERR ;SET UP RETURN FOR LOOP ON ERROR
1651 007346 104027 ERROR 27 ;ERROR: TEST OF TAG PARITY GENERATOR/CHECKER FAILED
1652 ; DID NOT GET PARITY TRAP FROM TAG FIELD
1653 ; WHEN WROTE WRONG PARITY
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 R0,-(SP) ;SAVE R0 FOR MED INST
1665 007370 076600 MED ;GET CONTENTS OF LOG REG
1666 007372 000022 .WORD RLOG
1667 007374 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
1668 007400 076600 MED ;UNLOCK ERROR LOG
1669 007402 000222 .WORD WLOG
1670 007404 012600 MOV (SP)+,R0 ;RESTORE R0
1671
1672 007406 022626 CMP (SP)+,(SP)+ ;RESTORE STACK
1673 007410 032737 000040 177744 BIT #40,EREGR ;TRAP DUE TO PARITY ERROR IN TAG?
1674 007416 001040 BNE T07L06 ;BRANCH IF YES
1675 007424 076600 MED ;GET LOG INFORMATION
1676 007422 000102 .WORD LOAD
1677 007424 010037 001162 MOV R0,AREG2 ;SAVE INFORMATION
1678 007430 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
1679 007432 000101 .WORD RSE
1680 007434 000300 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
```

```
1681 007436 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
1682 007442 010037 001160 MOV R0,$REG1 ;SAVE ADDRESS
1683 007446 076600 MED ;GET TAG LOG INFO.
1684 007450 000107 .WORD RTAG
1685 007452 000300 SWAB R0 ;PUT TAG IN LOW BYTE
1686 007454 042700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
1687 007460 010037 001164 MOV R0,$REG3 ;SAVE TAG
1688 007464 013737 172350 001172 MOV #KIPAR4,$TMP0 ;GET PAR FOR TAG CALC.
1689 007472 004737 033606 JSR PC,TAG ;FIND GOOD CONTENTS OF TAG
1690 007476 013737 001172 MOV $TMP0,$REG4 ;SAVE GOOD DATA
1691 007504 012737 007232 001166 MOV $T07L04,$$LPERR ;SET UP RETURN FOR ERROR LOOP
1692 007512 104030 001110 ERROR 30 ;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
; DID NOT GET PARITY TRAP FROM TAG FIELD
; WHEN WROTE WRONG PARITY
1693 ;GO TO END OF TEST
1694
1695 007514 000137 010214 JMP #T07L05
1696
1697 007520 013737 172350 001172 T07L06: MOV #KIPAR4,$TMP0 ;GET PAR FOR TAG CALC.
1698 007526 004737 033606 JSR PC,TAG ;CALC WHAT TAG SHOULD BE
1699 007532 076600 MED ;GET TAG LOG INFO.
1700 007534 000107 .WORD RTAG
1701 007536 000300 SWAB R0 ;PUT TAG IN LOW BYTE
1702 007540 042700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
1703 007544 020037 001172 CMP R0,$TMP0 ;DATA OK?
1704 007550 001432 BEQ T07L07 ;BRANCH IF YES
1705 007552 010037 001164 MOV R0,$REG3 ;SAVE TAG
1706 007556 076600 MED ;GET LOG INFORMATION
1707 007560 000107 .WORD LOADD
1708 007562 010037 001162 MOV R0,$REG2 ;SAVE INFORMATION
1709 007566 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
1710 007570 000107 .WORD RSER
1711 007572 000300 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
1712 007574 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
1713 007600 010037 001160 MOV R0,$REG1 ;SAVE ADDRESS
1714 007604 013737 001172 001166 MOV $TMP0,$$REG4 ;SAVE GOOD DATA
1715 007612 012737 007232 001110 MOV $T07L04,$$LPERR ;SET UP RETURN FOR ERROR LOOP
1716 007620 104031 ERROR 31 ;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
; TAG FIELD HELD WRONG DATA ON PARITY TRAP
; MORE THAN THREE ERRORS?
1717 ;BRANCH IF NO
1718 007622 123727 001103 000003 CMPB #$$ERFLG,$3
1719 007630 101402 BLOS T07L07
1720 007632 000137 010214 JMP T07L05 ;GO TO END OF TEST
1721
1722 007636 062737 000040 172350 T07L07: ADD #40,$KIPAR4 ;CALC NEXT TAG ADDRESS TO TEST
1723 007644 000137 007232 JMP T07L04 ;CONTINUE TEST
1724
1725 007650 T07L02:
1726
1727 ;RID CACHE OF BAD PARITY
1728 007650 012737 000214 177746 MOV #214,$$CCR ;CACHE OFF IF ON
1729 007656 004737 035134 JSR PC,SWEEP ;GO PURGE CACHE
1730
1731
1732 007662 012737 007734 000114 MOV $T07L00,$$PVEC ;SET UP FOR PARITY ERRORS
1733 007670 005037 172350 CLR #KIPAR4 ;INIT ADDRESSES
1734 007674 023737 172350 036322 T07L09: CMP #KIPAR4,$$LSTBK ;TESTED ALL POSSIBLE ADDRESSES?
1735 007702 003144 BGT T07L05 ;YES GO TO END OF TEST
1736 007704 012737 000210 177746 MOV #210,$$CCR ;TURN HALF CACHE ON
```

```
1737 007712 013737 100000 100000 MOV #100000,$$100000 ;GENERATE PARITY IN CACHE
1738 007720 005737 102000 TST #102000 ;CHECK PARITY IN CACHE
1739 007724 062737 000040 172350 ADD #40,$KIPAR4 ;CALC NEXT TAG ADDRESS TO TEST
1740 007732 000070 BR T07L09 ;CONTINUE TEST
1741
1742 007734 T07L08:
1743
1744 ;RID CACHE OF BAD PARITY
1745 007734 012737 000214 177746 MOV #214,$$CCR ;CACHE OFF IF ON
1746 007742 004737 035134 JSR PC,SWEEP ;GO PURGE CACHE
1747
1748
1749
1750 007746 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
1751 007750 076600 MED ;GET CONTENTS OF LOG REG
1752 007752 000022 .WORD RLOG
1753 007754 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
1754 007760 076600 MED ;UNLOCK ERROR LOG
1755 007762 000222 .WORD WLOG
1756 007764 012000 MOV (SP)+,R0 ;RESTORE R0
1757
1758 007766 022626 CMP (SP)+,(SP)+ ;RESTORE STACK
1759 007770 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
1760 007772 000107 .WORD RSER
1761 007774 000300 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
1762 007776 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
1763 010002 010037 001160 MOV R0,$REG1 ;SAVE ADDRESS
1764 010006 076600 MED ;GET LOG INFORMATION
1765 010010 000107 .WORD LOADD
1766 010012 010037 001162 MOV R0,$REG2 ;SAVE INFORMATION
1767 010016 032737 000040 177744 BIT #40,$$EREG ;ERROR DUE TO TAG ERROR?
1768 010024 001424 BEQ T07L10 ;BRANCH IF NO
1769 010026 076600 MED ;GET TAG LOG INFO.
1770 010030 000107 .WORD RTAG
1771 010032 000300 SWAB R0 ;PUT TAG IN LOW BYTE
1772 010034 042700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
1773 010040 010037 001164 MOV R0,$REG3 ;SAVE TAG
1774 010044 013737 172350 001172 MOV #KIPAR4,$TMP0 ;GET PAR FOR TAG CALC.
1775 010052 004737 033606 JSR PC,TAG ;CALC GOOD DATA
1776 010056 013737 001172 001166 MOV $TMP0,$$REG4 ;SAVE GOOD DATA
1777 010064 012737 007674 001110 MOV $T07L09,$$LPERR ;SET UP FOR ERROR LOOP
1778 010072 104034 ERROR 34 ;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
; PARITY ERROR OCCURRED IN TAG FIELD
1779 ;GO TO END OF TEST
1780 010074 000447 BR T07L05
1781
1782 010076 032737 000100 177744 T07L10: BIT #100,$$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 R0,$REG3 ;SAVE INFORMATION
1787 010116 013737 102000 001166 MOV #102000,$$REG4 ;SAVE GOOD DATA
1788 010124 012737 007674 001110 MOV $T07L09,$$LPERR ;INIT LOOP ON ERROR RETURN
1789 010132 104033 ERROR 33 ;ERROR: TEST OF TAG PARITY GEN/CHECKER FAILED
; PARITY ERROR IN LOW BYTE OF DATA
1790 ;GO TO END OF TEST
1791 010134 000427 BR T07L05
1792
```

```

1793 010136 032737 000200 177744 T07L11: BIT #200,##EREG ;ERROR IN HIGH BYTE?
1794 010144 001414 ;BRANCH IF NO
1795 010146 076600 ;GET LOG INFORMATION
1796 010150 000106 .WORD CDH
1797 010152 010037 001164 MOV R0,#REG3 ;SAVE INFORMATION
1798 010156 013737 102000 001166 MOV #102000,#REG4 ;SAVE GOOD DATA
1799 010164 012737 007674 001110 MOV #T07L09,##LPERR ;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 177744 001164 T07L12: MOV -4(SP),#REG3 ;SAVE PC OF ERROR
1805 010204 012737 007674 001110 MOV #T07L09,##LPERR ;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,##MMR0 ;TURN KI OFF
1809 010222 012737 033142 000114 MOV #UPERR,##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 (R0 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,##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,##PVEC ;SET UP PARITY TRAP HANDLER
1834 010254 012700 062000 MOV #BUFH,R0 ;GET TEST ADDRESS
1835 010260 005001 CLR R1 ;INIT FLAG TO INDIC. TESTING LOW BYTE
1836 010262 005037 001166 T08L06: CLR ##REG4 ;SAVE DATA IF ERROR
1837 010266 005037 001160 CLR ##REG1 ;SAVE ADDRESS IF ERROR
1838 010272 010037 001162 MOV R0,##REG2 ;SAVE ADDRESS IF ERROR
1839 010276 012737 000204 177746 MOV #204,##CCR ;TURN ON HALF OF CACHE
1840 010304 005737 062000 TST #BUFH ;PUT LOC IN CACHE
1841 010310 052737 000100 177746 BIS #100,##CCR ;ENABLE WRITE WRONG PARITY
1842 010316 112710 000000 MOVB #0,(R0) ;DO DATOB TO LOC & WWP
1843 010322 042737 000100 177746 BIC #100,##CCR ;WWP OFF
1844 010330 005737 062000 TST #BUFH ;FORCE PARITY TRAP
1845 010334 012737 000214 177746 MOV #214,##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

```

```

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 R0,-(SP) ;SAVE R0 FOR MED INST
1854 010360 076600 MED ;GET CONTENTS OF LOG REG
1855 010362 000022 .WORD RLOC
1856 010364 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
1857 010370 076600 MED ;UNLOCK ERROR LOG
1858 010372 000222 .WORD WLOG
1859 010374 012600 MOV (SP)+,R0 ;RESTORE R0
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 R0,#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 R0,#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 R0 ;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,##REG2 ;SAVE ADDRESS IF ERROR
1889 010502 005000 CLR R0 ;INIT. TEST DATA REG
1890 010504 010037 001166 T08L10: MOV R0,#REG4 ;SAVE DATA IF ERROR
1891 010510 012737 000304 177746 MOV #304,##CCR ;TURN HALF OF CACHE ON & WWP
1892 010516 010037 062000 MOV R0,##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,##CCP ;TURN CACHE OFF AFTER TRAP
1902
1903 010554 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
1904 010556 076600 MED ;GET CONTENTS OF LOG REG

```



```

1905 010560 000022      .WORD      RLOG
1906 010562 052700 100001  BIS      #100001,R0      ;ENABLE ERROR LOG & LOG FIRST MODE
1907 010566 076600      MED
1908 010570 000222      .WORD      WLOG      ;UNLOCK ERROR LOG
1909 010572 012600      MOV      (SP)+,R0      ;RESTORE R0
1910
1911 010574 022626      CMP      (SP)+,(SP)+      ;RESTORE STACK
1912 010576 032737 000100 177744 BIT      #100,0,0,EREG      ;TRAP FROM LOW BYTE?
1913 010604 001011      BNE      T08L09      ;BRANCH IF YES
1914
1915 010606 076600      MED
1916 010610 000106      .WORD      CDL      ;GET LOG INFORMATION
1917 010612 010037 001164  MOV      R0,0,REG3      ;SAVE INFORMATION
1918 010616 012737 010504 001110 MOV      #T08L10,0,0,LPERR ;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      T08L02      ;GO TO END OF TEST
1922
1923 010630 032737 000200 177744 T08L09: BIT      #200,0,0,EREG      ;TRAP FROM HIGH BYTE?
1924 010636 001011      BNE      T08L11      ;BRANCH IF YES
1925
1926 010640 076600      MED
1927 010642 000106      .WORD      CDH      ;GET LOG INFORMATION
1928 010644 010037 001164  MOV      R0,0,REG3      ;SAVE INFORMATION
1929 010650 012737 010504 001110 MOV      #T08L10,0,0,LPERR ;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      T08L02      ;GO TO NEXT TEST
1933
1934 010662 022700 177777 T08L11: CMP      #177777,R0      ;ALL WRITE WRONG PARITY PATTERNS CAED?
1935 010666 001403      BEQ      T08L12      ;BRANCH IF YES
1936 010670 062700 000401  ADD      #401,R0      ;GENERATE DATA FOR HIGH AND LOW BYTE
1937 010674 000703      BR      T08L10      ;GO TEST IT
1938
1939 010676 012737 010740 000114 T08L12: MOV      #T08L13,0,0,PVEC ;SET UP FOR PARITY ERRORS
1940 010704 005000      CLR      R0      ;INIT TEST DATA REG
1941 010706 012737 000204 177746 T08L14: MOV      #204,0,0,CCR ;TURN HALF OF CACHE ON
1942 010714 010037 062000      MOV      R0,0,0,BUFH ;GEN PARITY AND STORE IN CACHE
1943 010720 005737 062000      TST      0,0,BUFH ;TEST PARITY
1944 010724 022700 177777 T08L16: CMP      #177777,R0      ;ALL GOOD PARITY PATTERNS CKED?
1945 010730 001450      BEQ      T08L02      ;BRANCH YES TO END OF TEST
1946 010732 062700 000401  ADD      #401,R0      ;GENERATE DATA FOR HIGH & LOW BYTE
1947 010736 000763      BR      T08L14      ;TEST IT
1948
1949 010740 052737 000014 177746 T08L13: BIS      #14,0,0,CCR ;TURN CACHE OFF
1950
1951 010746 010046      MOV      R0,-(SP)      ;SAVE R0 FOR MED INST
1952 010750 076600      MED
1953 010752 000022      .WORD      RLOG      ;GET CONTENTS OF LOG REG
1954 010754 052700 100001  BIS      #100001,R0      ;ENABLE ERROR LOG & LOG FIRST MODE
1955 010760 076600      MED
1956 010762 000222      .WORD      WLOG      ;UNLOCK ERROR LOG
1957 010764 012600      MOV      (SP)+,R0      ;RESTORE R0
1958
1959 010766 022626      CMP      (SP)+,(SP)+      ;RESTORE STACK
1960 010770 010037 001166  MOV      R0,0,REG4      ;SAVE GOOD DATA

```

```

1961 010774 076600      MED
1962 010776 000106      .WORD      RDAT      ;GET LOG INFORMATION
1963 011000 010037 001164  MOV      R0,0,REG3      ;SAVE INFORMATION
1964 011004 013700 001166  MOV      0,REG4,R0      ;RESTORE R0
1965 011010 032737 000100 177744 BIT      #100,0,0,EREG      ;PARITY ERROR LOW BYTE?
1966 011016 001405      BEQ      T08L15      ;BRANCH IF NO
1967 011020 012737 010706 001110 MOV      #T08L14,0,0,LPERR ;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      T08L02      ;GO TO END OF TEST
1971
1972 011032 032737 000200 177744 T08L15: BIT      #200,0,0,EREG ;PARITY ERROR HIGH BYTE?
1973 011040 001731      BEQ      T08L16      ;TEST NEXT PATTERN IF NO
1974 011042 012737 010706 001110 MOV      #T08L14,0,0,LPERR ;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      T08L02:
1979
1980      ;RID CACHE OF BAD PARITY
1981 011052 012737 000214 177746 MOV      #214,0,0,CCR ;CACHE OFF IF ON
1982 011060 004737 035134 JSR      PC,SWEEP      ;GO PURGE CACHE
1983
1984
1985 011064 012737 033142 000114 MOV      #UPERR,0,0,114 ;RESTORE UNEXPECTED PARITY ERROR HANDLER
1986 011072 000137 012000 JMP      0,TST15      ;GO TO NEXT TEST
1987
1988
1989      ;=12000
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.

```

2017 ;*(I.E. STARTING WITH ALL LOC INVALIDATED AND THEN READING
2018 ;*AND WRITING THE V BIT.)
2019 ;*
2020 ;*R0 CONTAINS THE CACHE ADDRESS BEING TESTED.
2021
2022 ;*NOTE:TEST FOR DUAL ADDRESSING FOR LOCATIONS WHICH OVERLAP
2023 ;* THE PARITY TRAP ADDRESSES 114,116 IS NOT DONE
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 #R0,R0 ;INIT STARTING ADDRESS
2032 012034 012701 001000 MOV #1000,R1 ;INIT COUNT FOR 1/2 K
2033 012040 005020 1$: CLR (R0)+ ;WRITE CACHE
2034 012042 077102 SOB R1,1$ ;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 #T24L02,00PVEC ;SET UP PARITY HANDLER
2044
2045 012076 020027 060114 T24L05: CMP R0,#BUFL114 ;TESTING PARITY AREA?
2046 012102 001412 BEQ T24L22 ;DON'T TEST ADDRESS IF YES
2047 012104 020027 060116 CMP R0,#BUFL116 ;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 ;WVP 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,00PVEC ;RESTORE UNEXP. PARITY ERROR HANDLER
2076 012220 005740 1$: TST -(R0) ;WAS LOC INVALIDATED?
2077 012222 033727 177752 000004 BIT #HMR,#HMR2 ;LOC A MISS? (INVALIDATED?)
2078 012230 001402 BEQ 2$ ;BRANCH IF YES
2079 012232 000137 012712 JMP T24L06 ;REPORT ERROR
2080 012236 005010 2$: CLR (R0) ;WRITE LOC
2081 012240 020027 060000 CMP R0,#BUFL ;AT LAST LOC?
2082 012244 001365 BNE 1$ ;BRANCH IF NO
2083
2084 ;NOW WRITE/READ VALID BIT WITH PATTERN REVERSED
2085
2086 012246 012737 012316 000114 T24L10: MOV #T24L07,00PVEC ;SET UP FOR PARITY TRAP
2087 012254 012700 061776 MOV #BUFL+1776,R0 ;INIT TEST ADDR.
2088 012260 012737 000310 177746 T24L08: MOV #310,#CCR ;WRITE WRONG PARITY & CACHE ON
2089 012266 005010 CLR (R0) ;WRITE WRONG PARITY
2090 012270 012737 000210 177746 MOV #210,#CCR ;WVP OFF
2091 012276 005710 TST (R0) ;FORCE TRAP
2092 012300 012737 000214 177746 MOV #214,#CCR ;CACHE OFF
2093 012306 012737 012260 001110 MOV #T24L08,00LPERR ;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 R0,-(SP) ;SAVE R0 FOR MED INST
2099 012320 076600 MED ;GET CONTENTS OF LOG REG
2100 012322 000022 .WORD RLOG
2101 012324 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
2102 012330 076600 MED ;UNLOCK ERROR LOG
2103 012332 000222 .WORD WLOG
2104 012334 012600 MOV (SP)+,R0 ;RESTORE R0
2105
2106 012336 162700 000002 SUB #2,R0 ;LOOK AT NEXT ADDRESS
2107 012342 062706 000004 ADD #4,SP ;ADJUST STACK
2108 012346 020027 057776 CMP R0,#BUFL-2 ;HALF CACHE WRITTEN?
2109 012352 001342 BNE T24L08 ;BRANCH IF NO
2110
2111 012354 012737 033142 000114 T24L12: MOV #UPERR,00PVEC ;ADJUST ADDRESS
2112 012362 062700 000002 ADD #2,R0 ;READ LOC
2113 012366 005710 TST (R0) ;MISS? (LOC INVALIDATED?)
2114 012370 033727 177752 000004 BIT #HMR,#HMR2 ;BRANCH IF YES
2115 012376 001407 BEQ T24L09 ;CACHE OFF
2116 012400 012737 000214 177746 MOV #214,#CCR ;INIT RETURN FOR ERROR LOOP
2117 012406 012737 012246 001110 MOV #T24L10,00LPERR ;REPORT ERROR
2118 012414 000536 BR T24L06
2119
2120 012416 005010 T24L09: CLR (R0) ;WRITE LOC
2121 012420 020027 061776 CMP R0,#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 T24L17: MOV #T24L16,00PVEC ;SET UP PARITY HANDLER
2127 012434 020027 060114 CMP R0,#BUFL114 ;TESTING PARITY AREA?
2128 012440 001417 BEQ T24L13 ;DON'T TEST ADDRESS IF YES

```

```

2129 012442 020027 000116 CMP R0,#BUFL1116 ;TESTING PARITY AREA?
2130 012446 001414 BEQ T24L13 ;DON'T TEST ADDRESS IF YES
2131
2132 012450 005710 TST (R0) ;LOC IN CACHE?
2133 012452 033727 177752 000004 BIT 0#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 #T24L10,0#LPERR ;INIT RETURN FOR ERROR LOOP
2137 012476 000466 BR T24L14 ;REPORT ERROR
2138
2139 012500 052737 000100 177746 T24L13: BIS #100,0#CCR ;SET WRITE WRONG PARITY
2140 012506 005010 CLR (R0) ;WRITE WRONG PARITY
2141 012510 012737 000210 177746 MOV #210,0#CCR ;WMP OFF
2142 012516 005710 TST (R0) ;FORCE TRAP
2143 012520 012737 000214 177746 MOV #214,0#CCR ;CACHE OFF
2144 012526 012737 012246 001110 MOV #T24L10,0#LPERR ;INIT RETURN FOR ERROR LOOP
2145 012534 000466 BR T24L15 ;REPORT ERROR
2146
2147 012536 062706 000004 T24L16: ADD #4,SP ;RESTORE STACK
2148 012542 162700 000002 SUB #2,R0 ;LOOK AT NEXT ADDR.
2149
2150 012546 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
2151 012550 076600 MED ;GET CONTENTS OF LOG REG
2152 012552 000022 .WORD RLOG
2153 012554 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
2154 012560 076600 MED ;UNLOCK ERROR LOG
2155 012562 000222 .WORD WLOG
2156 012564 012600 MOV (SP)+,R0 ;RESTORE R0
2157
2158 012566 020027 057776 CMP R0,#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 #214,0#CCR ;CACHE OFF IF ON
2165 012602 004737 035134 JSR PC,SWEAP ;GO PURGE CACHE
2166
2167
2168 012606 012737 033142 000114 MOV #UPERR,0#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 #214,0#CCR ;CACHE OFF
2173 012632 005037 001100 CLR $REG1 ;SAVE FAILING ADDR
2174 012636 010037 001162 MOV R0,$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 #214,0#CCR ;CACHE OFF
2179 012654 005037 001100 CLR $REG1 ;SAVE FAILING ADDRESS
2180 012660 010037 001162 MOV R0,$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

```

```

2185 012670 012737 000214 177746 T24L04: MOV #214,0#CCR ;CACHE OFF
2186 012676 005037 001100 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 #214,0#CCR ;CACHE OFF
2192 012720 005037 001100 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 ;*TEST 16 TEST TAG PARITY BIT FOR LOW CACHE ADDRESSES
2200 ;*
2201 ;*
2202 ;* THE TEST OF THE TAG PARITY BIT IS NOT COMPLETE UNTIL
2203 ;*THE TAG P BIT TEST FOR THE SECOND HALF OF CACHE AND THE
2204 ;*MSB ADDRESS (A10) TO CACHE TAG FIELD TEST ARE RUN. TWO
2205 ;*TAG ADDRESSES ARE USED TO GENERATE A PARITY BIT OF 1 AND
2206 ;*0. THE FIRST ADDRESS IS CHOSEN FROM A TEST BUFFER AREA
2207 ;*AND THE SECOND IS CHOSEN TO LIE 1K AWAY. A WRITE/READ
2208 ;*PROCEDURE IS DONE WHICH CHECKS THE P BIT AND DUAL ADD-
2209 ;*RESSING FOR HALF OF CACHE. INITIALLY THE P BIT IS WRITTEN
2210 ;*WITH ONE PARITY PATTERN IN HALF OF CACHE. THEN STARTING
2211 ;*AT THE LOW HALF CACHE ADDRESS, THE LOC IS READ AND THEN
2212 ;*WRITTEN WITH THE OPPOSITE PARITY. THIS IS SEQUENTIALLY
2213 ;*REPEATED WITH INCREASING ADDRESSES UNTIL THE HIGH HALF
2214 ;*CACHE ADDRESS IS REACHED. THEN STARTING AT THE HIGH ADDR,
2215 ;*THE SECOND PARITY PATTERN IS READ AND THE LOC IS REWRITTEN
2216 ;*WITH THE FIRST. THIS IS SEQUENTIALLY REPEATED, DECREASING
2217 ;*THE ADDRESS, UNTIL THE LOW HALF CACHE ADDRESS IS REACHED.
2218 ;*A SECOND PASS IS THEN MADE WITH THE PARITY PATTERN RE-
2219 ;*VERSED. A PARITY ERROR HANDLER IS SETUP TO DETECT PARITY
2220 ;*ERRORS. ALSO, LOCS WHICH SHOULD BE HITS ARE CHECKED FOR
2221 ;*AND REPORTED IF NO HIT OCCURRED.
2222 ;*
2223 ;*R0, R1 CONTAIN ADDRESSES TO GENERATE COMPLIMENTARY TAG
2224 ;*PARITY BITS.
2225 ;*****
2226
2227 012734 012737 000214 177746 TST16: MOV #214,0#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 #T11L01,0#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 #210,0#CCR ;TURN HALF CACHE ON
2234 012774 012701 001000 T11L02: MOV #1000,R1 ;INIT COUNTER
2235 013000 005720 IS: 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

```

```

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 012700 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 #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 #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  MOV #210,#CCR      ;HALF CACHE ON IF OFF
2262 013112 012737 013104 001110  MOV #T11L12,#$LPERR ;SETUP RETURN FOR ERROR IF ONE OCCURS
2263 013120 012700 054000    MOV #BUFL-4000,R0    ;SET UP FOR SECOND PASS.
2264 013124 000723          BR T11L02       ;GO TEST SECOND PASS
2265
2266 013126          T11L01:
2267
2268          ;RID CACHE OF BAD PARITY
2269 013126 012737 000214 177746  MOV #214,#CCR      ;CACHE OFF IF ON
2270 013134 004737 035134    JSR PC,SWEEP      ;GO PURGE CACHE
2271
2272
2273
2274 013140 010046          MOV R0,-(SP)      ;SAVE R0 FOR MED INST
2275 013142 076000          MED              ;GET CONTENTS OF LOG REG
2276 013144 000022          .WORD RLOG
2277 013146 052700 100001    BIS #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
2278 013152 076000          MED              ;UNLOCK ERROR LOG
2279 013154 000222          .WORD WLOG
2280 013156 012000          MOV (SP)+,R0      ;RESTORE R0
2281
2282 013160 076000          MED              ;GET LOG INFOR FOR PHY. ADDR. A17,A16
2283 013162 000101          .WORD R5ER
2284 013164 000300          SWAB R0
2285 013166 042700 177776    BIC #177776,R0    ;PUT PHY. ADDR A17, A16 IN LOW BYTE
2286 013172 010037 001160    MOV R0,$REG1     ;ONLY LOOK AT A17, A16
2287 013176 076000          MED              ;SAVE ADDRESS
2288 013200 000102          .WORD LOADD
2289 013202 010037 001162    MOV R0,$REG2     ;GET LOG INFORMATION
2290 013206 076000          MED              ;SAVE INFORMATION
2291 013210 000100          .WORD RJAM
2292 013212 032700 000400    BIT #400,R0      ;GET LOG INFORMATION
2293 013216 001410          BEQ T11L07       ;ERROR IN BACKING STORE?
2294 013220 011637 001164    MOV (SP),$REG3   ;BRANCH IF NO
2295 013224 162737 000002 001166  SUB #2,$REG4   ;GET PC+2 WHERE ERROR OCCURRED
2296 013232 022626          CMP (SP)+,(SP)+   ;SAVE PC WHERE ERROR OCCURRED
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352

```

```

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,$EREG   ;ERROR IN TAG?
2302 013250 001411          BEQ T11L08       ;BRANCH NO
2303 013252 076000          MED              ;GET TAG LOG INFO.
2304 013254 000107          .WORD RTAG
2305 013256 000300          SWAB R0
2306 013260 042700 177400    BIC #177400,R0    ;PUT TAG IN LOW BYTE
2307 013264 010037 001164    MOV R0,$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,$EREG   ;ERROR: TAG PARITY ERROR WHEN TESTING TAG P BIT
2312 013302 001406          BEQ T11L09       ;GO TO NEXT TEST
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352

```

```

2353 ;*IS REWRITTEN WITH THE FIRST. THIS IS SEQUENTIALLY RE-
2354 ;*PEATED DECREASING THE ADDRESS UNTIL THE LOW HALF CACHE
2355 ;*ADDRESS IS REACHED. A SECOND PASS IS THEN MADE WITH
2356 ;*THE PARITY PATTERN REVERSED. A PARITY ERROR HANDLER IS
2357 ;*SETUP TO DETECT PARITY ERRORS. ALSO, LOGS WHICH SHOULD
2358 ;*BE HITS ARE CHECKED FOR AND REPORTED IF NO HIT OCCURRED.
2359 ;*
2360 ;*R0, R1 CONTAIN DATA WHICH GENERATE OPPOSITE PARITY. R3
2361 ;*INDICATES WHICH PASS IS BEING DONE.
2362
2363 ;*****
2364 013406 012737 000214 177746 TST17: 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 18: MOV R0,(R5)+ ;WRITE DATA PARITY PATTERN
2374 013456 077102 SOB R1,16 ;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 012704 000401 MOV #401,R0 ;SET UP PATTERN B FOR FIRST PASS
2379 013474 005703 TST R3 ;FIRST PASS?
2380 013476 001401 BEQ 28 ;BRANCH IF YES
2381 013500 005000 CLR R0 ;SET UP PARITY PATTERN A FOR SECOND PASS
2382 013502 005715 TST (R5) ;SEE IF PARITY UNCHANGED
2383 013504 033727 177752 000004 28: 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 R1,24 ;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 001401 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 18: 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 R1,10 ;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,#LPERR ;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

```

```

2409 013614 052737 000014 177746 BIS #14,#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,#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,#CCR ;CACHE OFF
2419
2420 013654 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
2421 013656 076600 MED ;GET CONTENTS OF LOG REG
2422 013660 000022 ,WORD RLOG
2423 013662 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
2424 013666 076600 MED ;UNLOCK ERROR LOG
2425 013670 000222 ,WORD WLOG
2426 013672 012600 MOV (SP)+,R0 ;RESTORE R0
2427
2428 013674 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
2429 013676 000101 ,WORD R8ER
2430 013700 000300 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
2431 013702 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
2432 013706 010037 001160 MOV R0,#REG1 ;SAVE ADDRESS
2433 013712 076600 MED ;GET LOG INFORMATION
2434 013714 000102 ,WORD LOADD
2435 013716 010037 001162 MOV R0,#REG2 ;SAVE INFORMATION
2436 013722 032737 000040 177744 BIT #40,#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 R0 ;PUT TAG IN LOW BYTE
2443 013752 042700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
2444 013756 010037 001164 MOV R0,#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 R0 ;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,#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 R0,#REG3 ;SAVE INFORMATION
2459 014030 104050 FRROR 50 ;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA P BITS
2460 014032 000665 BR T12L09 ;GO TO END OF TEST
2461
2462 014034 032777 000100 163702 T12L12: BIT #100,#EREG ;ERPOP IN LOW BYTE?
2463 014042 001406 BEQ T12L14 ;BRANCH IF NO
2464 014044 076600 MED ;GET LOG INFORMATION

```

```

2465 014046 000106      .WORD   CDL
2466 014050 010037 001164  MOV     R0,#REG3      ;SAVE INFORMATION
2467 014054 104051      ERROR    51          ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BITS
2468 014056 000653      BR       T12L08      ;GO TO END OF TEST
2469
2470 014060 016637 177774 001164 T12L14: MOV     -4(SP),#REG3 ;GET PC+2 OF TRAP
2471 014066 162737 000002 001164 SUP     #2,#REG3      ;SAVE PC OF TRAP
2472 014074 104001      ERROR    1          ;ERROR: UNEXP. PARITY ERROR IN BACKING STORE
2473 014076 000643      BR       T12L08      ;GO TO END OF TEST
2474
2475
2476
2477
2478 ;*****
2479 ;*TEST 20      TEST THE VALID BIT FOR HIGH HALF OF CACHE
2480 ;*
2481 ;* THE TEST OF THE VALID BIT IS NOT COMPLETE UNTIL THE
2482 ;*VALID TEST FOR THE SECOND HALF OF CACHE IS RUN. THIS
2483 ;*IS THE FIRST TEST WHERE THIS ENTIRE HALF OF CACHE ADDRESSES ARE
2484 ;*EXERCISED.
2485 ;* DURING THE ENTIRE TEST ONLY ONE TAG AND DATA VALUE IS
2486 ;*USED. INITIALLY, THE ENTIRE HALF OF CACHE WHICH IS
2487 ;*ENABLED (FORCE MISS OFF) IS WRITTEN AND CHECKED THAT ALL
2488 ;*ITS ADDRESSES CAN BE MADE HITS. FOLLOWING THIS, A WRITE/
2489 ;*READ PROCEDURE IS DONE WHICH VERIFIES THAT THE LOCATIONS
2490 ;*CAN BE VALIDATED/INVALIDATED AND THAT THERE IS NO DUAL
2491 ;*ADDRESSING PROBLEM FOR THE V BIT. FIRST THE VALID BIT
2492 ;*IS SET FOR HALF OF CACHE, THEN STARTING AT THE LOWEST
2493 ;*HALF CACHE ADDRESS, EACH LOC IS TESTED TO BE A HIT (VALID
2494 ;*SET) AND THEN INVALIDATED VIA WRITING WRONG PARITY AND
2495 ;*FORCING A TRAP. THIS IS DONE INCREASING THE ADDRESS
2496 ;*UNTIL HALF OF CACHE IS READ AND WRITTEN. NEXT, STARTING
2497 ;*AT THE HIGH HALF CACHE ADDRESS, EACH LOC IS READ, TESTED
2498 ;*TO BE A MISS (VALID=0) AND THEN WRITTEN TO SET THE VALID
2499 ;*BIT. THIS IS DONE, DECREASING THE ADDRESS EACH TIME,
2500 ;*TILL THE LOW ADDRESS IS REACHED. THIS PROCEDURE IS THEN
2501 ;*REPEATED FOR A SECOND PASS WITH THE PATTERN REVERSED.
2502 ;*(I.E. STARTING WITH ALL LOC INVALIDATED AND THEN READING
2503 ;*AND WRITING THE V BIT.)
2504 ;*
2505 ;*R0 CONTAINS THE CACHE ADDRESS BEING TESTED.
2506
2507 014100 012737 000214 177746 TST20: MOV     #214,#CCR      ;CACHE OFF FOR SCOPE
2508 014106 000004      SCOPE
2509 014110 012737 015000 001234      MOV     #TST21,SKTST      ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
2510 014116 012737 000204 177746      MOV     #204,#CCR      ;HALF CACHE ON
2511 014124 012700 062000      MOV     #BUFH,R0      ;INIT STARTING ADDRESS
2512 014130 012701 001000      MOV     #1000,R1      ;INIT COUNT FOR 1/2 K
2513 014134 005020      18:    CLR     (R0)+      ;WRITE CACHE
2514 014136 077102      SOB     R1,18      ;LOOP TILL HALF CACHE WRITTEN
2515
2516 014140 005740      T24H20: TST     -(R0)      ;SEE IF DATA IN CACHE
2517 014142 033727 177752 000004      BIT     #0HMR,#HMR2      ;HIT? (VALID BIT SET?)
2518 014150 001002      BNE     T24H19      ;BRANCH IF YES
2519 014152 000137 014670      JMP     T24H01      ;REPORT ERROR
2520 014156 020027 062000      T24H19: CMP     R0,#BUFH      ;HALF CACHE TESTED?

