EXTERNAL SPECIFICATION
(User Perspective)

TITLE : color.diag External Specification

AUTHOR : Bernard Bove

REPORT NO. : 900-1007-01

REVISION NO. : A @(#) color.diag.txt 1.2 3/21/85

DATE : April 5, 1985

APPROVALS :	DATE
Originator	----------
	Bernard Bove
Department Manager	----------
	Eddie Leung
Department Manager	----------
	Daryle Schei
V. P.	----------
	Scott McNealy

Document Review Form

Please make note and initial on this page all corrections and/or proposed amendments by page number and/or section number.

Recommendations, Differences, Construction Errors, and comments:

Typographical Errors:

Attach additional sheet(s) as needed.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Purpose

The purpose of this document is to describe the color video board diagnostic program designed and developed by Peter Costello. This diagnostic is to serve as the primary tool in determining the functionality of Sun's color video board.

1.2. Applicable Documents

For further information on the *Sun Color Video Board* please see

Sun Color Video Board
User's Manual
PM: 800-0898

2. SYSTEM OVERVIEW

2.1. General Description

The program *color.diag* is a diagnostic program dedicated totally for testing the Sun Color Video Board.

2.2. Features

The program *color.diag* is capable of performing

incoming inspection on multiple boards,
burn-in test, and
system test.

In addition, *color.diag* provides component test facilities for chip or component debugging.

2.3. Required Configuration

The incoming test inspection of Sun Color Video Board requires a Sun system containing

(1) a monitor (360-0484),
(2) a video board (501-0059),
(3) a processor board (501-0001-01),
(4) up to twelve Sun Color Video boards (501-0461-01) and
(5) optional cabling to connect color boards to optional (530-0495-01 & 530-0492-01)
(6) color monitor(s) (360-0595-01).

color.diag is a bootable program only, therefore it requires some means of booting.

2.4. Error Handling

Error messages are displayed upon discovery.

3. color.diag SPECIFICATION

3.1. User Interface

The operator/tester is required to set switches and insert board(s) into card cage of test system. The operator is familiar with booting diagnostics and noting messages appearing upon the screen.

3.2. Input/Output

3.3. Operation

color.diag may be ran with or without a color monitor. At sometime **before its shipment to customer it should be tested with a color monitor.**

(1) The color video board has four jumper *bergs* which need to set prior to insertion into card cage. For purpose of common reference bergs are counted from left to right with left most berg location specified as 1. Set the jumpers as specified below. **These are not final configuration jumper settings.**

box tab(;);
cfB s s s s s
c |c s s s |c
c |c |c |c |c |c.
Sun Color Video Board

_
Board;Jumper Location;MultiBus Mem Addr
;_;_;_;;
;J1;J2;J3;J4;;

_
0;All;6-7-8 jumped;6;jumped;0x1E0000

_
1;All;3 6-7-8 jumped;6;jumped;0x1E4000

_
2;All;4 6-7-8 jumped;6;jumped;0x1E8000

_
3;All;3-4 6-7-8 jumped;6;jumped;0x1EC000

If there is a monitor attached connect the *sync, blue, green, and red* internal cables to the board to be observed.

(2) Insert the color video board(s) into the card cage at any slot **not** having a **P2** connector.

(3) Boot *color.diag*.

(4) Upon booting, color.diag attempts to locate all color boards configured to the system. Cross check the address displayed against board setting(s) utilized from above. If there are any differences, power down system, check jumpers, reinsert board, power up, and again boot *color.diag*. If there is still a problem set aside board in question for further test and component replacement.

(5) *color.diag* nexts request whether to proceed with an automatic or manual test of the board(s) configured on the system. Respond with an **a**.

(6) From this point on *color.diag* performs an extensive set of diagnostics on all boards configured. The diagnostic loops continuously. Allow four passes.

Should an error occur the board does not pass incoming inspection (or system integration test). Send the board to MRB.

(7) If the color board is connected to a color monitor then the operator is directed to observe that at the completion of a pass the diagnostic displays the *prime* colors on the monitor and holds them there for seven seconds.

cidag displays red on the left third, green in the center third, and blue on the right. If these colors do not appear check cabling. If cabling does not fix the problem the board is to be rejected.

(8) Upon the completion of four passes, the test is complete.

3.4. Error Handling

color.diag performs extensive checks and tests upon the color board, below is a description of the tests performed and the possible error message(s) that can be generated during the test.

