Sun Microsystems

Sun-2 Firmware

Design Document

Mehrdad Mojgani

October 12, 1987

Revision A

Part Number: 800-1881-01

Contents

1. Preface

The Design Document associated with Sun-2 firmware is presented.

1.1. Purpose

This document presents the high level design of Sun-2 firmware. It informs the reader of the coverage that will be provided by Sun-2 firmware regarding the areas of power-up tests, machine initialization, boot sequences and monitor capabilities. A description of each item (i.e. test, initialization step, boot sequence, etc.) under these four categories is also given.

1.2. Audience

Members of any of the following five departments may find this document of interest for various reasons: Hardware, Software, Manufacturing, Field Service and Diagnostics.

Using the diagnostic tests associated with Sun-2 firmware, the Design Engineers of Sun-2 products will be able to confirm the correctness of their design. Software will depend on the boot sequence to load and begin execution of the UNIX kernel following machine initialization. In addition, the monitor will aid Software during the debugging process. Manufacturing personnel will use Sun-2 firmware for system burn-in, testing and/or trouble shooting purposes. Sun-2 firmware will also provide Field Service with testing and trouble shooting tools. Members of the Diagnostic department will review the design of Sun-2 firmware based on this document.

At a minimum, the reader is assumed to have had some exposure to the firmware of previous product lines of Sun Microsystems.

2. Revision History

Revision A	August 13, 1986
Revision B	October 12, 1987

3. Glossary

AMD	Advanced Micro Devices.
ASCII	American Standard Code for Information Interchange.
MMU Space	The core of the Sun-2 architecture. It includes the Context Register, Segment Map and Page Map, Bus Error Register, a System Enable Register, a Diagnostic Register and an ID PROM.
CPU	Central Processing Unit.
Device Space	All devices that are accessed through the MMU. At a minimum, this includes Main Memory, a set of EPROMs and a TOD Clock. One or more of the following devices may also be included: a Keyboard, a Mouse, Serial Ports, an Ethernet Interface, Video Memory.
Diagnostic Executive	Multi-tasking environment under which Sun Microsystems' diagnostics will eventually run.
DMA	Direct Memory Access.
DVMA	Direct Virtual Memory Access.
EPROM	Erasable Programmable Read Only Memory.
Ethernet	Communication link between systems by way of a coaxial cable.
Executive	A mnemonic for Diagnostic Executive.
FIFO	First In First Out.
FRU	Field Replaceable Unit.
Word	A 16-bit unit of memory.
ID PROM	Identification PROM which contains basic information, such as machine type, serial number, ethernet address, manufacturing date and a checksum.
I/O	Input Output.
LED	Light Emitting Diode.
MMU	Memory Management Unit.
Mouse	A switch-operated screen pointer/positioner.
Page	The smallest contiguous, selectable block of memory available through the MMU.
PMEG	Page Map Entry Group.
POR	Power-on Reset. The reset condition which follows powering on the system.
PROM	Programmable Read Only Memory.
RAM	Random Access Memory.
SCC	Serial Communications Controller.
SCSI	Small Computer System Interface.

SMD	Storage Module Device.
TOD	Time of Day Clock.
UART	Universal Asynchronous Receiver/Transmitter.
UNIX	An operating system.
VLSI	Very Large Scale Integration.
VME	Motorola's bus interface between the CPU board and peripherals.
Watchdog	A mnemonic for Watchdog Reset.
Watchdog Reset	The reset condition which follows the detection of a double bus error. It is equivalent to a power-on reset.
Long Word	A 32-bit unit of memory.

4. Introduction

Sun-2 firmware will provide the software foundation necessary to run programs on a Sun-2 work station. The UNIX operating system, the Diagnostic Executive and stand alone programs will all depend upon the firmware for their initial program load and invocation.

There will be four primary responsibilities associated with Sun-2 firmware. Those responsibilities are to "**perform the power-up test sequence**", "**initialize the machine**"," **execute the default boot sequence and , "provide a monitor", for more advanced trouble shooting.**

5. Problem Specification

Generally speaking, it is the objective of Sun-2 firmware to ensure that enough of the CPU board logic is working correctly such that, the default boot sequence can be carried out following machine initialization. Furthermore, a monitor must also be made available for more advanced diagnosis of the hardware. Each of these issues is discussed in more detail below.

To begin with, Sun-2 firmware will have to execute the power-up test sequence. The power-up test sequence will check the CPU board logic. The purpose of these tests is to ensure that enough of the system is functional such that the UNIX operating system, Diagnostic Executive or a stand alone program can be booted.