```

```

2521 014162 001366      BNE     T24H20      ;BRANCH IF NO
2522
2523 014164 012737 014242 000114      MOV     #T24H02,0#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     #0HMR,#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      ;WNP 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     R0,-(SP)      ;SAVE R0 FOR MED INST
2539 014244 076600      MED      ;GET CONTENTS OF LOG REG
2540 014246 000022      .WORD   RLOG
2541 014250 052700 100001      BIS     #100001,R0      ;ENABLE ERROR LOG & LOG FIRST MODE
2542 014254 076600      MED      ;UNLOCK ERROR LOG
2543 014256 000222      .WORD   WLOG
2544 014260 012600      MOV     (SP)+,R0      ;RESTORE R0
2545
2546 014262 062700 000002      ADD     #2,R0      ;LOOK AT NEXT ADDR.
2547 014266 062706 000004      ADD     #4,SP      ;RESTORE STACK
2548 014272 020027 064000      CMP     R0,#BUFH+2000      ;HALF ADDRESSES TESTED?
2549 014276 001340      BNE     T24H05      ;BRANCH IF NO
2550
2551 014300 012737 033142 000114      MOV     #UPERR,0#PVEC      ;RESTORE UNEXP. PARITY ERROR HANDLER
2552 014306 005740      TST     -(R0)      ;WAS LOC INVALIDATED?
2553 014310 033727 177752 000004      BIT     #0HMR,#HMR2      ;LOC A MISS? (INVALIDATED?)
2554 014316 001402      BEQ     28      ;BRANCH IF YES
2555 014320 000137 014756      JMP     T24H06      ;REPORT ERROR
2556 014324 005010      28:    CLR     (R0)      ;WRITE LOC
2557 014326 020027 062000      CMP     R0,#BUFH      ;AT LAST LOC?
2558 014332 001365      BNE     16      ;BRANCH IF NO
2559
2560 ;NOW WRITE/READ VALID BIT WITH PATTERN REVERSED
2561
2562 014334 012737 014404 000114 T24H10: MOV     #T24H07,0#PVEC      ;SET UP FOR PARITY TRAP
2563 014342 012700 063776      MOV     #BUFH+1776,R0      ;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      ;WNP OFF
2567 014364 005710      TST     (R0)      ;FORCE TRAP
2568 014366 012737 000214 177746      MOV     #214,#CCR      ;CACHE OFF
2569 014374 012737 014346 001110      MOV     #T24H08,0#LPERR      ;INIT RETURN FOR ERROR LOOP
2570 014402 000557      BP       T24H15      ;REPORT ERROR IF DID NOT TRAP
2571
2572 014404      T24H07:
2573
2574 014404 010046      MOV     R0,-(SP)      ;SAVE R0 FOR MED INST
2575 014406 076600      MED      ;GET CONTENTS OF LOG REG
2576 014410 000022      .WORD   RLOG

```

```

2577 014412 052700 100001    BIS      #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
2578 014416 076600            MED            ;UNLOCK ERROR LOG
2579 014420 000222            .WORD      WLOG
2580 014422 012600            MOV      (SP)+,R0    ;RESTORE R0
2581
2582 014424 162700 000002    SUB      #2,R0    ;LOOK AT NEXT ADDRESS
2583 014430 062700 000004    ADD      #4,SP    ;ADJUST STACK
2584 014434 020027 061776    CMP      R0,#BUFH-2 ;HALF CACHE WRITTEN?
2585 014440 001342            BNE      T24H08    ;BRANCH IF NO
2586
2587 014442 012737 033142 000114 T24H12: MOV      #UPERR,#PVEC
2588 014450 062700 000002    ADD      #2,R0    ;ADJUST ADDRESS
2589 014454 005710            TST      (R0)    ;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,#LPERR ;INIT RETURN FOR ERROR LOOP
2594 014502 000525            BR       T24H06    ;REPORT ERROR
2595
2596 014504 005010            T24H09: CLR      (R0)    ;WRITE LOC
2597 014506 020027 063776    CMP      R0,#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 T24H17: MOV      #T24H16,#PVEC ;SET UP PARITY HANDLER
2603 014522 005710            TST      (R0)    ;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,#LPERR ;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      (R0)    ;WRITE WRONG PARITY
2612 014562 012737 000204 177746    MOV      #204,#CCR    ;WNP OFF
2613 014570 005710            TST      (R0)    ;FORCE TRAP
2614 014572 012737 000214 177746    MOV      #214,#CCR    ;CACHE OFF
2615 014600 012737 014334 001110    MOV      #T24H10,#LPERR ;REPORT ERROR
2616 014606 000455            BR       T24H15
2617
2618 014610 062700 000004 T24H16: ADD      #4,SP    ;RESTORE STACK
2619 014614 162700 000002    SUB      #2,R0    ;LOOK AT NEXT ADDR.
2620
2621 014620 010046            MOV      R0,-(SP) ;SAVE R0 FOR MED INST
2622 014622 076600            MED            ;GET CONTENTS OF LOG REG
2623 014624 000022            .WORD      RLOG
2624 014626 052700 100001    BIS      #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
2625 014632 076600            MED            ;UNLOCK ERROR LOG
2626 014634 000222            .WORD      WLOG
2627 014636 012600            MOV      (SP)+,R0    ;RESTORE R0
2628
2629 014640 020027 061776    CMP      R0,#BUFH-2 ;ALL ADDR TESTED?
2630 014644 001326            BNE      T24H17    ;BRANCH IF NO
2631
2632 014646                                T24H18:

```

```

2633                                ;RID CACHE OF BAD PARITY
2634
2635 014646 012737 000214 177746    MOV      #214,#CCR    ;CACHE OFF IF ON
2636 014654 004737 035134    JSR      PC,SWEEP ;GO PURGE CACHE
2637
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 PAPIY 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-

```

```

2689 ;*VERSED. A PARITY ERROR HANDLER IS SETUP TO DETECT PARITY
2690 ;*ERRORS. ALSO, LOGS WHICH SHOULD BE HITS ARE CHECKED FOR
2691 ;*AND REPORTED IF NO HIT OCCURRED.
2692 ;*
2693 ;*R0, R1 CONTAIN ADDRESSES TO GENERATE COMPLIMENTARY TAG
2694 ;*PARITY BITS.
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 #T11H01,00PVEC ;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 #204,0#CCR ;TURN HALF CACHE ON
2704 015040 012701 001000 T11H02: MOV #1000,R1 ;INIT COUNTER
2705 015044 005720 18: TST (R0)+ ;PUT PARITY PATTERN IN TAG FIELD
2706 015046 077102 SOB R1,18 ;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 MOV #1000,R1 ;INIT COUNTER
2722 015122 005742 T11H11: 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 BIS #1,R3 ;SET FLAG TO INDIC. SECOND PASS
2731 015150 012737 000204 177746 T11H12: MOV #204,0#CCR ;HALF CACHE ON IF OFF
2732 015156 012737 015150 001110 MOV #T11H12,0#LPERR ;SETUP RETURN FOR ERROR IF ONE OCCURS
2733 015164 012700 056000 MOV #BUFH-4000,R0 ;SET UP FOR SECOND 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 #214,0#CCR ;CACHE OFF IF ON
2740 015200 004737 035134 JSR PC,SWEEP ;GO PURGE CACHE
2741
2742
2743
2744 015204 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST

```

```

2745 015206 076600 MED ;GET CONTENTS OF LOG REG
2746 015210 000022 .WORD RLOG
2747 015212 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
2748 015216 076600 MED ;UNLOCK ERROR LOG
2749 015220 000222 .WORD WLOG
2750 015222 012600 MOV (SP)+,R0 ;RESTORE R0
2751
2752 015224 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
2753 015226 000101 .WORD RSER
2754 015230 000300 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
2755 015232 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
2756 015236 010037 001160 MOV R0,$REG1 ;SAVE ADDRESS
2757 015242 076600 MED ;GET LOG INFORMATION
2758 015244 000102 .WORD LOADD
2759 015246 010037 001162 MOV R0,$REG2 ;SAVE INFORMATION
2760 015252 076600 MED ;GET LOG INFORMATION
2761 015254 000100 .WORD RJAM
2762 015256 032700 000400 BIT #400,R0 ;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,$REG4 ;SAVE PC WHERE ERROR OCCURRED
2766 015276 022626 CMP (SP)+,(SP)+ ;RESTORE STACK
2767 015300 104001 ERROR 1 ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
2768 015302 000460 BR T11H06 ;GO TO NEXT TEST
2769
2770 015304 022626 T11H07: CMP (SP)+,(SP)+ ;RESTORE STACK
2771 015306 032737 000040 177744 BIT #40,0#EREG ;ERROR IN TAG?
2772 015314 001411 BEQ T11H08 ;BRANCH NO
2773 015316 076600 MED ;GET TAG LOG INFO.
2774 015320 000107 .WORD RTAG
2775 015322 000300 SWAB R0 ;PUT TAG IN LOW BYTE
2776 015324 042700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
2777 015330 010037 001164 MOV R0,$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 #100,0#EREG ;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 R0,$REG3 ;SAVE INFORMATION
2786 015360 104046 ERROR 46 ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING TAG P BIT
2787 015362 000430 BR T11H06 ;NEXT TEST
2788
2789 015364 T11H09:
2790 015366 076600 MED ;GET LOG INFORMATION
2791 015368 000106 .WORD CDH
2792 015370 010037 001164 MOV R0,$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 #14,0#CCR ;CACHE OFF
2797 015406 162700 000002 SUB #2,R0 ;GET BAD ADDRESS
2798 015412 010037 001162 MOV R0,$REG2 ;SAVE BAD ADDRESS
2799 015416 000407 BR T11H10 ;REPORT ERROR
2800 015420 052737 000014 177746 T11H05: BIS #14,0#CCR ;CACHE OFF

```



```

2001 015476 010237 001162
2002 015432 062702 000002
2003 015436 005037 001160
2004 015442 104043
2005
2006 015444 012737 033142 000114
2007 015452 052737 000014 177746
2008 015460 000547
2009
2010
2011 016000
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044 016000 012737 000214 177746
2045 016006 000004
2046 016010 012737 016646 001234
2047 016016 032777 010000 163110
2048 016024 001402
2049 016026 000137 016646
2050 016032 052737 000200 036034
2051 016040 004737 035750
2052 016044 012737 016266 000114
2053 016052 013737 036322 172350
2054
2055
2056

```

```

MOV R2,#REG2 ;SAVE BAD ADDRESS
ADD #2,R2 ;RESTORE R2 TO FAILING ADDR.+2
T11H10: CLR #REG1 ;SAVE BAD ADDRESS
ERROR 43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT

T11H06: MOV #SUPERR,#PVEC ;RESTORE PARITY TRAP HANDLER
BIS #14,#CCR ;CACHE OFF WHEN CROSS CACHE ADDR. BOUNDARY
BR TST22 ;GO TO NEXT TEST

;=16000 ;ADJUST ADDRESS SPACE FOR NEXT TEST

;*****
;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.

;*****
TST22: MOV #214,#CCR ;CACHE OFF FOR SCOPE
SCOPE
MOV #TST23,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
BIT #SW12,#SWR ;INHIBIT TESTS USING KT?
BEQ 36 ;CONTINUE TEST IF NO
JMP #TST23 ;GO TO NEXT TEST
36: BIS #200,#SKT11 ;KT ON FOR $SIZE
JSR PC,$SIZE ;SIZE MEMORY
MOV #T13L01,#PVEC ;SET UP PARITY ERROR HANDLER
MOV #0#LSTBK,0#KIPAR4 ;SET UP PAR4 FOR ADDRESS PATTERN A

;CALC COMPLEMENT TAG PATTERN B

```

```

2057 016060 013700 036322
2058 016064 005100
2059 016066 005001
2060 016070 005201
2061 016072 006300
2062 016074 100775
2063 016076 006200
2064 016100 077102
2065 016102 042700 000037
2066
2067 016106 010037 172352
2068
2069 016112 012700 100000
2070 016116 012701 122000
2071 016122 005003
2072 016124 005004
2073 016126 012702 001000
2074 016132 052737 000001 177572
2075 016140 012737 000210 177746
2076
2077 016146 005720
2078 016150 077202
2079
2080 016152 012702 001000
2081 016156 005740
2082 016160 033727 177752 000004
2083 016166 001002
2084 016170 000137 016540
2085 016174 005741
2086 016176 077211
2087
2088 016200 005204
2089 016202 012702 001000
2090 016206 005711
2091 016210 033727 177752 000004
2092 016216 001002
2093 016220 000137 016606
2094 016224 005721
2095 016226 005720
2096 016230 077212
2097
2098 016232 005703
2099 016234 001402
2090 016236 000137 016634
2091 016242 052703 000001
2092 016246 012737 016254 001110
2093 016254 012700 120000
2094 016260 012701 102000
2095 016264 000717
2096
2097 016266 052737 000014 177746
2098
2099 016274 010046
2100 016276 076600
2101 016300 000022
2102 016302 052700 100001

```

```

MOV #0#LSTBK,R0 ;GET TEST PATTERN A AND
COM R0 ;CALC PATTERN B
CLR R1
18: INC R1
ASL R0
BMI 18
28: ASR R0
SOB R1,28
BIC #37,R0 ;ONLY COMPLEMENT TAG ADDR. BITS

MOV R0,#KIPAR5 ;SET UP PAR5 FOR ADDRESS PATTERN B

MOV #100000,R0 ;INIT R0 TO ADD PATTERN A
MOV #122000,R1 ;INIT R1 TO ADD PATTERN B
CLR R3 ;INIT FLAG FOR PASS 1
T13L02: CLR R4 ;INIT INDICATOR FOR ERROR LOOP 1

MOV #1000,R2 ;INIT ADDR. COUNTER
BIS #1,#MMR0 ;TURN KT ON
MOV #210,#CCR ;TURN HALF OF CACHE ON

18: TST (R0)+ ;WRITE PATTERN IN CACHE
SOB R2,18 ;ALL DONE? BRANCH IF NO

T13L03: MOV #1000,R2 ;INIT. ADDR. COUNTER
TST -(R0) ;READ CACHE TAG BITS
BIT #0#MMR,#HMR2 ;HIT?
BNE 28 ;BRANCH IF YES
JMP T13L04 ;REPORT ERROR
28: TST -(R1) ;WRITE NEW PATTERN IN TAG
SOB R2,T13L03 ;HALF ADDR. TESTED? BRANCH IF NO

INC R4 ;SET INDICATOR FOR ERROR LOOP 2
MOV #1000,R2 ;INIT. ADDR. COUNTER
T13L05: TST (R1) ;READ CACHE TAG BITS
BIT #0#MMR,#HMR2 ;HIT?
BNE 38 ;BRANCH IF YES
JMP T13L06 ;REPORT ERROR
38: TST (R1)+ ;UPDATE FOR NEXT ADDRESS
TST (R0)+ ;WRITE NEW PATTERN IN TAG
SOB R2,T13L05

TST R3 ;SECOND PASS?
BEQ 28 ;CONTINUE TEST IF NO
JMP T13L07 ;GO TO END OF TEST
28: BIS #1,R3 ;SET FLAG FOR SECOND PASS
MOV #T13L15,#0#LPERR ;INIT RETURN FOR ERROR LOOP IF ERROR OCCURS
T13L15: MOV #120000,R0 ;INIT. R0 TO ADDR. PATTERN B
MOV #102000,R1 ;INIT. R1 TO ADDR. PATTERN A
BR T13L02 ;GO TEST SECOND PASS

T13L01: BIS #14,#CCR ;CACHE OFF

MOV R0,-(SP) ;SAVE R0 FOR MED INST
MED ;GET CONTENTS OF LOG REG
.WORD RLOG
BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE

```

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
2971 016530 010037 001164    MOV     R0,#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     R0,#TMP0    ;GET VIRTUAL ADDRESS TESTED
2977 016552 004737 033434          JSR     PC,VIP     ;SAVE ADDRESS TESTED
2978 016556 062700 000002          ADD     #2,R0     ;ADJUST ADDRESS WHEN LOOP
2979 016562 004737 033634          JSR     PC,PAR     ;GET PAR TESTED
2980 016566 000000          ,WORD    0      ;INDICATOR FOR R0
2981 016570 004737 033606          T13L13: JSR     #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 000411          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     #MMR0    ;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 ;* R0 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,#KTST    ;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,R0     ;SET UP FLOAT 1 PATTERN
3018 016700 012702 000000          MOV     #BUFL,R2    ;SET UP TEST ADDRESS
3019 016704 012737 000210 177746 T14L02: MOV     #210,#CCR    ;HALF CACHE ON
3020 016712 010012          T14L06: MOV     R0,(R2)    ;WRITE CACHE
3021 016714 020012          CMP     R0,(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

```

```

3025 016721 005700      TST      R0      ;ALL SHIFTS FOR FLOAT 1 PATTERN DONE?
3026 016726 100482      BMI      T14L05  ;BRANCH IF YES
3027 016730 006300      ASL      R0      ;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,R0 ;SET UP FLOAT 0 PATTERN
3032 016744 000762      BR       T14L06  ;GO TEST IT
3033
3034 016746 005700      T14L04: TST      R0      ;ALL SHIFTS FOR FLOAT 0 PATTERN DONE?
3035 016750 100155      BPL      T14L07  ;GO TO END OF TEST IF YES
3036 016752 000261      SEC      R0      ;SET CARRY BIT FOR ROTATE
3037 016754 006100      ROL      R0      ;ROTATE FLOAT 0 PATTERN
3038 016756 000755      BR       T14L06  ;TEST IT
3039
3040 016760 052737 000014 177746 T14L01: BIS      #14,0*CCR ;CACHE OFF
3041
3042 016766 010046      MOV      R0,-(SP) ;SAVE R0 FOR MED INST
3043 016770 076600      MED      R0      ;GET CONTENTS OF LOG REG
3044 016772 000022      .WORD    RLOG
3045 016774 052700 100001      BIS      #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
3046 017000 076600      MED      R0      ;UNLOCK ERROR LOG
3047 017002 000222      .WORD    WLOG
3048 017004 012600      MOV      (SP)+,R0 ;RESTORE R0
3049
3050 017006 011637 001164      MOV      (SP),0*REG3 ;GET PC+2 OF ERROR
3051 017012 162737 000002 001164      SUB      #2,0*REG3 ;SAVE PC OF ERROR
3052 017020 022626      CMP      (SP)+,(SP)+ ;RESTORE STACK
3053 017022 010046      MOV      R0,-(SP) ;SAVE R0 FOR MED INST
3054 017024 076600      MED      R0      ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3055 017026 000101      .WORD    RSER
3056 017030 000300      SWAB     R0      ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3057 017032 042700 177776      BIC      #177776,R0 ;ONLY LOOK AT A17, A16
3058 017036 010037 001160      MOV      R0,0*REG1 ;SAVE ADDRESS
3059 017042 076600      MED      R0      ;GET LOG INFORMATION
3060 017044 000102      .WORD    LOADD
3061 017046 010037 001162      MOV      R0,0*REG2 ;SAVE INFORMATION
3062 017052 076600      MED      R0      ;GET LOG INFORMATION
3063 017054 000100      .WORD    RJAM
3064 017056 032700 000400      BIT      #400,R0 ;ERROR IN BACKING STORE?
3065 017062 001403      BEQ      T14L08  ;BRANCH IF NO
3066 017064 010026      MOV      R0,(SP)+ ;RESTORE R0
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      T14L09: MOV      (SP),0*REG4 ;SAVE GOOD DATA
3071 017076 012737 016704 001110      MOV      #14L02,0*0*LPERR ;INIT RETURN FOR ERROR LOOP
3072 017104 032737 000100 177744      BIT      #100,0*0*REG ;LOW BYTE PARITY ERROR?
3073 017112 001416      BEQ      T14L09  ;BRANCH IF NO
3074 017114 076600      MED      R0      ;GET LOG INFORMATION
3075 017116 000106      .WORD    CDL
3076 017120 010037 001164      MOV      R0,0*REG3 ;SAVE INFORMATION
3077 017124 012600      MOV      (SP)+,R0 ;RESTORE R0
3078 017126 104056      ERROR    56 ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD
3079 017130 123727 001103 000003 T14L12: CMPB     #0*0*ERFLG,#3 ;MORE THAN 3 ERRORS?
3080 017136 101062      BHI      T14L07  ;STOP TESTING IF YES

```

```

3081 017140 012737 000210 177746      MOV      #210,0*CCR ;HALF CACHE ON
3082 017146 000664      BR       T14L10  ;CONTINUE TEST
3083
3084 017150 033737 000200 177744 T14L09: BIT      200,0*0*EREG ;HIGH BYTE P.E.?
3085 017156 001407      BEQ      T14L11  ;BRANCH IF NO
3086 017160 076600      MED      R0      ;GET LOG INFORMATION
3087 017162 000106      .WORD    CDH
3088 017164 010037 001164      MOV      R0,0*REG3 ;SAVE INFORMATION
3089 017170 012600      MOV      (SP)+,R0 ;RESTORE R0
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: MED      R0      ;GET LOG INFORMATION
3094 017176 076600      .WORD    CTAG
3095 017200 000107      MOV      R0,0*REG3 ;SAVE INFORMATION
3096 017202 010037 001164      MOV      (SP)+,R0 ;RESTORE R0
3097 017206 012600      MOV      #BUFL,0*REG4 ;GET TESTED ADDRESS
3098 017210 012737 060000 001166      MOV      #13,R5 ;SETUP COUNTER
3099 017216 012705 000013      MOV      #REG4 ;PUT TAG ADDRESS BITS IN LSB 6-0
3100 017222 006737 001166      ASR      #R5,28 ;SHIFT NINE PLACES
3101 017226 077503      BIS      #200,0*REG4 ;SET VALID BIT
3102 017230 052737 000200 001166      ERROR    60 ;ERROR: TAG PARITY ERROR WHEN TESTING CACHE DATA FIELD
3103 017236 104060      BR       T14L12  ;SEE IF WANT TO CONTINUE TEST
3104 017240 000733
3105
3106 017242 011205      T14L03: MOV      (R2),R5 ;GET BAD DATA
3107 017244 052737 000014 177746      BIS      #14,0*CCR ;CACHE OFF
3108 017252 005037 001160      CLR      0*REG1 ;SAVE ADDRESS
3109 017256 010237 001162      MOV      R2,0*REG2 ;SAVE ADDRESS
3110 017262 010537 001164      MOV      R5,0*REG3 ;SAVE BAD DATA
3111 017266 010037 001166      MOV      R0,0*REG4 ;SAVE GOOD DATA
3112 017272 012737 016704 001110      MOV      #14L02,0*0*LPERR ;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,0*PVEC ;RESTORE HANDLER FOR UNEXP. PARITY ERRORS
3117 017312 052737 000014 177746      BIS      #14,0*CCR ;CACHE OFF WHEN CROSS CACHE ADDR. BOUNDARY
3118 017320 000137 020000      JMP      0*0*0*0 ;GO TO NEXT TEST
3119
3120
3121      .=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

```

```

3137 ;*HALF CACHE ADDRESS IS REACHED. THEN STARTING AT THE
3138 ;*HIGH ADDR. THE SECOND PARITY PATTERN IS READ AND THE LOC
3139 ;*IS REWRITTEN WITH THE FIRST. THIS IS SEQUENTIALLY RE-
3140 ;*PEATED DECREASING THE ADDRESS UNTIL THE LOW HALF CACHE
3141 ;*ADDRESS IS REACHED. A SECOND PASS IS THEN MADE WITH
3142 ;*THE PARITY PATTERN REVERSED. A PARITY ERROR HANDLER IS
3143 ;*SETUP TO DETECT PARITY ERRORS. ALSO, LOCs WHICH SHOULD
3144 ;*BE HITS ARE CHECKED FOR AND REPORTED IF NO HIT OCCURRED.
3145 ;*
3146 ;*R0, R1 CONTAIN DATA WHICH GENERATE OPPOSITE PARITY. R3
3147 ;*INDICATES WHICH PASS IS BEING DONE.
3148
3149 ;*****
3150 TST24: MOV #214,0*CCR ;CACHE OFF FOR SCOPE
3151 SCOPE
3152 MOV #TST25,SKT&T ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3153 MOV #T12H01,0*PVEC ;SET UP PARITY ERROR HANDLER
3154 CLR R3 ;INIT FLAG FOR FIRST PASS
3155 CLR R0 ;SET UP PARITY PATTERN A FOR FIRST PASS
3156 MOV #204,0*CCR ;HALF CACHE ON
3157 MOV #1000,R1 ;INIT ADDR. COUNTER
3158 MOV #BUFH,R5 ;INIT. TEST ADDRESS
3159 MOV R0,(R5)+ ;WRITE DATA PARITY PATTERN
3160 SOB R1,18 ;HALF ADDR. WRITTEN? BRANCH IF NO
3161
3162 MOV #1000,R1 ;INIT ADDR. COUNTER
3163 MOV #BUFH,R5 ;INIT. TEST ADDR
3164 MOV #401,R0 ;SET UP PATTERN B FOR FIRST PASS
3165 TST R3 ;FIRST PASS?
3166 BEQ 25 ;BRANCH IF YES
3167 CLR R0 ;SET UP PARITY PATTERN A FOR SECOND PASS
3168 TST (R5) ;SEE IF PARITY UNCHANGED
3169 BIT 0*HMR,HMR2 ;DATA FROM CACHE?
3170 BEQ T12H07 ;BRANCH TO ERROR IF NO
3171 MOV R0,(R5)+ ;WRITE NEW DATA PARITY PATTERN
3172 SOB R1,28 ;HALF ADDR. SPACE EXAMINED & WRITTEN?
3173
3174 MOV #1000,R1 ;INIT ADDR. COUNTER
3175 CLR R0 ;SET UP PARITY PATTERN A FOR FIRST PASS
3176 TST R3 ;FIRST PASS?
3177 BEQ T12H06 ;BRANCH IF YES
3178 MOV #401,R0 ;SET UP PARITY PATTERN B FOR SECOND PASS
3179 MOV #204,0*CCR ;HALF CACHE ON IF OFF FROM ERROR
3180 TST (R5) ;SEE IF PARITY UNCHANGED
3181 BIT 0*HMR,HMR2 ;DATA FROM CACHE
3182 BEQ T12H07 ;BRANCH IF NO TO ERROR
3183 MOV R0,(R5)+ ;WRITE NEW PARITY PATTERN IN CACHE
3184 SOB R1,18 ;HALF OF ADDRESS SPACE READ & WRITTEN? BRANCH IF NO
3185
3186 TST R3 ;SECOND PASS?
3187 BNE T12H08 ;GO TO END OF TEST IF YES
3188 MOV #401,R0 ;SET UP PARITY PATTERN B FOR SECOND PASS
3189 BIS #1,R3 ;SET FLAG FOR PASS 2
3190 MOV #T12H13,0*LPERR ;INIT RETURN FOR ERROR LOOP IF ERROR OCCURS
3191 BR T12H02 ;TEST DATA
3192

```

```

3193 T12H01: BIS #14,0*CCR ;CACHE OFF
3194
3195 MOV R0,-(SP) ;SAVE R0 FOR MED INST
3196 MED ;GET CONTENTS OF LOG REG
3197 .WORD RLOG
3198 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
3199 MED ;UNLOCK ERROR LOG
3200 .WORD WLOG
3201 MOV (SP)+,R0 ;RESTORE R0
3202
3203 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3204 .WORD RSER
3205 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3206 BIC #177776,R0 ;ONLY LOOK AT A17, A16
3207 MOV R0,RREG1 ;SAVE ADDRESS
3208 MED ;GET LOG INFORMATION
3209 .WORD LOADD
3210 MOV R0,RREG2 ;SAVE INFORMATION
3211 BIT #40,0*EREG ;ERROR IN TAG?
3212 BEQ T12H09 ;BRANCH IF NO
3213 MOV (SP),RREG4 ;GET PC+2 OF ERROR
3214 SUB #2,RREG4 ;GET PC OF ERROR
3215 MED ;GET TAG LOG INFO.
3216 .WORD RTAG
3217 SWAB R0 ;PUT TAG IN LOW BYTE
3218 BIC #177400,R0 ;LOOK AT TAG ONLY
3219 MOV R0,RREG3 ;SAVE BAD DATA
3220 CMP (SP)+,(SP)+ ;RESTORE THE STACK
3221 ERROR 2 ;ERROR: UNEXPECTED PARITY ERROR IN TAG FIELD
3222 BR T12H08 ;GO TO END OF TEST
3223
3224 T12H09: CMP (SP)+,(SP)+ ;RESTORE STACK
3225 CLR RREG4 ;SAVE GOOD DATA
3226 TST R0 ;WAS TEST DATA =0?
3227 BNE T12H11 ;BRANCH IF NO
3228 MOV #401,RREG4 ;SAVE GOOD DATA
3229 BIT #200,0*EREG ;ERROR IN HIGH BYTE?
3230 BEQ T12H12 ;BRANCH IF NO
3231 MED ;GET LOG INFORMATION
3232 .WORD CDH
3233 MOV R0,RREG3 ;SAVE INFORMATION
3234 ERROR 51 ;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA P BITS
3235 BR T12H08 ;GO TO END OF TEST
3236
3237 T12H12: BIT #100,0*EREG ;ERROR IN LOW BYTE?
3238 BEQ T12H14 ;BRANCH IF NO
3239 MED ;GET LOG INFORMATION
3240 .WORD CDL
3241 MOV R0,RREG3 ;SAVE INFORMATION
3242 ERROR 51 ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BITS
3243 BR T12H08 ;GO TO END OF TEST
3244
3245 T12H14: MOV -4(SP),RREG3 ;GET PC+2 OF TRAP
3246 SUB #2,RREG3 ;SAVE PC OF TRAP
3247 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 RREG1 ;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 ;*TEST 25 TEST TAG ADDRESS BITS FOR HIGH HALF OF CACHE
3260 ;*
3261 ;* THE TEST OF THE TAG BITS IS NOT COMPLETE UNTIL THE
3262 ;*TAG ADDRESS TEST FOR THE OTHER HALF OF CACHE AND THE
3263 ;*TEST OF THE MSB ADDRESS (A10) TO THE CACHE TAG FIELD
3264 ;*ARE RUN. A WRITE/READ PROCEDURE IS DONE WHICH CHECKS
3265 ;*THE TAG FIELD BITS AND DUAL ADDRESSING ON THEM FOR HALF
3266 ;*OF CACHE. MEMORY IS FIRST SIZED TO DETERMINE THE MAX-
3267 ;*IMUM TESTABLE ADDRESS. THE TAG ADDRESS BITS OF THIS
3268 ;*ADDRESS ARE USED AS PATTERN A AND STORED IN KIPAR4. A
3269 ;*PATTERN B IS NOW GENERATED WHICH HAS 'COMPLEMENT' TAG
3270 ;*BITS AND STORED IN KIPAR5. ON THE FIRST PASS, PATTERN
3271 ;*A IS WRITTEN THROUGH HALF OF CACHE. NEXT, STARTING AT
3272 ;*THE HIGH HALF CACHE ADDRESS, THE LOCATION IS READ,
3273 ;*CHECKED TO BE A HIT AND THEN WRITTEN WITH PATTERN B.
3274 ;*THIS IS SEQUENTIALLY REPEATED WITH DECREASING ADDRESSES
3275 ;*UNTIL THE LOW HALF CACHE ADDRESS IS REACHED. AT THE
3276 ;*LOW ADDRESS, THE SECOND PATTERN IS READ, CHECKED TO BE A
3277 ;*HIT AND REWRITTEN WITH THE FIRST PATTERN. THIS IS SE-
3278 ;*QUENTIALLY REPEATED WITH INCREASING ADDRESSES UNTIL THE
3279 ;*HIGH HALF CACHE ADDRESS IS REACHED. A SECOND PASS IS
3280 ;*THEN MADE WITH THE PATTERNS REVERSED.
3281 ;* ANY PARITY ERROR OR HIT ERROR IS REPORTED.
3282 ;* DURING THE PASSES, R0, R1 CONTAIN ADDRESSES WHICH
3283 ;*REFERENCE KIPAR5,5.
3284 ;* R3 INDICATES THE PASS NUMBER.
3285 ;*IF THE INHIBIT TESTS USING KT SWITCH (SW12) IS SET, THIS
3286 ;*TEST IS SKIPPED.
3287
3288 ;*****
3289 TST25: MOV #214,0#CCR ;CACHE OFF FOR SCOPE
3290 SCOPE
3291 MOV #TST26,0#KTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3292 BIT #SW12,0#SWR ;INHIBIT TESTS USING KT?
3293 BEQ 38 ;CONTINUE TEST IF NO
3294 JMP #TST26 ;GO TO NEXT TEST
3295 020510 052737 000200 036034 36: BIS #200,0#KT11 ;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 #06LSTBK,0#KIPAR4 ;SET UP PAR4 FOR ADDRESS PATTERN A
3299
3300 ;CALC COMPLEMENT TAG PATTERN B
3301
3302 020536 013700 036322 MOV #06LSTBK,R0 ;GET TEST PATTERN A AND
3303 020542 005100 COM R0 ;CALC PATTERN B
3304 020544 005001 CLR R1

```

```

3305 020546 005201 18: INC R1
3306 020550 006300 ASL R0
3307 020552 100775 BMI 18
3308 020554 006200 20: ASR R0
3309 020556 077102 SOB R1,28
3310 020560 042700 000037 BIC #37,R0 ;ONLY COMPLEMENT TAG ADDR. BITS
3311
3312 020564 010037 172352 MOV R0,0#KIPAR5 ;SET UP PAR5 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 #1,0#MNR0 ;TURN KT ON
3320 020616 012737 000204 177746 MOV #204,0#CCR ;TURN HALF OF CACHE ON
3321
3322 020624 005720 18: TST (R0)+ ;WRITE PATTERN IN CACHE
3323 020626 077202 SOB R2,18 ;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 #0#HMR,0#HMR2 ;HIT?
3328 020644 001002 BNE 28 ;BRANCH IF YES
3329 020646 000137 021216 JMP T13H04 ;REPORT ERROR
3330 020652 005741 28: 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 MOV #1000,R2 ;INIT. ADDR. COUNTER
3335 020664 005711 T13H05: TST (R1) ;READ CACHE TAG BITS
3336 020666 033727 177752 000004 BIT #0#HMR,0#HMR2 ;HIT?
3337 020674 001002 BNE 38 ;BRANCH IF YES
3338 020676 000137 021264 JMP T13H06 ;REPORT ERROR
3339 020702 005721 38: 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 28 ;CONTINUE TEST IF NO
3345 020714 000137 021312 JMP T13H07 ;GO TO END OF TEST
3346 020720 052703 000001 28: BIS #1,R3 ;SET FLAG FOR SECOND PASS
3347 020724 012737 020732 001110 MOV #T13H15,0#LPERR ;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 #14,0#CCR ;CACHE OFF
3353
3354 020752 010040 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 RIS #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

```

