SUN Microsystems

Sun2 Memory Diagnostic

User's Document

Gale Snow
revised by
Larry Laskowski

September 5, 1985

Revision A

Contents

1. Preface

This document uses the contents of the old "mem.diag.doc" and presents it in the new Diagnostics Document format. It describes the memory diagnostic in use on Sun-2 products. References may exist to the model(s) 120 and/or 120', but these may be removed over time in the interest of machine independence and generality.

1.1. Purpose

The purpose of this document is to describe the features and user interface to the latest memory diagnostic for the 120/170 product line.

1.2. Audience

The audience includes all users (internal) only including: design engineers, manufacturing test technicians, field service technicians, and is also intended as a starting document for the technical writing group.

2. Revision History

Revision A July 24, 1985 Initial release of this document.

3. Glossary

FRU - Field Replaceable Unit.

4. Introduction

The described program, *mem.diag* is a diagnostic tool written to help service personnel diagnose and repair memory problems in Sun-2 products.

5. Requirements

5.1. Hardware Requirements

The minimum configuration of hardware required to test memory boards is as follows:

1). multibus card cage, power supply

2). processor board

3). memory board(s)

4). terminal, one of : sun video console (need video bd) or a televideo-like terminal

5). boot device, one of : local disk (need disk ctlr), local tape (need tape ctlr), or a remote disk via ethernet (need ethernet)

Note: this memory test runs out of ram and is not relocatable. The diag is loaded at location 0x4000 and occupies memory up to 0xa800. So in the first meg of memory only addrs 0xa800-0xfffff is tested. For a board test station it is recommended that a known good memory board is used and kept for the first meg of memory, while the board under test should be configured for the second (third and fourth) meg of memory.

5.2. Software Requirements

mem.diag requires standard power up diag / monitor / boot ROMs (firmware) as well as the diagnostic itself : *mem.diag*

5.3. Firmware Requirements

6. General Information

6.1. Hardware-Related Information

memory board organization (cross ref theory of ops)

a. board 1 megabyte = 0x100000 = 1048576 bytes

b. 8 rows @ 0x20000 = 131072 bytes (128K)

physical rows are named 0 - 7 as follows

row 0 : locations U1200 - U1218 } rows 0-1 addrs 0x00000 - 0x3ffff
row 1 : locations U1220 - U1238 }
row 2 : locations U1300 - U1218 } rows 2-3 addrs 0x40000 - 0x7ffff
row 3 : locations U1320 - U1238 }
row 4 : locations U1400 - U1218 } rows 4-5 addrs 0x80000 - 0xbffff
row 5 : locations U1420 - U1238 }
row 6 : locations U1500 - U1218 } rows 6-7 addrs 0xc0000 - 0xfffff
row 7 : locations U1520 - U1238 }

note that addresses within a given row range don't stay within physical row because of a9, but switch back and forth between 2 rows every 0x200 bytes (eg addrs 0x000-0x1ff in row 1, addrs 0x200-0x3ff in row 0, 0x400-0x5ff in row 1, etc, addrs 0x40000-0x401ff in row 3, addrs 0x40200-0x403ff in row 2, addrs 0x40400-0x405ff in row 3, etc)

c. 18 64Kx1 chips in each row for 16 bits (1 word or 2 bytes) + 2 parity bits for upper/lower

byte of each word

d. board select addr A20, A21, A22 compared with dip switch, memory board can be configured for any one of the following address ranges:

0x000000 - 0x0fffff (switch 1 on, everything else off)
0x100000 - 0x1fffff (switch 2 on, everything else off)
0x200000 - 0x2fffff (switch 3 on, everything else off)
0x300000 - 0x3fffff (switch 4 on, everything else off)
0x400000 - 0x4fffff (switch 5 on, everything else off)
0x500000 - 0x5fffff (switch 6 on, everything else off)
0x600000 - 0x6fffff (switch 7 on, everything else off)
0x700000 - 0x7fffff (switch 8 on, everything else off)

6.2. Software-Related Information

This diagnostic runs standalone, UNIX† is not present within the system when the diagnostic is running, hence the name standalone. This is necessary since UNIX is rather picky about allowing free access to resources. All i/o support is provided by the standalone libraries linked to the diagnostic.

mem.diag features include:

- usefulness to engineering, manufacturing, field service,
- flexibility for engineers (in terms of sequencing and parameterization), and
- default parameterization and sequencing for manufacturing (ie hit one key to start recommended test auto sequencing and default parameters).