The firmware's next responsibility will be to perform machine initialization. This will include such tasks as initializing segment maps, page maps, keyboard, mouse, serial ports, frame buffer, memory, etc. Additional initialization tasks include sizing memory and setting up interrupt vectors, trap vectors, exception vectors, entry points corresponding to support routines, and so on.

The third area of firmware concern is the booting of programs. In particular, following successful initialization of the machine and assuming *no* operator intervention, the UNIX operating system is to be loaded and executed. It should be noted that, if no bootable device exists the monitor will be invoked. On the other hand, if the power-up tests are successfully completed and the operator interrupts the automatic (default) boot sequence, an operator-specified boot command can be issued which either will load a bootblock or the Loader and execute it.

Finally, due to the need for trouble shooting tools, Sun-2 firmware must also provide a set of comprehensive tests which go beyond the ability of the power-up tests mentioned earlier. As a consequence of interrupting the normal boot sequence, the monitor will be invoked. Through interaction with the monitor, the user will be able to execute the additional tests.

All four of these areas are more fully described in the section titled "Functional Description Of Sun-2 Firmware".

6. Requirements

During the process of designing the firmware for Sun-2 products, the following performance, functional, hardware and environmental requirements were assumed.

6.1. Performance Requirements

The main memory is sized during the powerup diagnostic tests and the size will be kept in the global variable "**gp-> g_memorysize**" which is used by other routines. The amount of time required to bring the system up is a function of main memory size and since the self-test performs simple memory tests and the maximum memory on Sun-2 machines is 6Mbyte, it would take less than a minute . Default boot devices are not defined and the F/W itself is responsible to map devices, probe them and boot from the first existing one in order.

6.2. Functional Requirements

The area of functional requirements can be broken down into four categories. Those categories are the "**power-up test sequence**", "**host machine initialization**", "**the boot strap sequence**" and "**monitor functions**". The functional requirements for each of these areas is expanded upon below.

The goal of the power-up test sequence will be to check the CPU board logic to the point that an attempt can be made to boot UNIX. As these "crawl-out" tests execute, their *test number* must be displayed in bits zero through four of the diagnostic LEDs. If one of these tests should fail, bit seven of the diagnostic LEDs must also light up. Bit seven will indicate that there is a hardware problem. Given an error condition, the booting process will not be attempted. Instead, upon detection of a failure, the unsuccessful test will enter an infinite scope loop. Continuously re-executing the failing test should enhance the trouble shooter's ability to study the problem with test equipment. For more information regarding the power-up test sequence, see the section titled "The Power-Up Test Sequence".

Sun-2 firmware's second area of functional requirements concerns host machine initialization. Basically speaking, the work station must be initialized to the point that a program can be successfully loaded and executed. With a few exceptions, all components of MMU Space and Device Space (see Glossary) are to be initialized by the firmware. Additional initialization tasks include setting up entry points to support routines, interrupt vectors, trap vectors, exception vectors, and so on. The subject of initialization will be covered more completely in the section titled "Host Machine Initialization".

The boot strap sequence is the third area under functional requirements which must be addressed by Sun-2 firmware. Assuming *no* operator intervention, either the UNIX operating system is to be loaded. following the *error-free* completion of the power-up test sequence and subsequent host machine initialization. If there is no bootable device in the configuration the monitor will be invoked. On the other hand, if the operator interrupts the automatic (default) boot sequence, an operator-specified stand alone program can be loaded. In order to obtain further information related to the boot sequence, see the section titled "The Boot Sequence".

The monitor is the final topic under functional requirements that must be taken into account by Sun-2 firmware. The monitor will be invoked following the *error-free* completion of the power-up tests and machine initialization. It will be the underlying software environment for programs that will run on Sun-2 work stations.

The monitor will have two primary areas of concern. The initial concern will be to provide a set of low level commands. These commands will carry out a variety of debugging/trouble shooting functions, provide a number of utilities, etc. These low level commands will be presented in the section titled "Basic Monitor Commands".

The section titled "Extended Test Menu Structure" will discuss all of the non-power-up tests which will be available with Sun-2 firmware. These more comprehensive tests will give the user more flexibility when testing. Each test and the argument(s) associated with it will be covered in the section titled "Extended Test Menu Structure".

6.3. Hardware Requirements

Since there are lots of different configurations along with CPU boards under the Sun-2 name a single configuration cannot be considered as a test environment for the firmware.A Minimum hardware requirement consists of the CPU board with Boot PROMs installed; as the Boot PROM has no hardware feedback to assertain if the video display is working. Additional hardware requirements are as follows:

(1) a minimum of 64 Kbytes of programmable read-only memory (PROM),

(2) eight diagnostic, LEDs conveniently viewable by the user in the event of self test errors.