```

3361
3362 020772 011637 001164 MOV (SP),R03 ;GET PC+2 OF TRAP
3363 020776 162737 000002 001164 SUB #2,R03 ;SAVE PC FOR MAIN PARITY ERROR
3364 021004 022626 CMP (SP)+,(SP)+ ;RESTORE STACK
3365 021006 010046 MOV R0,-(SP) ;SAVE R0 ON STACK FOR MED INST.
3366 021010 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3367 021012 000101 .WORD R03R ;
3368 021014 000300 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3369 021016 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
3370 021022 010037 001160 MOV R0,R03 ;SAVE ADDRESS
3371 021026 076600 MED ;GET LOG INFORMATION
3372 021030 000102 .WORD LOADD ;
3373 021032 010037 001162 MOV R0,R03 ;SAVE INFORMATION
3374 021036 076600 MED ;GET LOG INFORMATION
3375 021040 000100 .WORD RJAM ;
3376 021042 012600 MOV (SP)+,R0 ;RESTORE R0
3377 021044 032700 000400 BIT #400,R0 ;ERROR BACKING STORE?
3378 021050 001402 BEQ T13H08 ;BRANCH IF NO
3379 021052 104001 ERROR 1 ;ERROR: UNEXPECT. PARITY ERROR IN BACKING STORE
3380 021054 000516 BR T13H07 ;GO TO END OF TEST
3381
3382 021056 011137 001166 T13H08: MOV (R1),R03 ;SAVE GOOD DATA
3383 021062 005704 TST R4 ;ERROR IN LOOP 2?
3384 021064 001002 BNE T13H09 ;BRANCH IF YES
3385 021066 011037 001166 MOV (R0),R03 ;SAVE GOOD DATA
3386
3387 021072 032737 000040 177744 T13H09: BIT #40,R03 ;TAG PARITY ERROR?
3388 021100 001426 BEQ T13H10 ;BRANCH IF NO
3389 021102 004737 033634 JSR PC,PAR ;GET PAR USED
3390 021106 000000 .WORD 0 ;INDICATOR FOR R0
3391 021110 005704 TST R4 ;ERROR FROM LOOP 1?
3392 021112 001403 BEQ T13H11 ;BRANCH IF YES
3393 021114 004737 033634 JSR PC,PAR ;GET PAR USED
3394 021120 000001 .WORD 1 ;INDICATOR FOR R1
3395 021122 004737 033606 T13H11: JSR PC,TAG ;CALC TAG CONTENTS
3396 021126 013737 001172 001166 MOV $TMP0,R03 ;SAVE GOOD DATA
3397 021134 076600 MED ;GET TAG LOG INFO.
3398 021136 000107 .WORD RTAG ;
3399 021140 000300 SWAB R0 ;PUT TAG IN LOW BYTE
3400 021142 042700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
3401 021146 010037 001164 MOV R0,R03 ;SAVE BAD DATA
3402 021152 104052 ERROR 52 ;ERROR: TAG PARITY ERROR ON TEST OF TAG ADDRESS BITS
3403 021154 000456 BR T13H07 ;GO TO END OF TEST
3404
3405 021156 032737 000100 177744 T13H10: BIT #100,R03 ;LOW BYTE P.E.?
3406 021164 001406 BEQ T13H12 ;BRANCH IF NO
3407 021166 076600 MED ;GET LOG INFORMATION
3408 021170 000106 .WORD CDL ;
3409 021172 010037 001164 MOV R0,R03 ;SAVE INFORMATION
3410 021176 104053 ERROR 53 ;ERROR: LOW BYTE PARITY ERROR ON TEST OF TAG ADDR. BITS
3411 021200 000444 BR T13H07 ;GO TO END OF TEST
3412
3413 021202 T13H12: ;
3414 021202 076600 MED ;GET LOG INFORMATION
3415 021204 000106 .WORD CDH ;
3416 021206 010037 001164 MOV R0,R03 ;SAVE INFORMATION

```

```

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,R03 ;CACHE OFF
3421 021224 010037 001172 MOV R0,$TMP0 ;GET VIRTUAL ADDRESS TESTED
3422 021230 004737 033434 JSR PC,VIP ;SAVE ADDRESS TESTED
3423 021234 062700 000002 ADD #2,R0 ;ADJUST ADDRESS WHEN LOOP
3424 021240 004737 033634 JSR PC,PAR ;GET PAR TESTED
3425 021244 000000 .WORD 0 ;INDICATOR FOR R0
3426 021246 004737 033606 T13H13: JSR PC,TAG ;CALC TAG FROM PAR
3427 021252 013737 001172 001164 MOV $TMP0,R03 ;SAVE TAG
3428 021260 104055 ERROR 55 ;ERROR: TEST OF TAG ADDRESS BITS FAILED
3429 ; ADDR. COULD NOT BE MADE A HIT
3430 021262 000413 BR T13H07 ;GO TO NEXT TEST
3431
3432 021264 052737 000014 177746 T13H06: BIS #14,R03 ;CACHE OFF
3433
3434 021272 010137 001172 MOV R1,$TMP0 ;GET VIRTUAL ADDRESS TESTED
3435 021276 004737 033434 JSR PC,VIP ;SAVE PHYSICAL ADDRESS TESTED
3436 021302 004737 033634 JSR PC,PAR ;GET PAR TESTED
3437 021310 000756 .WORD 1 ;INDICATOR FOR R1
3438 BR T13H13 ;REPORT ERROR
3439
3440 021312 005037 177572 T13H07: CLR #MMR0 ;KI OFF
3441 021316 012737 033142 000114 MOV $UPERR,$PVEC ;RESTORE UNEXPECTED PARITY ERROR HANDLER
3442 021332 000137 022000 BIS #14,R03 ;CACHE OFF WHEN CROSS CACHE ADDR. BOUNDARY
3443 JMP #TST26 ;GO TO NEXT TEST
3444
3445 .=22000 ;ADJUST ADDRESS SPACE FOR NEXT TEST
3446
3447
3448
3449
3450 ;*****
3451 ;*TEST 26 TEST DATA FIELD FOR LOW HALF OF CACHE
3452 ;*
3453 ;* THE TEST OF THE DATA FIELD IS NOT COMPLETE UNTIL THE
3454 ;*TEST OF THE DATA FIELD FOR THE OTHER HALF OF CACHE AND
3455 ;*THE TEST OF THE MSB ADDRESS (A10) TO THE CACHE DATA
3456 ;*FIELD ARE RUN. A WRITE/READ PROCEDURE IS DONE WHICH
3457 ;*CHECKS ALL THE DATA FIELD BITS AND DUAL ADDRESSING ON
3458 ;*THEM FOR HALF OF CACHE. ON THE FIRST PASS ONE PATTERN
3459 ;*(CONTAINED IN R0) IS WRITTEN IN ALL THE DATA FIELDS.
3460 ;*FOR HALF OF CACHE. NEXT, STARTING AT THE HIGH HALF
3461 ;*CACHE ADDRESS, THE LOCATION IS TESTED TO BE A HIT, ITS
3462 ;*DECREASING ADDRESSES UNTIL THE LOW HALF CACHE ADDRESS IS
3463 ;*REACHED. AT THE LOW ADDRESS, THE SECOND PATTERN IS READ,
3464 ;*TESTED TO BE A HIT AND REWRITTEN WITH THE FIRST PATTERN.
3465 ;*THIS IS SEQUENTIALLY REPEATED WITH INCREASING ADDRESSES
3466 ;*UNTIL THE HIGH HALF CACHE ADDRESS IS REACHED. A SECOND
3467 ;*PASS IS THEN MADE WITH THE PATTERNS REVERSED.
3468 ;*
3469 ;* ANY PARITY ERROR OR HIT ERROR IS REPORTED.
3470 ;* R0, R1 CONTAIN THE TEST PATTERN
3471 ;* R2 CONTAINS THE TEST ADDRESS
3472 ;* R4 CONTAINS THE PASS NUMBER

```

```
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 #7ST27,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3478 022016 012737 022210 000114 MOV #715L01,0#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 #RUF1,R2 ;INIT STARTING TEST ADDRESS
3484 022050 012703 001000 MOV #1000,R3 ;INIT ADDRESS COUNTER
3485 022054 010022 18: MOV R0,(R2)+ ;WRITE CACHE WITH PATTERN
3486 022056 077302 SOB R3,18 ;LOOP TILL HALF CACHE WRITTEN
3487
3488 ;NOW READ AND WRITE PATTERN, DECREASING ADDRESS
3489
3490 022060 012703 001000 MOV #1000,R3 ;INIT ADDRESS COUNTER
3491 022064 005742 T15L21: TST -(R2) ;READ CACHE
3492 022066 033727 177752 000004 BIT #4HMR,#HMR2 ;HIT?
3493 022074 001002 BNE 18 ;BRANCH IF YES
3494 022076 000137 022466 JMP T15L02 ;REPORT ERROR
3495 022102 021200 18: 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 #4HMR,#HMR2 ;HIT?
3507 022136 001002 BNE 18 ;BRANCH IF YES
3508 022140 000137 022466 JMP T15L02 ;REPORT ERROR
3509 022144 021201 18: CMP (R2),R1 ;DATA OK?
3510 022146 001402 BEQ T15L10 ;BRANCH IF YES
3511 022150 000137 022526 JMP T15L15 ;REPORT ERROR
3512 022154 010022 T15L10: 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 #715L12,0#8LPERR ;INIT RETURN FOR ERROR LOOP IF ERROR
3521 022206 000712 T15L05: 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)+,R0 ;RESTORE R0
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 R0,-(SP) ;SAVE R0 FOR MED INST
3537 022254 076600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3538 022256 000101 .WORD RSER
3539 022260 000100 R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3540 022262 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
3541 022266 010037 001160 MOV R0,#REG1 ;SAVE ADDRESS
3542 022272 076600 MED ;GET LOG INFORMATION
3543 022274 000102 .WORD LOADD
3544 022276 010037 001162 MOV R0,#REG2 ;SAVE INFORMATION
3545 022302 076600 MED ;GET LOG INFORMATION
3546 022304 000100 .WORD RJAM
3547 022306 010005 MOV R0,R5 ;SAVE INFORMATION
3548 022310 012600 MOV (SP)+,R0 ;RESTORE R0
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 R0,#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 R0,#REG4 ;SAVE GOOD DATA
3561
3562 022350 032737 000100 177744 T15L08: BIT #100,0#EREG ;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 R0,#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 #200,0#EREG ;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 R0,#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 28: ASR #REG4 ;PUT TAG ADDRESS BITS IN LSB 6-0
3581 022434 077503 SOB R5,28 ;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
```

```

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.
;* R0, R1 CONTAIN THE TEST PATTERN

```

```

3641
3642
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          MOV R0,(R2)+          ;WRITE CACHE WITH PATTERN
3656 024056 077302          SOB R3,10              ;LOOP TILL HALF CACHE WRITTEN
3657
3658
3659
3660
3661 024060 012703 001000          MOV #1000,R3        ;INIT ADDRESS COUNTER
3662 024064 005742          T15H21: TST -(R2)         ;READ CACHE
3663 024074 001002 177752 000004    BIT #HMR,#HMR2    ;HIT?
3664 024076 000137 024466          BNE 10           ;BRANCH IF YES
3665 024102 021200          JMP T15H02             ;REPORT ERROR
3666 024104 001402          10: CMP (R2),R0         ;IS DATA CORRECT?
3667 024106 000137 024510          BEQ T15H17        ;BRANCH IF YES
3668 024112 010112          JMP T15H03             ;REPORT ERROR
3669 024114 077315          T15H17: MOV R1,(R2)        ;WRITE NEW PATTERN IN CACHE
3670
3671
3672
3673 024116 052704 000001          SOB R3,T15H21      ;LOOP TILL HALF CACHE READ & WRITTEN
3674 024122 012703 001000          ;NOW READ AND WRITE PATTERN, INCREASING ADDRESS
3675 024126 005712          MOV #1,R4              ;SET FLAG FOR ERROR LOOP 2
3676 024130 033727 177752 000004    MOV #1000,R3        ;INIT. ADDRESS COUNTER
3677 024136 001002          T15H22: TST (R2)          ;READ CACHE
3678 024140 000137 024466          BIT #HMR,#HMR2    ;HIT?
3679 024144 021201          BNE 10                 ;BRANCH IF YES
3680 024146 001402          JMP T15H02             ;REPORT ERROR
3681 024150 000137 024526          10: CMP (R2),R1         ;DATA OK?
3682 024154 010022          BEQ T15H18             ;BRANCH IF YES
3683 024156 077315          JMP T15H15             ;REPORT ERROR
3684
3685 024160 005700          T15H18: MOV R0,(R2)+          ;WRITE NEW TEST PATTERN
3686 024162 100402          SOB R3,T15H22          ;LOOP TILL HALF OF CACHE READ & WRITTEN
3687 024164 000137 024560          TST R0            ;DOES R0 HAVE DATA FOR FIRST PASS?
3688 024170 012700 052525          BMI T15H12        ;BRANCH IF YES
3689 024174 012701 125252          JMP T15H04        ;GO TO END OF TEST
3690 024200 012737 024170 001110    MOV #52525,R0      ;SET UP DATA PATTERN B FOR PASS 2.
3691 024206 000712          MOV #125252,R1         ;SET UP DATA PATTERN A FOR PASS 2
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

```



```

3697 024222 000022      .WORD RLOG      ;ENABLE ERROR LOG & LOG FIRST MODE
3698 024224 052700 100001 BIS      #100001,R0      ;UNLOCK ERROR LOG
3699 024230 076600      MED                      ;
3700 024232 000222      .WORD WLOG      ;
3701 024234 012600      MOV      (SP)+,R0      ;RESTORE R0
3702
3703 024236 011637      MOV      (SP),R0      ;GET PC+2 OF PARITY ERROR
3704 024242 162737 000002 001164 SUB      #2,R0      ;SAVE PC OF PARITY ERROR
3705 024250 022626      CMP      (SP)+,(SP)+    ;RESTORE STACK
3706 024252 010046      MOV      R0,-(SP)      ;SAVE R0 FOR MED INST
3707 024254 076600      MED                      ;GET LOG INFOR FOR PHY. ADDR. A17,A16
3708 024256 000101      .WORD RSER      ;
3709 024260 000300      SWAB      R0      ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3710 024262 042700 177776 BIC      #177776,R0    ;ONLY LOOK AT A17, A16
3711 024266 010037 001160 MOV      R0,$REG1    ;SAVE ADDRESS
3712 024272 076600      MED                      ;GET LOG INFORMATION
3713 024274 000102      .WORD LOADD      ;
3714 024276 010037 001162 MOV      R0,$REG2    ;SAVE INFORMATION
3715 024302 076600      MED                      ;GET LOG INFORMATION
3716 024304 000100      .WORD RJAM      ;
3717 024306 010005      MOV      R0,R5      ;SAVE INFORMATION
3718 024310 012600      MOV      (SP)+,R0      ;RESTORE R0
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 055616      .WORD BSD      ;
3723 024324 010037 001164 MOV      R0,$REG3    ;SAVE INFORMATION
3724 024330 104001      ERROR      1      ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
3725 024332 000512      BR      T15H04      ;GO TO END OF TEST
3726
3727 024334 010137 001166 T15H06: MOV      R1,$REG4    ;SAVE GOOD DATA
3728 024340 005704      TST      R4      ;ERROR LOOP 1?
3729 024342 001002      BNE      T15H08      ;BRANCH IF NO
3730 024344 010037 001166 MOV      R0,$REG4    ;SAVE GOOD DATA
3731
3732 024350 032737 000100 177744 T15H08: BIT      #100,$$REG      ;LOW BYTE PARITY ERROR?
3733 024356 001406      BEQ      T15H13      ;BRANCH IF NO
3734 024360 076600      MED                      ;GET LOG INFORMATION
3735 024362 000106      .WORD CDL      ;
3736 024364 010037 001164 MOV      R0,$REG3    ;SAVE INFORMATION
3737 024370 104056      ERROR      56      ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD
3738 024372 000472      BR      T15H04      ;GO TO END OF TEST
3739
3740 024374 032737 000200 177744 T15H13: BIT      #200,$$REG      ;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      R0,$REG3    ;SAVE INFORMATION
3745 024414 104057      ERROR      57      ;ERROR: HIGH BYTE PARITY ERROR WHEN TEST DATA FIELD
3746 024416 000460      BR      T15H04      ;GO TO END OF TEST
3747
3748 024420 010237 001166 T15H14: MOV      R2,$REG4    ;GET FAILING ADDRESS
3749 024424 012705 000013      MOV      #13,R5      ;SET UP COUNTER
3750 024430 006237 001166 28:  ASR      $REG4      ;PUT TAG ADDRESS BITS IN LSB 6-0
3751 024434 077503      SOB      R5,R5      ;LOOP TILL DONE
3752 024436 052737 000200 001166 BIS      #200,$REG4    ;SET VALID BIT

```

```

3753 024444 076600      MED                      ;GET TAG LOG INFO.
3754 024446 000107      .WORD RTAG      ;
3755 024450 000300      SWAB      R0      ;PUT TAG IN LOW BYTE
3756 024452 042700 177400 BIC      #177400,R0    ;LOOK AT TAG ONLY
3757 024456 010037 001164 MOV      R0,$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,$$CCR      ;CACHE OFF
3762 024474 005037 001160      CLR      $REG1      ;SAVE ADDRESS
3763 024500 010237 001162      MOV      R2,$REG2    ;SAVE ADDRESS
3764 024504 104043      ERROR      43      ;ERROR: ADDRESS COULD NOT BE MADE A HIT
3765 024506 000424      BR      T15H04      ;GO TO END OF TEST
3766
3767 024510 011205      .WORD T15H03: MOV      (R2),R5      ;GET BAD DATA
3768 024512 005737 000014 177746 BIS      #14,$$CCR      ;CACHE OFF
3769 024520 010037 001166      MOV      R0,$REG4    ;SAVE GOOD DATA
3770 024524 000406      BR      T15H16      ;REPORT ERROR
3771
3772 024526 011205      .WORD 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 ;*
3794 ;*      THE FIRST ADDRESS IS INVALIDATED VIA WWP AND FORCING A PARITY
3795 ;*TRAP. THE SECOND IS THEN MADE VALID AND CHECKED TO BE A HIT. THE FIRST IS
3796 ;*THEN EXAMINED TO STILL BE INVALID (NOT A HIT). ANY PARITY OR HIT
3797 ;*ERROR IS REPORTED.
3798 ;*****
3799 024566 012737 000214 177746 TST30: MOV      #214,$$CCR    ;CACHE OFF FOR SCOPE
3800 024574 000001      SCOPE
3801 024576 012737 025244 001234 MOV      #TST31,SKTST    ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3802 024604 012737 024646 000114 MOV      #T30L01,$$PVEC    ;SET UP FOR PARITY TRAP
3803 024612 004737 033714 JSR      PC,HAD      ;CALC CONGRUENT ADDR. IN TEST BUFFER
3804 024616 025242      .WORD TAD2
3805 024620 013700 001172      MOV      $TMP0,R0      ;SAVE ADDR.
3806 024624 012737 000300 177746 MOV      #300,$$CCR      ;CACHE ON & WWP
3807 024632 005010      CLR      (R0)      ;WWP IN TEST ADDR.
3808 024634 012737 000200 177746 MOV      #200,$$CCR      ;WWP OFF

```

```

3809 024642 005710      TST      (R0)      ;FORCE PARITY TRAP
3810 024644 000465      BR       T30L02      ;REPORT FAILURE TO TRAP
3811
3812 024646      T30L01:
3813
3814 024646 010046      MOV      R0,-(SP)      ;SAVE R0 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
3819 024662 000222      .WORD    WLOG
3820 024664 012600      MOV      (SP)+,R0      ;RESTORE R0
3821
3822 024666 062706 000004  ADD      #4,SP      ;RESTORE STACK
3823 024672 012737 025042 000114  MOV      #T30L06,0#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      0#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      0#HMR,#HMR2      ;MISS?
3829 024726 001011      BNE      T30L04      ;REPORT ERROR IF NO
3830 024730
3831
3832      ;RID CACHE OF BAD PARITY
3833 024730 012737 000214 177746  MOV      #214,0#CCR      ;CACHE OFF IF ON
3834 024736 004737 035134  JSR      PC,SWEEP      ;GO PURGE CACHE
3835
3836
3837 024742 012737 033142 000114  MOV      #UPERR,0#PVEC      ;RESTORE UNEXP. PARITY ERROR HANDLER
3838 024750 000535      BR       TST31      ;GO TO NEXT TEST
3839
3840 024752 012737 000214 177746  T30L04: MOV      #214,0#CCR      ;CACHE OFF
3841 024760 005037 001160      CLR      #REG1      ;SAVE BAD ADDRESS
3842 024764 010037 001162      MOV      R0,#REG2      ;SAVE BAD ADDRESS
3843 024770 104121      ERROR    121      ;ERROR:TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
3844      ; LOC. NOT INVALIDATED
3845 024772 000756      BR       T30L05      ;GO TO END OF TEST
3846
3847 024774 012737 000214 177746  T30L03: MOV      #214,0#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    43      ;ERROR:ADDRESS COULD NOT BE MADE A HIT
3851 025016 000744      BR       T30L05      ;GO TO END OF TEST
3852
3853 025020 012737 000214 177746  T30L02: MOV      #214,0#CCR      ;CACHE OFF
3854 025026 005037 001160      CLR      #REG1      ;SAVE BAD ADDRESS
3855 025032 010037 001162      MOV      R0,#REG2      ;SAVE BAD ADDRESS
3856 025036 104042      ERROR    42      ;ERROR:NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PAR.
3857 025040 000733      BR       T30L05      ;GO TO END OF TEST
3858
3859 025042 012737 000214 177746  T30L06: MOV      #214,0#CCR      ;CACHE OFF
3860
3861 025050 010046      MOV      R0,-(SP)      ;SAVE R0 FOR MED INST
3862 025052 076600      MED      ;GET CONTENTS OF LOG REG
3863 025054 000022      .WORD    RLOG
3864 025056 052700 100001  BIS      #100001,R0      ;ENABLE ERROR LOG & LOG FIRST MODE

```

```

3865 025062 076600      MED      ;UNLOCK ERROR LOG
3866 025064 000222      .WORD    WLOG
3867 025066 012600      MOV      (SP)+,R0      ;RESTORE R0
3868
3869 025070 011637 001164      MOV      (SP),#REG3      ;GET PC+2 OF ERROR
3870 025074 162737 000002 001164  SUB      #2,#REG3      ;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    R0      ;PUT PHY. ADDR A17, A16 IN LOW BYTE
3875 025112 042700 177776 001160  RIC      #177776,R0      ;ONLY LOOK AT A17, A16
3876 025116 010037 001160      MOV      R0,#REG1      ;SAVE ADDRESS
3877 025122 076600      MED      ;GET LOG INFORMATION
3878 025124 000102      .WORD    LOADD
3879 025126 010037 001162      MOV      R0,#REG2      ;SAVE INFORMATION
3880 025132 076600      MED      ;GET LOG INFORMATION
3881 025134 000100      .WORD    RJAM
3882 025136 032700 000400      BIT      #400,R0      ;ERROR IN BACKING STORE?
3883 025142 001402      BEQ      10      ;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 10:  BIT      #40,0#EREG      ;PARITY ERROR TAG?
3888 025156 001411      BEQ      20      ;BRANCH IF NO
3889 025160 076600      MED      ;GET TAG LOG INFO.
3890 025162 000107      .WORD    RTAG
3891 025164 000300      SWAB    R0      ;PUT TAG IN LOW BYTE
3892 025166 042700 177400 001164  BIC      #177400,R0      ;LOOK AT TAG ONLY
3893 025172 010037 001164      MOV      R0,#REG3      ;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 20:  BIT      #100,0#EREG      ;PARITY ERROR LOW BYTE?
3899 025210 001406      BEQ      30      ;BRANCH IF NO
3900 025212 076600      MED      ;GET LOG INFORMATION
3901 025214 000106      .WORD    CDL
3902 025216 010037 001164      MOV      R0,#REG3      ;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      30:  MED      ;GET LOG INFORMATION
3908 025226 076600      .WORD    CDH
3909 025230 000106      MOV      R0,#REG3      ;SAVE INFORMATION
3910 025232 010037 001164      ERROR    124      ;ERROR:TEST OF MSB ADDR. (A10) TO VALID BIT FAILED
3911 025236 104124      BR       T30L05      ; PARITY ERROR HIGH BYTE
3912      ; 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 ;*FIELD FOR THE MSB ADDRESS (A10) TO CACHE. THERE ARE TWO
3922 ;*PASSES. THE FIRST EXERCISES THE ADDRESS BITS IN THE TAG
3923 ;*FIELD AND THE SECOND EXERCISES THE TAG P BIT. INITIALLY
3924 ;*THE MEMORY IS SIZED TO DETERMINE THE MAXIMUM TESTABLE
3925 ;*ADDRESS. THE TAG FIELD OF THE MAX ADDR. IS USED AS THE
3926 ;*FIRST TEST VALUE AND ITS COMPLEMENT AS THE SECOND. THESE
3927 ;*TAG VALUES ARE THEN PUT INTO CACHE LOCATIONS WITH THE
3928 ;*SAME CACHE ADDRESS (A1-A9) EXCEPT FOR THEIR ADDRESS BIT
3929 ;*A10 COMPLEMENT. THE LOCS IN CACHE ARE CHOSEN SO THAT
3930 ;*THEY DON'T OVERLAP THE TEST INSTRUCTION ADDRESS SPACE.
3931 ;*THIS IS TO PREVENT THEIR BEING SWAP OUT WHEN THE INSTRU-
3932 ;*CTIONS ARE BEING EXECUTED. AFTER THE LOCATIONS ARE
3933 ;*WRITTEN THEY ARE EXAMINED AND CHECKED TO BE HITS.
3934 ;*FOLLOWING THIS THE SECOND PASS IS DONE FOR THE TAG P BIT.
3935 ;*TWO NEW TAG VALUES ARE CHOSEN WITH OPPOSITE P BITS. THEY ARE
3936 ;*THEN WRITTEN, READ AND TESTED FOR HITS. ANY PARITY ERRORS
3937 ;*OR HIT ERRORS ARE REPORTED.
3938 ;* KIPAR4, 5 CONTAIN THE TAG VALUES WHICH ARE STORED IN
3939 ;*CACHE.
3940 ;* R0, R1 CONTAIN THE CACHE TEST LOC THAT DON'T OVERLAP
3941 ;*THE INSTRUCTION ADDRESS SPACE
3942 ;* R5 CONTAINS THE PASS #.
3943 ;* IF THE INHIBIT TEST USING KT SWITCH (SW12) IS SET,
3944 ;*THIS TEST IS SKIPPED.
3945
3946 ;*****
3947 TST31: MOV #214,CCR ;CACHE OFF FOR SCOPE
3948 SCOPE
3949 MOV #TST32,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
3950 BIT #SW12,SWR ;INHIBIT TESTS USING KT?
3951 BEQ 38 ;CONTINUE TEST IF NO
3952 JMP #TST32 ;GO TO NEXT TEST
3953 MOV #T16L01,PPVEC ;SET UP PARITY ERROR HANDLER
3954 BIS #200,KT1 ;USE KT FOR SIZE
3955 JSR PC,SIZE ;SIZE MEMORY
3956 MOV #T16L01,R0 ;GET LAST ADDRESS AND
3957 COM R0 ;CALC. ITS COMPLEMENT
3958 CLR R1 ;KEEPING THE MSB THAT ARE 0
3959 16: INC R1 ;A 0
3960 ASL R0
3961 BMI 18
3962 28: ASR R0
3963 SOB R1,28
3964 BIC #37,R0 ;ONLY COMPLEMENT TAG ADDRESS BITS
3965 MOV R0,KTIPAR5 ;SET UP PAR5 WITH COMPLEMENT ADDRESS BITS
3966 MOV #T16L01,KTIPAR4 ;SET UP PAR4 WITH COMPLEMENT ADDRESS BITS
3967
3968 ;SET UP R0,R1 TO ADDR. LOCS WHICH DON'T OVERLAP THIS TEST'S INSTRUCTION SPACE
3969
3970 MOV #LAST1,R0 ;GET ADDR. OF LAST IN THIS TEST
3971 BIC #174000,R0 ;SAVE LOWER ADDR BITS A10-A0
3972 MOV R0,R1 ;COPY ADDRESS
3973 ADD #100000,R0 ;HAVE R0 ADDR PAR4
3974 ADD #122000,R1 ;HAVE R1 ADDR PAR5 & HAVE A10 COMP OF R0
3975 CLR R5 ;INDICATE PASS 1
3976 BIS #1,MMMR0 ;KT ON

```

```

3977 025410 012737 000200 177746 T16L05: MOV #200,CCR ;CACHE ON
3978 025416 021011 CMP (R0),(R1) ;GET LOC IN CACHE VIA DAT1
3979 025420 005110 TST (R0) ;READ CACHE
3980 025422 033727 177752 000004 BIT 00HMR,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 00HMR,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 005705 000001 BIS #1,R5 ;SET FLAG FOR SECOND PASS
3988 025454 005037 172350 CLR 00KIPAR4 ;SET UP PAR4 TO TEST P BIT
3989 025460 012737 000040 172352 MOV #40,KTIPAR5 ;SET UP PAR5 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,STMP0 ;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,STMP0 ;GET VIRTUAL ADDR.
3998 025516 004737 033434 T16L06: JSR PC,VIP ;CHANGE VIRTUAL ADDRESS INTO PHYSICAL
3999 025522 012737 025410 001110 MOV #T16L05,00LPERR ;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 ;ERRPOP 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 RP T16L04 ;GO TO END OF TEST
4031
4032 025642 032737 000040 177744 T16L07: BIT #40,EREGER ;ERRPOP IN TAG FIELD?

```

```
0033 025650 001411 BEQ T16L0R ;BRANCH IF NO
0034 025652 076600 MED ;GET TAG LOG INFO.
0035 025654 000107 .WORD RTAG
0036 025656 000300 SWAB R0 ;PUT TAG IN LOW BYTE
0037 025660 042700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
0038 025664 010037 001164 MOV R0,#REG3 ;SAVE BAD DATA
0039 025670 104070 ERROR 70 ;ERROR: TEST OF MSB ADDR. (A10) TO ADDRESS FIELD FAILED
0040 ; TAG PARITY ERROR
0041 025672 000417 BR T16L04 ;GO TO END OF TEST
0042
0043 025674 032737 000100 177744 T16L08: BIT #100,##REG ;LOW BYTE P.E.?
0044 025702 001406 BEQ T16L09 ;BRANCH IF NO
0045 025704 076600 MED ;GET LOG INFORMATION
0046 025706 000106 .WORD CDL
0047 025710 010037 001164 MOV R0,#REG3 ;SAVE INFORMATION
0048 025714 104071 ERROR 71 ;ERROR: TEST OF MSB ADDR. (A10) TO ADDRESS FIELD FAILED
0049 ; LOW BYTE PARITY ERROR
0050 025716 000405 BR T16L04 ;GO TO END OF TEST
0051
0052 025720 T16L09: MED ;GET LOG INFORMATION
0053 025722 076600 .WORD CDH
0054 025724 000106 MOV R0,#REG3 ;SAVE INFORMATION
0055 025724 010037 001164 ERROR 72 ;ERROR: TEST OF MSB ADDR. (A10) TO TAG FIELD FAILED
0056 025730 104072 ERROR ; HIGH BYTE PARITY ERROR
0057
0058
0059 025732 005037 177572 T16L04: CLR #MMR0 ;KT OFF
0060 025736 012737 033142 000114 MOV #UPERR,0#PVEC ;RESTORE PARITY ERROR HANDLER
0061 025744 000401 BR TSI32 ;GO TO NEXT TEST
0062
0063 025746 000000 LAST1: .WORD 0 ;TEST ADDRESS LOCATION
0064
0065 ;*****
0066 ;TEST 32 TEST OF MSB ADDRESS (A10) TO CACHE DATA FIELD
0067 ;*
0068 ;* THIS TEST CHECKS FOR DUAL ADDRESSING ON THE DATA FIELD
0069 ;*FOR THE MSB (A10) ADDRESS TO CACHE. THERE ARE TWO PASSES.
0070 ;*THE FIRST EXERCISES THE DATA BITS AND THE SECOND EXERCISES
0071 ;*THE DATA PARITY BITS. THE TEST DATA IS STORED IN A TABLE
0072 ;*(TPAT) AT THE END OF THE TEST. INITIALLY TEST ADDRESSES
0073 ;*ARE CALCULATED WHICH DON'T OVERLAP THE TEST INSTRUCTIONS
0074 ;*AND WHICH HAVE THE SAME CACHE ADDRESS (A1-A9) EXCEPT FOR
0075 ;*A10. ONE ADDRESS IS THE LAST LOC IN THIS TEST (TAD1)
0076 ;*AND THE SECOND LIES IN A 1K BUFFER AT THE END OF THE
0077 ;*PROGRAM. A SUBROUTINE, HAD, GENERATES THIS SECOND ADDRESS.
0078 ;*ON THE FIRST PASS DIFFERENT TEST DATA IS WRITTEN IN THE
0079 ;*CONGRUENT ADDRESS AND THEN CHECKED TO BE A HIT AND TO
0080 ;*BE THE CORRECT VALUE. ON THE SECOND PASS NEW DATA IS
0081 ;*CHOSEN, WHICH GENERATES OPPOSITE PARITY IN THE DATA FIELD.
0082 ;*IS WRITTEN IN THE ADDRESSES AND THEN CHECKED TO BE A HIT
0083 ;*AND TO BE THE CORRECT VALUE. ANY PARITY ERRORS OR HIT
0084 ;*ERRORS ARE REPORTED.
0085 ;* R2 CONTAINS THE CONGRUENT ADDRESS FOR TAD1
0086 ;*****
0087
0088 025750 012737 000214 177746 TST32: MOV #214,0#CCR ;CACHE OFF FOR SCOPE
```