7. Operating Instructions

7.1. Loading And Starting

Turn on the system. After power-on ROM diagnostics are run, the system automatically begins booting UNIX. Break out of the boot and return to the ROM monitor by typing L1-a (hold down the L1 key while typing a) on the sun2 console or by typing <break> on the televideo-type terminal.

At this point type k1 to the ROM monitor to reset memory maps to initial state. We are now ready to boot the memory diagnostic. Remember there are 3 ways of booting:

† UNIX is a trademark of Bell Laboratories.

7.1.1. from local disk

Assuming the diagnostic *mem.diag* exists on your local disk in the /pub/stand (fileserver) or the /stand (standalone) directory, the diagnostic is loaded by typing

> b stand/mem.diag

to the ROM monitor prompt

7.1.2. from remote disk via ethernet

Assuming the network fileserver has a partition reserved for the system under test (ie it is a legitimate client) and that the diagnostic exists in the /pub/stand on the fileserver, the diag is loaded by typing

> b stand/mem.diag

to the ROM monitor prompt

If the system under test is not a client of the fileserver where the diagnostic lives in the /pub/stand directory, the use the following boot command to the ROM monitor

>b ec(,<fileserver_host_net_#>)stand/mem.diag

7.1.3. from tape

(I've still got to verify this info)

Assuming you have a tape with a bootable image of the memory diagnostic on tape, use the following commands to the ROM monitor to load it.

b st() <from scsi tape 1/4" cartridge>

b ar() <from archive tape 1/4" cartridge>

b mt() <from tape master 1/2" tape reel>

Note: No matter how the diagnostic is booted, once it is loaded, it is automatically started.

7.2. User Interface

7.2.1. Command Line

The menu is displayed whenever a single <cr> is typed at the command line prompt (which is the first thing one sees after booting the diag).

Command : (this is the prompt)

(This is so you're not forced to wade thru a menu if you know what you're doing.)

The menu is displayed as follows:

```
Sun 120' Memory Test REV 1.9 10/11/84

Start of memory under test : 0xb800
Size  of memory under test : 0x1f4800

Command selections:

A  -  set default address
S  -  set default size
f  -  fill memory
d  -  display memory
t  -  default test
c  -  constant pattern test
r  -  random pattern test
a  -  address test
u  -  uniqueness test
x  -  checker pattern test
g  -  diagonal pattern test
m  -  byte, word, long mode
p  -  parity check
w  -  wait on error
s  -  scopeloop on error
e  -  error message mode
E  -  display error log
l  -  loop
h  -  help
q  -  quit

Command :
```

The following is displayed for "help". (type h at the command prompt) (Note the third column with the parameters required for each command indicated)

```
A       set default address  A address
S       set default size     S size
f       fill memory          f address size pattern
d       display memory               d address size
t       default test         t
c       constant pattern test        c address size pattern passcnt
r       random pattern testr address size seed passcnt
a       address test         a address size passcnt
u       uniqueness test              u address size incr passcnt
x       checker pattern testx address size pattern passcnt
g       diagonal pattern test        g address size passcnt
m       byte, word, long mode        m [0|1|2]
p       parity check                 p [0|1]
w       wait on error                w [0|1]
s       scopeloop on error   s [0|1]
e       error message mode e [0|1|2]
E       display error log    E
l       loop                         l loopcount
h       help                         h
q       quit                         q
```

7.2.2. Modes

Before using tools provided by this diagnostic, and before running any of the tests, there are several operating modes that should be considered. these modes are explained below:

7.2.2.1. byte, word, long data mode

to set the mode to byte:	m 0
to set the mode to word:	m 1 (or just m)
to set the mode to long:	m 2
default :	m 1

applies to the following tests:
constant, random, address, uniqueness
also applies to the fill tool

in the initial state the data mode is set to word

7.2.2.2. parity

to turn parity on:	p 1 (or just p)
to turn parity off:	p 0
default :	p 1

in the initial state parity checking is enabled

7.2.2.3. errors : continue, wait or scopeloop

to wait (or stop) on error:	w 1	
no wait		w 0
default :	w 0	
to scopeloop on error:		s 1
no scopeloop:		s 0
default :	s 0	

after on error has occurred with either of these modes on, the test may be continued by typing any key.

scopeloop continually writes/reads the data causing the failure to the address where the failure occurred (also taking into account the byte, word, or long mode of the test)

with scopeloop on, when an error occurs (no matter what the error message mode), the following is displayed:
.. looping
(<obs>)+ *or* (<obs>)-
where <obs> is the observed value, and the + message is printed when the observed value matches the expected value and the - message is printed when the observed and expected values don't match.

in the initial state, wait on error off
in the initial state, scopeloop on error off

7.2.2.4. error messages on or off

to print or display error messages:	e 1	
to turn off all message output:		e 0
default :	e 1	

error messages are printed in the initial state

7.2.3. Parameters

In this section, some of the general rules about setting parameters for the various tests provided by this diagnostic are described.

7.2.3.1. address & size

All of the tools and the tests require address and size parameters. (the address being the beginning address (low) of memory to be tested and the size being the number of bytes in the block of memory to be tested.) The address and size parameters are always specified in hex notation. (eg 0x100000). The size parameter always specifies a number of bytes (no matter what the data mode - see further discussion under modes - byte, word, long)). If these parameters are not supplied (see further discussion of this technique under the special notations heading below) then the default address and size are used. The default address and size are the address and size

printed at the head of the menu, set up in the following ways:

1. at the beginning of the diag (right after boot) the page boundary (pages are 2K = 0x800 bytes) following the end of the diagnostic itself is taken to be the start address of memory under test, while the size is taken to be the total memory size of the system (determined and offered by the diag/mon/boot ROM) minus the start address. *or*

2. the addr and size parameters may also be set using the commands 'A' and 'S'.

7.2.3.2. passcnt

The passcnt parameter specifies how many times to run through the test, before continuing on to the next test specified on the command line or before prompting for another command. It is always specified in decimal and it is always the last parameter supplied to the tests.

7.2.3.3. others

Some of the tools and tests require an additional parameter that has to do with the data used. If present, these parameters are always the 3rd paramter supplied.

pattern

The pattern parameter is the data to be written and read from each memory location in the block of memory to be tested. It is always specified in hexadecimal notation. The pattern parameter is used in the following tools or utilities and tests: fill, constant, checker

seed

The seed parameter is the seed used to generate the random sequence of data for the random pattern test. It is always specified in decimal notation.

incr

The incr parameter is the increment used to generate a series of numbers used in the uniqueness test. The series is as follows {n, 2n, 3n, 4n, ... } where n=incr. It is always specified in hexadecimal notation.

7.2.3.4. special notations

separator';'

The separator is used between commands at the command line prompt. It must be isolated on the line (i.e. surrounded by spaces). It is this feature that allows tests to be flexibly sequenced. There is a further discussion on sequencing below.

for example:

Command : c 8000 1000 aaaa 2 ; c 8000 1000 5555 2

default '.'

This symbol is used as a place holder to indicate default values for some parameters while specifying others occurring later in the command line.

for example:

Command : c . . aaaa 3

forever'*'

Forever is approximately infinty or 0xffffffff or 4294967295 (decimal) times. Use the forever symbol for the passcnt parameter to loop on a single test "forever".

for example:

Command : c 100000 100000 ffff *

null <nothing>

Null is simply nothingness if a parameter is not supplied then the default values are used.

for example:

Command : c 100000 100000 or simply

Command : c

Note : For more information on the default values see the section on the specific test as well as the section on default parameters below.