(3) Ethernet, disk, tape, or communications line to boot Unix, the Diagnostic Executive, or a standalone program, and,

(4) a correctly programmed IDPROM to boot Unix via Ethernet.

6.3.1. Loopback Connector for Ports A and B

A loopback connector is required to test Serial Ports A and B at the handle edge of the CPU card using the Extended Menu Tests provided by the "X" Monitor command. This shorting connector is type DB25P. A list of interconnects is provided below 1.

From pin	To pin
2(TxD)	3(RxD)
4(RTS)	5(CTS)
6(DSR)	20(DTR)

6.3.2. Loopback Connector for Keyboard/Mouse Ports

A loopback connector is required to test the keyboard/mouse connector at the handle edge of the CPU card using the Extended Menu Tests provided by the "X" Monitor command. This shorting connector is a 15-pin DB15P connector. A list of interconnects is provided below 1.

From pin	To pin
1(RxD)mouse	3(TxD)mouse
5(RxD)keybd	7(TxD)keybd

6.4. Environmental Requirements

Sun-2 firmware will run on fully-equipped Sun-2 CPU boards. The current CPU boards under consideration are the Sun-2/50/130/160 ,Sun-2/120/170 and Sun-1/100U/150U CPU boards.

7. Limitations

The following are limitations inherent to the Boot PROM design:

(1) A working video display and keyboard, while not absolutely necessary to observe the execution state of the Boot PROM program from the diagnostic LEDs, or a terminal connected to serial port A at the rear panel of the workstation is required to interact with the Boot PROM program .In case during the booting there is problem with the terminal a dumb terminal could be hooked up to serial port Athen by typing **uaio** on the keyboard all the I/O will be directed to it.

(2) The Boot PROM diagnostics are designed to only test the CPU board hardware sufficiently to bring the operating system or any loader up.In other words It is not within the charter of Sun-2 firmware to perform exhaustive tests on all of the hardware components available in any specific configuration. If for no other reason, the size limitations imposed by the EPROMs themselves prevent an exhaustive test set from being included. The amount of testing performed by Sun-2 firmware will, however, be sufficient enough to allow more comprehensive diagnostics to be loaded.

The firmware will *not* require any underlying software environment. In fact, the firmware itself will serve as the software basis for to-be-executed programs.

8. Functional Description

As stated previously, Sun-2 firmware will have four functional requirements. Each of these four areas of responsibility are covered below in the sections titled "The Power-Up Test Sequence", "Host Machine Initialization", "The Boot Sequence" and "The Monitor", respectively.

8.1. The Power-Up Test Sequence

In order to perform the power-up tests, two assumptions must be met. The 68010 Processor must be functional and the ability to fetch instructions from the EPROM must be intact.

Powering up a Sun-2 work station will reset the 68010 Processor to boot state. As a result of reseting the CPU to boot state, all instruction fetches will be forced to the EPROM. Execution of the minimum-confidence power-up tests will begin immediately. These tests will not employ any memory until memory has been successfully checked.

The objective of the power-up test sequence will be to determine whether or not the CPU board logic is functional. Following the successful completion of the power-up tests and subsequent machine initialization, an attempt will be made to boot UNIX.

If the power-up tests execute successfully and the user does *not* terminate the default boot sequence, an attempt will be made to load UNIX. The following display will appear on the *work station's* screen to indicate this.

```
D'l |2808u 0'

                Selftest Completed Successfully.


(The )  Sun Workstation, Model Sun-2/xxx..., type of keyboard
(Sun )  ROM Rev xxx, x MB memory installed, Serial # xxxx
(Logo)  Ethernet address XX:XX:XX:XX:XX:XX

Autobooting in progress...
Boot: sd(0,0,0)
Boot: sd(0,0,0)vmunix

D'l [2 8 0 8 u 0 f v '                                          D'[
```

One requirement of Sun-2 firmware will be to assign a unique test number to each of the power-up tests and display that number in bits *zero* through *seven* of the diagnostic LEDs as the test is running. If one of these tests should fail, the LEDs will keep displaying the same pattern to enable the trouble shooter to conclude which power-up test is failing.

For the sake of completeness, LED *four* will be the *heart beat* LED. After the power-up tests have been completed, but prior to invocation of UNIX , LED 4 will blink on and off to indicate that the 68010 Processor is actually executing (It is not hung).

Upon detection of a failure, the unsuccessful test will enter an infinite scope loop. Continuously re-executing the failing test should enhance the trouble shooter's ability to study the problem with test equipment.