```
0089 025756 000004 SCOPE
0090 025760 012737 026444 001234 MOV #TST33,0#TST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
0091 025766 012737 026224 000114 MOV #T17L01,0#PVEC ;SET UP FOR PARITY ERRORS
0092 025774 004737 033714 JSR PC,HAD ;CALC CONGRUENT ADDRESS IN TEST BUFFER
0093 026000 026442 .WORD TAD1 ;TEST ADDRESS
0094 026002 013702 001172 MOV #TMP0,R2 ;SAVE CONGRUENT ADDRESS
0095 026006 005000 CLR R0 ;INIT TEST PATTERN ADDRESS REG
0096 026010 012737 000200 177746 T17L08: MOV #200,0#CCR ;ALL CACHE ON
0097 026016 016037 026432 026442 T17L07: MOV TPAT(R0),0#TAD1 ;WRITE CACHE LOCS WITH
0098 026024 016012 026436 MOV TPAT+4(R0),(R2) ;ADDRESS BIT A10 COMPLEMENTED
0099 026030 013701 026442 MOV #TAD1,R1 ;SEE IF DATA IN CACHE
0100 026034 033727 177752 000004 BIT #MMR,0#MR2 ;HIT?
0101 026042 001420 BEQ T17L02 ;BRANCH IF NO TO ERROR
0102 026044 020160 026432 CMP R1,TPAT(R0) ;DATA CORRECT?
0103 026050 001051 BNE T17L03 ;BRANCH IF NO TO ERROR
0104 026052 011201 MOV (R2),R1 ;SEE IF NEXT DATA IN CACHE
0105 026054 033727 177752 000004 BIT #MMR,0#MR2 ;HIT?
0106 026062 001425 BEQ T17L04 ;BRANCH IF NO TO ERROR
0107 026064 020160 026436 CMP R1,TPAT+4(R0) ;DATA OK?
0108 026070 001033 BNE T17L05 ;BRANCH IF NO TO ERROR
0109 026072 005760 026436 TST TPAT+4(R0) ;TEST IF FIRST PASS
0110 026076 100151 BPL T17L06 ;BRANCH TO END OF TEST IF NO
0111 026100 005720 TST (R0)+ ;UPDATE ADDRESS
0112 026102 000745 BR T17L07 ;GO TEST NEW DATA
0113
0114 026104 052737 000014 177746 T17L02: BIS #14,0#CCR ;CACHE OFF
0115 026112 012737 026442 001162 MOV #TAD1,0#REG2 ;SAVE ADDRESS
0116 026120 012737 026010 001110 T17L09: MOV #T17L08,0#LPERR ;INIT. RETURN FOR ERROR LOOP
0117 026126 005037 001160 CLR #REG1 ;SAVE ADDRESS
0118 026132 104062 ERROR 62 ;ERROR: TEST OF MSB ADDRESS (A10) TO DATA FIELD FAILED
0119 ; ADDRESS COULD NOT BE MADE A HIT
0120 026134 000532 BR T17L06 ;GO TO END OF TEST
0121
0122 026136 052737 000014 177746 T17L04: BIS #14,0#CCR ;CACHE OFF
0123 026144 010237 001162 MOV R2,0#REG2 ;SAVE ADDRESS
0124 026150 000763 BR T17L09 ;REPORT ERROR
0125
0126 026152 052737 000014 177746 T17L05: BIS #14,0#CCR ;CACHE OFF
0127 026160 016037 026436 001166 MOV TPAT+4(R0),0#REG4 ;SAVE GOOD DATA
0128 026166 010237 001162 MOV R2,0#REG2 ;SAVE BAD ADDRESS
0129 026172 000406 BR T17L10 ;REPORT ERROR
0130
0131 026174 052737 000014 177746 T17L03: BIS #14,0#CCR ;CACHE OFF
0132 026202 016037 026432 001166 MOV TPAT(R0),0#REG4 ;SAVE GOOD DATA
0133 026210 010137 001164 T17L10: MOV R1,0#REG3 ;SAVE BAD DATA
0134 026214 005037 001160 CLR #REG1 ;SAVE BAD ADDRESS
0135 026220 104063 ERROR 63 ;ERROR: TEST OF MSB ADDR. (A10) TO DATA FIELD FAILED
0136 ; ADDRESS HELD WRONG DATA
0137 026222 000477 BR T17L06 ;GO TO END OF TEST
0138
0139 026224 052737 000014 177746 T17L01: BIS #14,0#CCR ;CACHE OFF
0140
0141 026232 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
0142 026234 076600 MED ;GET CONTENTS OF LOG REG
0143 026236 000022 .WORD RLOG
0144 026240 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
```

```

4145 026244 076600 MED ;UNLOCK ERROR LOG
4146 026246 000222 .WORD WLOG
4147 026250 012600 MOV (SP)+,R0 ;RESTORE R0
4148
4149 026252 011637 001164 MOV (SP),#REG3 ;GET PC+2 OF ERROR
4150 026256 162737 000002 001164 SUB #2,#REG3 ;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 R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
4155 026274 042700 177776 BIC #177776,R0 ;ONLY LOOK AT A17, A16
4156 026300 010037 001160 MOV R0,#REG1 ;SAVE ADDRESS
4157 026304 076600 MED ;GET LOG INFORMATION
4158 026306 000102 .WORD LOADD
4159 026310 010037 001162 MOV R0,#REG2 ;SAVE INFORMATION
4160 026314 076600 MED ;GET LOG INFORMATION
4161 026316 000100 .WORD RJAM
4162 026320 032700 000400 BIT #400,R0 ;ERROR IN BACKING STORE
4163 026324 001402 BEQ T17L11 ;BRANCH IF NO
4164 026326 104001 EPROR 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 R0,#REG3 ;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 R0,#REG3 ;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: MED ;GET TAG LOG INFO.
4186 026406 076600 .WORD RTAG
4187 026408 000107 SWAB R0 ;PUT TAG IN LOW BYTE
4188 026406 000300 RIC #177400,R0 ;LOOK AT TAG ONLY
4189 026410 042700 177400 MOV R0,#REG3 ;SAVE BAD DATA
4190 026414 010037 001164 ERROR 66 ;ERROR: TEST OF MSB ADDR. (A10) TO DATA FIELD FAILED
4191 026420 104066 ; PARITY ERROR TAG
4192
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 .WORD 1403 ;TEST DATA FOR PARITY BIT TEST
4202
4203 026442 000000 TAD1: .WORD 0 ;TEST ADDRESS
4204
4205 ;*****
4206 ;*TEST 33 TEST CACHE IS NOT ALLOCATED DURING ODD ADDRESS TRAP
4207 ;*
4208 ;*THIS TEST FIRST PUTS DATA IN A CACHE LOC. THEN A WORD
4209 ;*INSTRUCTION TO AN ODD BYTE ADDRESS TRIES TO CLEAR THE
4210 ;*LOC AND FORCE AN ODD ADDRESS ERROR. UPON TRAPPING, THE
4211 ;*LOC IN CACHE IS LOOKED AT AND VERIFIED TO NOT HAVE CHANGED.
4212
4213 ;*****
4214 026444 012737 000214 177746 TST33: MOV #214,#CCR ;CACHE OFF FOR SCOPE
4215 026452 000004 SCOPE
4216 026454 012737 026634 001234 MOV #TST34,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4217 026462 012737 000200 177746 MOV #200,#CCR ;CACHE ON
4218 026470 012737 026510 000004 MOV #T27L01,#4 ;SETUP FOR ODD ADDRESS TRAP
4219 026476 012737 177777 060000 MOV #177777,#BUFL ;PUT DATA IN CACHE
4220 026504 005037 060001 CLR #BUFL+1 ;FORCE ODD ADDRESS ERROR
4221
4222 026510 022626 T27L01: CMP (SP)+,(SP)+ ;RESTORE THE STACK
4223
4224 026512 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
4225 026514 076600 MED ;GET CONTENTS OF LOG REG
4226 026516 000022 .WORD RLOG
4227 026520 052700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
4228 026524 076600 MED ;UNLOCK ERROR LOG
4229 026526 000222 .WORD WLOG
4230 026530 012600 MOV (SP)+,R0 ;RESTORE R0
4231
4232 026532 013700 060000 MOV #BUFL,R0 ;GET DATA
4233 026536 033727 177752 000004 BIT #HMR,HMR2 ;HIT?
4234 026544 001407 BEQ T27L02 ;BRANCH TO ERROR IF NO
4235 026546 020027 177777 CMP R0,#177777 ;DATA UNCHANGED?
4236 026552 001016 BNE T27L03 ;BRANCH IF YES TO ERROR
4237 026554 012737 033352 000004 T27L04: MOV #UT4,#4 ;RESTORE HANDLER FOR UNEXPECTED TRAPS TO 4
4238 026562 000424 BR TST34 ;GO TO NEXT TEST
4239
4240 026564 052737 000014 177746 T27L02: BIS #14,#CCR ;CACHE OFF
4241 026572 005037 001160 CLR #REG1 ;SAVE FAILING ADDRESS
4242 026576 012737 060000 001162 MOV #BUFL,#REG2 ;SAVE FAILING ADDRESS
4243 026604 104043 EPROR 43 ;ERROR! ADDRESS COULD NOT BE MADE A HIT
4244 026606 000762 BR T27L04 ;GO TO END OF TEST
4245
4246 026610 032737 000014 177746 T27L03: BIS #14,#CCR ;CACHE OFF
4247 026616 005037 001160 CLR #REG1 ;SAVE BAD ADDRESS
4248 026622 012737 060001 001162 MOV #BUFL+1,#REG2 ;SAVE BAD ADDRESS
4249 026630 104116 EPROR 116 ;ERROR! CACHE ALLOCATED DURING ODD ADDRESS TRAP
4250 026632 000750 BR T27L04 ;GO TO END OF TEST
4251
4252 ;*****
4253 ;*TEST 34 TEST CACHE NOT ALLOCATED DURING RED ZONE TRAP
4254 ;*
4255 ;* THIS TEST FIRST PUTS DATA IN A CACHE LOC WHICH COR-
4256 ;*RESPONDS TO A RED ZONE ADDRESS. A STACK OPERATION IS

```

```

4257 ;*DONE TO THIS ADDRESS WHICH WILL CHANGE THE DATA IF
4258 ;*COMPLETED, UPON TRAPPING, THE DATA IN CACHE IS LOOKED
4259 ;*AT AND VERIFIED TO NOT HAVE CHANGED.
4260
4261 ;*****
4262 TST34: MOV #214,#CCR ;CACHE OFF FOR SCOPE
4263 SCOPE
4264 MOV #TST35,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4265 MOV #200,#CCR ;CACHE ON
4266 MOV #T28L01,#4 ;SET UP FOR RED ZONE TRAPS
4267 CLR #17774 ;INITIALIZE THE STACK LIMIT REG
4268 CLR #0336 ;INITIALIZE TEST LOC
4269 MOV #336,SP ;PUT RED ZONE TRAP ADDRESS IN STACK PTER
4270 MOV #17777,(SP) ;FORCE RED ZONE TRAP
4271
4272 T28L01: MOV #1100,SP ;RESTORE THE STACK
4273
4274 MOV R0,-(SP) ;SAVE R0 FOR MED INST
4275 MED ;GET CONTENTS OF LOG REG
4276 .WORD RLOG
4277 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
4278 MED ;UNLOCK ERROR LOG
4279 .WORD WLOG
4280 MOV (SP)+,R0 ;RESTORE R0
4281
4282 MOV #0336,R0 ;GET DATA
4283 BIT #0HMR,#HMR2 ;HIT?
4284 BEQ T28L02 ;BRANCH IF NO
4285 TST R0 ;DATA UNCHANGED?
4286 BNE T28L03 ;BRANCH IF NO TO ERROR
4287 MOV #044,#4 ;RESTORE HANDLER FOR UNEXP. TRAPS TO 4
4288 CLR #0 ;RESTORE LOC 0
4289 CLR #2 ;RESTORE LOC 2
4290 BR TST35 ;GO TO NEXT TEST
4291
4292 T28L02: BIS #14,#CCR ;CACHE OFF
4293 CLR #REG1 ;SAVE FAILING ADDR.
4294 MOV #336,#REG2 ;SAVE FAILING ADDR.
4295 ERROR 43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT
4296 BR T28L04 ;GO TO END OF TEST
4297
4298 T28L03: BIS #14,#CCR ;CACHE OFF
4299 ERROR 117 ;ERROR: CACHE ALLOCATED DURING RED ZONE TRAP
4300 BR T28L04 ;GO TO END OF TEST
4301
4302 ;*****
4303 ;*TEST 35 TEST CACHE NOT ALLOCATED DURING KT ABORT
4304 ;*
4305 ;* DATA IS PUT IN CACHE IN A TEST BUFFER ADDRESS. KIPAR4
4306 ;*IS SET UP TO REFERENCE THAT ADDRESS AND KIPDR4 IS SET
4307 ;*UP TO ABORT ACCESSES TO NON RESIDENT PAGE. THE KT IS
4308 ;*TURNED ON AND A MEMORY REFERENCE THROUGH KIPAR4 IS MADE
4309 ;*WHICH WOULD MODIFY THE TEST LOCATION IF COMPLETED. UPON
4310 ;*TRAPPING, THE LOCATION IS LOOKED AT AND VERIFIED TO NOT
4311 ;*HAVE CHANGED.
4312 ;* IF THE INHIBIT TEST USING KT SWITCH (SW12) IS SET,

```

```

4313 ;*THIS TEST IS SKIPPED.
4314
4315 ;*****
4316 TST35: MOV #14,#CCR ;CACHE OFF FOR SCOPE
4317 SCOPE
4318 MOV #TST36,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4319 BIT #SW12,0SWR ;INHIBIT TESTS USING KT?
4320 BNE TST36 ;YES, GO TO NEXT TEST
4321 BIS #200,#KT1 ;USE KT FOR $SIZE
4322 JSR PC,$SIZE ;USE $SIZE TO SET UP PAR'S AND PDR'S
4323 MOV #T29L01,#250 ;SET UP FOR KT ABORTS
4324 MOV #77400,KIPDR4 ;SET UP PDR4 TO ABORT ACCESS TO NON RESIDENT PAGE
4325 MOV #0BUFL,R0 ;GET TEST ADDRESS
4326 BIC #100000,R0 ;MASK ITS PAR ADDRESS
4327 BIS #100000,R0 ;HAVE IT ADDRESS PAR4
4328 MOV #0KIPAR3,#KIPAR4 ;INIT PAR4 TO HAVE SAME OFFSET AS PAR3 FOR THE BUFFER
4329 MOV #200,#CCR ;CACHE ON
4330 MOV #17777,#0BUFL ;INIT TEST ADDRESS
4331 BIS #1,#HMR0 ;KT ON
4332 CLR (R0) ;FORCE KT ABORT
4333
4334 T29L01: CMP (SP)+,(SP)+ ;RESTORE STACK
4335
4336 MOV R0,-(SP) ;SAVE R0 FOR MED INST
4337 MED ;GET CONTENTS OF LOG REG
4338 .WORD RLOG
4339 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
4340 MED ;UNLOCK ERROR LOG
4341 .WORD WLOG
4342 MOV (SP)+,R0 ;RESTORE R0
4343
4344 MOV #0BUFL,R1 ;GET ADDRESS
4345 BIT #0HMR,#HMR2 ;HIT?
4346 BEQ T29L02 ;BRANCH IF NO
4347 CMP R1,#17777 ;DATA OK?
4348 BNE T29L03 ;BRANCH IF NO
4349 T29L04: BIC #1,#HMR0 ;KT OFF
4350 BIS #6,#KIPDR4 ;ALLOW READ OR WRITE TO PAGE
4351 MOV #252,#250 ;RESTORE KT TRAP CATCHER
4352 BR TST36 ;GO TO NEXT TEST
4353
4354 T29L02: BIS #14,#CCR ;CACHE OFF
4355 CLR #REG1 ;SAVE FAILING ADDRESS
4356 MOV #0BUFL,#REG1 ;SAVE FAILING ADDRESS
4357 ERROR 43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT
4358 BR T29L04 ;GO TO END OF TEST
4359
4360 T29L03: BIS #14,#CCR ;CACHE OFF
4361 ERROR 120 ;ERROR: CACHE ALLOCATED DURING KT ABORT
4362 BR T29L04 ;GO TO END OF TEST
4363
4364 ;*****
4365 ;*TEST 36 DYNAMIC TEST OF CACHE
4366 ;*
4367 ;* THIS TEST CREATES A GREAT DEAL OF ACTIVITY IN CACHE
4368 ;*TO TRY TO FIND ANY NOISE OR TIMING PROBLEMS. THESE

```

```

4369 ;*PROBLEMS WILL BE DETECTED VIA THE PARITY ERRORS, ILLEGAL
4370 ;*INSTRUCTION TRAPS OR DATA CHANGES THEY CAUSE. FIRST
4371 ;*CACHE IS LOADED WITH AN ALTERNATING DATA PATTERN (525,252).
4372 ;*THEN IT IS REFERENCED AS QUICKLY AS POSSIBLE IN OPPOSITE
4373 ;*DIRECTIONS TO CAUSE LARGE CHANGES IN THE ADDRESS LINES AND
4374 ;*RAPID CHANGES IN THE DATA LINES. THIS IS THEN REPEATED
4375 ;*WITH A DIFFERENT DATA PATTERN AND THE CACHE IS MODIFIED
4376 ;*AS THE REFERENCES OCCUR. AFTER THIS THE LOCATIONS ARE
4377 ;*CHECKED TO CONTAIN THEIR PROPER VALUES.
4378 ;* FOLLOWING THIS, THE TAG FIELD IS WRITTEN WITH A
4379 ;*CHANGING PATTERN. THEN THE CACHE IS REFERENCED AS QUICKLY
4380 ;*AS POSSIBLE IN OPPOSITE DIRECTIONS TO CAUSE LARGE CHANGES
4381 ;*IN THE ADDRESS LINES AND RAPID CHANGES IN THE TAG FIELD.
4382 ;*THIS LAST PART IS SKIPPED IF THE INHIBIT TEST USING KT
4383 ;*SWITCH (SW12) IS SET.
4384
4385 ;*****
4386 027300 012737 000214 177746 TST36: MOV #214,#CCR ;CACHE OFF FOR SCOPE
4387 027306 000004 SCOPE
4388 027310 012737 030260 001234 MOV #TST37,SKTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4389 027316 012737 030036 000114 MOV #T18L01,#PVEC ;SETUP FOR PARITY ERRORS
4390 027324 012737 027710 000010 MOV #T18L02,#10 ;SETUP FOR TRAPS TO ILLEGAL INST
4391 027332 012737 027340 001110 MOV #T18L11,#LPERR ;INIT RETURN FOR ERROR LOOPS
4392 027340 012737 000200 177746 T18L11: MOV #200,#CCR ;CACHE ON
4393
4394 ;GENERATE TEST DATA IN A 1K BUFFER
4395
4396 027346 012703 060000 MOV #BUFL,R3 ;GET STARTING ADDRESS OF BUFFER
4397 027352 012702 002000 MOV #200,R2 ;INIT REG FOR 1K COUNT
4398 027356 012701 176540 MOV #176540,R1 ;PUT RANDOM # IN REG
4399 027362 012700 023456 MOV #023456,R0 ;PUT RANDOM # IN REG
4400 027366 060001 18: ADD R0,R1 ;GENERATE NEW RANDOM DATA
4401 027370 010123 MOV R1,(R3)+ ;SAVE DATA
4402 027372 000261 SEC ;GENERATE MORE
4403 027374 006101 ROL R1 ;RANDOM DATA
4404 027376 006000 ROR R0
4405 027400 077206 SOB R2,18 ;LOOP TILL 1K BUFFER FULL
4406
4407 ;LOAD CACHE WITH PATTERN AND TEST CACHE
4408
4409 027402 012700 060000 MOV #BUFL,R0 ;SET UP TO ADDRESS BUFFER
4410 027406 012701 060000 MOV #BUFL,R1 ;ASET UP TO ADDRESS BUFFER
4411 027412 012702 002000 MOV #200,R2 ;INIT REG FOR 1K COUNT
4412 027416 012703 002000 MOV #200,R3 ;INIT REG FOR 1K COUNT
4413 027422 005721 28: TST (R1)+ ;GET DATA IN CACHE
4414 027424 077202 SOB R2,28 ;LOOP TILL 1K REFERENCED
4415 027426 022041 38: CMP (R0)+,-(R1) ;REFERENCE CACHE QUICKLY AND WITH COMPLEMENT ADDR
4416 027430 077302 SOB R3,38 ;LOOP TILL ALL CACHE REFERENCED
4417
4418 ;GENERATE SECOND TEST PATTERN IN BUFFER AND TEST IT
4419
4420 027432 012700 060000 MOV #BUFL,R0 ;SET UP TO ADDRESS BUFFER
4421 027436 012701 060000 MOV #BUFL,R1 ;SET UP TO ADDRESS BUFFER
4422 027442 012702 002000 MOV #200,R2 ;INIT REG FOR 1K COUNT
4423 027446 012703 002000 MOV #200,R3 ;INIT REG FOR 1K COUNT
4424 027452 005004 CLR R4 ;INIT DATA

```

```

4425 027454 010421 58: MOV R4,(R1)+ ;LOAD BUFFER WITH PATTERN
4426 027456 005204 INC R4 ;CHANGE DATA
4427 027460 077203 SOB R2,58 ;LOOP TILL 1K LOADED
4428
4429 027462 062041 68: ADD (R0)+,-(R1) ;REFERENCE CACHE QUICKLY
4430 027464 077302 SOB R3,68 ;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 78: MOV -(R0),R3 ;GET DATA
4437 027500 020103 CMP R1,R3 ;DATA OK?
4438 027502 001110 BNE T18L03 ;BRANCH IF NO TO ERROR
4439 027504 077204 SOB R2,78 ;LOOP TILL 1/2K REFERRED
4440 027506 012702 001000 MOV #1000,R2 ;INIT REG FOR 1/2K COUNT
4441 027512 012701 002776 108: MOV -(R0),R3 ;INIT REG WITH 'GOOD' DATA
4442 027516 014003 CMP R1,R3 ;GET DATA
4443 027520 020103 BNE T18L03 ;DATA OK?
4444 027522 001130 DEC R1 ;BRANCH IF NO TO ERROR
4445 027524 005301 DEC R1 ;ADJUST GOOD DATA
4446 027526 077205 SOB R2,108 ;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 118 ;CONTINUE TEST IF NO
4452 027540 000137 030260 JMP #TST37 ;GO TO NEXT TEST
4453 027544 052737 000200 036034 118: BIS #200,#KT11 ;KT ON FOR $SIZE
4454 027552 004737 035750 JSR PC,$SIZE ;SIZE MEMORY
4455 027556 013737 027564 001110 MOV T18L05,#LPERR ;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 #06LSTBK,R2 ;GET LAST BANK
4462 027616 010215 MOV R2,(R5) ;SET UP PAR5
4463 027620 010714 T18L06: MOV R2,(R4) ;SET UP PAR4
4464 027622 052737 000001 177572 BIS #1,#MMR0 ;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 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 BIT #3776,R1 ;ALL CACHE TESTED?
4474 027656 001002 BNE 28 ;BRANCH IF NO TO CONTINUE
4475 027660 000137 030234 JMP T18L10 ;GO TO END OF TEST
4476 027664 162715 28: SUB #40,(R5) ;ADJUST PAR5 FOR NEXT TEST ADDR. REF.
4477 027670 100001 BPL 18 ;TAG > OR EQUAL 0, BRANCH IF YES
4478 027672 010215 MOV R2,(R5) ;NO, INIT PAR5 FOR HIGHEST TAG ADDR
4479 027674 062714 18: AND #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      #14,0*CCR      ;CACHE OFF
4486 027716 011637 001164          MOV     (SP),R0      ;GET PC+2 OF TRAP
4487 027722 162737 000002 001164 SUB     #2,R0      ;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     R0      ;PUT PHY. ADDR A17, A16 IN LOW BYTE
4492 027740 042700 177776 BIC      #177776,R0    ;ONLY LOOK AT A17, A16
4493 027744 010037 001160          MOV     R0,R0      ;SAVE ADDRESS
4494 027750 076600          MED     ;GET LOG INFORMATION
4495 027752 000102          .WORD  LOADD     ;SAVE INFORMATION
4496 027754 010037 001162          MOV     R0,R0      ;
4497
4498 027760 010046          MOV     R0,-(SP)    ;SAVE R0 FOR MED INST
4499 027762 076600          MED     ;GET CONTENTS OF LOG REG
4500 027764 000022          .WORD  RLOG      ;
4501 027766 052700 100001          BIS     #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
4502 027772 076600          MED     ;UNLOCK ERROR LOG
4503 027774 000222          .WORD  WLOG      ;
4504 027776 012600          MOV     (SP)+,R0    ;RESTORE R0
4505
4506 030000 104074          ERROR    74      ;ERROR: DYNAMIC TEST OF CACHE FAILED
4507
4508 030002 000514          BR       T18L10      ;
4509
4510 030004 052737 000014 177746 T18L03: BIS      #14,0*CCR      ;CACHE OFF
4511 030012 005037 001160          CLR      R0      ;SAVE ADDRESS
4512 030016 010037 001162          MOV     R0,R0      ;SAVE ADDRESS
4513 030022 010037 001164          MOV     R3,R0      ;SAVE BAD DATA
4514 030026 010137 001166          MOV     R1,R0      ;SAVE GOOD DATA
4515 030032 104073          ERROR    73      ;ERROR: DYNAMIC TEST OF CACHE FAILED
4516
4517 030034 000477          BR       T18L10      ;
4518
4519 030036 052737 000014 177746 T18L01: BIS      #14,0*CCR      ;CACHE OFF
4520
4521 030044 010046          MOV     R0,-(SP)    ;SAVE R0 FOR MED INST
4522 030046 076600          MED     ;GET CONTENTS OF LOG REG
4523 030050 000022          .WORD  RLOG      ;
4524 030052 052700 100001          BIS     #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
4525 030056 076600          MED     ;UNLOCK ERROR LOG
4526 030060 000222          .WORD  WLOG      ;
4527 030062 012600          MOV     (SP)+,R0    ;RESTORE R0
4528
4529 030064 011637 001164          MOV     (SP),R0      ;GET PC+2 OF TRAP
4530 030070 162737 000002 001164 SUB     #2,R0      ;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     R0      ;PUT PHY. ADDR A17, A16 IN LOW BYTE
4535 030106 042700 177776 BIC      #177776,R0    ;ONLY LOOK AT A17, A16
4536 030112 010037 001160          MOV     R0,R0      ;SAVE ADDRESS

```

```

4537 030116 076600          MED     ;GET LOG INFORMATION
4538 030120 000102          .WORD  LOADD     ;SAVE INFORMATION
4539 030122 010037 001162          MOV     R0,R0      ;GET LOG INFORMATION
4540 030126 076600          MED     ;
4541 030130 000100          .WORD  RJAM      ;
4542 030132 032700 000400          BIT     #400,R0    ;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     #100,0*EREG    ;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     R0,R0      ;SAVE INFORMATION
4552 030164 104075          ERROR    75      ;ERROR: DYNAMIC TEST OF CACHE FAILED
4553
4554 030166 000422          BR       T18L10      ;
4555
4556 030170 032737 000200 177744 T18L13: BIT     #200,0*EREG    ;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     R0,R0      ;SAVE INFORMATION
4561 030210 104076          ERROR    76      ;ERROR: DYNAMIC TEST OF CACHE FAILED
4562
4563 030212 000410          BR       T18L10      ;
4564
4565 030214          T18L14:
4566 030214 076600          MED     ;GET TAG LOG INFO.
4567 030216 000107          .WORD  RTAG      ;
4568 030220 000300          SWAB     R0      ;PUT TAG IN LOW BYTE
4569 030222 042700 177400          BIC     #177400,R0    ;LOOK AT TAG ONLY
4570 030226 010037 001164          MOV     R0,R0      ;SAVE BAD DATA
4571 030232 104077          ERROR    77      ;ERROR: DYNAMIC TEST OF CACHE FAILED
4572
4573
4574 030234 005037 177572          T18L10: CLR     #MMR0    ;KT OFF
4575 030240 012737 000012 000010 MOV     #12,0*10    ;RESTORE TRAP CATCHER
4576 030246 005037 000012          CLR     #12      ;RESTORE TRAP CATCHER
4577 030252 012737 033142 000114 MOV     #UPERR,0*PVEC ;RESTORE HANDLER FOR PARITY ERRORS
4578
4579
4580
4581 ;*****
4582 ;*TEST 37      TEST RETRIES TO BACKING STORE DONE ON CACHE PARITY TRAP
4583 ;*
4584 ;* THE JAMUPP ON CACHE PARITY ERROR BIT IS CLEARED AND
4585 ;*THE CACHE CONTROL REG IS TESTED TO CONTAIN THE CORRECT
4586 ;*VALUE. A CACHE LOC IS THEN WRITTEN WITH WRONG PARITY
4587 ;*AND A TRAP IS FORCED. THE LOC IS THEN REFERENCED TO SEE
4588 ;*IF IT STILL IS IN CACHE (RETRY DONE).
4589 ;*****
4590 030260 012737 000214 177746 TST37: MOV     #214,0*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

```



```
4593 030276 012737 030350 000114      MOV    #18,0#PVEC    ;SET UP FOR PARITY TRAP
4594 030304 042737 000200 177746      BIC    #200,0#CCR    ;ENABLE RETRIES
4595 030312 032737 000200 177746      BIT    #200,0#CCR    ;WAS BIT CLEARED?
4596 030320 001045                      BNE    20             ;BRANCH IF NO TO ERROR
4597 030322 012737 000100 177746      MOV    #100,0#CCR    ;CACHE ON, WRITE WRONG PARITY, DO RETRIES
4598 030330 005037 060000              CLR    0#BUFL        ;WRITE WRONG PARITY
4599 030334 012737 000000 177746      MOV    #0,0#CCR     ;WMP OFF
4600 030342 005737 060000              TST    0#BUFL        ;FORCE TRAP
4601 030346 000445                      BR     38             ;REPORT ERROR IF NO TRAP
4602
4603 030350 062706 000004      18:    ADD    #4,SP    ;RESTORE THE STACK
4604
4605 030354 010046              MOV    R0,-(SP)     ;SAVE R0 FOR MED INST
4606 030356 076600              MED                      ;GET CONTENTS OF LOG REG
4607 030360 000022              .WORD  RLOG
4608 030362 052700 100001      BIS    #100001,R0    ;ENABLE ERROR LOG & LOG FIRST MODE
4609 030366 076600              MED                      ;UNLOCK ERROR LOG
4610 030370 000222              .WORD  WLOG
4611 030372 012600              MOV    (SP)+,R0    ;RESTORE R0
4612
4613 030374 005737 060000              TST    0#BUFL        ;SEE IF DATA IN CACHE
4614 030400 033727 177752 000004      BIT    0#HMR,#HMR2  ;HIT?
4615 030406 001036              BNE    T23L01       ;GO TO END OF TEST IF YES
4616 030410 012737 000214 177746      MOV    #214,0#CCR    ;CACHE OFF
4617
4618                      ;RID CACHE OF BAD PARITY
4619 030416 012737 000214 177746      MOV    #214,0#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 26:    MOV    0#CCR,#REG1  ;SAVE BAD DATA
4627 030442 012737 000214 177746      MOV    #214,0#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 38:    MOV    #214,0#CCR    ;CACHE OFF
4633 030470 005037 001160              CLR    #REG1         ;SAVE ADDR. OF TESTED LOC
4634 030474 012737 060000 001162      MOV    0#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    #214,0#CCR    ;CACHE OFF IF ON
4641 030512 004737 035134      JSR    PC,SWEEP      ;GO PURGE CACHE
4642
4643
4644 030516 012737 033142 000114      MOV    0#UPERR,0#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                      ;* THE TEST INSTRUCTION ADDRESSES ARE FIRST EXAMINED TO
4650                      ;* DETERMINE IF THEY OVERLAP THE TEST LOCATION ADDRESS IN
4651                      ;* CACHE. IF THEY DO, THE TEST IS RUN IN A NON OVERLAPPING
4652                      ;* ADDRESS SPACE. A LOC IS PUT IN CACHE WHICH HAS THE SAME
4653                      ;* 11 LEAST SIGNIFICANT ADDRESS BITS AS THE MEMORY MANAGEMENT
4654                      ;* REG KIPAR0. A DATO IS THEN DONE TO KIPAR0 AND THE LOC
4655                      ;* IS CHECKED TO STILL BE IN CACHE.
4656
4657                      ;*****
4658 030524 012737 000214 177746      TST40:  MOV    #214,0#CCR    ;CACHE OFF FOR SCOPE
4659 030532 000004              SCOPE
4660 030534 012737 030674 001234      MOV    #TST41,SKTST  ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4661 030542 012737 000200 177746      MOV    #200,0#CCR    ;TURN ON ALL OF CACHE
4662 030550 012737 030610 001172      MOV    #T19L01,0#TMP0 ;SAVE ADDRESS OF TEST INSTRUCTION
4663 030556 042737 174000 001172      BIC    #174000,0#TMP0 ;LOOK AT ITS CACHE ADDRESS
4664 030564 023727 001172 002326      CMP    0#TMP0,#2326  ;INSTRUCTION AT TEST LOC?
4665 030572 002404              BLT    T19L02       ;BRANCH IF NO
4666 030574 033727 001172 002340      CMP    0#TMP0,#2340  ;INSTRUCTION AT TEST LOC?
4667 030602 003422              BGE    T19L03       ;BRANCH IF YES
4668 030604 005737 002340              TST    #2340         ;PUT TEST LOC IN CACHE
4669 030610 005037 172340      T19L02: CLR    0#KIPAR0    ;DO DATO TO I/O
4670 030614 005737 002340      T19L01: TST    0#KIPAR0    ;DATA STILL IN CACHE
4671 030620 033727 177752 000004      BIT    0#HMR,#HMR2  ;WAS IT A HIT?
4672 030626 001022              BNE    TST41         ;GO TO NEXT TEST IF YES
4673 030630 012737 000003 001160      T19L04: MOV    #3,#REG1  ;SAVE PHYSICAL ADDRESS HIGH
4674 030636 012737 172340 001162      MOV    #172340,#REG2 ;SAVE PHYSICAL ADDRESS LOW
4675 030644 104025              ERROR  25         ;ERROR: DATO TO I/O ADDRESS WRITTEN IN CACHE
4676 030646 000412              BR     TST41         ;GO TO NEXT TEST
4677
4678 030650 005737 002340      T19L03: TST    #2340         ;PUT TEST LOC IN CACHE
4679 030654 005037 172340      CLR    0#KIPAR0    ;DO DATO TO I/O
4680 030660 005737 002340      TST    #2340         ;DATA STILL IN CACHE?
4681 030664 033727 177752 000004      BIT    0#HMR,#HMR2  ;STILL A HIT?
4682 030672 001756              BEQ    T19L04       ;BRANCH TO ERROR IF NO
4683
4684                      ;*****
4685                      ;*TEST 41      TEST CONSOLE INITIATED SWEEP INVALIDATES ALL CACHE
4686                      ;*
4687                      ;* A LOC IS PUT IN CACHE, CHECKED TO BE A HIT AND THEN
4688                      ;* A CONSOLE SWEEP IS INITIATED. THE LOC IS AGAIN REF-
4689                      ;* ERENCED TO SEE IF IT WAS INVALIDATED (NOT A HIT). THIS
4690                      ;* IS DONE FOR ALL OF CACHE. BEFORE THE CONSOLE SWEEP IS
4691                      ;* STARTED, THE TEST LOC IS VERIFIED TO NOT OVERLAP THE
4692                      ;* PROGRAM INSTRUCTION ADDRESSES IN CACHE. IF THEY DO, THE
4693                      ;* TEST IS RUN OUT OF A DIFFERENT ADDRESS SPACE.
4694                      ;* 00 CONTAINS THE ADDRESS UNDER TEST.
4695
4696                      ;*****
4697 030674 012737 000214 177746      TST41:  MOV    #214,0#CCR    ;CACHE OFF FOR SCOPE
4698 030702 000004              SCOPE
4699 030704 012737 031132 001234      MOV    #TST42,SKTST  ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4700 030712 012737 000200 177746      MOV    #200,0#CCR    ;CACHE ON
4701 030720 012702 060000      MOV    #0#BUFL,R1    ;INIT REG FOR TEST ADDRESS
4702 030724 012701 002000      MOV    #2000,R1     ;INIT LOOP COUNT
4703
4704                      ;DOES THE TEST ADDR OVERLAP THE SAME ADDR SPACE IN CACHE AS THE PROGRAM INSTRUCT
```