7.2.4. Tools

The use of the tools, fill and display, is described.

7.2.4.1. fill

Command usage is : f addr size pattern

The fill command allows one to fill a specified block of memory with a specified pattern. Memory can be filled with bytes, words, or long words of the given data pattern depending on the setting of the data mode of the diagnostic (see the discussion of modes - byte, word, long below).

example:

Command : f 80000 80000 0

Clears the 1/2 megabyte of memory at 0x80000 to 0.

defaults:

f daddr dsize fffffff

7.2.4.2. display

Command usage is : d addr size

The display command allows one to display a specified block of memory on the console or terminal. Data is always (no matter what the data mode) read and displayed a byte at a time. Each line of the display contains the hexadecimal address (always a multiple of 0x10 except for the first line if the specified block doesn't begin on a multiple of 0x10 boundary) followed by 16

bytes of data grouped in long words (by 4s). While there is no paging of the display, ^s pauses the command and display for inspection of the data. After a ^s, any key typed causes the command and display to continue. Any key other than ^s interrupts the command and returns (the user) to the command line prompt.

example:

Command : d 100000 20

100000: 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f

100010: 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f

Displays 32 bytes of data starting at 0x100000 as shown.

defaults:

d daddr dsize

7.2.5. Test Descriptions

7.2.5.1. constant pattern

Command usage is: c addr size pattern passcnt

The constant pattern test tests the specified block of memory using the specified data pattern. First the memory block is filled with data, then it is read back and compared with the specified data pattern. If the data read from an address location does not match the original data pattern an error is flagged. This test can be run in byte, word, or long mode (see discussion under mode - byte, word, long).

example :

Command : c 100000 100000 aaaa 10

Tests the single megabyte of memory at 0x100000 using the data pattern 0xaaaa. The test is repeated 10 times.

defaults:

c daddr dsize 55555555 1

7.2.5.2. random pattern

Command usage is: r addr size seed passcnt

The random pattern test tests the specified block of memory using a sequence of random numbers generated from the specified seed. The random number generator is the same as that in the 'C' run-time libraries. First the block of memory is filled with data, the random sequence is reseeded, and the data is then read back and compared with data that was originally written. If the data read from an address location does not match the original data pattern an error is flagged. This test can be run in byte, word, or long mode (see discussion under mode - byte, word, long).

example:

Command : r 10000 400 3 5

Tests the 1k block of memory at 0x10000 using a random series of numbers seeded by the number 3 as data. The test is repeated 5 times.

defaults:

r daddr dsize 7 1

7.2.5.3. address

Command usage is: a addr size passcnt

The address test tests the specified block of memory using the low order bits of the address of each location as data. The number of bits used depends on the data mode set up as this test can be run in byte, word, or long mode (see discussion under mode - byte, word, long). There are two passes through the block of memory, the first pass uses the lower address bits as is, the second pass uses the complement of the lower address bits (see example for further clarification).

example:

Command : a 100000 100000 1

Tests the 1m block of memory at 0x100000 using the low order bits of the address of each location as data as well as the compliment of those bits. The test executed only one time. For the sake of clarity, the series of numbers used as test data is detailed below for this example for each of the modes:

byte

{00, 01, 02, 03, 04, ... } and {ff, fe, fd, fc, fb, ... }

word

{0000, 0002, 0004, 0006, 0008, ... } and {ffff, fffd, fffb, fff9, fff7, ... }

long

{00000000, 00000004, 00000008, 0000000c, ... } and {fffffff, ffffffb, ffffff7, ffffff3, ... }

defaults:

a daddr dsize 1

7.2.5.4. uniqueness

Command usage is: u addr size incr passcnt

The test for address uniqueness tests the specified block of memory using the sequence {incr, 2 * incr, 3 * incr, 4 * incr, ... } for the test data. This test can be run in byte, word, or long mode (see discussion under mode - byte, word, long).