At this point, each power-up test will be documented. The name of each power-up test and its function is given below.

Diagnostic Register Test

This test will sequentially light up each of the LEDs. It indirectly tests the Processor's ability to fetch instructions from the EPROM and transfer data across the data bus.

Context Register Test

This test will perform write-read-compare cycles on the Context Register.

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
2	0x11	○○○●○○○●	okay

Segment Map Address Test

Should this test fail means that Segment map address lines are shorted or miswired.

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
3	0x22	○○●○○○●○	okay

Segment Map data lines Test

Should this test fail means that Segment map data wires are shorted or bits are bad.

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
4	0x23	○○●○○○●●	okay

Segment Map constant data Test

Should this test fail means that Segment map data wires are shorted .

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
5	0x21	○○●○○○○●	okay

Page Map Address dependency Test

Should this test fail means that Page map address lines are miswires are shorted .

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
6	0x32	○○●●○○●○	okay

Page Map data lines Test

Should this test fail means that Page map data lines are miswires are shorted .

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
7	0x33	○○●●○○●●	okay

Page Map Constant Data Test

Should this test fail means There exists a hardware problem.

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
8	0x31	○○●●○○○●	okay

Memory sizing Test

Prior to testing memory, the memory will be sized by first mapping it and then trying to write and read back.

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
9	0x70	●●●●○○○○	okay

Memory Constant Data Test

the memory will be tested by write-read-compare steps.

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
10	0x71	●●●●○○○●	okay

Memory Address Dependency Test

the memory address lines will be checked against any miswiring or defected chips.

Test Number	Hexadecimal Value Of LEDs	Visual Representation	Condition
11	0x72	●●●●○○●○	okay

The following table summarizes the potential LED displays.

LEDs (● = ON, o = OFF) 7	6	5	4	3	2	1	0	What the system is doing when these LEDs are cycling.	What might be bad if these LEDs stay on
●	●	●	●	●	●	●	●	A reset will set LEDs to this state.	CPU Board/ EPROM.
o	o	o	o	o	●	o	o	EPROM checksum test.	EPROM/ CPU Board.
o	o	●	o	o	o	●	o	Segment Map Address Test.	CPU Board (MMU).
o	o	●	o	o	o	●	●	Segment Map data lines Test.	CPU Board (MMU).
o	o	●	o	o	o	o	●	Segment Map constant data Test.	CPU Board (MMU).
o	o	●	●	o	o	●	o	Checking Page Map Address dependency.	CPU Board (MMU).
o	o	●	●	o	o	●	●	Checking Page Map Data lines .	CPU Board (MMU).
o	o	●	●	o	o	●	●	Checking Page Constant Data lines .	CPU Board (MMU).
o	●	●	●	o	o	o	o	Checks memory size.	CPU or Memory board.
o	●	●	●	o	o	o	●	Constant Data Memory Test.	CPU or Memory board.
o	●	●	●	o	o	●	o	Memory Address Dependency Test.	CPU or Memory board.
●	●	●	●	o	o	●	●	Setting up frame buffer.	
●	●	●	●	o	●	o	o	Setting up NMI or keyboard.	
o	o	o	●	o	o	o	o	Self-Test done, UNIX in boot-state.	CPU Board
o	o	o	o	o	o	o	o	(LED is blinking).	

8.2. Host Machine Initialization

Beyond the *error-free* completion of the power-up tests but before the the execution of the default boot sequence, the machine will require initialization. The initialization steps are listed below in an arbitrary order.

1. The segment map in each context will be initialized.
2. All page map entries will be initialized.
3. Memory will be sized.
   A. The total amount of main memory which is physically present, both working and non-working, will be stored in **romp-> v_memorysize*.
   B. Variable **romp-> v_memoryavail* will contain the amount of available memory, excluding nonworking pages and pages reserved for the monitor itself.

4. All of the available main memory will be initialized.
5. All available pages of *working* memory will be mapped in ascending order, starting at location zero.
6. Assuming that they exist, the devices which will be initialized and mapped are presented below.
   A. Keyboard.
   B. Mouse.
   C. Serial Ports A and B.
   D. Video Memory.
   E. TOD Clock.
   F. Ethernet.
7. The system is initialized such that the monitor's interrupt vectors will be employed.
8. Entry points to support routines will be set up.

8.3. The Boot Sequence

Following the initialization of the work station, the default boot sequence will be executed. There are two issues which must be taken into consideration here. One has to do with *what is to be loaded* while, the other has to do with *where it is to be loaded from*.