3.4.1. Testing Function Register

All possible values are written to and read from the function register. An error occurring during this test appears as follows:

Device #*d* @ *x* Wrote *valu1* to Function Reg. Read *valu2*

d - specifies the board causing the error, x is the board's multibus address, valu1 and valu2 are the hex values written and read respectively.

3.4.2. Testing Mask Register

All possible values are written to and read from the mask register. An error occurring during this test appears as follows:

Device #*d* @ *x* Wrote *valu1* to Mask Reg. Read *valu2*

d - specifies the board causing the error, x is the board's multibus address, valu1 and valu2 are the hex values written and read respectively.

3.4.3. Testing Status Register

Interrupts are disabled and all possible values are written to and read from the status register. An error occurring during this test appears as follows:

Device #*d* @ *x* Wrote *valu1* to Status Reg. Read *valu2*

d - specifies the board causing the error, x is the board's multibus address, valu1 and valu2 are the hex values written and read respectively.

3.4.4. Testing Interrupt Logic

The software controlled vertical retrace interrupt is enabled, and the diagnostic waits on an retrace interrupt. If an interrupt doe not occur the following message is displayed:

Device #*d* @ *x* No Interrupt received when expected.

d - specifies the board causing the error, x is the board's multibus address.

3.4.5. Testing Address Registers

The two X-address registers and the two Y-address registers are tested via setting the address and reading it back for all possible locations. Error(s) occurring during this test appear as follows:

Device #*d* @ *x* Wrote X-Address *valu1* Set 1 Read *valu2*
Device #*d* @ *x* Wrote Y-Address *valu1* Set 1 Read *valu2*

d - specifies the board causing the error, x is the boards multibus address, valu1 and valu2 are the hex values written and read respectively, and i is the x-y address pair under

test (0 or 1).

3.4.6. Testing Color Map

These tests include first writing out the "check box" image to the monitor, which contains all possible color combinations (256), then performs a constant data check on the color map buffers. Finally it performs the old check board test (*id est* writing inverted data to alternate locations, then every second location, etc.). An error occurring during these tests appear as follows:

Device *d* **@** *x* **Error** *s* **Color Map** *i* **Color** *valu1*.
Wr: *valu2* **RD:** *valu3*.

d - specifies the board causing the error, x is the boards multibus address, s is **green, red, or blue**, i is one of the four color maps available (0 through 3), valu1 is the color index (0 through 255), valu2 and valu3 are the hex values for what was written and what was read respectively.

3.4.7. Testing Function Unit

These tests include placing various data patterns against various function values and checking on the corresponding result placed in the frame buffer. If the expected transformation does not take place then the following error message is displayed:

Device *d* **@** *x* **Function Unit write** *s*.
Wrote: *valu1* **Read:** *valu2*.

d - specifies the board causing the error, x is the boards multibus address, s is a function from the set of {**Source Data, Zeros, Ones, Inverted Source Data, Old Data Inverted, Mask, or Inverted Mask**}, valu1 and valu2 are the hex values for what was written and what was read respectively.

3.4.8. Testing Frame Buffer Memory

These tests include the regular memory test routines (id est constant data check, address line check, and the *checker board check*. An error occurring during these tests appear as follows:

Device *d* **@** *x* **s Test X =** *i* **Y =** *j*.
Wr: *valu1* **Rd:** *valu2*.

d - specifies the board causing the error, x is the boards multibus address, s is memory test from the set of {**Constant Data, Address, Checker**}, i and j are the x and y locations respectively, valu1 and valu2 are the hex values for what was written and what was read respectively.

3.4.9. Testing 8-Pixel-Wide Mode

This tests the "paint" provided by the Sun Color Video Board. The background area is set to one color and then painted over. Frame buffer locations are examined to ensure the paint to affect. An error occurring during these tests appear as follows:

Device *d* **@** *x* **Paint-Mode Error Y = 0. Wrote =** *valu1* **to** *valu2* **with** *valu3*.
Read valu4 at X = valu5
Wrote paint-mode pixel at Xaddr = valu6

d - specifies the board causing the error, x is the boards multibus address, valu1 and valu2 are the x coordinates (in hex) of where valu3 was written respectively. valu4 is the incorrect value read at location valu5. valu6 is where the paint started.