example:

Command : u c0000 40000 5 100

Tests the 1/4m block of memory at 0xc0000 using the series {5, a, f, 14, ... } as data. The test repeated 100 times.

defaults:

u daddr dsize 1 1

7.2.5.5. checker

Command usage is: x addr size pattern passcnt

Checkertest writes patterns of pattern and ~pattern in a series of write-read scans as follows: Pass 0 writes every other word with the alternating patterns. Pass 1 writes two words of each in sequence every four words. Pass 2 writes four words of each in sequence every eight words. And so on until the cell size is half the block size.

The checker test requires a number of write-read scans over the block of memory under test (which explains why it takes so long to run!). The data used is an alternating sequence of pattern and ~pattern (the complement of pattern), and hence the name checker (short for checker board). The test scans are as follows:

the first scan writes alternating words of pattern and ~pattern throughout the block of memory,

the second scan writes two words of pattern and two words of ~pattern (ditto)

the third writes 4 words of pattern and 4 of ~pattern

the fourth 8 pattern then 8 ~pattern and so on

the last scan will write size/2 words of pattern and size/2 words of ~pattern

(how do you explain that in words?!) (of course each write scan is followed by a read scan checking for the pattern that was just written flagging errors where something else is found)

the following equation applies:

size = 2 ^ n

where n is the number of scans required for this test

the size is assumed to be a power of 2

this test only runs in word mode

example:

Command : x 100000 100000 aaaa 10

Tests the 1m block of memory at 0x100000 using the patterns 0xaaaa and 0x5555 in a checkerboard pattern as data. The test repeated 10 times.

defaults:

x daddr dsize 5555 1

7.2.5.6. diagonal

Command usage is: g addr size passcnt

The moving diagonal test requires a number of write-read scans through the entire memory bank. On the first scan, all bits are written to DATA. The read scan verifies the correct operation of the array under these conditions. On each succeeding scan, the position of the diagonal

of DATA is shifted until, on the last scan, it has occupied every possible position in the array. Each cell has once been the only DATA cell in a row and column of DATA.

The diagonal test is actually a modified galpat test (hence the 'g' to call up the command). This test also requires a number of write-read scans thru each bank (or row depending on karl's verbiage) of the block of memory under test. At the beginning of the test the memory is initialized to 1's (every bit!). Then the test proceeds in the following sequence:

on the first scan a word of zeroes is written at particular locations in memory causing a diagonal of 0's in each memory chip's memory array.

on each successive scan the diagonal is shifted until it has occupied all of the diagonal positions of each memory chip's array. in other words each cell of the memory array has been the only 0 cell in a row and column of the array.

each read scan checks for the diagonal of 0's in a field of 1's.

the size is assumed to be a multiple of 0x40000

this test only runs in word mode

example:

Command : g 100000 80000 1

Tests the 1/2m block of memory at 0x100000. The test executed only one time.

defaults:

g daddr dsize 1

7.2.6. Sequencing

The syntax used to specify loops is:

syntax :l loopcnt

(where loopcnt is a decinal number in the range 1 - 4294967295)

default :l 4294967295

7.2.7. Defaults

This section describes the default values used in each test as well as the sequence used as the default test sequence.

7.2.7.1. Default parameters

The default parameters for each of the commands are summarized below:

```
Command : A daddr
Command : S dsize

Command : f daddr dsize ffffffff
Command : d daddr dsize

Command : c daddr dsize 55555555 1
Command : g daddr dsize 1
Command : x daddr dsize 5555 1
Command : r daddr dsize 7 1
Command : a daddr dsize 1
Command : u daddr dsize 1 1

Command : m 1
Command : p 3
Command : w 0
Command : s 0
Command : e 1

Command : l 4294967295
```

Where daddr and dsize above are the default address and size parameters. The current default address and size parameters are always printed at the head of the menu and are set in the following ways:

1. at the beginning of the diagnostic (right after boot) the page boundary (pages are 2K = 0x800 bytes) following the end of the diagnostic itself is taken to be the default address parameter, while the default size parameter is taken to be the total memory size of the system (determined and offered by the diag/mon/boot ROM) minus the default address. -or-

2. the default addr and size parameters may also be set using the commands 'A' and 'S'.