If the operator interrupts the automatic (default) boot sequence, the monitor will be invoked. At that point, the user will have the ability to specify *what* is to be booted and *where* it is to be booted from. See command "b" (boot) below in the section titled "Basic Monitor Commands" for a description of how to boot user-specified programs from user-specified devices.

If the operator is interacting with a Sun-2 machine through the console, s/he will be able to interrupt the default boot sequence by typing "*L1-a*". That is, hold down the "*L1*" key while pressing the "*a*" key. On the other hand, if the user is interacting with the work station through a dumb terminal, the automatic boot sequence can be terminated by pressing the "*break*" key.

The firmware must also determine where the to-be-loaded program is to be loaded from.

In the default boot sequence Sun-2 firmware will attempt to boot UNIX using the following boot device polling sequence.

1. Xylogics Disk.
2. SCSI Disk.
3. Ethernet.

In contrast, if the operator intervenes the default boot sequence or either the file name or the boot device is not present or is in error, the monitor will be invoked.

8.4. The Monitor

Following the *error-free* completion of the power-up tests and machine initialization, the monitor will be invoked. The monitor will serve as the underlying software environment for programs that will run on Sun-2 work stations.

The monitor can be broken down into two parts. The initial section, titled "Basic Monitor Commands", will cover the low level monitor commands. The section titled "Extended Test Menu Structure" will discuss all of the non-power-up tests which will be available with Sun-2 firmware.

8.4.1. Basic Monitor Commands

The low level monitor commands will provide a number of debugging/trouble shooting functions.
To invoke the Monitor the operator presses the break or L1-A key. The command line syntax and a description of each of these commands is discussed below.

Some of the commands are not shown on the help menu due to space considerations.

8.4.2. Monitor Command Syntax

The monitor command syntax is:

< verb> < space> *[< argument>]< return>

where:

< verb> is always one alphabetic character-- case does not matter.

< space> * is any number of spaces.

< argument> is normally a hexadecimal number or a single letter; again, case does not matter. Square brackets indicate that the argument portion is optional.

< return> is a carriage return.

When typing commands, < backspace> and < delete> (also called < rubout>) erase one character; control-U erases the entire line.

8.4.3. Syntax for Memory and Register Access

Several of the commands open a memory location, map register, or processor register, so that you can examine and/or modify the contents of the specified location. These commands include A, D, E, L, M, O, P, Q, and R. *A "+ " or "-" in front of commands E, L, M, O, P and Q will determine the direction of traversing the addresses for the next location. A "+ " causes the address be incremented and a "-" decrements the address for the next location. The default is*

address increment for these commands.

Each of these commands takes the form of a command letter, possibly followed by a hexadecimal memory address or register number, followed by a sequence of zero or more 'actions specifier' arguments. The various options are illustrated below, using the "e" command as an example. You type the parts as shown with a < return> at the end of each command.

If no action specifier arguments are present, the address or register name is displayed along with its current contents. You may then type a new hexadecimal value, or simply < return> to go on to the next address or register. Typing any non-hexadecimal character and < return> gets you back to command level. for register, 'next' means within the sequence of registers:

D0-D7	the data registgers,
A0-A6	the address registers,
SS	the system stack pointer,
US	the user stack pointer,
SF	the source function code register,
DF	the destination function code register,
VB	the vector base register,
SC	the system context register,
UC	the user context register,
SR	the status register,
PC	the program counter.

For example, the following command sets consecutive locations 0x1234 and 0x1236 to the values 0x5678 and 0x0000 respectively. To terminate the input enter any non-hexadecimal character or a space followed by a carriage return.

```
001234: 007F? 5678
001236: 51A4? 0
001238: C022? q
>
```

A non-hexadecimal character (such as question mark) on the command line means read-only:

```
> e 1000 ?
001000: 007F
>
```

Multiple nonhexadecimal characters read multiple locations:

```
> e 1000 1 2 3
001000 ->  0001
001002 ->  0002
001004 ->  0003
>
```

Finally, reads and write can be interspersed:

```
> e 1000 ? 1 ? ? 3 4
001000: 007F -> 0001
001002: 0064
001004: 1234 -> 0003
001006 -> 0004
>
```

Spaces are optional except between two consecutive numbers. When actions are specified on the command line after the address, no further input is taken from the keyboard for that command; after executing the specified actions, a new command is prompted for. Note that these commands provide the ability to write to a location (such as an I/O registger) without reading from it: and provide the ability to query a location without having to interact.

8.4.3.1. Monitor Command Descriptions

A [n][actions]

Open A-register n (0= < n= < 7, default zero). A7 is the system stack pointer; to see the user stack pointer, use the "r" command. For further explanation see section above, "Syntax for Memory and Register Access".