EXTERNAL SPECIFICATION
(User Perspective)

TITLE : color.diag External Specification

AUTHOR : Bernard Bove

REPORT NO. : 900-1007-01

REVISION NO. : A @(#) color.diag.txt 1.2 3/21/85

DATE : July 2, 1985

APPROVALS :		DATE
Originator	Bernard Bove	
Department Manager	Eddie Leung	
Department Manager	Daryle Schei	
V. P.	Scott McNealy	

Document Review Form

Please make note and initial on this page all corrections and/or proposed amendments by page number and/or section number.

Recommendations, Differences, Construction Errors, and comments:

Typographical Errors:

Attach additional sheet(s) as needed.

TABLE OF CONTENTS

1. INTRODUCTION

1.1. Purpose

The purpose of this document is to describe the color video board diagnostic program designed and developed by Peter Costello. This diagnostic is to serve as the primary tool in determining the functionality of Sun's color video board.

1.2. Applicable Documents

For further information on the *Sun Color Video Board* please see

Sun Color Video Board
User's Manual
PM: 800-0898

2. SYSTEM OVERVIEW

2.1. General Description

The program *color.diag* is a diagnostic program dedicated totally for testing the Sun Color Video Board.

2.2. Features

The program *color.diag* is capable of performing

incoming inspection on multiple boards,
burn-in test, and
system test.

In addition, *color.diag* provides component test facilities for chip or component debugging.

2.3. Required Configuration

The incoming test inspection of Sun Color Video Board requires a Sun system containing

(1) a monitor (360-0484),
(2) a video board (501-0059),
(3) a processor board (501-0001-01),
(4) up to twelve Sun Color Video boards (501-0461-01) and
(5) optional cabling to connect color boards to optional (530-0495-01 & 530-0492-01)
(6) color monitor(s) (360-0595-01).

color.diag is a bootable program only, therefore it requires some means of booting.

2.4. Error Handling

Error messages are displayed upon discovery.

3. color.diag SPECIFICATION

3.1. User Interface

The operator/tester is required to set switches and insert board(s) into card cage of test system. The operator is familiar with booting diagnostics and noting messages appearing upon the screen.

3.2. Input/Output

3.3. Operation

color.diag may be ran with or without a color monitor. At sometime **before its shipment to customer it should be tested with a color monitor.**

(1) The color video board has four jumper *bergs* which need to set prior to insertion into card cage. For purpose of common reference bergs are counted from left to right with left most berg location specified as 1. Set the jumpers as specified below. **These are not final configuration jumper settings.**

box tab(;);
cfB s s s s s
c |c s s s |c
c |c |c |c |c |c.
Sun Color Video Board

_
Board;Jumper Location;MultiBus Mem Addr
;_;_;_;_;;
;J1;J2;J3;J4;;

_
0;All;6-7-8 jumped;6;jumped;0x1E0000

_
1;All;3 6-7-8 jumped;6;jumped;0x1E4000

_
2;All;4 6-7-8 jumped;6;jumped;0x1E8000

_
3;All;3-4 6-7-8 jumped;6;jumped;0x1EC000

If there is a monitor attached connect the *sync, blue, green, and red* internal cables to the board to be observed.

(2) Insert the color video board(s) into the card cage at any slot **not** having a **P2** connector.

(3) Boot *color.diag.*

(4) Upon booting, color.diag attempts to locate all color boards configured to the system. Cross check the address displayed against board setting(s) utilized from above. If there are any differences, power down system, check jumpers, reinsert board, power up, and again boot *color.diag.* If there is still a problem set aside board in question for further test and component replacement.

(5) *color.diag* nexts request whether to proceed with an automatic or manual test of the board(s) configured on the system. Respond with an **a.**

(6) From this point on *color.diag* performs an extensive set of diagnostics on all boards configured. The diagnostic loops continuously. Allow four passes.

Should an error occur the board does not pass incoming inspection (or system integration test). Send the board to MRB.

(7) If the color board is connected to a color monitor then the operator is directed to observe that at the completion of a pass the diagnostic displays the *prime* colors on the monitor and holds them there for seven seconds.

cidag displays red on the left third, green in the center third, and blue on the right. If these colors do not appear check cabling. If cabling does not fix the problem the board is to be rejected.

(8) Upon the completion of four passes, the test is complete.