(For more information, see the section on parameters, address and size above.)

7.2.7.2. default test sequence

type "t" to get the default test sequence as follows :

```
m 0                     byte mode
c . . 0___              constant pattern 00
c . . 55          constant pattern 55
c . . aa          constand pattern aa
c . . ff          constant pattern ff
a                       address test
u . . 5                 uniqueness test
r                       random test
m 1                     word mode
c . . 0                 constant pattern 0000
c . . 5555        constant pattern 5555
c . . aaaa        constant pattern aaaa
c . . ffff        constant pattern ffff
a                       address test
u . . 5                 uniqueness test
r                       random test
m 2                     long mode
c . . 0                 constant pattern 00000000
c . . 55555555          constant pattern 55555555
c . . aaaaaaaa          constant pattern aaaaaaaa
c . . ffffffff    constant pattern ffffffff
a                       address test
u . . 5                 uniqueness test
r                       random test
m 1                     mode word
l *                     loop on the above sequence "forever"
```

note : remember that the .'s above mean that the default address
and size are used (see discussion under parameters above)

8. Error Handling

8.1. Message Interpretation

8.1.1. bus errors and parity errors

Note that parity errors are just a special case of bus errors. Bus errors are trapped and the message is displayed as follows:

bus error: 0x<buserrorreg>

sr=<statusreg> pc=<progctr> vss=<vector&specstat> @<addr>

<[VALID|INVALID] | BUSMASTER | PROTECTION | TIMEOUT |
UPPER PARITY | LOWER PARITY>

The first line flags the error as a bus error and displays the contents of the CPU bus error register. The second line displays the information saved on the stack by the 68010 when it processes exceptions (bus errors). Note that the last line is simply an English interpretation of the bits in the bus error register.

Refer to both The Theory of Ops for the Sun-2/120 CPU as well as the M68000 16/32-bit Microprocessor Programmer's Reference Manual (4th edition) for more information on understanding the meaning of the displayed information.

8.1.2. data errors

Data compare errors are trapped and the message is displayed as follows:

>> <testname> failed @ <addr> exp (<writedata>) obs (<readdata>)

>> failure in row <row#> : location U<location#>

Anomaly: Currently, if an error is detected, the memory location where the error occurred is read twice (the first time when the error is detected, the second time to pass the observed value to the error handler). The problem with this is that on the second read the data read will occasionally match the expected value. It is the second value read that is displayed in the error message. This explains confusing error messages similar to the following: constant pattern test failed @ 0x140000 exp (0x5555) obs (0x5555)

8.1.3. chip mapping

This diagnostic has the intelligence to report bad chip locations based on addresses that fail with data errors (this is not done for parity errors, since the failing address is not neccessarily latched by the hardware). Note that the chip locations are valid for the 120prime CPU layout ONLY.

8.1.4. error logging

Both bus errors as well as data errors are logged. The first 100 of each type are logged separately. To display the error logs, type E at the command line prompt.

8.2. Failure Analysis

8.3. Field Replaceable Units

The only field replaceable unit addressed by this test is the memory board(s) under test.

9. Recommended Test Procedure

This is determined by TE, but one possible sequence would be:

1. testing the memory board circuits
2. p2bus interface
3. board select
4. bank select
5. memory array
6. parity

10. Future Considerations

11. Summary

This document is a reformatted version of "mem.diag.doc", and as such inherits any inaccuracies. This has been reformatted so that users of diagnostic programs are presented the information in a consistent manner.

12. References

Theory of Operations for the Sun-2/120

M68000 16/32-bit Microprocessor Programmer's Reference Manual (4th edition)