B [device([c],[u],[p])]

where device may be any of the following:

ic - Intel Ethernet
sd - SCSI disk
st - SCSI tape
mt - Tape Master 9-track tape
xt - Xylogics 1/4" tape
xy - Xylogics 440/450 disk
ar - Archieve Tape

and where:
c = controller number (0 if only one controller),
u = unit number (0 if only one driver), and,
p = partition.

Boot. Resets appropriate parts of the system then bootstraps the system. This allows bootstrap loading of programs from various devices such as disk, tape, or Ethernet.

Entering B without any command arguments will cause a default boot: if the system is disk based, then from the disk; if the system is diskless, then from the Ethernet controller.

Entering 'B?' will cause a display of all possible boot devices and their device specifier

arguments.

The boot command line will be echo displayed in the next line with the actual command arguments filled in, whether operator specified or not.

C [addr]

Continue a program. The address addr, if given, is the address at which execution will begin: default is the current PC. The registers will be restored to the values shown by the A, D, and R commands. A continue command should be executed after executing a breakpoint where a breakpoint has been set and executed under the Monitor.

! Command

This command will print and execute the last non blank command entered. It will prevent retyping of long commands.

D [n][actions]

Open D-register n (0= < n= < 7, default 0). This command is provided to examine and modify any of the 8 data registers D0 - D7 within the 68020 CPU. For a detailed explanation, see the section, "Syntax for memory and Register Access" above.

[+ / -]E [addr][actions][+ / -]

Open the 16 bit word at memory address addr (default zero) in the address space defined by the 'S' command. For a detailed explanation, see the section, "Syntax for Memory and Register Access" above.

For the function description of [+ / -] refer to Monitor "L" command.

F [addr1] [addr2] [param] [size]

Fill address space from lower address addr1 to higher address addr2 with the constant, param, of size b, w, or l where b specifies a byte pattern, w, a word pattern, and l, a long word pattern. If not size is specified, l for long word will be used.

For example, the following monitor command fills the address block from 0x1000 to 0x2000 with the word pattern, 0xABCD:

F 1000 2000 ABCD W

G [addr][param]

Start the program by executing a subroutine call to the access addr if given, or else to the current PC. The values of the address and data registers are undefined; the status register will contain 0x2700. One parameter is passed to the subroutine of the stack, it is the address of the remainder of the command line following the last digit of addr (and possible blanks).

H

Displays the menu of Monitor commands and their descriptions.

K [number]

If number is 0 or not given, this commands does a 'reset instruction': it resets the system without affecting main memory or maps. If number is 1, this does a 'Medium Reset', which re-initializes most of the system without clearing memory. If number is 2, a hard reset is done and memory is cleared. This is equivalent to a power-on reset and runs the PROM-based diagnostic self test, which can take from 5 to 180 seconds depending upon how much memory is being tested.

[+ / -]L [addr][actions][+ / -]

Open the 32 bit long word at memory address addr (default zero) in the address space defined by the 'S' command. For a detailed explanation, see the section, "Syntax for Memory and Register Access" above. *By preceding the command with a "+ " or "-" address increment or decrement for the next location will be specified.*

While traversing a range of address, direction can be changed by typing a "+ " or "-" after the machine displays the content and waits for input.
For example:
l 0 < ret>
00000000 00000000 ? < ret>
00000004 00000001 ? < ret>
00000008 00000002 ? - < ret>
00000004 00000001 ? < ret>
00000000 00000000 ? + < ret>
00000004 00000001 ? < ret>
00000008 00000002 ? < ret>

[+ / -]M [addr][actions][+ / -]

Opens the Segment Map entry which maps virtual address addr (default zero) in the current context. The Segment Map address is the virtual address field from address bit 27 thru bit 17 of the virtual address presented by the CPU to the Memory Management Unit. The choice of supervisor or user context is determined by the 'S' command setting, but is of no consequence in address translation in the Sun-2 MMU implementation. *For the function description of [+ / -] refer to Monitor "L" command.*

[+ / -]O [addr][actions][+ / -]

Opens the byte (8 bit) location specified (default zero in the address space defined by the 'S' command. For a detailed explanation, see the section, "Syntax for Memory and Register Access" above.

For the function description of [+ / -] refer to Monitor "L" command.

[+ / -]P [addr][actions][+ / -]

Opens the Page Map entry which maps virtual address addr (default zero) in the current context. For further syntax information, see the section "Syntax for Memory and Register Access" above.