```

4705
4706 030730 010237 001172 T25L04: MOV R2,$TMP0 ;GET TEST ADDR.
4707 030734 012737 031006 001174 MOV #16,$TMP1 ;GET PROGRAM TEST INSTRUCTION ADDR.
4708 030742 042737 174000 001172 BIC #174000,$TMP0 ;CALC ADDRESSES CORRESP. CACHE ADDR
4709 030750 042737 174000 001174 BIC #174000,$TMP1 ;CALC ADDRESSES CORRESP. CACHE ADDR
4710 030756 023737 001174 001172 CMP $TMP1,$TMP0 ;DO THE CACHE ADDRESSES OVERLAP?
4711 030764 101010 10 BHI ;BRANCH IF NO
4712 030766 062737 000012 001174 ADD #12,$TMP1 ;CALC LAST PROG. TEST INSTRUCTION ADDR
4713 030774 023737 001172 001174 CMP $TMP0,$TMP1 ;DO THE CACHE ADDRESSES OVERLAP?
4714 031002 101415 BLOS T25L01 ;BRANCH IF YES
4715
4716 031004 005012 18: CLR (R2) ;PUT THE DATA IN CACHE
4717 031006 MOV #200,R0 ;SET BIT IN R0 FOR CONSOLE CACHE SWEEP
4718 031006 012700 000200 MED ;CONSOLE CACHE SWEEP
4719 031012 076600 .WORD WINIT ;SEE IF LOC STILL IN CACHE
4720 031014 000352 TST (R2) ;HIT?
4721 031016 005712 177752 000004 BIT #0HMR,$HMR2 ;BRANCH TO ERROR IF YES
4722 031020 033727 BNE T25L02 ;UPDATE ADDRESS
4723 031026 001016 T25L03: TST (R2) ;BRANCH IF ALL CACHE NOT TESTED
4724 031030 005722 SOB R1,T25L04 ;GO TO NEXT TEST
4725 031032 077142 BR TST42
4726 031034 000436
4727
4728 031036 005012 T25L01: CLR (R2) ;PUT DATA IN CACHE
4729 031040 012700 000200 MOV #200,R0 ;SET BIT IN R0 FOR CONSOLE CACHE SWEEP
4730 031044 076600 MED ;CONSOLE CACHE SWEEP
4731 031046 000352 .WORD WINIT ;SEE IF LOC STILL IN CACHE
4732 031050 005712 TST (R2) ;HIT?
4733 031052 033727 177752 000004 BIT #0HMR,$HMR2 ;BRANCH TO ERROR IF YES
4734 031060 001001 BNE T25L02 ;LOOK AT NEXT ADDRESS
4735 031062 000762 BR T25L03
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,$$LPERR ;INIT RETURN FOR ERROR LOOP
4741 031110 194113 ERROR 113 ;ERROR: ADDR. NOT INVALIDATED BY CONSOLE SWEEP
4742 031112 123727 001103 000003 CMPB #0$ERFLG,$3 ;MORE THAN 3 ERRORS?
4743 031120 101004 BHI TST42 ;GO TO NEXT TEST IF YES
4744 031122 012737 000200 177746 MOV #200,$CCR ;CACHE ON
4745 031130 000737 BR T25L03 ;CONTINUE TEST
4746
4747 ;*****
4748 ;*TEST 42 TEST POWER UP INVALIDATES CACHE AND CLEARS CACHE CONTROL REG
4749 ;*
4750 ;* THIS TEST IS ONLY RUN IF SW07=1. THIS IS BECAUSE
4751 ;*OPERATOR INTERVENTION IS NEEDED TO POWER DOWN AND THEN
4752 ;*UP THE MACHINE WHEN THE MESSAGE IS TYPED ON THE TTY.
4753 ;*IF RUNNING UNDER APT AND SW07=1,THE PROGRAM ASSUMES
4754 ;*THAT A UNIBUS EXERCISOR (M7055) IS AVAILABLE
4755 ;*TO POWER DOWN AND THEN UP THE MACHINE.
4756 ;*AFTER THE MACHINE HAS DONE THIS, THE CACHE CONTROL REG
4757 ;*IS EXAMINED TO HAVE BEEN PROPERLY INITIALIZED BY POWER
4758 ;*UP. AFTER THIS ALL CACHE IS REFERENCED. THERE IS A
4759 ;*VERY HIGH PROBABILITY THAT CACHE WILL HAVE PARITY ERRORS
4760 ;*IF THE POWER UP FAILED TO SWEEP CACHE. ANY CACHE PARITY

```

```

4761
4762 ;*ERROR THEREFORE IS REPORTED AS THE POWER UP FAILING TO
4763 ;*INVALIDATE CACHE.IT SHOULD BE POINTED OUT THAT
4764 ;*THE SWEEP MECHANISM IS CHECKED IN THE PREVIOUS TEST.THIS
4765 ;*TEST VERIFIES THAT THE MECHANISM CAN BE INITIATED BY
4766 ;*THE POWER UP SEQUENCE.
4767 ;*
4768 ;*NOTE:IF MACHINE HAS VOLATILE MEMORY, THE SWITCH
4769 ;* SETTINGS WILL HAVE TO BE RESTORED AFTER THIS TEST
4770
4771 ;*****
4772 031132 012737 000214 177746 TST42: MOV #214,$CCR ;CACHE OFF FOR SCOPE
4773 031140 000004 SCOPE
4774 031142 012737 031524 001234 MOV #TST43,$KTST ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4775 031150 032777 000200 147756 BIT #SW07,$SWR ;RUN THIS TEST?
4776 031156 001562 BEQ TST43 ;BRANCH TO NEXT TEST IF NO
4777 031160 012737 031412 000114 MOV #T20L01,$$PVEC ;SET UP FOR PARITY ERRORS
4778 031166 012737 031226 000024 MOV #T20L02,$$PWRVEC ;SET UP FOR POWER DOWN
4779 031174 012737 000314 177746 MOV #314,$CCR ;SET ALL BITS IN CCR
4780 031202 105737 001256 TSTB #0$ENV ;RUNNING UNDER APT?
4781 031206 001404 BEQ 18 ;BRANCH IF NO
4782 031210 012777 000020 147776 MOV #20,$CREG2 ;SET UP UBE TO POWER FAIL
4783 031216 000777 BR . ;WAIT FOR POWER FAIL
4784
4785 031220 104401 040625 18: TYPE ,MSG2 ;POWER DOWN AND THEN UP
4786 031224 000777 BR . ;WAIT FOR POWER DOWN
4787
4788 031226 012737 031246 000024 T20L02: MOV #T20L03,$$PWRVEC ;SET UP FOR POWER UP
4789 031234 022626 CMP (SP)+,(SP)+ ;RESTORE STACK
4790 031236 017737 147672 001172 MOV #SWR,$TMP0 ;SAVE (SWR)
4791 031244 000777 BR . ;WAIT FOR POWER UP
4792
4793 031246 012706 001100 T20L03: MOV $STACK,SP ;RESTORE STACK
4794 031252 105737 001256 TSTB #0$ENV ;RUNNING UNDER APT?
4795 031256 001402 BEQ 18 ;BRANCH IF NO
4796 031260 005077 147730 CLR $CREG2 ;STOP UBE POWER FAIL
4797
4798 031264 013700 001172 18: MOV $TMP0,R0 ;GET (SWR)
4799 031270 076600 MED ;RESTORE SWR
4800 031272 000226 .WORD WSW
4801 031274 012701 177000 38: MOV #177000,R1 ;INIT DELAY
4802 031300 012700 177400 28: MOV #177400,R0 ;INIT DELAY COUNTER FOR TTY
4803 031304 063737 060000 ADD #0$BUFL,$$BUFL ;DELAY
4804 031312 005200 INC R0
4805 031314 001373 BNE 28 ;WAIT FOR TTY
4806 031316 005201 INC R1
4807 031320 001367 BNE 38 ;CONTINUE DELAY
4808 031322 013737 177746 001160 MOV #0CCR,$REG1 ;SEE IF CCR INITIALIZED
4809 031330 001491 BEQ T20L04 ;BRANCH IF CCR CLEARED
4810 031332 104101 ERROR 101 ;ERROR: CACHE CONTROL REG NOT INIT BY POWER FAIL
4811
4812 031334 T20L04:
4813
4814 031334 010046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
4815 031336 076600 MED ;GET CONTENTS OF LOG REG
4816 031340 000022 .WORD RLOG

```

```

4817 031342 052700 100001      BIS      #100001,R0      ;ENABLE ERROR LOG & LOG FIRST MODE
4818 031346 076600      MED      ;UNLOCK ERROR LOG
4819 031350 000222      .WORD    WLOG
4820 031352 012600      MOV      (SP)+,R0      ;RESTORE R0
4821
4822 031354 012737 000200 177746  MOV      #200,#CCR      ;JAMUPP ON PARITY ERRORS
4823 031362 012701 002000      MOV      #200,R1      ;INIT LOOP COUNT
4824 031366 005000      CLR      R0      ;INIT ADDRESS
4825 031370 005720      TST      (R0)+      ;REFERENCE ALL CACHE LOC
4826 031372 077102      SOB      R1,10      ;LOOP TILL DONE
4827 031374 012737 033142 000114 T20L06: MOV      #UPERR,#PVEC      ;RESTORE PARITY ERROR HANDLER
4828 031402 012737 040364 000024  MOV      #PMRDN,#PMRVEC      ;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      R0,-(SP)      ;SAVE R0 FOR MED INST
4834 031422 076600      MED      ;GET CONTENTS OF LOG REG
4835 031424 000022      .WORD    RLOG
4836 031426 052700 100001      BIS      #100001,R0      ;ENABLE ERROR LOG & LOG FIRST MODE
4837 031432 076600      MED      ;UNLOCK ERROR LOG
4838 031434 000222      .WORD    WLOG
4839 031436 012600      MOV      (SP)+,R0      ;RESTORE R0
4840
4841 031440 011637 001164      MOV      (SP),REG3      ;GET PC+2 OF ERROR
4842 031444 162737 000002 001164  SUB      #2,REG3      ;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,R0      ;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      R0      ;PUT PHY. ADDR A17, A16 IN LOW BYTE
4851 031474 042700 177776      BIC      #177776,R0      ;ONLY LOOK AT A17, A16
4852 031500 010037 001160      MOV      R0,REG1      ;SAVE ADDRESS
4853 031504 076600      MED      ;GET LOG INFORMATION
4854 031506 000102      .WORD    LOADD
4855 031510 010037 001162      MOV      R0,REG2      ;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 ;THE FOLLOWING TESTS ARE RUN ONLY IF SW08=1 AT THE
4867 ;BEGINNING OF THE PROGRAM. AND THE NPR DEVICE WAS SELECTED.
4868 ;THESE TESTS USE DATA SUPPLIED BY THE USER WHEN THE PROGRAM
4869 ;IS STARTED TO SETUP VARIOUS CONTROL REGISTERS TO RUN THE
4870 ;NPR DEVICE. CREG1 ALWAYS CONTAINS THE DEVICES GO ADDRESS,
4871 ;EAD CONTAINS THE DEVICES ERROR REG ADDRESS, IVEC CONTAINS
4872 ;THE DEVICE'S INTERRUPT VECTOR (IF USED), SETUP CONTAINS THE

```

```

4873      ;ADDRESS OF THE DEVICE'S HANDLER AND THE REMAINING CREG2-5
4874      ;CONTAIN VARIOUS CONTROL INFORMATION NEEDED FOR THE PARTICULAR
4875      ;DEVICE USED. THE SETUP HANDLERS ARE USED TO INITIALIZE THE
4876      ;DEVICE TO DO NPR DATOS TO THE STARTING ADDRESS FOLLOWING
4877      ;THE SUBROUTINE CALL.
4878
4879 ;*****
4880
4881
4882 ;*****
4883 ;*TEST 43      TEST NPR DATO INVALIDATES CACHE FOR PHYSICAL ADDRESS BITS A1-A10
4884 ;*
4885 ;* THIS TEST IS ONLY RUN IF SW08=1. THE PREVIOUSLY SEL-
4886 ;* ECTED DEVICE IS SETUP TO DO NPR DATO'S TO ADDRESSES IN
4887 ;* A 1K BUFFER AREA. ON THE FIRST PASS A '1' IS FLOATED
4888 ;* THROUGH THE ADDRESS LINES A1-A10. AT EACH BIT POSITION,
4889 ;* THE LOC IS PUT IN CACHE AND THEN A NPR DATO IS DONE TO
4890 ;* THAT LOC. A MINIMUM TIME IS THEN WAITED TO ALLOW THE
4891 ;* SLOWEST DEVICE SELECTABLE TO FINISH ITS TRANSFERS. THE
4892 ;* LOC IS THEN CHECKED TO BE A MISS.
4893 ;* FOR THE SECOND PASS, A '0' IS FLOATED THROUGH ADDRESS
4894 ;* BITS A1-A10 AND THE SAME PROCEDURE IS REPEATED. BEFORE
4895 ;* THE DEVICE'S GO BIT IS SET, THE TRANSFER ADDRESS IS CHECKED
4896 ;* TO SEE IF IT OVERLAPS THE INSTRUCTION ADDRESS IN CACHE.
4897 ;* IF IT DOES, THE INSTRUCTIONS ARE EXECUTED OUT OF A NON
4898 ;* OVERLAPPING ADDRESS SPACE.
4899 ;* R0 CONTAINS THE DEVICES GO ADDRESS
4900 ;* R1 CONTAINS THE PASS INDICATOR
4901 ;* R2 IS THE DELAY COUNTER
4902 ;* R3 CONTAINS THE TRANSFER ADDRESS
4903 ;* R4 USED TO CALCULATE NEXT TRANSFER ADDRESS
4904
4905 ;*****
4906 031524 012737 000214 177746  TST43: MOV      #214,#CCR      ;CACHE OFF FOR SCOPE
4907 031532 000004      SCOPE
4908 031534 012737 032240 001234  MOV      #TST44,SKTST      ;SAVE POINTER TO NEXT TEST IF UNEXPECTED PARITY ERROR
4909 031542 032777 000400 147364  BIT      #SW08,SWP      ;NPR DEVICE AVAILABLE?
4910 031550 001002      BNE      10      ;BRANCH IF YES
4911 031552 000137 032240      JMP      #TST44      ;NO GO TO NEXT TEST
4912 031556 012737 000200 177746 1s: MOV      #200,#CCR      ;CACHE ON
4913 031564 004737 034000      JSR      PC,VEC      ;SEE IF UBE NEW USED AND SETUP INTERRUPT VECTOR
4914 031570 013700 001212      MOV      CREG1,R0      ;GET DEVICE'S GO ADDRESS
4915 031574 005001      CLR      R1      ;CLEAR FLAG FOR PASS 1 (FLOAT 1 PATTERN)
4916 031576 012704 000002      MOV      #2,R4      ;INIT REG FOR ADDR. CALC.
4917 031602 012737 060002 031620  MOV      #0UFL+2,#ADD1L      ;INIT ADDRESS LOWER FOR TEST
4918 031610 005037 031622      CLR      #ADD1H      ;INIT ADDRESS HIGHER FOR TEST
4919 031614 004777 147412      T21L09: JSR      PC,#SETUP      ;SETUP DEVICE TO DO NPR DATO TO FOLLOWING ADDRESS
4920 031620 000000      ADD1L: .WORD    0      ;TEST ADDRESS LOWER 16 BITS
4921 031622 000000      ADD1H: .WORD    0      ;TEST ADDRESS UPPER 2 BITS
4922 031624 005002      CLR      R2      ;INIT R2 FOR TIME DELAY COUNT
4923
4924 ;FIND OUT IF THE TEST INSTRUCTION ADDRESS IN CACHE
4925 ;OVERLAP THE XFER ADDRESS IN CACHE. IF THEY DO, USE THE
4926 ;TEST INSTRUCTIONS AT NON OVERLAPPING ADDRESS. THIS IS TO
4927 ;ENSURE THAT A MISS IS DUE TO A INVALIDATE RATHER THAN
4928 ;THE TEST INSTRUCTION SWAPPING OUT OF CACHE THE XFER LOCATION.

```

```
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 BTL 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 MOV (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 10: INC R2 ;DELAY TILL THE SLOWEST DEVICE
4945 031726 001376 BNE 10 ;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 T21L00: 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 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 ASL 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 IVEC,STMP0 ;GET UBE 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 ;TEST INST TO TEST XFER ADDRESSES WHICH OVERLAP TEST CODE
4983
4984 032112 013703 031620 T21L03: MOV ADDIL,R3 ;GET XFER ADDRESS
```

```
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 10: INC R2 ;DELAY TILL THE SLOWEST DEVICE
4990 032134 001376 BNE 10 ;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,##LPERR ;SET UP RETURN FOR ERROR LOOP
4997 032160 013737 031622 001160 ADDIL,#REG1 ;SAVE 'BAD' ADDRESS
4998 032166 013737 031620 001162 MOV ADDIL,#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,##LPERR ;SET UP RETURN FOR ERROR LOOP
5003 032206 013737 031622 001160 ADDIL,#REG1 ;SAVE BAD ADDRESS
5004 032214 013737 031620 001162 MOV ADDIL,#REG2 ;SAVE BAD ADDRESS
5005 032222 005777 147002 TST #EAD ;DID DEVICE HAVE ERROR?
5006 032226 100002 BPL 10 ;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 10: 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 ;*TEST 44 TEST NPR DATO INVALIDATES CACHE FOR PHYSICAL ADDRESS BITS A17-A11
5014 ;*
5015 ;* THIS TEST IS RUN ONLY IF SW08=1 AND THE INHIBIT TESTS
5016 ;*USING KT (SW12)=0. THE PREVIOUSLY SELECTED DEVICE IS
5017 ;*SETUP TO DO NPR DATOS TO LOCATIONS LYING OUTSIDE THE
5018 ;*PROGRAM AND MONITOR ADDRESS SPACE. (THE MONITOR, IF IT
5019 ;*EXISTS, LIES IN THE LAST 1.5K OF MEMORY).
5020 ;* MEMORY IS FIRST SIZED TO DETERMINE THE MAXIMUM TESTABLE
5021 ;*ADDRESS.A VIRTUAL ADDRESS IS GENERATED AND STORED IN KIPAR4.
5022 ;*THEN ITS PHYSICAL ADDRESS IS CALCULATED FOR THE DEVICES
5023 ;*NPR TRANSFER. THE ADDRESS IS MADE A HIT IN CACHE AND THEN
5024 ;*AN NPR DATO IS DONE TO IT. A MINIMUM TIME IS THEN WAITED
5025 ;*TO ALLOW THE SLOWEST SELECTABLE DEVICE TO FINISH ITS
5026 ;*TRANSFERS. THE LOCATION IS THEN CHECKED TO BE A MISS
5027 ;*(INVALIDATED). A NEW TAG VALUE IS THEN GENERATED AND
5028 ;*THE PROCEDURE REPEATS TO THE MAXIMUM ALLOWABLE ADDRESS.
5029 ;* BEFORE THE DEVICE'S GO BIT IS SET, THE ADDRESS IS
5030 ;*CHECKED TO SEE IF IT OVERLAPS THE INSTRUCTION ADDRESSES
5031 ;*IN CACHE. IF IT DOES, THE INSTRUCTIONS ARE EXECUTED
5032 ;*OUT OF NON OVERLAPPING ADDRESSES.
5033 ;*
5034 ;* R0 CONTAINS THE DEVICE'S GO ADDRESS
5035 ;* R2 IS THE DELAY COUNTER
5036 ;* R3 IS USED TO GENERATE THE TEST ADDRESS
5037
5038 ;*****
5039 032240 012737 000214 177746 TST44: MOV #214,#CCR ;CACHE OFF FOR SCOPE
5040 032246 000004 SCOPE
```

```

5041 032250 012737 033020 001234      MOV    #EOP,SKTST
5042 032256 032777 000400 146650      BIT    $SW08,0SWR      ;NPR DEVICE AVAILABLE?
5043 032264 001002                      BNE    18              ;BRANCH IF YES
5044 032266 000137 033020                JMP    0##EOP
5045 032272 032777 010000 146634 18:    BIT    $SW12,0SWR      ;INHIBIT TESTS USING KT?
5046 032300 001402                      BEQ    28              ;BRANCH IF NO
5047 032302 000137 033020                JMP    0##EOP
5048 032306 012737 000200 177746 28:    MOV    $200,0#CCR      ;CACHE ON
5049 032314 052737 000200 036034          BIS    $200,0#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,R0        ;GET THE GO ADDRESS OF THE DEVICE
5053 032340 005237 177572                INC    0#MMR0         ;TURN KT ON
5054
5055
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    R3,0#KIPAR4    ;SETUP PAR WITH TEST ADDR
5060 032356 023727 172350 001000          CMP    0#KIPAR4,$1000 ;PAST INSTRUCTION SPACE?
5061 032364 002003                      BGE    18              ;BRANCH IF YES
5062 032366 052737 000600 172350          BIS    $600,0#KIPAR4 ;MAKE SURE TEST ADDR. LIES OUTSIDE OF TEST CODE
5063 032374 023737 172350 036322 18:    CMP    0#KIPAR4,$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
5068
5069 032410 012737 101776 001172          MOV    $101776,0TMP0   ;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
5079
5080
5081
5082
5083
5084 032450 013737 032442 001172          MOV    ADD2L,0TMP0     ;GET XFER ADDRESS
5085 032456 042737 174000 001172          BIC    $174000,0TMP0  ;CALC ITS CACHE ADDRESS
5086 032464 012737 032534 001174          MOV    $T22L01,0TMP1 ;GET TEST INST ADDRESS
5087 032472 042737 174000 001174          BIC    $174000,0TMP1  ;CALC ITS CACHE ADDRESS
5088 032500 023737 001172 001174          CMP    0TMP0,0TMP1    ;DOES XFER ADDRESS OVERLAP TEST INST?
5089 032506 002407                      BLT    T22L03         ;BRANCH IF NO
5090 032510 062737 000024 001174          ADD    $24,0TMP1      ;CALC ADDR OF LAST OVERLAPPING TEST INST.
5091 032516 023737 001172 001174          CMP    0TMP0,0TMP1    ;DOES XFER ADDR STILL OVERLAP TEST INST?
5092 032524 003440                      BLE    T22L04         ;BRANCH IF YES
5093
5094 032526 023737 101776 101776 T22L03: CMP    0#101776,0#101776 ;MAKE ADDR A HIT
5095 032534 033727 177572 000004 T22L01: BIT    0#HMR,0HMR2 ;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                18:    INC    R2             ;DELAY TILL THE SLOWEST DEVICE
5099 032550 001376                      BNE    18              ;HAS FINISHED ITS XFERS
5100 032552 005737 101776          TST    0#101776      ;SEE IF NPR DATO HAS INVALIDATED THE XFER ADDR IN CACHE
5101 032556 033727 177572 000004          BIT    0#HMR,0HMR2   ;LOC NOW A MISS? (CACHE INVALIDATED?)
5102 032564 001054                      ENE    T22L06         ;GO REPORT ERROR IF LOC A HIT
5103 032566 005777 146436          T22L10: 0EAD        ;SEE IF DEVICE HAS AN ERROR
5104 032572 100451                      RMI    T22L06         ;REPORT DEVICE ERROR IF YES
5105 032574 023727 172350 004000          CMP    0#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    0#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    0#101776,0#101776 ;MAKE XFER ADDRESS A HIT
5114 032634 033727 177572 000004          BIT    0#HMR,0HMR2   ;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                18:    INC    R2             ;DELAY TILL THE SLOWEST DEVICE
5118 032650 001376                      BNE    18              ;HAS FINISHED
5119 032652 005737 101776          TST    0#101776      ;SEE IF NPR DATO HAS INVALID THE XFER ADDR. IN CACHE
5120 032656 033727 177572 000004          BIT    0#HMR,0HMR2   ;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    $T22L11,0#0LPERR ;INIT RETURN FOR ERROR LOOP
5125 032676 013737 032444 001160          MOV    ADD2H,$REG1    ;SAVE BAD ADDRESS
5126 032704 013737 032442 001162          MOV    ADD2L,$REG2    ;SAVE BAD ADDRESS
5127 032712 104043                      EPROR  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    $T22L11,0#0LPERR ;SETUP RETURN FOR ERROR LOOPS
5131 032724 013737 032444 001160          MOV    ADD2H,$REG1    ;SAVE BAD ADDRESS
5132 032732 013737 032442 001162          MOV    ADD2L,$REG2    ;SAVE BAD ADDRESS
5133 032740 005777 146264          TST    0EAD          ;DID DEVICE HAVE AN ERROR?
5134 032744 100002                      BPL    18              ;BRANCH IF NO
5135 032746 104103                      RPROR  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                18:    ERROR  104         ;ERROR: CACHE LOC NOT INVALID BY NPR DATO TO ADDR.
5139 032754 042737 000001 177572 T22L02: BIC    $1,0#MMR0     ;KT OFF
5140 032762 023727 001232 034046          CMP    SETUP,$HUBEN   ;WAS THE NEW UBE USED?
5141 032770 001013                      RNE    $EOP          ;IF NO, GO TO END-OF-PASS
5142 032772 013737 001226 001172          MOV    IVEC,0TMP0     ;GET UBE INTERRUPT VECTOR
5143 033000 062737 000002 001172          ADD    $2,0TMP0       ;AND RESTORE
5144 033006 013777 001172 146212          MOV    0TMP0,0IVEC    ;THE TRAP CATCHER
5145 033014 005077 146152                      CLR    0TMP0
5146
5147
5148
5149
5150
5151
5152

```

.SBTTL END OF PASS ROUTINE

;;*****
;INCREMENT THE PASS NUMBER (\$PASS)

```

5153 ;*TYPE "END PASS #XXXXX" (WHERE XXXX IS A DECIMAL NUMBER)
5154 ;*IF THERE'S A MONITOR GO TO IT
5155 ;*IF THERE ISN'T JUMP TO START1
5156
5157 #33020 #00004 SCOPE
5158 #33020 #00004 CLR $TSTNM ;ZERO THE TEST NUMBER
5159 #33022 #05037 001102 CLR $TIMES ;ZERO THE NUMBER OF ITERATIONS
5160 #33026 #05037 035406 INC $PASS ;INCREMENT THE PASS NUMBER
5161 #33032 #05237 001244 BIC #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
5162 #33036 #42737 100000 001244 DEC (PC)+ ;LOOP?
5163 #33044 #05327
5164 #33046 #00001 $EOPCT: .WORD 1
5165 #33050 #03022 BGT $DOAGN ;YES
5166 #33052 #12737 MOV (PC)+,$(PC)+ ;RESTORE COUNTER
5167 #33054 #00001 $ENDCT: .WORD 1
5168 #33056 #33046 $EOPCT
5169 #33060 #14401 033125 TYPE , $ENDMG ;TYPE "END PASS #"
5170 #33064 #13746 #01244 MOV $PASS,-(SP) ;SAVE $PASS FOR TYPEOUT
5171 #33070 #14405 TYPDS ;GO TYPE--DECIMAL ASCII WITH SIGN
5172 #33072 #14401 033122 TYPE , $ENULL ;TYPE A NULL CHARACTER
5173 #33076 #13700 000042 $GET42: MOV #42,R0 ;GET MONITOR ADDRESS
5174 #33102 #01405 BEQ $DOAGN ;BRANCH IF NO MONITOR
5175 #33104 #00005 RESET ;CLEAR THE WORLD
5176 #33106 #04710 $ENDAD: JSR PC,(R0) ;GO TO MONITOR
5177 #33110 #00240 NOP ;SAVE ROOM
5178 #33112 #00240 NOP ;FOR
5179 #33114 #00240 NOP ;ACT11
5180 #33116
5181 #33116 #00137 $DOAGN: JMP 0(PC)+ ;RETURN
5182 #33120 #03056 $RTNAD: .WORD START1
5183 #33122 377 377 000 $ENULL: .BYTE -1,-1,0 ;NULL CHARACTER STRING
5184 #33125 #15 042412 042116 $ENDMG: .ASCIZ <15><12>/END PASS #/
5185 #33132 #05040 #51501 020123
5186 #33143 #00043
5187
5188 ;////////////////////////
5189 ;SUBROUTINE TO REPORT AN UNEXPECTED PARITY ERRORS
5190 ;////////////////////////
5191
5192 #33142 #12737 000214 177746 UPERR: MOV #214,$CCR ;TURN OFF CACHE TO PREVENT OTHER ERRORS
5193 #33150 #11637 001166 MOV (SP), $REG4 ;SAVE PC+2 WHERE PARITY ERROR OCCURRED
5194 #33154 162737 001166 SUB #2,$REG4 ;CALC. PC WHERE PARITY ERROR OCCURRED
5195 #33162 #22626 CMP (SP)+,(SP)+ ;RESTORE STACK
5196 #33164 #76600 MED ;GET LOG INFOR FOR PHY. ADDR. A17,A16
5197 #33166 #00101 .WORD RSER
5198 #33170 #00300 SWAB R0 ;PUT PHY. ADDR A17, A16 IN LOW BYTE
5199 #33172 #42700 177776 RIC #177776,R0 ;ONLY LOOK AT A17, A16
5200 #33176 #10037 001160 MOV R0,$REG1 ;SAVE ADDRESS
5201 #33202 #76600 MED ;GET LOG INFORMATION
5202 #33204 #00102 .WORD LOADD
5203 #33206 #10037 001162 MOV R0,$REG2 ;SAVE INFORMATION
5204 #33212 #76600 MED ;GET LOG INFORMATION
5205 #33214 #00100 .WORD RJAM
5206 #33216 #32700 000400 BIT #400,R0 ;WAS ERROR IN BACKING STORE?
5207 #33222 #01016 BNE UP1 ;BRANCH IF YES
5208 #33224 #32737 000040 177744 BIT #40,$EREG ;WAS ERROR IN CACHE TAG FIELD?

```

```

5209 #33232 #01017 BNE UP2 ;BRANCH IF YES
5210 #33234 #32737 000100 177744 BIT #100,$EREG ;WAS ERROR IN CACHE LOW BYTE?
5211 #33242 #01024 BNE UP3 ;BRANCH IF YES
5212 #33244 #76600 MED ;GET LOG INFORMATION
5213 #33246 #00106 .WORD CDH
5214 #33250 #10037 001164 MOV R0,$REG3 ;SAVE INFORMATION
5215 #33254 #14004 ERROR 4 ;ERROR: UNEXPECTED PARITY ERROR IN CACHE DATA HIGH
5216 #33256 #000423 BR UP1 ;RETURN
5217
5218 #33260 #11737 001166 001164 UP1: MOV $REG4,$REG3 ;SAVE PC OF TRAP
5219 #33266 #14001 ERROR 1 ;ERROR: UNEXPECTED PARITY ERROR IN BACKING STORE
5220 #33270 #000416 BR UP1 ;RETURN
5221
5222 #33272
5223 #33272 #76600 UP2: MED ;GET TAG LOG INFO.
5224 #33274 #00107 .WORD RTAG
5225 #33276 #00300 SWAB R0 ;PUT TAG IN LOW BYTE
5226 #33300 #42700 177400 BIC #177400,R0 ;LOOK AT TAG ONLY
5227 #33304 #10037 001164 MOV R0,$REG3 ;SAVE CACHE TAG DATA
5228 #33310 #10002 ERROR 2 ;ERROR: UNEXPECTED PARITY ERROR IN CACHE TAG
5229 #33312 #000405 BR UP1 ;RETURN
5230
5231 #33314
5232 #33314 #76600 UP3: MED ;GET LOG INFORMATION
5233 #33316 #00106 .WORD CDL
5234 #33320 #10037 001164 MOV R0,$REG3 ;SAVE INFORMATION
5235 #33324 #104003 ERROR 3 ;ERROR: UNEXPECTED PAPITY ERROR IN CACHE DATA LOW
5236
5237 #33326
5238
5239 #33326 #10046 UP4: MOV R0,-(SP) ;SAVE R0 FOR MED INST
5240 #33330 #76600 MED ;GET CONTENTS OF LOG REG
5241 #33332 #00022 .WORD RLOG
5242 #33334 #52700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
5243 #33340 #76600 MED ;UNLOCK ERROR LOG
5244 #33342 #00022 .WORD WLOG
5245 #33344 #12600 MOV (SP)+,R0 ;RESTORE R0
5246
5247 #33346 #00177 145662 JMP $SKTST ;START SUBTEST FOLLOWING ONE WHERE ERROR OCCURRED
5248
5249 ;ROUTINE TO REPORT UNEXPECTED TRAPS TO VECTOR 4
5250 ;////////////////////////
5251 #33352 #12737 000214 177746 UT4: MOV #214,$CCR ;TURN OFF CACHE
5252 #33360 #11637 001162 MOV (SP), $REG2 ;SAVE FAILING PC
5253 #33364 #13737 177766 001160 MOV #0CER,$REG1 ;GET CPU ERROR REG (CER)
5254 #33372 #32777 001000 145534 BIT $SW09,$SWP ;LOOP ON ERROR?
5255 #33400 #01401 BEQ 15 ;BRANCH IF NO
5256 #33402 #22626 CMP (SP)+,(SP)+ ;RESTORE STACK
5257 #33404 #104016 18: ERROR 16 ;ERROR: UNEXPECTED TRAP TO VECTOR 4
5258
5259 #33406 #10046 MOV R0,-(SP) ;SAVE R0 FOR MED INST
5260 #33410 #76600 MED ;GET CONTENTS OF LOG REG
5261 #33412 #00022 .WORD RLOG
5262 #33414 #52700 100001 BIS #100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
5263 #33420 #76600 MED ;UNLOCK ERROR LOG
5264 #33422 #00022 .WORD WLOG

```