3.4. Error Handling

color.diag performs extensive checks and tests upon the color board, below is a description of the tests performed and the possible error message(s) that can be generated during the test.

3.4.1. Testing Function Register

All possible values are written to and read from the function register. An error occurring during this test appears as follows:

Device #*d* @ *x* Wrote *valu1* to Function Reg. Read *valu2*

d - specifies the board causing the error, x is the board's multibus address, valu1 and valu2 are the hex values written and read respectively.

3.4.2. Testing Mask Register

All possible values are written to and read from the mask register. An error occurring during this test appears as follows:

Device #*d* @ *x* Wrote *valu1* to Mask Reg. Read *valu2*

d - specifies the board causing the error, x is the board's multibus address, valu1 and valu2 are the hex values written and read respectively.

3.4.3. Testing Status Register

Interrupts are disabled and all possible values are written to and read from the status register. An error occurring during this test appears as follows:

Device #*d* @ *x* Wrote *valu1* to Status Reg. Read *valu2*

d - specifies the board causing the error, x is the board's multibus address, valu1 and valu2 are the hex values written and read respectively.

3.4.4. Testing Interrupt Logic

The software controlled vertical retrace interrupt is enabled, and the diagnostic waits on an retrace interrupt. If an interrupt doe not occur the following message is displayed:

Device #*d* @ *x* No Interrupt received when expected.

d - specifies the board causing the error, x is the board's multibus address.

3.4.5. Testing Address Registers

The two X-address registers and the two Y-address registers are tested via setting the address and reading it back for all possible locations. Error(s) occurring during this test appear as follows:

Device #*d* @ *x* Wrote X-Address *valu1* Set I Read *valu2*
Device #*d* @ *x* Wrote Y-Address *valu1* Set I Read *valu2*

d - specifies the board causing the error, x is the boards multibus address, valu1 and valu2 are the hex values written and read respectively, and i is the x-y address pair under

test (0 or 1).

3.4.6. Testing Color Map

These tests include first writing out the "check box" image to the monitor, which contains all possible color combinations (256), then performs a constant data check on the color map buffers. Finally it performs the old check board test (*id est* writing inverted data to alternate locations, then every second location, etc.). An error occurring during these tests appear as follows:

Device *d* **@** *x* **Error** *s* **Color Map** *i* **Color** *valu1*.
Wr: *valu2* **RD:** *valu3*.

d - specifies the board causing the error, x is the boards multibus address, s is **green, red, or blue**, i is one of the four color maps available (0 through 3), valu1 is the color index (0 through 255), valu2 and valu3 are the hex values for what was written and what was read respectively.

3.4.7. Testing Function Unit

These tests include placing various data patterns against various function values and checking on the corresponding result placed in the frame buffer. If the expected transformation does not take place then the following error message is displayed:

Device *d* **@** *x* **Function Unit write** *s*.
Wrote: *valu1* **Read:** *valu2*.

d - specifies the board causing the error, x is the boards multibus address, s is a function from the set of {**Source Data, Zeros, Ones, Inverted Source Data, Old Data Inverted, Mask, or Inverted Mask**}, valu1 and valu2 are the hex values for what was written and what was read respectively.

3.4.8. Testing Frame Buffer Memory

These tests include the regular memory test routines (id est constant data check, address line check, and the *checker board check*. An error occurring during these tests appear as follows:

Device *d* **@** *x* **s Test X =** *i* **Y =** *j*.
Wr: *valu1* **Rd:** *valu2*.

d - specifies the board causing the error, x is the boards multibus address, s is memory test from the set of {**Constant Data, Address, Checker**}, i and j are the x and y locations respectively, valu1 and valu2 are the hex values for what was written and what was read respectively.

3.4.9. Testing 5-Pixel-Wide Mode

This tests the "paint" provided by the Sun Color Video Board. The background area is set to one color and then painted over. Frame buffer locations are examined to ensure the paint to affect. An error occurring during these tests appear as follows:

Device *d* **@** *x* **Paint-Mode Error Y = 0. Wrote =** *valu1* **to** *valu2* **with** *valu3*.
Read valu4 at X = valu5
Wrote paint-mode pixel at Xaddr = valu6

d - specifies the board causing the error, x is the boards multibus address, valu1 and valu2 are the x coordinates (in hex) of where valu3 was written respectively. valu4 is the incorrect value read at location valu5. valu6 is where the paint started.