For the function description of [+ / -] refer to Monitor "L" command.

R [actions]

Opens the miscellaneous registers: SS (Supervisor Stack Pointer), US (User Stack Pointer), SF (Source Function code), DF (Destination Function code), VB (Vector Base), SC (System Context), UC (User Context), SR (Status Register), and PC (Program Counter). Alterations made to these registers except SC and UC do not take effect until the next 'C' command. For further syntax information, see the section "Syntax for Memory and Register Access" above.

S [number]

Sets or queries the address space to be used by subsequent memory access commands. The number argument is the function code to be used, ranging from 1 to 7. Useful values are 1 (user data), 2 (user program), 3(memory maps), 5 (supervisor data), 6 (supervisor program). If no number argument is specified, the current setting is printed. Upon entry to the monitor, this is set to 5 if the program was in supervisor state, or to 1 if the program was in user state.

U [arg]

The U command manipulates the serial ports and switches the current operator input or output device. The argument may have the following values ('[AB]' means that either 'A' or 'B' is specified):

{AB}	Select serial port A (or B) as input and output device
{AB}io	Select serial port A (or B) as input and output device
{AB}i	Select serial port A (or B) for input only
{AB}o	Select serial port A (or B) for output only
k	Select keyboard for input
ki	Select keyboard for input
s	Select screen for output
so	Select screen for output
ks,sk	Select keyboard for input and screen for output
{AB}#	Set speed of serial port A (or B) to # (such as 1200,9600,..)
e	Echo input to output
ne	Don't echo input to output
u addr	Set virtual serial port address

If no argument is specified, the U command reports the current values of the settings. If no serial port is specified when changing speeds, the 'current' input device is changed.

At power-up, the following default settings are used: the default console input device is the Sun keyboard or if the keyboard is unavailable, serial port A. The default console output device is the Sun screen or if the graphics board is unavailable, serial port A. All serial ports are set to 9600 Baud.

V [addr1] [addr2] [size]

Display the contents of addresses from address addr1 to address addr2 in format specified by size where b specifies byte format, w, word format, and l, long word format. Enter a return character to cause the display to pause for viewing; enter another return character to resume the display. To terminate the display at any time press the space bar.

For example, the following command displays the contents of virtual address space from address 0x1000 to 0x2000 in word format:

V 1000 2000 W

^T [Address] Command

This command will display the Virtual to Physical address transform along with a detailed description of all the bits in the Page table entry, Segment and Page Ram addresses and their space.

For example:
^T 1000 < ret>
will result:
D'l |2808u 0'

Virtual Addr 1000 is mapped to Physical Addr 1000
Seg Map = 0x0, Page Map = 0xC0000000.
Page 0 has these attributes:

Valid bit = 0
Permission = 1
Type = 0
Access bit = 0
Modify bit = 0

D'l [2 8 0 8u 1 0' **D'**

^I Command

This command will display compilation informations. It includes the date, host name and the build directory path.
For Example:
Compiled at 6/7/87 on hostname in /directoryname/build

^C [source] [destination] [n] Command

Will copy a block of length [n] from [source] address to [destination] address byte by byte. There is enough delay to copy to EEPROM too.

X

Entering the X command character will cause a menu of extended menu tests to be presented with loop and print options also selectable. These test commands are provided to permit additional testing of such things as the I/O port connectors at the handle edge of the CPU board, Video memory, workstation memory and the workstation keyboard as well as permit the boot device paths to be tested.

8.4.4. Extended Test Menu Structure

In addition to the power-up tests, Sun-2 firmware will also provide a list of more comprehensive tests. Unlike the power-up tests, these tests will allow the user the ability to exercise a much larger degree of control over the behavior of the tests.

The user interface of the monitor's extended test menu structure will consist of a Main Menu and multiple sub-menus.

8.4.4.1. Main Menu

The Main Menu depending on the machine type will be displayed as one of the three listings below. A discussion for each of these options will be presented here.

D'l |1883u 0'

Extended Test Menu: (Enter 'q' to return to Monitor)

Cmd - Test

kb - Keyboard Input Test
me - Memory Test
vi - Video Test
mk - Mouse/ Keyboard Test
rs - Serial Ports Test
ie - Intel Ethernet Test
mt - TapeMaster Bootpath Test
sd - SCSI Disk Bootpath Test
st - SCSI Tape Bootpath Test
xt - Xylogics Tape Bootpath Test
xy - xylogics Disk Bootpath Test

Cmd= >

D'l [illegible]

D'l 0 |885u-1v'

Extended Test Menu for Sun-2/ 50/ 130/ 160.