```

5265 033424 012600      MOV      (SP)+,R0      ;RESTORE R0
5266
5267 033426 000000      HALT
5268 033430 000137 000200      JMP      01200      ;RESTART TEST IF CONTINUE
5269
5270 ;SUBROUTINE TO CONVERT VIRTUAL ADDRESS IN $TMP0 TO A PHYSICAL ADDRESS IN $REG2, $REG1
5271 ;
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      $TMP0,R1      ;GET VIRTUAL ADDRESS
5276 033444 005002      CLR      R2      ;INT SHIFT COUNTER
5277 033446 006201      18:  ASR      R1      ;SHIFT BLOCK # TO LSB 0-6
5278 033450 005202      INC      R2      ;COUNT SHIFTS
5279 033452 020227 000006      CMP      R2,#6      ;ALL DONE?
5280 033456 001373      BNE      18      ;BRANCH IF NO
5281 033460 010137 001162      MOV      R1,$REG2      ;SAVE BLOCK #
5282 033464 042737 177600 001162      BIC      0177600,$REG2      ;MASK BLOCK #
5283 033472 006201      28:  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      28      ;BRANCH IF NO
5287 033504 042701 177761      BIC      0177761,R1      ;CALC. APFX2
5288 033510 062701 172340      ADD      $KIPAR0,R1      ;CALC ADDRESS OF PAR REFERENCING
5289 033514 011141      MOV      (R1),R1      ;GET (PAR)
5290 033516 060137 001162      ADD      R1,$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      38:  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      38      ;BRANCH IF NO
5297 033546 005002      CLR      R2      ;INIT. SHIFT COUNT
5298 033550 006337 001162      48:  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      48      ;BRANCH IF NO
5302 033564 013701 001172      MOV      $TMP0,R1      ;GET VIRTUAL ADDRESS
5303 033570 042701 177700      BIC      0177700,R1      ;MASK OFF BLOCK COUNT
5304 033574 060137 001162      ADD      R1,$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 $TMP0
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      18:  ASR      $TMP0      ;CALC TAG CONTENTS
5316 033620 077103      SOB      R1,18      ;ALL DONE?
5317 033622 052737 000200 001172      BIS      $200,$TMP0      ;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 $TMP0
5324
5325 033634 010037 001172      PAR:  MOV      R0,$TMP0      ;GET VIRTUAL ADDRESS
5326 033640 005776 000000      TST      0(SP)      ;WAS R0 USED?
5327 033644 001402      BEQ      18      ;BRANCH IF YES
5328 033646 010137 001172      MOV      R1,$TMP0      ;GET VIRTUAL ADDRESS
5329 033652 062716 000002      18:  ADD      $2,(SP)      ;ADJUST PC
5330 033656 012705 000014      MOV      $14,R5      ;INIT COUNT
5331 033662 006237 001172      28:  ASR      $TMP0      ;SHIFT ADDRESS TO GET ACTIVE PAGE FIELD
5332 033666 077503      SOB      R5,28      ;APF IN LSB 1-3? BRANCH IF NO
5333 033670 042737 177761 001172      BIC      0177761,$TMP0      ;MASK APF X 2
5334 033676 062737 172340 001172      ADD      $KIPAR0,$TMP0      ;PUT PAR ADDRESS IN $TMP0
5335 033704 017737 145262 001172      MOV      0$TMP0,$TMP0      ;GET CONTENTS OF PAR
5336 033712 000207      RTS      PC      ;RETURN
5337
5338 ;SUBROUTINE TO GENERATE A TEST BUFFER ADDRESS 512(10) LOCATIONS FROM GIVEN
5339 ;ADDRESS FOLLOWING ITS CALL
5340
5341 033714 017637 000000 001172      HAD:  MOV      0(SP),$TMP0      ;GET ADDRESS TO BE USED
5342 033722 062716 000002      ADD      $2,(SP)      ;ADJUST PC
5343 033726 062737 002000 001172      ADD      $2000,$TMP0      ;CALC. ADDR WITH ADDRESS BIT A10 COMPLEMENTED
5344 033734 042737 174000 001172      RIC      0174000,$TMP0      ;MASK A15-A11
5345 033742 032737 002000 001172      BIT      $2000,$TMP0      ;BIT 10 SET?
5346 033750 001004      BNE      18      ;BRANCH IF YES
5347 033752 062737 060000 001172      ADD      $BUFL,$TMP0      ;CALC TEST BUFFER ADDR.
5348 033760 000406      BR      28
5349
5350 033762 042737 002000 001172      18:  BIC      $2000,$TMP0      ;ADJUST ADDRESS BIT A10
5351 033770 062737 062000 001172      ADD      $BUFH,$TMP0      ;CALC TEST BUFFER ADR.
5352 033776 000207      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 034000 023727 001232 034046      VEC:  CMP      $SETUP,$HUBEN      ;NEW USE USED?
5359 034006 001016      BNE      18      ;BRANCH IF NO
5360 034010 013737 001226 001172      MOV      IVEC,$TMP0      ;GET ITS INTERRUPT VECTOR
5361 034016 062737 000002 001172      ADD      $2,$TMP0
5362 034024 013777 001172 145174      MOV      $TMP0,$IVEC      ;PUT ON RTI
5363 034032 012777 000002 145132      MOV      $RTI,$TMP0      ;IN ITS INTERRUPT AREA
5364 034040 005037 177776      CLR      $PSW      ;LOWER PRIORITY LEVEL FOR INTERRUPTS
5365 034044 000207      18:  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 034046 005037 001204      HUBEN:  CLR      $TMP5      ;INIT COUNTER TO WAIT FOR RDY BIT
5373 034052 105777 145134      28:  TSTB      $CREG1      ;READY BIT SET?
5374 034056 100421      BMI      18      ;BRANCH IF YES
5375 034060 005237 001204      INC      $TMP5      ;WAIT FOR RDY TO SET
5376 034064 001172      BNE      28      ;BRANCH IF HAVEN'T WAITED MAX TIME

```

```

5377 034066 032777 020000 145040      BIT      #SW13,0SWR      ;INHIBIT TYPEOUTS?
5378 034074 001004                      BNE      30             ;BRANCH IF YES
5379 034076 104401 041645                TYPE     ,MSG13        ;DEVICE READY 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        BIC      #1,0MMR0      ;KT OFF IF ON
5383 034116 000137 033020                JMP      $EOP         ;GO TO END OF PROGRAM
5384 034122 005077 145070                CLR      0CREG3       ;CLEAR ANY ERROR BITS SET
5385 034126 012777 003040 145056        MOV      03040,0CREG1 ;HAVE UBE DO INPR DATO DATA XFER
5386 034134 005077 145054                CLR      0CREG2       ;HAVE UBE DO INPR DATO DATA XFER
5387 034140 012777 177777 145054        MOV      017777,0CREG5 ;CYCLE COUNT=1 XFER
5388 034146 017677 000000 145044        MOV      0(SP),0CREG4 ;GET ADDRESS FOR XFER
5389 034154 062716 000002                ADD      #2,(SP)      ;GET HIGH ADDRESS BITS A17, A16
5390 034160 057677 000000 145026        BIS      0(SP),0CREG2 ;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 ;////////////////////////////////////
5396 ;SUBROUTINE TO SETUP THE OLD UNIBUS EXERCISOR TO DO 1 NPR DATO
5397 ;TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL.
5398 ;////////////////////////////////////
5399
5400 HUBFO: MOV      #50200,00170006        ;HAVE UBE DO 1 NPR DATO AND RELEASE BUS
5401        MOV      #2,00170004            ;SET BYTE COUNT FOR 1 WORD XFER
5402        MOV      0(SP),00170002        ;SET UP XFER ADDRESS
5403        ADD      #4,(SP)                ;ADJUST PC FOR RETURN
5404        RTS      PC                     ;RETURN
5405 FAKE:   _WORD 0                        ;FAKE ERROR REG. MSB=0 FOR NO ERRORS
5406
5407 ;////////////////////////////////////
5408 ;SUBROUTINE TO SETUP AN RK05 FOR 1 NPR DATO
5409 ;TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL
5410 ;////////////////////////////////////
5411
5411 HRK05: CLR      0TMP5                  ;INIT COUNTER TO WAIT FOR RDY BIT
5412        TSTB     0RKCS                 ;IS CONTROLLER RDY?
5413        BMI      10                     ;BRANCH IF YES
5414        INC      0TMP5                 ;WAIT FOR RDY TO SET
5415        BNE      20                     ;BRANCH IF HAVEN'T WAITED MAX TIME
5416        BIC      #1,0MMR0             ;KT OFF IF ON
5417        BIT      #SW13,0SWR           ;INHIBIT TYPEOUTS?
5418        BNE      00                     ;BRANCH IF YES
5419        TYPE     ,MSG13                ;DEVICE RDY BIT DOES NOT SET
5420        TYPE     ,MSG14                ;FURTHER NPR TESTS ABORTED
5421        TST      (SP)+                 ;RESTORE STACK FROM SUBROUTINE CALL
5422        JMP      $EOP                 ;GO TO END OF PROGRAM
5423
5424        CLR      0TMP5                  ;INIT COUNTER TO WAIT FOR RDY BIT
5425        BIT      #100,00RKDS           ;IS DRIVE RDY?
5426        BNE      30                     ;BRANCH IF YES
5427        INC      0TMP5                 ;WAIT FOR RDY TO SET
5428        BNE      40                     ;BRANCH IF HAVEN'T WAITED MAX TIME
5429        BR       50                     ;REPORT DEVICE NOT READY
5430
5431        MOV      #1,00RKCS             ;RESET CONTROLLER
5432        CLR      0TMP5                  ;INIT COUNTER TO WAIT FOR RDY BIT

```

```

5433 034340 105737 177404                TSTB     00RKCS        ;CONTROLLER RDY?
5434 034344 100404                      BMI      60             ;BRANCH IF YES
5435 034346 005237 001204                INC      0TMP5        ;WAIT FOR RDY TO SET
5436 034352 001372                      BNE      70             ;BRANCH IF HAVEN'T WAITED MAX TIME
5437 034354 000734                      BR       50             ;REPORT DEVICE NOT RDY
5438
5439 034356 012737 177777 177406        MOV      #1,00RKWC        ;SET WORD COUNT FOR 1 XFER
5440 034364 013737 001214 177412        MOV      00CREG2,00RKDA ;SET UP DISK ADDRESS REG
5441 034372 012737 000004 177404        MOV      #4,00RKCS        ;SET UP DISK TO DO DATO
5442 034400 017637 000000 177410        MOV      0(SP),00RKBA    ;SET UP XFER ADDRESS
5443 034406 062716 000002                ADD      #2,(SP)        ;LOOK AT HIGH ADDRESS BITS
5444 034412 017637 000000 001204        MOV      0(SP),0TMP5     ;GET HIGH ADDRESS BITS
5445 034420 062716 000002                ADD      #2,(SP)        ;ADJUST PC FOR RETURN
5446 034424 005037 001202                CLR      0TMP4          ;INIT COUNT FOR SHIFT
5447 034430 006337 001204                ASL      0TMP5         ;SHIFT ADDRESS BITS TO RKCS ADDR, BIT'S POSITION
5448 034434 005237 001202                INC      0TMP4          ;COUNT SHIFTS
5449 034440 023727 001202 000004        CMP      0TMP4,#4       ;ALL DONE?
5450 034446 001370                      BNE      80             ;BRANCH IF NO
5451 034450 057337 001204 177404        BIS      0TMP5,00RKCS   ;SET UP THE EXTENDED MEMORY BITS
5452 034456 000207                      RTS      PC             ;RETURN
5453
5454 ;////////////////////////////////////
5455 ;SUBROUTINE TO SETUP AN RP03 TO DO 1 NPR DATO
5456 ;TO THE ADDRESS FOLLOWING THE SUBROUTINE CALL
5457 ;////////////////////////////////////
5458
5459 HRP03: TST      00RPCS                 ;ANY ERRORS?
5460        BEQ      10                     ;BRANCH IF NO
5461        BIC      #1,0MMR0             ;KT OFF IF ON
5462        BIT      #SW13,0SWR           ;INHIBIT TYPEOUTS?
5463        BNE      20                     ;BRANCH IF YES
5464        TYPE     ,MSG12                ;DEVICE ERROR BIT SET
5465        TYPE     ,MSG14                ;FURTHER NPR DEVICE TEST ABORTED
5466        TST      (SP)+                 ;RESTORE STACK FROM SUBROUTINE CALL
5467        JMP      $EOP                 ;GO TO END OF PROG
5468
5469        CLR      0TMP5                  ;INIT COUNTER TO WAIT FOR RDY BIT
5470        TSTB     00RPCS                 ;CONTROLLER RDY?
5471        BMI      30                     ;BRANCH IF YES
5472        INC      0TMP5                 ;WAIT FOR RDY TO SET
5473        BNE      40                     ;BRANCH IF HAVEN'T WAITED MAX TIME
5474        BIC      #1,0MMR0             ;KT OFF IF ON
5475        BIT      #SW13,0SWR           ;INHIBIT TYPEOUTS?
5476        BNE      50                     ;BRANCH IF YES
5477        TYPE     ,MSG13                ;DEVICE RDY BIT DID NOT SET
5478        TYPE     ,MSG14                ;FURTHER NPR DEVICE TEST ABORTED
5479        TST      (SP)+                 ;RESTORE STACK FROM SUBROUTINE CALL
5480        JMP      $EOP                 ;GO TO END OF PROG
5481
5482        CLR      0TMP5                  ;INIT COUNTER TO WAIT FOR RDY BIT
5483        TST      00PPDS                 ;IS DEVICE RDY?
5484        BMI      60                     ;BRANCH IF YES
5485        INC      0TMP5                 ;WAIT FOR RDY TO SET
5486        BNE      70                     ;BRANCH IF HAVEN'T WAITED MAX TIME
5487        BR       80                     ;REPORT RDY DID NOT SET
5488

```



```
5489 034620 012737 177776 176716 68: MOV 0-2,0#RPMC ;SET UP TO DO MIN # OF XFERS(2)
5490 034626 013737 001214 176714 MOV CREG2,0#RPCS ;START TO SETUP CONTROLLER FOR NPR DATO
5491 034634 017637 000000 176720 MOV 0(SP),0#RPBA ;SETUP XFER ADDRESS
5492 034642 062716 000002 ADD 02,(SP) ;LOOK AT HIGH XFER ADDRESS
5493 034646 017637 000000 001202 MOV 0(SP),0#TMP4 ;GET HIGH XFER ADDR.
5494 034654 062716 000002 ADD 02,(SP) ;ADJUST PC FOR RETURN
5495 034660 005037 001204 CLR 0#MTC ;INIT SHIFT COUNTER
5496 034664 006337 001202 98: ASL 0#TMP4 ;SHIFT ADDR. BITS TO COINCIDE WITH RPCS EXTENDED ADDR. B
5497 034670 005237 001204 INC 0#TMP5 ;COUNT SHIFTS
5498 034674 022737 000004 001204 CMP 04,0#TMP5 ;FINISHED?
5499 034702 001370 BNE 98 ;BRANCH IF NO
5500 034704 053737 001202 176714 BIS 0#TMP4,0#RPCS ;SETUP THE EXTENDED MEM ADDR.
5501 034712 000207 RTS PC ;RETURN
5502
5503 ;////////////////////
5504 ;SUBROUTINE TO SETUP A TU10 TO DO NPR DATO XFERS
5505 ;TO THE STARTING ADDRESS FOLLOWING THE SUBROUTINE CALL
5506 ;////////////////////
5507
5508 034714 052737 010000 172522 HTU10: BIS 010000,0#MTC ;POWER CLEAR THE UNIT
5509 034722 000240 NOP ;WAIT FOR POWER CLEAR
5510 034724 000240 NOP
5511 034726 012737 177777 172524 MOV 0-1,0#MTBRC ;PREPARE TO BACKSPACE ONE RECORD
5512 034734 013737 001214 172522 MOV CREG2,0#MTC ;GET CONTROL MASK
5513 034742 052737 000012 172522 BIS 012,0#MTC ;SET UP BACKSPACE COMMAND
5514 034750 005237 172522 INC 0#MTC ;BACKSPACE
5515 034754 005037 001204 CLR 0#TMP5 ;INIT COUNTER TO WAIT FOR RDY
5516 034760 012737 000001 172520 28: BIT 01,0#MTS ;UNIT DONE?
5517 034766 001021 BNE 18 ;BRANCH IF YES
5518 034770 005237 001204 INC 0#TMP5 ;WAIT TILL UNIT DONE
5519 034774 001371 BNE 28 ;BRANCH IF HAVEN'T WAITED MAX TIME
5520 034776 005726 TST 0(SP)+ ;RESTORE STACK FROM SUBROUTINE CALL
5521 035000 042737 000001 177572 BIT 01,0#MMNR0 ;KT OFF IF ON
5522 035006 032777 020000 144120 BIT 05W13,0#SWR ;INHIBIT TYPEOUTS?
5523 035014 001004 BNE 38 ;BRANCH IF YES
5524 035016 104401 041645 TYPE 0#MSG13 ;RDY BIT DID NOT SET
5525 035022 104401 041705 TYPE 0#MSG14 ;ABORT ALL TESTS USING NPR DEVICE
5526 035026 000137 033020 38: JMP 0#EOP ;GO TO END OF PROGRAM
5527
5528 035032 013737 001214 172522 18: MOV CREG2,0#MTC ;GET CONTROL MASK
5529 035040 052737 000002 172522 BIS 02,0#MTC ;SETUP CONTROL TO DO READ
5530 035046 012737 177776 172524 MOV 0-20,0#MTBRC ;PREPARE TO READ MIN # OF BYTES (20(8))
5531 035054 017637 000000 172526 MOV 0(SP),0#MTCNA ;SETUP XFER ADDRESS
5532 035062 062716 000002 ADD 02,(SP) ;LOOK AT HIGH ADDRESS
5533 035066 017637 000000 001202 MOV 0(SP),0#TMP4 ;GET HIGH ADDRESS
5534 035074 005037 001204 CLR 0#TMP5 ;INIT SHIFT COUNTER
5535 035100 006337 001202 48: ASL 0#TMP4 ;SHIFT ADDR. BITS TO COINCIDE WITH MTC ADDR BITS
5536 035104 005237 001204 INC 0#TMP5 ;COUNT SHIFTS
5537 035110 022737 000004 001204 CMP 04,0#TMP5 ;DONE?
5538 035116 001370 BNE 48 ;BRANCH IF NO
5539 035120 053737 001202 172522 BIS 0#TMP4,0#MTC ;SETUP HIGH ADDRESS BITS
5540 035126 062716 000002 ADD 02,(SP) ;ADJUST PC FOR RETURN
5541 035132 000207 RTS PC ;RETURN
5542
5543 ;////////////////////
5544 ;SUBROUTINE TO RID CACHE OF BAD PARITY BY
;OVERWRITTING IT WHEN CACHE IS OFF
```

```
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 0#BUFL,R5 ;GET STARTING ADDRESS
648: MOV (R5),(R5)+ ;WRITE ALL CACHE WITH GOOD PARITY
CMP R5,0#BUFL+4000 ;ALL CACHE WRITTEN?
BNE 648 ;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(0#STSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;AND LOAD THE ERROR FLAG (0#ERFLG) INTO DISPLAY<15:08>
;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;SW14=1 LOOP ON TEST
;SW11=1 INHIBIT ITERATIONS
;SW09=1 LOOP ON ERROR
;CALL
;= SCOPE ;SCOPE=10T

0#SCOPE:
18: BIT 0#BIT14,0#SWR ;LOOP ON PRESENT TEST?
BNE 0#OVER ;YES IF SW14=1
;*****START OF CODE FOR THE XOR TESTER*****
0#XTSTRI: BR 68 ;IF RUNNING ON THE "XOR" TESTER CHANGE
;THIS INSTRUCTION TO A "NOP" (NOP=240)
MOV 0#ERRVEC,-(SP) ;SAVE THE CONTENTS OF THE ERROR VECTOR
MOV 0#0,0#ERRVEC ;SET FOR TIMEOUT
TST 0#177060 ;TIME OUT ON XOR?
MOV (SP)+,0#ERRVEC ;RESTORE THE ERROR VECTOR
BR 0#SVLAD ;GO TO THE NEXT TEST
58: CMP (SP)+,(SP)+ ;CLEAR THE STACK AFTER A TIME OUT
MOV (SP)+,0#ERRVEC ;RESTORE THE ERROR VECTOR
BR 78 ;LOOP ON THE PRESENT TEST
68: ;*****END OF CODE FOR THE XOR TESTER*****
28: TSTB 0#ERFLG ;HAS AN ERROR OCCURRED?
REQ 38 ;BR IF NO
CMPB 0#ERMAX,0#ERFLG ;MAX. ERRORS FOR THIS TEST OCCURRED?
BHI 38 ;BR IF NO
BIT 0#BIT09,0#SWR ;LOOP ON ERROR?
BEQ 48 ;BR IF NO
78: MOV 0#LPERR,0#LPADR ;SET LOOP ADDRESS TO LAST SCOPE
BR 0#OVER
48: CLR 0#ERFLG ;ZERO THE ERROR FLAG
CLR 0#TIMES ;CLEAR THE NUMBER OF ITERATIONS TO MAKE
BR 18 ;ESCAPE TO THE NEXT TEST
38: BIT 0#BIT11,0#SWR ;INHIBIT ITERATIONS?
BNE 18 ;BR IF YES
TST 0#PASS ;IF FIRST PASS OF PROGRAM
BEQ 18 ; INHIBIT ITERATIONS
INC 0#ICNT ;INCREMENT ITERATION COUNT
CMP 0#TIMES,0#ICNT ;CHECK THE NUMBER OF ITERATIONS MADE
RGE 0#OVER ;BR IF MORE ITERATION REQUIRED
18: MOV 0#1,0#ICNT ;REINITIALIZE THE ITERATION COUNTER
MOV 0#XCNT,0#TIMES ;SET NUMBER OF ITERATIONS TO DO
0#SVLAD: INCH 0#STSTNM ;COUNT TEST NUMBERS
```

MACY11 27(1006) 09-FEB-77 15:33 PAGE 102
ERROR HANDLER ROUTINE

5657	035572			58:	CMP	#ENDAD,0#42	;;ACT-11 AUTO-ACCEPT?
5658	035572	022737	033106	000042	BNE	68	;;BRANCH IF NO
5659	035600	001001			HALT		;;YES
5660	035602	000000					
5661	035604			68:	RTI		;;RETURN
5662	035604	000002			\$ESCAPE:	.WORD 0	;;ESCAPE ON ERROR ADDRESS
5663	035606	000000			\$BELL:	.ASCIIZ <207><377><377>	;;ASCII CODE FOR BELL
5664	035610	177607	000377		.SBTTL	ERROR MESSAGE TYPEOUT ROUTINE	
5665							
5666							
5667							
5668							
5669							
5670							
5671							
5672	035614						
5673	035614	104401	001207		TYPE	, \$CRLF	;;"CARRIAGE RETURN" & "LINE FEED"
5674	035620	010046			MOV	R0,-(SP)	;;SAVE R0
5675	035622	005000			CLR	R0	;;PICKUP THE ITEM INDEX
5676	035624	153700	001114		BISB	0#ITEMB,R0	
5677	035630	001004			BNE	18	;;IF ITEM NUMBER IS ZERO, JUST
5678							;;TYPE THE PC OF THE ERROR
5679	035632	013746	001116		MOV	\$ERRPC,-(SP)	;;SAVE \$ERRPC FOR TYPEOUT
5680							;;ERROR ADDRESS
5681	035636	104402			TYPOC		;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5682	035640	000426			BP	68	;;GET OUT
5683	035642	005300		14:	DEC	R0	;;ADJUST THE INDEX SO THAT IT WILL
5684	035644	006300			ASL	R0	;; WORK FOR THE ERROR TABLE
5685	035646	006300			ASL	R0	
5686	035650	006300			ASL	R0	
5687	035652	062700	055074		ADD	\$ERRTB,R0	;;FORM TABLE POINTER
5688	035656	012037	035666		MOV	(R0)+,28	;;PICKUP "ERROR MESSAGE" POINTER
5689	035662	001404			BEQ	30	;;SKIP TYPEOUT IF NO POINTER
5690	035664	104401			TYPE		;;TYPE THE "ERROR MESSAGE"
5691	035666	000000		24:	.WORD	0	;;"ERROR MESSAGE" POINTER GOES HERE
5692	035670	104401	001207		TYPE	, \$CRLF	;;"CARRIAGE RETURN" & "LINE FEED"
5693	035674	012037	035704	36:	MOV	(R0)+,48	;;PICKUP "DATA HEADER" POINTER
5694	035700	001404			BEG	58	;;SKIP TYPEOUT IF 0
5695	035702	104401			TYPE		;;TYPE THE "DATA HEADER"
5696	035704	000000		48:	.WORD	0	;;"DATA HEADER" POINTER GOES HERE
5697	035706	104401	001207		TYPE	, \$CRLF	;;"CARRIAGE RETURN" & "LINE FEED"
5698	035712	011000		58:	MOV	(R0),R0	;;PICKUP "DATA TABLE" POINTER
5699	035714	001004			BNE	78	;;GO TYPE THE DATA
5700	035716	012600			MOV	(SP)+,R0	;;RESTORE R0
5701	035720	104401	001207		TYPE	, \$CRLF	;;"CARRIAGE RETURN" & "LINE FEED"
5702	035724	000207			RTS	PC	;;RETURN
5703	035726			78:			
5704	035726	013046			MOV	0(R0)+,-(SP)	;;SAVE 0(R0)+ FOR TYPEOUT
5705	035730	104402			TYPOC		;;GO TYPE--OCTAL ASCII(ALL DIGITS)
5706	035732	005710			TST	(R0)	;;IS THERE ANOTHER NUMBER?
5707	035734	001770			RFQ	68	;;BR IF NO
5708	035736	104401	035744		TYPE	,R\$	;;TYPE TWO(2) SPACES
5709	035742	000771			BR	78	;;LOOP
5710	035744	020040	000	88:	.ASCIIZ	/ /	;;TWO(2) SPACES
5711		035750			.EVEN		
5712					.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 18:
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 022026
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
;*
;* $LSTAD WILL CONTAIN:
;* WITH KT11--LAST VIRTUAL ADDRESS OF THE LAST BANK
;* WITHOUT KT11 --LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
;* $LSTBK WILL CONTAIN THE LAST BANK AS A SAF
;*
;* $KT11 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 R0,-(SP) ;SAVE R0 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,R0 ;SAVE THE STACK POINTER
;SET THE ERRVEC PS TO THE PRESENT PS
MOV $TRAPVEC,-(SP) ;SAVE CURRENT TRAP VECTOR
MOV $64,$TRAPVEC ;SETUP NEW TRAP VECTOR
TRAP ;PUSH OLD PSW AND PC ON STACK
64: MOV 2(SP),$ERRVEC+2 ;SAVE PSW IN $ERRVEC+2
MOV $65,(SP) ;REPLACE OLD PC WITH NEW
RTI ;RESTORE PSW
65: MOV (SP)+,$TRAPVEC ;RESTORE OLD TRAP VECTOR
MOV $3776,R1 ;SETUP ADDRESS
TSTB (PC)+ ;USE MEMORY MANAGEMENT?
200 ;SET TO USE MEMORY MANAGEMENT
$KT11: .WORD
BPL $CORE ;BR IF NO
MOV $KTNEX,$ERRVEC ;SET FOR TIMEOUT
TST $SR0 ;KT11 ARE YOU THERE?
BIS $10000,$KT11 ;YES--SET KT11 KEY
CLR -(SP) ;INITIALIZE FOR "PAR" LOADING
MOV $KIPAR0,R2 ;ADDRESS OF FIRST "PAR"
MOV $D8,R3 ;LOAD EIGHT "PAR,"S" AND EIGHT "PDR,"S"
MOV $77406,-40(R2) ;PDR = 4K, UP, READ/WRITE
MOV (SP),(R2)+ ;LOAD "PAR"
ADD $200,(SP) ;UPDATE FOR NEXT "PAR"
SOB R3,18 ;LOOP UNTIL ALL EIGHT ARE LOADED
MOV $177600,-(R2) ;SETUP KIPAR7 FOR I/O
CLR -(R2) ;SETUP KIPAR6 FOR TESTING
MOV $28,$ERRVEC ;CATCH TIMEOUT IF NO SR3
MOV $20,$SR3 ;ENABLE 22 BIT MODE
BR 38 ;THIS PDP-11 HAS A SR3 REGISTER
CMP (SP)+,(SP)+ ;CLEAN OFF THE STACK--NO SR3
INC $SR0 ;TURN ON MEMORY MANAGEMENT
MOV $KTOUT,$ERRVEC ;SET FOR TIME OUT
TST $0143776 ;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 010746
5820 036332 010346
5821 036334 010546
5822 036336 012746 020200
5823 036342 016605 000020
5824 036346 100004

BHI 48 ;NO--TRY IT
$KTOUT: MOV (R2),R2 ;GET LAST BANK+1
CLR $SR0 ;TURN OFF MEMORY MANAGEMENT
BR $SIZE
$KTNEX: BIC $10000,$KT11 ;KT11 NON-EXISTENT
$CORE: MOV $CROUT,$ERRVEC ;SET FOR TIMEOUT
R2 ;SET UP BANK
18: ADD $4000,R1 ;INCREMENT BY 1K
ADD $40,R2 ;1K STEP
TST (R1) ;TRAP ON TIME OUT
CMP $177776,R1 ;LAST ONE
BNE 18 ;NO--TRY AGAIN
$CROUT: SUB $4000,R1
$SIZE: SUB $40,R2 ;DROP BACK
MOV R0,SP ;RESTORE THE STACK
MOV (SP)+,$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)+,R0 ;RESTORE R0
MOV R0,-(SP) ;SAVE R0 FOR MED INST
MED ;GET CONTENTS OF LOG REG
.WORD RLOG
BIS $100001,R0 ;ENABLE ERROR LOG & LOG FIRST MODE
MED ;UNLOCK ERROR LOG
.WORD WLOG
MOV (SP)+,R0 ;RESTORE R0
RTS PC
$LSTAD: .WORD 0 ;CONTAINS THE LAST ADDRESS
$LSTBK: .WORD 0 ;CONTAINS THE LAST BANK
$BTTL 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
$TYPDS: MOV R0,-(SP) ;PUSH R0 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
RPL 18 ;BR IF INPUT IS POS.

```

```

5025 036350 005405      NEG      R5          ;MAKE THE BINARY NUMBER POS.
5026 036352 112766 000055 000001      MOV     #'-,(SP)      ;MAKE THE ASCII NUMBER NEG.
5027 036360 005000      CLR      R0          ;ZERO THE CONSTANTS INDEX
5028 036362 012703 036540      MOV     $DBLK,R3      ;SETUP THE OUTPUT POINTER
5029 036366 112723 000040      MOV     #' ,(R3)+      ;SET THE FIRST CHARACTER TO A BLANK
5030 036372 005002      CLR      R2          ;CLEAR THE BCD NUMBER
5031 036374 016001 036530      MOV     $DTBL(R0),R1 ;GET THE CONSTANT
5032 036400 160105      SUB      R1,R5      ;FORM THIS BCD DIGIT
5033 036402 002402      BLT      46          ;BR IF DONE
5034 036404 005202      INC      R2          ;INCREASE THE BCD DIGIT BY 1
5035 036406 000774      BR       38          ;
5036 036410 060105      40: ADD     R1,R5      ;ADD BACK THE CONSTANT
5037 036412 005702      TST      R2          ;CHECK IF BCD DIGIT=0
5038 036414 001002      BNE      58          ;FALL THROUGH IF 0
5039 036416 105716      TSTB     (SP)      ;STILL DOING LEADING 0'S?
5040 036420 100407      BMI      74          ;BR IF YES
5041 036422 106316      50: ASLB     (SP)      ;MSD?
5042 036424 103003      BCC      68          ;BR IF NO
5043 036426 116663 000001 177777      MOV     1(SP),-(R3) ;YES--SET THE SIGN
5044 036434 052702 000060      BIS      #'0,R2      ;MAKE THE BCD DIGIT ASCII
5045 036440 052702 000040      BIS      #' ,R2      ;MAKE IT A SPACE IF NOT ALREADY A DIGIT
5046 036444 110223      MOV     R2,(R3)+      ;PUT THIS CHARACTER IN THE OUTPUT BUFFER
5047 036446 005720      TST      (R0)+      ;JUST INCREMENTING
5048 036450 020027 000010      CMP     R0,#10      ;CHECK THE TABLE INDEX
5049 036454 002746      BLT      28          ;GO DO THE NEXT DIGIT
5050 036456 003002      BGT      08          ;GO TO EXIT
5051 036460 010502      MOV     R5,R2      ;GET THE LSD
5052 036462 000764      BR       68          ;GO CHANGE TO ASCII
5053 036464 105726      80: TSTB     (SP)+      ;WAS THE LSD THE FIRST NON-ZERO?
5054 036466 100003      BPL      98          ;BR IF NO
5055 036470 116663 177777 177776      MOV     -(1(SP),-2(R3)) ;YES--SET THE SIGN FOR TYPING
5056 036476 105013      90: CLRB     (R3)      ;SET THE TERMINATOR
5057 036500 012605      MOV     (SP)+,R5      ;POP STACK INTO R5
5058 036502 012603      MOV     (SP)+,R3      ;POP STACK INTO R3
5059 036504 012602      MOV     (SP)+,R2      ;POP STACK INTO R2
5060 036506 012601      MOV     (SP)+,R1      ;POP STACK INTO R1
5061 036510 012600      MOV     (SP)+,R0      ;POP STACK INTO R0
5062 036512 104401 036540      TYPE     $DBLK      ;NOW TYPE THE NUMBER
5063 036516 116666 000002 000004      MOV     2(SP),4(SP) ;ADJUST THE STACK
5064 036524 012616      MOV     (SP)+,(SP) ;
5065 036526 000002      RTI          ;RETURN TO USER
5066 036530 023420      $DTBL: 10000.
5067 036532 001750      1000.
5068 036534 000144      100.
5069 036536 000012      10.
5070 036540 000004      $DBLK: .BLKW 4
5071      .SBTTL TYPE ROUTINE
5072
5073 ;*****
5074 ;ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
5075 ;THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
5076 ;NOTE1: 0NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
5077 ;NOTE2: 0FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
5078 ;NOTE3: 0FILLC CONTAINS THE CHARACTER TO FILL AFTER.
5079 ;
5080 ;CALL:

```

```

5081 ;*1) USING A TRAP INSTRUCTION
5082 ;* TYPE ,MESADR ;MESADR IS FIRST ADDRESS OF AN ASCII STRING
5083 ;*OR
5084 ;* TYPE
5085 ;* MESADR
5086 ;*
5087
5088 036550 105737 001153      0TYPE: TSTB     $TPFLG      ;IS THERE A TERMINAL?
5089 036554 100002      BPL      18          ;BR IF YES
5090 036556 000000      HALT     38          ;HALT HERE IF NO TERMINAL
5091 036560 000430      BR       38          ;LEAVE
5092 036562 010046      10: MOV     R0,-(SP) ;SAVE R0
5093 036564 017600 000002      MOV     02(SP),R0 ;GET ADDRESS OF ASCII STRING
5094 036570 122737 000001 001256      CMPB     $APTEV,$ENV ;RUNNING IN APT MODE
5095 036576 001011      BNE      628      ;NO,GO CHECK FOR APT CONSOLE
5096 036600 132737 000100 001257      BITB     $APTSPOOL,$ENV ;SPOOL MESSAGE TO APT
5097 036606 001405      BEQ      628      ;NO,GO CHECK FOR CONSOLE
5098 036610 010037 036620      MOV     R0,618 ;SETUP MESSAGE ADDRESS FOR APT
5099 036614 004737 037040      JSR      PC,$ATY3 ;SPOOL MESSAGE TO APT
5100 036620 000000      .WORD     0 ;MESSAGE ADDRESS
5101 036622 132737 000040 001257      618: BITB     $APTCUP,$ENV ;APT CONSOLE SUPPRESSED
5102 036630 001003      BNE      608      ;YES,SKIP TYPE OUT
5103 036632 112046      20: MOV     (R0)+,-(SP) ;PUSH CHARACTER TO BE TYPED ONTO STACK
5104 036634 001005      BNE      46          ;BR IF IT ISN'T THE TERMINATOR
5105 036636 005726      TST      (SP)+      ;IF TERMINATOR POP IT OFF THE STACK
5106 036640 012600      608: MOV     (SP)+,R0 ;RESTORE R0
5107 036642 062716 000002      30: ADD     #2,(SP) ;ADJUST RETURN PC
5108 036646 000002      RTI          ;RETURN
5109 036650 122716 000011      40: CMPB     $HT,(SP) ;BRANCH IF <HT>
5110 036654 001430      BEQ      88          ;
5111 036656 122716 000200      CMPB     $CRLF,(SP) ;BRANCH IF NOT <CRLF>
5112 036662 001006      BNE      58          ;
5113 036664 005726      TST      (SP)+      ;POP <CR><LF> EQUIV
5114 036666 104401      TYPE     TYPE ;TYPE A CR AND LF
5115 036670 001207      $CRLF
5116 036672 105037 037026      CLRB     CLRB ;CLEAR CHARACTER COUNT
5117 036676 000755      BR       28          ;GET NEXT CHARACTER
5118 036700 004737 036762      50: JSR      PC,$TYPEC ;GO TYPE THIS CHARACTER
5119 036704 123726 001152      60: CMPB     $FILLC,(SP)+ ;IS IT TIME FOR FILLER CHARS.?
5120 036710 001350      BNE      28          ;IF NO GO GET NEXT CHAR.
5121 036712 013746 001150      MOV     $NULL,-(SP) ;GET # OF FILLER CHARS. NEEDED
5122      ;AND THE NULL CHAR.
5123 036716 105366 000001      70: DECB     1(SP) ;DOES A NULL NEED TO BE TYPED?
5124 036722 002770      BLT      68          ;BR IF NO--GO POP THE NULL OFF OF STACK
5125 036724 004737 036762      JSR      PC,$TYPEC ;GO TYPE A NULL
5126 036730 105337 037026      DECB     $CHARCNT ;DO NOT COUNT AS A COUNT
5127 036734 000770      BR       76          ;LOOP
5128
5129 ;HORIZONTAL TAB PROCESSOR
5130
5131 036736 112716 000040      80: MOV     #' ,(SP) ;REPLACE TAB WITH SPACE
5132 036742 004737 036762      90: JSR      PC,$TYPEC ;TYPE A SPACE
5133 036746 132737 000007 037026      BITB     $7,$CHARCNT ;BRANCH IF NOT AT
5134 036754 001372      BNE      96          ;TAB STOP
5135 036756 005726      TST      (SP)+      ;POP SPACE OFF STACK
5136 036760 000724      BR       28          ;GET NEXT CHARACTER

```

```

5937 036762 105777 142156 $TYPE: TSTB $STPS ;;WAIT UNTIL PRINTER IS READY
5938 036766 100375 BPL $TYPE
5939 036770 116677 000002 142150 MOV 2(SP),0$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
5940 036776 122766 000015 000002 CMPB $CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
5941 037004 001003 BNE 10 ;;BRANCH IF NO
5942 037006 105037 037026 CLR 0$CHARCNT ;;YES--CLEAR CHARACTER COUNT
5943 037012 000406 BR $TYPEX ;;EXIT
5944 037014 122766 000012 000002 10: 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 $ATY1: MOV 01,$FFLG ;;TO REPORT FATAL ERROR
5954 037040 112737 000001 037274 $ATY3: MOV 01,$MFLG ;;TO TYPE A MESSAGE
5955 037046 000403 BR $ATYC
5956 037050 112737 000001 037276 $ATY4: MOV 01,$FFLG ;;TO ONLY REPORT FATAL ERROR
5957 037056 $ATYC:
5958 037056 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
5959 037060 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
5960 037062 105737 037274 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
5961 037066 001450 BEQ 50 ;;IF NOT: BR
5962 037070 122737 000001 001256 CMPB $APENV,$ENV ;;OPERATING UNDER APT?
5963 037076 001031 BNE 30 ;;IF NOT: BR
5964 037100 132737 000100 001257 BITB $APTSPOOL,$ENV ;;SHOULD SPOOL MESSAGES?
5965 037106 001425 BEQ 30 ;;IF NOT: BR
5966 037110 017600 000004 MOV 04(SP),R0 ;;GET MESSAGE ADDR.
5967 037114 062766 000002 000004 ADD 02,4(SP) ;;BUMP RETURN ADDR.
5968 037122 005737 001236 10: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
5969 037126 001375 BNE 10 ;;IF NOT: WAIT
5970 037130 010037 001252 MOV R0,$MSGAD ;;PUT ADDR IN MAILBOX
5971 037134 105720 20: TSTB (R0)+ ;;FIND END OF MESSAGE
5972 037136 001376 BNE 20
5973 037140 163700 SUB $MSGAD,R0 ;;SUB START OF MESSAGE
5974 037144 006200 ASH R0 ;;GET MESSAGE LGTH IN WORDS
5975 037146 010037 001254 MOV R0,$MSGGLT ;;PUT LENGTH IN MAILBOX
5976 037152 012737 000004 001236 MOV 04,$MSGTYPE ;;TELL APT TO TAKE MSG.
5977 037160 000413 BR 50
5978 037162 017637 000004 037206 30: MOV 04(SP),40 ;;PUT MSG ADDR IN JSR LINKAGE
5979 037170 062766 000002 000004 ADD 02,4(SP) ;;BUMP RETURN ADDRESS
5980 037176 013746 177776 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
5981 037202 004737 036550 JSR PC,$TYPE ;;CALL TYPE MACRO
5982 037206 000000 40: .WORD 0
5983 037210 50:
5984 037210 105737 037276 100: TSTB $FFLG ;;SHOULD REPORT FATAL ERROR?
5985 037214 001416 BEQ 120 ;;IF NOT: BR
5986 037216 005737 001256 TST $ENV ;;RUNNING UNDER APT?
5987 037222 001413 BEQ 120 ;;IF NOT: BR
5988 037224 005737 001236 110: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
5989 037230 001375 BNE 110 ;;IF NOT: WAIT
5990 037232 017637 000004 001240 MOV 04(SP),$FATAL ;;GET ERROR
5991 037240 062766 000002 000004 ADD 02,4(SP) ;;BUMP RETURN ADDR.
5992 037246 005237 001236 INC $MSGTYPE ;;TELL APT TO TAKE ERROR

```

```

5993 037252 105037 037276 120: CLRB $FFLG ;;CLEAR FATAL FLAG
5994 037256 105037 037275 CLRB $LFLG ;;CLEAR LOG FLAG
5995 037262 105037 037274 CLRB $MFLG ;;CLEAR MESSAGE FLAG
5996 037266 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
5997 037270 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
5998 037272 000207 RTS PC ;;RETURN
5999 037274 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
6000 037275 000 $LFLG: .BYTE 0 ;;LOG FLAG
6001 037276 000 $FFLG: .BYTE 0 ;;FATAL FLAG
6002 037300 .EVEN
6003 APTSIZE=200
6004 APTENV=001
6005 APTSPOOL=100
6006 APTCSUP=040
6007 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
6008
6009 ;*****
6010 ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
6011 ;OCTAL (ASCII) NUMBER AND TYPE IT.
6012 ;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
6013 ;CALL:
6014 ;* MOV NUM,-(SP) ;;NUMBER TO BE TYPED
6015 ;* TYPOS ;;CALL FOR TYPEOUT
6016 ;* .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
6017 ;* .BYTE M ;;M=1 OR 0
6018 ;* ;;1=TYPE LEADING ZEROS
6019 ;* ;;0=SUPPRESS LEADING ZEROS
6020 ;*
6021 ;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
6022 ;$TYPOS OR $TYPOC
6023 ;CALL:
6024 ;* MOV NUM,-(SP) ;;NUMBER TO BE TYPED
6025 ;* TYPON ;;CALL FOR TYPEOUT
6026 ;*
6027 ;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
6028 ;CALL:
6029 ;* MOV NUM,-(SP) ;;NUMBER TO BE TYPED
6030 ;* TYPOC ;;CALL FOR TYPEOUT
6031 ;*
6032 037300 017646 000000 $TYPOS: MOV 0(SP),-(SP) ;;PICKUP THE MODE
6033 037304 116637 000001 037523 MOV 1(SP),0$FILL ;;LOAD ZERO FILL SWITCH
6034 037312 112637 037525 MOV 0$FILL,0$MODE+1 ;;NUMBER OF DIGITS TO TYPE
6035 037316 062716 000002 ADD 02,(SP) ;;ADJUST RETURN ADDRESS
6036 037322 000406 BR $TYPON
6037 037324 112737 000001 037523 $TYPOC: MOV 01,$0FILL ;;SET THE ZERO FILL SWITCH
6038 037332 112737 000006 037525 MOV 06,$0MODE+1 ;;SET FOR SIX(6) DIGITS
6039 037340 112737 000005 037522 $TYPON: MOV 05,$0CNT ;;SET THE ITERATION COUNT
6040 037346 010346 MOV R3,-(SP) ;;SAVE R3
6041 037350 010446 MOV R4,-(SP) ;;SAVE R4
6042 037352 010546 MOV R5,-(SP) ;;SAVE R5
6043 037354 113704 037525 MOV 0$MODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
6044 037360 005404 NEG R4
6045 037362 062704 000006 ADD 06,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
6046 037366 110437 037524 MOV R4,$0MODE ;;SAVE IT FOR USE
6047 037372 113704 037523 MOV 0$FILL,R4 ;;GET THE ZERO FILL SWITCH
6048 037376 016605 000012 MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER

```

```

0049 037402 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
0050 037404 006105          ROL      R5          ;;ROTATE MSB INTO "C"
0051 037406 000404          BR       30          ;;GO DO MSB
0052 037410 006105          ROL      R5          ;;FORM THIS DIGIT
0053 037412 006105          ROL      R5
0054 037414 006105          ROL      R5
0055 037416 010503          MOV      R5,R3
0056 037420 006103          ROL      P3          ;;GET LSB OF THIS DIGIT
0057 037422 105337 037524    DECB     $OMODE     ;;TYPE THIS DIGIT?
0058 037426 100016          BPL      70          ;;BR IF NO
0059 037430 042703 177770    BIC      #177770,R3    ;;GET RID OF JUNK
0060 037434 001002          BNE      40          ;;TEST FOR 0
0061 037436 005704          TST      R4          ;;SUPPRESS THIS 0?
0062 037440 001403          BEQ      50          ;;BR IF YES
0063 037442 005204          INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
0064 037444 052703 000060    BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
0065 037450 052703 000040    BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
0066 037454 110337 037520    MOVB     R3,R8      ;;SAVE FOR TYPING
0067 037460 104401 037520    TYPE     ,R8          ;;GO TYPE THIS DIGIT
0068 037464 105337 037522    DECB     $OCNT      ;;COUNT BY 1
0069 037470 003347          BGT      20          ;;BR IF MORE TO DO
0070 037472 002402          BLT      60          ;;BR IF DONE
0071 037474 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
0072 037476 000744          BR       20          ;;GO DO THE LAST DIGIT
0073 037500 012605          MOV      (SP)+,R5    ;;RESTORE R5
0074 037502 012604          MOV      (SP)+,R4    ;;RESTORE R4
0075 037504 012603          MOV      (SP)+,R3    ;;RESTORE R3
0076 037506 016666 000002 000004    MOV      2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
0077 037514 012616          MOV      (SP)+,(SP)
0078 037516 000002          RTI              ;;RETURN
0079 037520 000          .BYTE     0          ;;STORAGE FOR ASCII DIGIT
0080 037521 000          .BYTE     0          ;;TERMINATOR FOR TYPE ROUTINE
0081 037522 000          $OCNT: .BYTE 0          ;;OCTAL DIGIT COUNTER
0082 037523 000          $OFILL: .BYTE 0        ;;ZERO FILL SWITCH
0083 037524 000000          $OMODE: .WORD 0       ;;NUMBER OF DIGITS TO TYPE
0084          .SBTTL TTY INPUT ROUTINE

;*****
;ENABL LSB

;DSABL LSB

;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;CALL:
; RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
; RETURN HERE    ;;CHARACTER IS ON THE STACK
;               ;;WITH PARITY BIT STRIPPED OFF
;
$RDCHR: MOV      (SP),-(SP)    ;;PUSH DOWN THE PC
        MOV      4(SP),2(SP)  ;;SAVE THE PS
10:      TSTB     $TKS         ;;WAIT FOR
        BPL      10          ;;A CHARACTER
        MOVB     $TKB,4(SP)   ;;READ THE TTY

```

```

6105 037552 042766 177600 000004    BIC      #'C<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      30          ;;BRANCH IF NO
6108 037570 105777 141344          TSTB     $TKS         ;;WAIT FOR A CHARACTER
6109 037574 100375          BPL      20          ;;LOOP UNTIL ITS THERE
6110 037576 117746 141340          MOVB     $TKB,-(SP)    ;;GET CHARACTER
6111 037602 042716 177600          BIC      #'C177,(SP)    ;;MAKE IT 7-BIT ASCII
6112 037606 022627 000021          CMP      (SP)+,#21    ;;IS IT A CONTROL-Q?
6113 037612 001366          BNE      20          ;;IF NOT DISCARD IT
6114 037614 000750          BR       10          ;;YES, RESUME
6115 037616 026627 000004 000140    CMP      4(SP),#140     ;;IS IT UPPEP CASE?
6116 037624 002407          BLT      40          ;;BRANCH IF YES
6117 037626 026627 000004 000175    CMP      4(SP),#175     ;;IS IT A SPECIAL CHAR?
6118 037634 003003          BGT      40          ;;BRANCH IF YES
6119 037636 042766 000040 000004    BIC      #40,4(SP)     ;;MAKE IT UPPER CASE
6120 037644 000002          RTI              ;;GO BACK TO USER
6121          ;*****
6122          ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
6123          ;CALL:
6124          ; RDLIN          ;;INPUT A STRING FROM THE TTY
6125          ; RETURN HERE    ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
6126          ;               ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
6127
$RDLIN: MOV      R3,-(SP)      ;;SAVE R3
        CLR      -(SP)        ;;CLEAR THE RUBOUT KEY
6130 037650 005046          MOV      $TTYIN,R3    ;;GET ADDRESS
6131 037656 022703 040102          CMP      $TTYIN+8,,R3    ;;BUFFER FULL?
6132 037662 101456          BLOS     40          ;;BR IF YES
6133 037664 104406          RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
6134 037666 112613          MOVB     (SP)+,(R3)    ;;GET CHARACTER
6135 037670 122713 000177          CMPB     #177,(R3)    ;;IS IT A RUBOUT
6136 037674 001022          BNE      50          ;;BR IF NO
6137 037676 005716          TST      (SP)          ;;IS THIS THE FIRST RUBOUT?
6138 037700 001007          BNE      60          ;;BR IF NO
6139 037702 112737 000134 040100    MOVB     #' \,90       ;;TYPE A BACK SLASH
6140 037710 104401 040100          TYPE     ,90
6141 037714 112716 177777          MOV      #-1,(SP)    ;;SET THE RUBOUT KEY
6142 037720 005303          DEC      R3          ;;BACKUP BY ONE
6143 037722 020327 040102          CMP      R3,$TTYIN    ;;STACK EMPTY?
6144 037726 103434          BLO      40          ;;BR IF YES
6145 037730 111337 040100          MOVB     (R3),90      ;;SETUP TO TYPEOUT THE DELETED CHAR.
6146 037734 104401 040100          TYPE     ,90
6147 037740 000746          BR       20          ;;GO TYPE
6148 037742 005716          TST      (SP)          ;;RUBOUT KEY SET?
6149 037744 001406          BEQ      70          ;;BR IF NO
6150 037746 112737 000134 040100    MOVB     #' \,90       ;;TYPE A BACK SLASH
6151 037754 104401 040100          TYPE     ,90
6152 037760 005016          CLR      (SP)          ;;CLEAR THE RUBOUT KEY
6153 037762 122713 000025          CMPB     #25,(R3)    ;;IS CHARACTER A CTRL U?
6154 037766 001003          RNE      80          ;;BR IF NO
6155 037770 104401 040112          TYPE     , $CNTLU    ;;TYPE A CONTROL "U"
6156 037774 000726          BR       10          ;;GO START OVER
6157 037776 122713 000022          CMPB     #22,(R3)    ;;IS CHARACTER A "R"?
6158 040002 001011          RNE      30          ;;BRANCH IF NO
6159 040004 105013          CLRB     (R3)        ;;CLEAR THE CHARACTER
6160 040006 104401 001207          TYPE     , $CRLF    ;;TYPE A "CR" & "LF"

```

```

6161 040012 104401 040102      TYPE      ,8TTYIN      ;;TYPE THE INPUT STRING
6162 040016 000717              BR          28              ;;GO PICKUP ANOTHER CHARACTER
6163 040020 104401 001206      48:      TYPE      ,8QUES      ;;TYPE A "7"
6164 040024 000712              BR          18              ;;CLEAR THE BUFFER AND LOOP
6165 040026 111337 040100      38:      MOVB      (R3),98      ;;ECHO THE CHARACTER
6166 040032 104401 040100      TYPE      ,98              ;;CHECK FOR RETURN
6167 040036 122723 000015      CMPB      #15,(R3)+      ;;LOOP IF NOT RETURN
6168 040042 001305              BNE          28              ;;CLEAR RETURN (THE 15)
6169 040044 105663 177777      CLRB      -(R3)          ;;TYPE A LINE FEED
6170 040050 104401 001210      TYPE      ,8LF              ;;CLEAN RUBOUT KEY FROM THE STACK
6171 040054 005726              TST          (SP)+          ;;RESTORE R3
6172 040056 012603              MOV          (SP)+,R3      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
6173 040060 011646              MOV          (SP),-(SP)      ;; FIRST ASCII CHARACTER ON IT
6174 040062 016666 000004 000002  MOV          4(SP),2(SP)
6175 040070 012766 040102 000004  MOV          #8TTYIN,4(SP)
6176 040076 000002              RTI                      ;;RETURN
6177 040100 000              .BYTE      0              ;;STORAGE FOR ASCII CHAR. TO TYPE
6178 040101 000              .BYTE      0              ;;TERMINATOR
6179 040102 000010              .BLKB      8              ;;RESERVE 8 BYTES FOR TTY INPUT
6180 040112 052536 005015 000      %CNTLU: .ASCIIZ  /"U"<15><12>  ;;CONTROL "U"
6181 040117 136 006507 000012  %CNTLG: .ASCIIZ  /"G"<15><12>  ;;CONTROL "G"
6182 040124 005015 053523 020122  %MSWR: .ASCIIZ  <15><12>/5WR = /
6183 040132 020075 000
6184 040135 040 047040 053505  %MNEW: .ASCIIZ  / NEW = /
6185 040142 036440 000040
6186
6187 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
6188
6189 ;;*****
6190 ;;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
6191 ;;CHANGE IT TO BINARY.
6192 ;;THE INPUT CHARACTERS WILL BE CHECKED TO INSURE THEY ARE LEGAL
6193 ;;OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "7" WILL BE TYPED
6194 ;;FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
6195 ;;THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
6196 ;;CALL:
6197 ;*      RDOCT              ;;READ AN OCTAL NUMBER
6198 ;*      RETURN HERE      ;;LOW ORDER BITS ARE ON TOP OF THE STACK
6199 ;*                      ;;HIGH ORDER BITS ARE IN %HI OCT
6200
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 010746      MOV          R2,-(SP)      ;;PUSH R2 ON STACK
6205 040164 104407      18:      RDLIN              ;;READ AN ASCII LINE
6206 040166 012600      MOV          (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
6207 040170 010037 040274      MOV          R0,56      ;;AND SAVE IT
6208 040174 005001      CLR          R1              ;;CLEAR DATA WORD
6209 040176 005002      CLR          R2              ;;PICKUP THIS CHARACTER
6210 040200 112046      28:      MOVB      (R0)+,-(SP)      ;;IF ZERO GET OUT
6211 040202 001420      BEQ          38              ;;MAKE SURE THIS CHARACTER
6212 040204 127716 000060      CMPB      #'0,(SP)      ;;IS AN OCTAL DIGIT
6213 040210 003026      BGT          48
6214 040212 127716 000067      CMPB      #'7,(SP)
6215 040216 002423      BLT          48
6216 040220 006301      ASL          R1              ;;*2

```

```

6217 040222 006102      POL          R2              ;;*4
6218 040224 006301      ASL          R1
6219 040226 006102      POL          R2              ;;*8
6220 040230 006301      ASL          R1
6221 040232 006102      POL          R2              ;;*16
6222 040234 042716 177770      BIC          #'C7,(SP)      ;;STRIP THE ASCII JUNK
6223 040240 062601      ADD          (SP)+,R1      ;;ADD IN THIS DIGIT
6224 040242 000756      BR          28              ;;LOOP
6225 040244 005726      38:      TST          (SP)+          ;;CLEAN TERMINATOR FROM STACK
6226 040246 010166 000012      MOV          R1,12(6P)      ;;SAVE THE RESULT
6227 040252 010237 040304      MOV          R2,%HI OCT
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                      ;;RETURN
6232 040266 005726      48:      TST          (SP)+          ;;CLEAN PARTIAL FROM STACK
6233 040270 105010      CLRB      (R0)              ;;SET A TERMINATOR
6234 040272 104401      TYPE          0              ;;TYPE UP THRU THE BAD CHAR.
6235 040274 000000      56:      .WORD      0
6236 040276 104401 001206      TYPE          ,8QUES      ;;"? "CR" & "LF"
6237 040302 000730      BR          18              ;;TRY AGAIN
6238 040304 000000      %HI OCT: .WORD      0              ;;HIGH ORDER BITS GO HERE
6239 .SBTTL TRAP DECODER
6240
6241 ;;*****
6242 ;;THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
6243 ;;AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
6244 ;;OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
6245 ;;GO TO THAT ROUTINE.
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 ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
6258
6259 040330 011646      %TRAP2: MOV          (SP),-(SP)      ;;MOVE THE PC DOWN
6260 040332 016666 000004 000002  MOV          4(SP),2(SP)      ;;MOVE THE PSW DOWN
6261 040340 000002      RTI                      ;;RESTORE THE PSW
6262
6263 .SBTTL TRAP TABLE
6264
6265 ;;THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
6266 ;;BY THE "TRAP" INSTRUCTION.
6267
6268 ;      POUTINE
6269 ;      -----
6270 %TRPAD: .WORD      %TRAP2      TRAP+1(104401)  TTY TYPEOUT ROUTINE
6271          .TYPE      %CALL=TYPE  TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
6272          .STYPOC      %CALL=STYPOC  TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)

```

6273	040352	037340		\$TYPON	;;CALL=TYPON	TRAP+4(104404)	TTY OCTAL NUMBER (AS PER LAST CALL)
6274	040354	036324		\$TYPDS	;;CALL=TYPDS	TRAP+5(104405)	TTY DECIMAL NUMBER (WITH SIGN)
6275							
6276							
6277	040356	037526		\$RDCHR	;;CALL=RDCHR	TRAP+6(104406)	TTY TYPEIN CHARACTER ROUTINE
6278	040360	037646		\$RDLIN	;;CALL=RDLIN	TRAP+7(104407)	TTY TYPEIN STRING ROUTINE
6279	040362	040146		\$RDOCT	;;CALL=RDOCT	TRAP+10(104410)	READ AN OCTAL NUMBER FROM TTY
6280				.SBTTL POWER DOWN AND UP ROUTINES			
6281				;*****			
6282				;POWER DOWN ROUTINE			
6283				\$PWRDN:	MOV	\$ILLUP,00PWRVEC	;;SET FOR FAST UP
6284	040364	012737	040524	000024	MOV	R0,-(SP)	;;PUSH R0 ON STACK
6285	040372	012737	000340	000026	MOV	R1,-(SP)	;;PUSH R1 ON STACK
6286	040400	010046			MOV	R2,-(SP)	;;PUSH R2 ON STACK
6287	040402	010146			MOV	R3,-(SP)	;;PUSH R3 ON STACK
6288	040404	010246			MOV	R4,-(SP)	;;PUSH R4 ON STACK
6289	040406	010346			MOV	R5,-(SP)	;;PUSH R5 ON STACK
6290	040410	010446			MOV	\$SWR,-(SP)	;;PUSH \$SWR ON STACK
6291	040412	010546			MOV	SP,\$SAVR6	;;SAVE SP
6292	040414	017746	140514		MOV	\$PWRUP,00PWRVEC	;;SET UP VECTOR
6293	040420	010637	040530		HALT		
6294	040424	012737	040436	000024	BR	.-2	;;HANG UP
6295	040432	000000					
6296	040434	000776					
6297							
6298							
6299							
6300	040436	012737	040524	000024	;*****		
6301	040444	013706	040530		;POWER UP ROUTINE		
6302	040450	005037	040530		\$PWRUP:	MOV	\$ILLUP,00PWRVEC
6303	040454	005237	040530		MOV	\$SAVR6,SP	;;GET SP
6304	040460	001375			CLR	\$SAVR6	;;WAIT LOOP FOR THE TTY
6305	040462	012677	140446		INC	\$SAVR6	;;WAIT FOR THE INC
6306	040466	012635			BNE	18	;;OF WORD
6307	040470	012604			MOV	(SP)+,\$SWR	;;POP STACK INTO \$SWR
6308	040472	012603			MOV	(SP)+,R5	;;POP STACK INTO R5
6309	040474	012602			MOV	(SP)+,R4	;;POP STACK INTO R4
6310	040476	012601			MOV	(SP)+,R3	;;POP STACK INTO R3
6311	040500	012600			MOV	(SP)+,R2	;;POP STACK INTO R2
6312	040502	012737	040364	000024	MOV	(SP)+,R1	;;POP STACK INTO R1
6313	040510	012737	000340	000026	MOV	(SP)+,R0	;;POP STACK INTO R0
6314	040516	104401			MOV	\$PWRDN,00PWRVEC	;;SET UP THE POWER DOWN VECTOR
6315	040520	040532			MOV	\$340,00PWRVEC+2	;;PRI0:7
6316	040522	000002			TYPE		;;REPORT THE POWER FAILURE
6317	040524	000000			\$PWRMG:	WORD	;;POWER FAIL MESSAGE POINTER
6318	040526	000776			RTI		
6319	040530	000000			\$ILLUP:	HALT	;;THE POWER UP SEQUENCE WAS STARTED
6320	040532	005015	047520	042527	BR	.-2	;; BEFORE THE POWER DOWN WAS COMPLETE
6321	040540	000122			\$SAVR6:	0	;;PUT THE SP HERE
6322					\$POWER:	.ASCIZ	<15><12>"POWER"
6323							
6324							
6325							
6326							
6327	040542	005015	005015	040515	;*****		
6328	040550	047111	042504	026503	;*****		
					;*****		
					MSG1:	.ASCIZ	<15><12><15><12>*MAINDEC-11-DQKKA-1 11/6X CACHE DIAGNOSTIC*<15><12><15><12>

6329	040556	030461	042055	045521			
6330	040564	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	MSG2:	.ASCIZ	<15><12>*POWER MACHINE DOWN AND THEN UP*<15><12>
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	MSG3:	.ASCII	<CR><LF>*TYPE WHICH DEVICE SHOULD BE USED*<CR><LF>
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	070122	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	070122	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	MSG4:	.ASCIZ	<CR><LF>*? INVALID ENTRY, TRY AGAIN*<CR><LF>
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	MSG5:	.ASCIZ	<CR><LF>*TYPE THE URE'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	070124	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 R0Y 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 DATAB 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 DATAB*
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 WPITTEN 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	EM26: .ASCIZ*ERROR:CACHE CONTROL REG HELD WRONG DATA*
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	EM27: .ASCIZ*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*<15><12>
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	.ASCIZ* DID NOT GET PARITY TRAP FROM TAG FIELD WHEN WROTE WRONG PARITY*
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	EM31: .ASCIZ*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*<15><12>
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	.ASCIZ* TAG FIELD HELD WRONG DATA ON PARITY TRAP*
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	EM32: .ASCIZ*ERROR:TEST OF TAG PARITY GENERATOR/CHECKER FAILED*<15><12>
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	

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: .ASCIZ*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: .ASCIZ*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: .ASCIZ*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 050015
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 050015
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

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	EM47: .ASCIZ=ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG P BIT*
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	042124	052123	047111	
6840	046244	020107	040524	020107	
6841	046252	020120	044502	000124	
6842	046260	051105	047522	035122	EM50: .ASCIZ=ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA P BIT*
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	EM51: .ASCIZ=ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BIT*
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	EM52: .ASCIZ=ERROR: TAG PARITY ERROR WHEN TESTING TAG ADDRESS BITS*
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	EM53: .ASCIZ=ERROR: LOW BYTE PARITY ERROR WHEN TESTING TAG ADDRESS BITS*
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	EM54: .ASCIZ=ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG ADDRESS BITS*
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	

6889	046702	041040	052111	000123	
6890	046710	051105	047522	035122	EM55: .ASCIZ=ERROR: TEST OF TAG ADDRESS BITS FAILED<15><12>
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	.ASCIZ= ADDRESS COULD NOT BE MADE A HIT*
6898	046766	040440	042104	042522	
6899	046774	051523	041440	052517	
6900	047002	047114	047040	052117	
6901	047010	041040	020105	040515	
6902	047016	042504	040440	044040	
6903	047024	052111	000		
6904	047027	105	051122	051117	EM56: .ASCIZ=ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA FIELD*
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	EM57: .ASCIZ=ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA FIELD*
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	EM60: .ASCIZ=ERROR: TAG PARITY ERROR WHEN TESTING DATA FIELD*
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	EM61: .ASCIZ=ERROR: CACHE DATA LOC HELD WRONG DATA*
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	EM62: .ASCIZ=ERROR:TEST OF M8B ADDRESS (A10) TO CACHE DATA FIELD FAILED<15><12>
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	

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	.ASCIZ*	HIGH BYTE PARITY ERROR*
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	EM77:	.ASCII*ERROR:DYNAMIC TEST OF CACHE FAILED*<15><12>
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	.ASCIZ*	TAG PARITY ERROR*
7130	051456	052040	043501	050040		
7131	051464	051101	052111	020131		
7132	051472	051105	047522	000122		
7133	051500	051105	047522	035122	EM101:	.ASCIZ*ERROR:CACHE CONTROL REG NOT INITIALIZED BY POWER FAIL*
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	EM102:	.ASCIZ*ERROR:POWER UP FAILED TO INVALIDATE CACHE*
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	EM103:	.ASCIZ*ERROR:DEVICE ERROR BIT SET WHEN DOING NPR, DATO TO ADDRESS*
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	EM104:	.ASCIZ*ERROR:CACHE LOCATION NOT INVALIDATED BY NPR, DATO TO ADDRESS*
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		

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	EM113: .ASCIZ*ERROR:ADDRESS NOT INVALIDATED BY CONSOLE SWEEP*<CR><LF>
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	EM114: .ASCIZ*ERROR:LOC WRITTEN WITH WRONG PARITY NOT INVALIDATED VIA NPR DATO*
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	EM115: .ASCII*ERROR:PARITY TRAP WHILE TESTING LOC WRITTEN WITH WRONG PARITY*<CR><LF>
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	.ASCIZ* AND INVALIDATING IT VIA NPR DATO*
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	EM116: .ASCIZ*ERROR:CACHE ALLOCATED DURING ODD ADDRESS TRAP*
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	EM117: .ASCIZ*ERROR:CACHE ALLOCATED DURING RED ZONE TRAP*
7277	053164	040503	044103	020105	
7278	053172	046101	047514	040503	
7279	053200	042524	020104	052504	
7280	053206	044522	043516	051040	

7281	053214	042105	055040	047117	
7282	053222	020105	051124	050101	
7283	053230	000			
7284	053231	105	051122	051117	EM120: .ASCII=ERROR: CACHE ALLOCATED DURING KT ABORT*
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	EM121: .ASCII=ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED* <CR> <LF>
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	.ASCII* LOC NOT INVALIDATED*
7301	053372	047514	020103	047516	
7302	053400	020124	047111	040526	
7303	053406	044514	040504	042524	
7304	053414	000104			
7305	053416	051105	047522	035122	EM122: .ASCII=ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED* <CR> <LF>
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	.ASCII* PARITY ERROR TAG*
7315	053510	050040	051101	052111	
7316	053516	020131	051105	047522	
7317	053524	020122	040524	000107	EM123: .ASCII=ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED* <CR> <LF>
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	.ASCII* PARITY ERROR LOW BYTE*
7328	053624	050040	051101	052111	
7329	053632	020131	051105	047522	
7330	053640	020122	047514	020127	
7331	053646	054502	042524	000	EM124: .ASCII=ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED* <CR> <LF>
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	

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	.ASCII* 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: .ASCII*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: .ASCII*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: .ASCII*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: .ASCII*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: .ASCII*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: .ASCII*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: .ASCII*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	\$ERRPC,\$REG1,\$REG2,\$REG3,\$REG4,0
7444	055012	001116	001166	000000			
7445	055020	001116	001160	001162	DT5:	.WORD	\$ERRPC,\$REG1,\$REG2,0
7446	055026	000000					
7447	055030	001116	001160	001162	DT12:	.WORD	\$ERRPC,\$REG1,\$REG2,\$REG3,0
7448	055036	001164	000000				

7449	055042	001116	001160	000000	DT16:	.WORD	\$ERRPC,\$REG1,0
7450	055050	001116	001160	001162	DT35:	.WORD	\$ERRPC,\$REG1,\$REG2,\$REG4,0
7451	055056	001166	000000				
7452	055062	001116	001160	000000	DT100:	.WORD	\$ERRPC,\$REG1,0
7453	055070	001116	000000		DT107:	.WORD	\$ERRPC,0
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							

;*****
 ;*****
 .SBTTL ERROR POINTER TABLE
 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 ;* THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 ;*LOCATION \$ITEMB, THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 ;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
 ;* FM ;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	;ERRRPC,\$REG1,\$REG2
7508	055162	000000	0	
7509				
7510	055164	042460	ITEM 10	
7511	055166	054226	EM10	;ERROR: DATA NOT CACHED ON DATOB TO A HIT LOC.
7512	055170	055020	DH7	;PC=/P ADDH/P ADDL
7513	055172	000000	DT5	;ERRRPC,\$REG1,\$REG2
7514			0	
7515	055174	042541	ITEM 11	
7516	055176	054256	EM11	;ERROR: CACHE DID NOT CONTAIN PROPER DATA ON DATOB
7517	055200	055004	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7518	055202	000000	DT1	;ERRRPC,\$REG1,\$REG2,\$REG3,\$REG4
7519			0	
7520	055204	042623	ITEM 12	
7521	055206	054336	EM12	;ERROR: FORCE MISS BIT FAILED TO CAUSE MISS
7522	055210	055030	DH12	;PC=/(CCR)/P ADDH/P ADDL
7523	055212	000000	DT12	;ERRRPC,\$REG1,\$REG2,\$REG3
7524			0	
7525	055214	042330	ITEM 13	
7526	055216	054144	EM6	;FATAL ERROR: HIT MISS REG HELD WRONG DATA
7527	055220	055020	DH6	;PC=/DATA IS/DATA EXPECTED SET (0= DON'T CARE)
7528	055222	000000	DT5	;ERRRPC,\$REG1,\$REG2
7529			0	
7530	055224	042676	ITEM 14	
7531	055226	054226	EM14	;ERROR: ADDRESS COULD NOT BE MADE A HIT AFTER DATO TO IT
7532	055230	055020	DH7	;PC=/P ADDH/P ADDL
7533	055232	000000	DT5	;ERRRPC,\$REG1,\$REG2
7534			0	
7535	055234	000000	ITEM 15	
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	;ERRRPC,\$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	;ERRRPC,\$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	;ERRRPC,\$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/(ERE)
7558	055300	055030	DT12	;ERRRPC,\$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	;ERRRPC,\$REG1,\$REG2,\$REG3
7564	055312	000000	0	
7565			ITEM 23	
7566	055314	043316	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	;ERRRPC,\$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	;ERRRPC,\$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	;ERRRPC,\$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	;ERRRPC,\$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	;ERRRPC,\$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	;ERRRPC,\$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	;ERRRPC,\$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	;ERRRPC,\$REG1,\$REG2,\$REG3
7614	055422	000000	0	
7615			ITEM 34	
7616	055424	044514	EM34	;ERROR: TEST OF TAG PARITY GEN/CKER FAILED

```

7617
7618 055426 045601 DH30
7619 055430 055004 DT1
7620 055432 000000 0
7621 ; ITEM 35
7622 055434 044637 EM35 ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
7623 ; NO PARITY TRAP OCCURRED
7624 055436 054660 DH35
7625 055440 055050 DT35
7626 055442 000000 0
7627 ; ITEM 36
7628 055444 045000 EM36 ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
7629 ; NO PARITY TRAP FROM LOW BYTE WHEN WWP
7630 055446 054256 DH11
7631 055450 055004 DT1
7632 055452 000000 0
7633 ; ITEM 37
7634 055454 045157 EM37 ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
7635 ; NO PARITY TRAP FROM HIGH BYTE WHEN WWP
7636 055456 054256 DH11
7637 055460 055004 DT1
7638 055462 000000 0
7639 ; ITEM 40
7640 055464 045337 EM40 ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
7641 ; PARITY ERROR LOW BYTE
7642 055466 054256 DH11
7643 055470 055004 DT1
7644 055472 000000 0
7645 ; ITEM 41
7646 055474 045462 EM41 ;ERROR: TEST OF DATA PARITY GEN/CKER FAILED
7647 ; PARITY ERROR HIGH BYTE
7648 055476 054256 DH11
7649 055500 055004 DT1
7650 055502 000000 0
7651 ; ITEM 42
7652 055504 045606 EM42 ;ERROR: NO PARITY TRAP FROM LOC WRITTEN WITH WRONG PARITY
7653 055506 054226 DH7
7654 055510 055020 DT5
7655 055512 000000 0
7656 ; ITEM 43
7657 055514 045676 EM43 ;ERROR: ADDRESS COULD NOT BE MADE A HIT
7658 055516 054226 DH7
7659 055520 055020 DT5
7660 055522 000000 0
7661 ; ITEM 44
7662 055524 045745 EM44 ;ERROR: ADDRESS NOT INVALIDATED BY PARITY TRAP
7663 055526 054226 DH7
7664 055530 055020 DT5
7665 055532 000000 0
7666 ; ITEM 45
7667 055534 046023 EM45 ;ERROR: TAG PARITY ERROR WHEN TESTING TAG P BIT
7668 055536 054725 DH45
7669 055540 055030 DT12
7670 055542 000000 0
7671 ; ITEM 46
7672 055544 046102 EM46 ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING TAG P BIT

```

```

7673 055546 054725 DH45 ;PC=/P ADDH/P ADDL/DATA=
7674 055550 055030 DT12 ;ERRPC,$REG1,$REG2,$REG3
7675 055552 000000 0
7676 ;ITEM 47
7677 055554 046173 EM47 ;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING TAG P BIT
7678 055556 054725 DH45 ;PC=/P ADDH/P ADDL/DATA=
7679 055560 055030 DT12 ;ERRPC,$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 054725 DH11 ;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7684 055570 055004 DT1 ;ERRPC,$REG1,$REG2,$REG3
7685 055572 000000 0
7686 ;ITEM 51
7687 055574 046346 EM51 ;ERROR: LOW BYTE PARITY ERROR WHEN TESTING DATA P BIT
7688 055576 054256 DH11 ;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7689 055600 055004 DT1 ;ERRPC,$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 ;ERRPC,$REG1,$REG2,$REG3
7695 055612 000000 0
7696 ;ITEM 53
7697 055614 046521 EM53 ;ERROR: LOW BYTE PAR. ERROR WHEN TESTING TAG ADDR. BITS
7698 055616 054256 DH11 ;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7699 055620 055004 DT1 ;ERRPC,$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 ;ERRPC,$REG1,$REG2,$REG3
7705 055632 000000 0
7706 ;ITEM 55
7707 055634 046710 EM55 ;ERROR: TEST OF TAG ADDR. BITS FAILED
7708 ; ADDR. COULD NOT BE MADE A HIT
7709 055636 054535 DH27 ;PC=/P ADDH/P ADDL/(TAG) SHOULD=
7710 055640 055030 DT12 ;ERRPC,$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 ;ERRPC,$REG1,$REG2,$REG3
7716 055652 000000 0
7717 ;ITEM 57
7718 055654 047114 EM57 ;ERROR: HIGH BYTE PARITY ERROR WHEN TESTING DATA FIELD
7719 055656 054256 DH11 ;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7720 055660 055004 DT1 ;ERRPC,$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 ;ERRPC,$REG1,$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	;ERRRPC,\$REG1,\$REG2,\$REG3
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	;ERRRPC,\$REG1,\$REG2
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	;ERRRPC,\$REG1,\$REG2,\$REG3
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	;ERRRPC,\$REG1,\$REG2,\$REG3
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	;ERRRPC,\$REG1,\$REG2,\$REG3
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	;ERRRPC,\$REG1,\$REG2,\$REG3
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	055030	DT5	;ERRRPC,\$REG1,\$REG2
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	;ERRRPC,\$REG1,\$REG2,\$REG3
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	;ERRRPC,\$REG1,\$REG2,\$REG3
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	;ERRRPC,\$REG1,\$REG2,\$REG3

7785	056012	000000	0	
7786			ITEM 73	
7787	056014	051010	EM73	;ERROR: DYNAMIC TEST OF CACHE FAILED
7788				; LOC HELD WRONG DATA
7789	056016	054256	DH11	;PC=/P ADDH/P ADDL/DATA IS/DATA SHOULD BE
7790	056020	055004	DT1	;ERRRPC,\$REG1,\$REG2,\$REG3
7791	056022	000000	0	
7792			ITEM 74	
7793	056024	051106	EM74	;ERROR: DYNAMIC TEST OF CACHE FAILED
7794				; TRAP TO 10 OCCURRED
7795	056026	053775	DH1	;PC=/P ADDH/P ADDL/PC OF PE
7796	056030	055004	DT1	;ERRRPC,\$REG1,\$REG2,\$REG3
7797	056032	000000	0	
7798			ITEM 75	
7799	056034	051204	EM75	;ERROR: DYNAMIC TEST OF CACHE FAILED
7800				; LOW BYTE PARITY ERROR
7801	056036	054725	DH45	;PC=/P ADDH/P ADDL/DATA=
7802	056040	055030	DT12	;ERRRPC,\$REG1,\$REG2,\$REG3
7803	056042	000000	0	
7804			ITEM 76	
7805	056044	051304	EM76	;ERROR: DYNAMIC TEST OF CACHE FAILED
7806				; HIGH BYTE PARITY ERROR
7807	056046	054725	DH45	;PC=/P ADDH/P ADDL/DATA=
7808	056050	055030	DT12	;ERRRPC,\$REG1,\$REG2,\$REG3
7809	056052	000000	0	
7810			ITEM 77	
7811	056054	051405	EM77	;ERROR: DYNAMIC TEST OF CACHE FAILED
7812				; TAG PARITY ERROR
7813	056056	054472	DH22	;PC=/P ADDH/P ADDL/TAG FIELD=
7814	056060	055030	DT12	;ERRRPC,\$REG1,\$REG2,\$REG3
7815	056062	000000	0	
7816			ITEM 100	
7817	056064	000000	0	
7818	056066	000000	0	
7819	056070	000000	0	
7820	056072	000000	0	
7821			ITEM 101	
7822	056074	051500	EM101	;ERROR: CACHE CONTROL REG NOT INITIALIZED BY POWER FAIL
7823	056076	054763	DH100	;PC=/DATA=
7824	056100	055062	DT100	;ERRRPC,\$REG1
7825	056102	000000	0	
7826			ITEM 102	
7827	056104	051566	EM102	;ERROR: POWER UP FAILED TO INVALIDATE CACHE
7828	056106	055000	DH107	;PC=
7829	056110	055070	DT107	;ERRRPC
7830	056112	000000	0	
7831			ITEM 103	
7832	056114	051640	EM103	;ERROR: DEVICE ERROR RIT SET WHEN DOING NPR,DATO TO ADDR
7833	056116	054726	DH7	;PC=/P ADDH/P ADDL
7834	056120	055020	DT5	;ERRRPC,\$REG1,\$REG2
7835	056122	000000	0	
7836			ITEM 104	
7837	056124	051733	EM104	;ERROR: CACHE LOC NOT INVALIDATED BY NPR, DATO
7838	056126	054726	DH7	;PC=/P ADDH/P ADDL
7839	056130	055020	DT5	;ERRRPC,\$REG1,\$REG2
7840	056132	000000	0	

7841					
7842	056134	052030	ITEM 105	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	ERRPC,\$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	ERRPC
7856	056162	000000		0	
7857			ITEM 110		
7858	056164	052742		EM110	ERROR: RETRY TO BACKING STORE NOT DONE ON CACHE PARITY
7859	056166	055000		DH107	PC=
7860	056170	055070		DT107	ERRPC
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	ERRPC,\$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	ERRPC,\$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	ERRPC,\$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	ERRPC,\$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	ERRPC,\$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	ERRPC,\$REG1,\$REG2
7896	056252	000000		0	

7897					
7898	056254	053156	ITEM 117	EM117	ERROR: CACHE ALLOCATED DURING RED ZONE TRAP
7899	056256	055000		DH107	PC=
7900	056260	055070		DT107	ERRPC
7901	056262	000000		0	
7902			ITEM 120		
7903	056264	053231		EM120	ERROR: CACHE ALLOCATED DURING KT ABORT
7904	056266	055000		DH107	PC=
7905	056270	055070		DT107	ERRPC
7906	056272	000000		0	
7907			ITEM 121		
7908	056274	053277		EM121	ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
7909					LOC NOT INVALIDATED
7910	056276	054226		DH7	PC=/P ADDH/P ADDL
7911	056300	055020		DT5	ERRPC,\$REG1,\$REG2
7912	056302	000000		0	
7913			ITEM 122		
7914	056304	053416		EM122	ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
7915					PARITY ERROR TAG
7916	056306	054725		DH45	PC=/P ADDH/P ADDL/DATA=
7917	056310	055030		DT12	ERRPC,\$REG1,\$REG2,\$REG3
7918	056312	000000		0	
7919			ITEM 123		
7920	056314	053532		EM123	ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
7921					PARITY ERROR LOW BYTE
7922	056316	054725		DH45	PC=/P ADDH/P ADDL/DATA=
7923	056320	055030		DT12	ERRPC,\$REG1,\$REG2,\$REG3
7924	056322	000000		0	
7925			ITEM 124		
7926	056324	053653		EM124	ERROR: TEST OF MSB ADDRESS (A10) TO VALID BIT FAILED
7927					PARITY ERROR HIGH BYTE
7928	056326	054725		DH45	PC=/P ADDH/P ADDL/DATA=
7929	056330	055030		DT12	ERRPC,\$REG1,\$REG2,\$REG3
7930	056332	000000		0	
7931					
7932					
7933					
7934		000001			

;TEST BUFFER

;END

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC MACY11 27(1006) 09-FEB-77 15:33 PAGE 145
DQKKA.P11 07-FEB-77 11:01 CROSS REFERENCE TABLE -- USER SYMBOLS

[illegible]

		4701	4803*	4917	4957	4963	4972	5347	5546	5548							
B1	001734	641	645*														
B2	001710	636*	643														
CCR	= 177746	286*	354*	646*	896*	1103*	1105*	1107	1142*	1145*	1146	1149*	1162*	1188*			
		1195*	1207*	1210*	1213*	1217	1233*	1236*	1240*	1242*	1247*	1251*	1266*	1269*			
		1275*	1286*	1295*	1301*	1308*	1324*	1327*	1335*	1341*	1346*	1367*	1370*	1371			
		1374*	1384*	1390*	1394	1421*	1457*	1469*	1475*	1484*	1512*	1516*	1517	1520*			
		1524*	1527*	1532*	1544*	1559*	1561*	1567*	1576*	1581*	1618*	1633*	1636*	1641*	1659*		
		1728*	1736*	1745*	1830*	1839*	1841*	1843*	1845*	1851*	1891*	1893*	1896*	1901*			
		1941*	1949*	1981*	2026*	2030*	2054*	2056*	2080*	2090*	2092*	2116*	2135*	2139*			
		2144*	2143*	2164*	2172*	2178*	2185*	2191*	2227*	2233*	2261*	2269*	2326*	2330*			
		2364*	2370*	2393*	2409*	2412*	2418*	2507*	2510*	2524*	2530*	2532*	2564*	2566*			
		2568*	2592*	2639*	2610*	2612*	2614*	2635*	2642*	2648*	2655*	2661*	2697*	2703*			
		2731*	2739*	2796*	2800*	2807*	2844*	2875*	2907*	2978*	2987*	3012*	3019*	3040*			
		3001*	3107*	3117*	3156*	3179*	3179*	3193*	3249*	3289*	3320*	3352*	3420*	3432*			
		3441*	3475*	3481*	3523*	3591*	3598*	3603*	3611*	3645*	3651*	3693*	3761*	3768*			
		3773*	3799*	3806*	3800*	3833*	3840*	3847*	3853*	3859*	3947*	3977*	3992*	3996*			
		4004*	4088*	4096*	4114*	4122*	4126*	4131*	4139*	4214*	4217*	4240*	4246	4262*			
		4265*	4292*	4298*	4316*	4329*	4354*	4360*	4386*	4392*	4456*	4485*	4510*	4519*			
		4590*	4594*	4595	4597*	4599*	4616*	4619*	4626	4627*	4632*	4640*	4658*	4661*			
		4697*	4700*	4737*	4744*	4772*	4779*	4800	4822*	4831*	4906*	4912*	5039*	5048*			
		5192*	5251*														
CDH	= 000106	287*	1796	1877	1927	2321	2457	2791	2970	3087	3232	3415	3573	3743			
		3909	4054	4559	5213												
CDL	= 000106	288*	1785	1868	1916	2314	2465	2784	2963	3075	3240	3400	3565	3735			
		3901	4046	4170	4179	4550	5233										
CER	= 177766	291*	5253														
CR	= 000015	39*	334*	5940	5950	6342	6340	6356	6363	6360	6373	6370	6384	6391			
		6396	6403	6409	6414	6420	6427	6432	6438	7170	7205	7217	7230	7250			
		7291	7305	7318	7332												
CREG1	001212	444*	690*	696	711*	743*	785*	834*	4914	5052	5373	5385*					
CREG2	001214	445*	695*	741*	746	781*	782*	836*	837*	841	4782*	4796*	5386*	5390*			
		5440	5490	5512	5528												
CPEG3	001216	446*	692*	5384*													
CREG4	001220	447*	688*	5380*													
CREG5	001222	448*	686*	5387*													
CREG6	001224	449*	684*														
CR1F	= 000200	40*	5911	5950													
CTAG	= 000107	289*	3095														
DDISP	= 177570	46*	418	600													
DH1	053775	7347*	7476	7795													
DH100	054763	7438*	7823														
DH107	055000*	7441*	7820	7854	7859	7899	7904										
DH11	054256	7379*	7516	7552	7606	7612	7630	7636	7642	7648	7683	7688	7698	7703			
		7714	7719	7729	7741	7789											
DH12	054336	7387*	7521														
DH16	054376	7393*	7541														
DH2	054036	7353*	7481	7486	7491												
DH21	054433	7398*	7557														
DH22	054472	7404*	7562	7567	7771	7813											
DH27	054535	7410*	7588	7709													
DH30	054601	7417*	7594	7600	7618	7693	7724										
DH35	054660	7425*	7624														
DH45	054725	7432*	7660	7673	7678	7747	7753	7759	7777	7783	7801	7807	7916	7922			
		7928															
DH5	054105	7360*	7496	7501	7502												

[illegible]

		4671	4681	4722	4733	4941	4947	4986	4992	5095	5101	5114	5120	
HMR0	= 000001	320#												
HMR1	= 000002	329#	1163											
HMR2	= 000004	330#	1170	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	2802	2891	3169	3181	3327	3336	3492	3506	3662	3676
		3825	3828	3980	3983	4100	4105	4233	4283	4345	4614	4671	4681	4722
		4733	4941	4947	4986	4992	5095	5101	5114	5120				
HMR3	= 000010	331#	1176											
HMR4	= 000020	332#	1182											
HMR5	= 000040	333#	1193											
HRK05	034226	745	5411#											
HRP03	034460	787	5459#											
HT	= 000011	37#	5909	5950										
HTU10	034714	833	5508#											
HUBFN	034046	697	872	4975	5140	5358	5372#							
HUBEO	034174	713	5107	5399#										
IOTVCC	= 000020	132#	5#1*	582*										
IVEC	001226	450#	683*	874	876*	4977	5142	5144*	5360	5362*				
KDPA0#	172360	275#												
KDPA1#	172362	276#												
KDPA2#	172364	277#												
KDPA3#	172366	278#												
KDPA4#	172370	279#												
KDPA5#	172372	280#												
KDPA6#	172374	281#												
KDPA7#	172376	282#												
KDPA0#	172320	253#												
KDPA1#	172322	254#												
KDPA2#	172324	255#												
KDPA3#	172326	256#												
KDPA4#	172330	257#												
KDPA5#	172332	258#												
KDPA6#	172334	259#												
KDPA7#	172336	260#												
KIPAR0#	172340	264#	962	4669*	4679*	5288	5334	5752						
KIPAR1#	172342	265#												
KIPAR2#	172344	266#												
KIPAR3#	172346	267#	4328											
KIPAR4#	172350	268#	914*	916	925*	926	933*	934	940*	941	989*	997*	999*	1008*
		1014*	1021*	1027*	1036*	1044*	1052*	1054*	1429	1441	1443*	1446*	1447	1449
		1451	1460	1470	1620*	1630	1647	1688	1697	1722*	1733*	1734	1739*	1774
		2853*	3298*	3966*	3988*	4328*	4459	5059*	5060	5062*	5063	5105	5109	
KIP														

LF	= 000012	38#	335#	5944	5950	6342	6348	6356	6363	6368	6373	6378	6384	6391
		6396	6403	6409	6414	6420	6427	6432	6438	6443	6448	6453	6458	6463
LOADD	= 000102	7291	7305	7318	7332									
		293#	1070	1676	1707	1765	2288	2434	2758	2927	3060	3209	3372	3543
LOC	= 000200	3713	3878	4023	4158	4495	4538	4854	5202					
MED	= 076600	348#	356											
		297#	865	868	1066	1069	1080	1083	1550	1553	1586	1589	1665	1668
		1675	1678	1683	1699	1706	1709	1751	1754	1759	1764	1769	1784	1795
		1854	1857	1867	1876	1904	1907	1915	1926	1952	1955	1961	2063	2066
		2099	2102	2151	2154	2275	2278	2282	2287	2290	2303	2313	2320	2421
		2424	2428	2433	2440	2456	2464	2539	2542	2575	2578	2622	2625	2745
		2748	2752	2757	2760	2773	2783	2790	2910	2913	2921	2926	2929	2952
		2962	2969	3043	3046	3054	3059	3062	3074	3086	3094	3196	3199	3203
		3208	3215	3231	3239	3355	3358	3366	3371	3374	3397	3407	3414	3526
		3529	3537	3542	3545	3551	3564	3572	3583	3696	3699	3707	3712	3715
		3721	3734	3742	3753	3815	3818	3862	3865	3872	3877	3880	3889	3900
		3908	4007	4010	4017	4022	4025	4034	4045	4053	4142	4145	4152	4157
		4160	4169	4178	4186	4225	4228	4275	4278	4337	4340	4489	4494	4499
		4502	4522	4525	4532	4537	4540	4549	4550	4566	4606	4609	4719	4730
		4799	4815	4818	4834	4837	4844	4848	4853	5196	5201	5204	5212	5223
		5232	5240	5243	5260	5263	5794	5797						
MMR0	= 177572	308#	861*	915*	1059*	1077*	1468*	1478*	1487*	1498*	1629*	1808*	2074*	2995*
		3319*	3439*	3976*	4059*	4331*	4349*	4464*	4574*	5053*	5139*	5382*	5416*	5461*
		5474*	5521*											
MMR2	= 177576	309#												
MMVEC	= 000250	143#												
MSG1	040542	616	6327*											
MSG10	041507	822	6414*											
MSG11	041546	827	6420*											
MSG12	041614	855	5464	6427*										
MSG13	041645	753	850	5379	5419	5477	5524	6432*						
MSG14	041705	5380	5420	5465	5478	5525	6438*							
MSG2	040625	4785	6336*											
MSG3	040670	635	6342*											
MSG4	041202	642	671	733	775	807	6378*							
MSG5	041242	660	6384*											
MSG6	041312	670	708	725	767	799	6391*							

[illegible]

MD-11-DQKKA-A 11/6X CACHE DIAGNOSTIC		MACY11 27(1996)		09-FEB-77 15:33		PAGE 153	
DQKKA,P11 07-FEB-77 11:01		CROSS REFERENCE TABLE -- USER SYMBOLS					
SW5	= 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	1699	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#					
TPAPVE=	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	1480	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	3294	3442	3475#		
TST26	022000	3291	3294	3442	3475#		
TST27	024000	3477	3612	3645#			
TST3	004320	1106	1142#				
TST30	024566	3647	3709#				
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#	
TYPOS	= 104405	5171	6274#				
TYPE	= 104401	616	635	642	660	670	671 708 725

TYPOC = 104402	6163	6166	6170	6234	6236	6270#	6314
TYPON = 104404	5681	5705	6271#				
TYPOS = 104403	6273#						
T01L01 004242	6272#						
T01L02 004264	1109	1114#					
T01L03 004252	1115	1120#					
T02L01 004404	1116#	1123	1127				
T02L02 004462	1148	1155#					
T02L03 004504	1164	1170#					
T02L04 004526	1171	1176#					
T02L05 004550	1177	1182#					
T02L06 004454	1183	1188#					
T04L01 005144	1167#	1174	1180	1186	1190		
T04L02 005220	1274	1281#					
T04L03 005300	1285	1292#					
T05L01 005472	1293	1303#					
T05L02 005514	1330	1341#					
T06L01 006260	1334	1339	1346#				
T06L02 006366	1437#	1494	1496				
T06L03 006276	1440	1454	1457#				
T06L04 006620	1438	1441#	1491				
T06L05 006316	1448	1450	1498#				
T06L06 006324	1442	1446#					
T06L07 006360	1444	1447#					
T06L08 006440	1452	1456#					
T06L09 006522	1469#	1476	1485				
T06L10 006566	1471	1481#					
T06L11 006610	1474	1483	1490#				
T06L12 006266	1492	1495#					
T07L01 007354	1435	1439#					
T07L02 007650	1626	1656#					
T07L03 007254	1632	1725#					
T07L04 007232	1635#						
T07L05 010214	1630#	1650	1691	1715	1723		
T07L06 007520	1654	1695	1720	1735	1780	1791	1802
T07L07 007636	1674	1697#					1808#
T07L08 007734	1674	1719	1722#				
T07L09 007674	1704	1732					
T07L10 010076	1732	1742#					
T07L11 010136	1734#	1740	1777	1788	1799	1805	
T07L12 010176	1768	1782#					
T08L01 010350	1783	1793#					
T08L02 011052	1794	1804#					
T08L03 010432	1833	1851#					
T08L04 010460	1849	1872	1881	1899	1921	1932	1945
T08L05 010466	1863	1874#					1970
T08L06 010262	1865	1883#					1978#
T08L07 010546	1875	1887#					
T08L08 010630	1836#	1885					
T08L09 010504	1887	1901#					
T08L10 010662	1913	1923#					
T08L11 010676	1890#	1918	1929	1937			
T08L12 010740	1924	1934#					
T08L13 010706	1935	1939#					
T08L14 011032	1939	1949#					
T08L15 011032	1941#	1947	1967	1974			
	1966	1972#					

T08L16 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#					

T13H07	021312	3345	3380	3403	3411	3418	3430	3439#
T13H08	021056	3378	3382#					
T13H09	021072	3384	3387#					
T13H10	021156	3388	3405#					
T13H11	021122	3392	3395#					
T13H12	021202	3406	3413#					
T13H13	021246	3426#	3437					
T13H15	020732	3347	3348#					
T13L01	016266	2852	2907#					
T13L02	016124	2872#	2905					
T13L03	016156	2881#	2896					
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	3034#					
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			
T15H01	024210	3648	3693#					
T15H02	024466	3664	3678	3761#				
T15H03	024510	3667	3767#					
T15H04	024560	3687	3725	3738	3746	3759	3765	3780#
T15H05	024034	3651#	3691					
T15H06	024334	3720	3727#					
T15H08	024350	3729	3732#					
T15H12	024170	3686	3688#	3690				
T15H13	024374	3733	3740#					
T15H14	024420	3741	3748#					
T15H15	024526	3681	3772#					
T15H16	024542	3770	3775#					
T15H17	024112	3666	3668#					
T15H18	024154	3680	3682#					
T15H21	024064	3661#	3669					
T15H22	024126	3675#	3683					
T15L01	022210	3478	3523#					
T15L02	022166	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	4130#					
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	4990	4993	5002#		
T21L07	032024	4952	4966#				
T21L08	032000	4954	4968#				
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	2520	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	2050	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	

T24L19	012062	2037	2039#				
T24L20	012044	2035#	2040				
T24L22	012130	2046	2048	2051	2054#		
T25L01	031036	4714	4728#				
T25L02	031064	4723	4734	4737#			
T25L03	031030	4724#	4735	4745			
T25L04	030730	4706#	4725	4740			
T27L01	026510	4218	4222#				
T27L02	026564	4234	4240#				
T27L03	026610	4236	4246#				
T27L04	026554	4237#	4244	4250			
T28L01	026706	4266	4272#				
T28L02	026772	4284	4292#				
T28L03	027016	4286	4298#				
T28L04	026752	4287#	4296	4300			
T29L01	027152	4323	4334#				
T29L02	027242	4346	4354#				
T29L03	027266	4348	4360#				
T29L04	027216	4349#	4358	4362			
T30L01	024646	3802	3812#				
T30L02	025020	3810	3853#				
T30L03	024774	3826	3847#				
T30L04	024752	3829	3840#				
T30L05	024730	3830#	3845	3851	3857	3885	3896
T30L06	025042	3823	3859#				3905
UBEAPT	002050	627	629	678#			3913
UDPAR0=	177660	187#					
UDPAR1=	177662	188#					
UDPAR2=	177664	189#					
UDPAR3=	177666	190#					
UDPAR4=	177670	191#					
UDPAR5=	177672	192#					
UDPAR6=	177674	193#					
UDPAR7=	177676	194#					
UDPDR0=	177620	165#					
UDPDR1=	177622	166#					
UDPDR2=	177624	167#					
UDPDR3=	177626	168#					
UDPDR4=	177630	169#					
UDPDR5=	177632	170#					
UDPDR6=	177634	171#					
UDPDR7=	177636	172#					
UIPAR0=	177640	176#					
UIPAR1=	177642	177#					
UIPAR2=	177644	178#					
UIPAR3=	177646	179#					
UIPAR4=	177650	180#					
UIPAR5=	177652	181#					
UIPAR6=	177654	182#					
UIPAR7=	177656	183#					
UIPDR0=	177600	154#					
UIPDR1=	177602	155#					
UIPDR2=	177604	156#					
UIPDR3=	177606	157#					
UIPDR4=	177610	158#					
UIPDR5=	177612	159#					

[illegible]

MACY11 27(1006) 09-FEB-77 15:33 PAGE 166
CROSS REFERENCE TABLE -- MACRO NAMES

JP	529#	1426	1623	1986	2170	2849	3118	3294	3442	3612	3952	4452	4911	
COMMEN	139#													
CSPW	532#	4717	4729											
ENDCOM	139#													
ERROR	33#	993	1075	1111	1117	1152	1167	1220	1248	1254	1278	1289	1300	1313
	1344	1377	1387	1397	1477	1486	1523	1538	1581	1651	1692	1716	1778	1789
	1806	1847	1870	1879	1897	1919	1930	1968	1975	2175	2181	2188	2194	2297
	2316	2323	2334	2415	2446	2459	2467	2472	2645	2651	2658	2664	2767	2778
	2793	2804	2934	2957	2965	2972	2983	3067	3070	3090	3103	3113	3221	3234
	3247	3252	3379	3402	3410	3417	3428	3554	3567	3575	3588	3594	3608	3724
	3745	3758	3764	3778	3843	3850	3856	3884	3894	3903	3911	4000	4029	4039
	4056	4118	4135	4164	4172	4181	4191	4243	4249	4295	4299	4357	4361	4506
	4544	4552	4561	4571	4623	4629	4635	4675	4741	4810	4856	4859	4999	5007
	5127	5135	5138	5215	5219	5228	5235	5257						5009
ESCAPE	139#													
GET	530#	1675	1706	1764	1784	1795	1867	1876	1915	1926	1961	2287	2290	2313
	2433	2456	2464	2757	2760	2783	2789	2926	2929	2962	2968	3059	3062	3074
	3093	3208	3231	3239	3371	3374	3407	3413	3542	3545	3551	3564	3572	3712
	3721	3734	3742	3877	3880	3900	3907	4022	4025	4045	4052	4157	4160	4169
	4494	4537	4540	4549	4558	4844	4853	5201	5204	5212	5231			4178
GETPRI	139#													
GETSWR	139#													
GHI	533#	1678	1709	1759	2202	2428	2752	2921	3054	3203	3366	3537	3707	3872
	4152	4499	4532	4848	5196									4017
GTA	534#	1693	1699	1769	2303	2440	2773	2952	3215	3397	3583	3753	3889	4034
	4565	5222												4185
JP	528#	1426	1623	1986	2170	2849	3118	3294	3442	3612	3952	4452	4911	
MSG1	536#	882												
MSG10	543#	1317												
MSG11	544#	1401												
MSG12	545#	1502												
MSG13	546#	1599												
MSG14	547#	1813												
MSG15	548#	1995	2479											
MSG16	549#	2200	2670											
MSG17	550#	2340	3126											
MSG2	537#	1091												
MSG20	551#	2815	3260											
MSG21	552#	3000												

MSG42	569#	3784																	
MSG43	569#	1995																	
MSG5	540#	1355																	
MSG6	541#	1225																	
MSG7	542#	1258																	
MULT	139#																		
NEWTST	139#	880	1089	1129	1200	1223	1256	1315	1353	1399	1500	1597	1811	1993	2198				
	2338	2477	2668	2813	2998	3124	3258	3449	3619	3782	3917	4065	4205	4252	4302				
	4364	4580	4646	4684	4747	4882	5012												
POP	139#	5857	5996	5997	6228	6305	6306												
PUSH	139#	5816	5957	5959	5980	6202	6286	6292											
RFPOR	139#																		
SAV	525#	1150	1217	1276	1287	1296	1309												
SCOPE	34#	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	5158											
SFTPRI	139#																		
SETTRA	6262#	6271	6272	6273	6274	6277	6278	6279											
SETUP	139#	572																	
SKIP	139#	1061	1194	1216	1245	1249	1279	1290	1306	1334	1339	1347	1388	1393	1479				
	1488	1594	2410	2640	2808	3038	4061	4195	4238	4290	4320	4352	4672	4676	4726				
	4743	4776	4829	4976	4980														
SKT	535#	898	1106	1144	1209	1235	1268	1326	1369	1423	1514	1620	1832	2028	2229				
	2366	2509	2699	2846	3014	3152	3291	3477	3647	3801	3949	4090	4216	4264	4318				
	4388	4592	4660	4699	4774	4908													
SLASH	139#	618	621	656	658	700	702	716	718	758	760	790	792	5188	5190				
	5248	5250	5269	5271	5309	5311	5321	5324	5337	5340	5353	5356	5367	5370	5394				
	5397	5406	5409	5454	5457	5503	5506	5542	5545										
SPACE	139#																		
STARS	27	139#	337	347	360	362	369	382	457	569	880	895	1089	1102	1129				
	1141	1200	1206	1221	1232	1256	1265	1315	1323	1353	1366	1399	1420	1500	1511				
	1597	1617	1811	1829	1993	2025	2198	2226	2338	2363	2477	2506	2668	2696	2813				
	2843	2998	3011	3124	3149	3258	3288	3449	3474	3619	3644	3782	3798	3917	3946				
	4065	4087	4205	4213	4252	4261	4302	4315	4364	4385	4580	4589	4646	4657	4684				
	4696	4747	4771	4863	4879	4882	4905	5012	5038	5151	5554	5613	5667	5806	5873				
	5952	6009	6086	6092	6121	6188	6241	6282	6298	6323	7455	7931	7933						
SWP	531#	1530	1541	1564	1573	1639	1656	1725	1742	1978	2161	2266	2632	2736	3830				
	4617	4637																	
SWRSU	139#	595#																	
TPMTRP	6262#																		
TYPBIN	139#																		
TYPDEC	139#	5170																	
TYPNAM	139#																		
TYPNUM	139#																		
TYPOCS	139#																		
TYPOCT	139#	5679	5703																
TYPTXT	139#																		
UNLK	526#	863	1078	1548	1583	1663	1749	1852	1902	1950	2060	2096	2149	2273	2419				
	2536	2572	2620	2743	2908	3041	3194	3353	3524	3694	3812	3860	4005	4140	4223				
	4273	4335	4497	4520	4604	4812	4832	5237	5258	5792									
ESCA	139#																		
ENWT	139#	880	1089	1129	1200	1223	1256	1315	1353	1399	1500	1597	1811	1993	2198				
	2338	2477	2668	2813	2998	3124	3258	3449	3619	3782	3917	4065	4205	4252	4302				
	4364	4580	4646	4684	4747	4882	5012												
ESSET	6262#	6271	6272	6273	6274	6277	6278	6279											
ESSETM	611#																		

SSSKIP	139#	1061	1194	1216	1245	1249	1279	1290	1306	1347	1388	1393	1479	1488	1594				
	2410	2640	2808	3038	4061	4195	4238	4290	4320	4352	4672	4676	4726	4743	4776				
	4829	4976	4980																
.EQUAT	2#	29																	
.HEADE	2#																		
.KIT1	2#	139																	
.SETUP	2#	337																	
.SWRHI	2#	13																	
.SWPLO	25#																		
.SACT1	2#	380																	
.SAPT8	455																		
.SAPTH	2#	358																	
.SAPTY	2#	5950																	
.SCATC	2#	338																	
.SCMTA	2#																		
.SEOP	2#	5149																	
.SERRO	2#	5611																	
.SEPR	2#	5665																	
.SPOWE	2#	6280																	
.SROOC	2#	6186																	
.SREAD	2#	6084																	
.SSCOP	2#	5552																	
.STRAP	2#	6239																	
.STYPD	2#	5804																	
.STYPE	2#	5871																	
.STYPO	2#	6007																	

. ABS. 056334 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DSK2:DQKKA,DSK2:DQKKA/SOL/CRF=DSK2:DQKKA.P11
RUN-TIME: 25 22 2 SECONDS
RUN-TIME RATIO: 487/51=9.5
CORE USED: 34K (68 PAGES)