D'l |1995u 0'

Extended Test Menu: (Enter 'q' to to return to Monitor)

Cmd - Test

kb - Keyboard Input Test
me - Memory Test
vi - Video Test
mk - Mouse/ Keyboard Test
rs - Serial Ports Test
ar - Archieve Tape Bootpath Test
mt - TapeMaster Bootpath Test
sd - SCSI Disk Bootpath Test
st - SCSI Tape Bootpath Test
xt - Xylogics Tape Bootpath Test
xy - xylogics Disk Bootpath Test

Cmd= >

D'l [1995u 0]v' **D'l 0 |444u-1v'**

Extended Test Menu for Sun-2/ 120/ 170.

1. Chosing option "ie" from the Main Menu will present the "Ethernet Menu", complete with all of the ethernet tests.(Sun-2/ 50/ 130/ 160 only)
2. Option "mk", will invoke the Keyboard and Mouse Menu. This menu will contain a list of keyboard and mouse tests.
3. Option "me" will invoke the memory sub-menu which contains all of the memory tests .
4. Option "rs" will invoke the Serial Ports Menu. This menu contains all of the serial ports tests.
6. Chosing option "vi" from the Main Menu will present the "Video Menu", complete with all of the video tests.
7. Option "kb" has no local menu and allows to test the keys. Keyboard Menu. This menu will contain a list of all the keyboard specific tests.
8. The rest of the tests will exercise the functionality of the bootpath to the specific device.

8.4.4.2. Ethernet Menu(VME systems only)

The Ethernet Menu will contain three *local* options. The three ethernet tests check the functionality of the Intel 82501 Serial Interface Unit Chip, the Intel 82586 Ethernet LAN Coprocessor Chip and the connection to the ethernet network. These options are explained below.

1. Option "local" will test the Intel 82586 Ethernet LAN co-processor chip. Also will initiate the internal tests which are built into the chip. Prior to the test, the Intel 82586 Ethernet LAN co-processor chip will disconnect itself from the Intel 82501 Serial Interface Unit Chip.
2. Option "Encoder" will perform an internal loop back test of the Intel 82501 Serial Interface Unit Chip. In particular, the transmitter line, receiver line, noise filters and Manchester encoding/decoding logic are tested. Prior to executing this test, the transmitter and receiver lines of the chip will be connected.
3. Option "External" will execute the external loop back test. This test checks the Intel 82586 Ethernet LAN Co-processor Chip, the Intel 82501 Serial Interface Unit Chip and the ethernet transceiver and receiver lines. The test will send data out onto the ethernet and receive them back before comparing the sent and received data. In order to run this test, an ethernet transceiver cable must be attached to the CPU board and the terminator assemblies of the transceiver black box.

D'l |1963u 0'

Intel Ethernet Tests: (Enter 'q' to return to Test Menu)

Cmd - Test

l - Local Loopback Test
e - Encoder Loopback Test
x - External Loopback Test

Cmd= >

D'l [illegible] D'l 0 |1803u-1v'

Ethernet tests Menu.

8.4.4.3. Keyboard and Mouse Menu

The Keyboard and Mouse Menu will contain a total of four *local* options. These options are presented next.

1. Option "Wr/Rd SCC Reg" will perform write-read-compare cycles to register 12 of the port under test. Again, the same three optional arguments which will apply to the "External" testwill apply here.
2. If option "Xmit" is chosen, a pattern will be written to the port under test. As is true for the "External" option, the "Transmit" option will accept the three optional arguments covered earlier.
3. Chosing option "Internal" will perform internal loop back write-read-compare cycles on the port under test. The transmitter and receiver lines of the requested port will be connected internally prior to the test. Similar to the "External" option, option "Internal" will also accept the three optional arguments described previously.

4. The option "External" will execute the external loop back test on a user-specified port. Basically, write-read-compare cycles are performed on the port under test. In order to run this test, the within-port external loop back cable must be installed.

   Option "External" will accept as many three arguments. Argument "Channel" will determine which port the test is performed on. By default, the value of "Channel" will be "Keyboard". The "Baud" argument will indicate the baud rate at which the test is executed. The default baud rate will be 1200 and "Baud" will expect a base 10 number. Finally, the "Pattern" argument specifies which pattern is written to the port. By default, the to-be-written pattern will be 0xaa and "Pattern" will be looking for a base 16 number.

```
D'l |2721u 0'


 Mouse/ Keyboard Ports Tests: (Enter 'q' to return to Test Menu)

 Enter port cmd: Cmd [port(M or K)] [Baud rate (decimal # 1)] [hex byte pattern]

 Cmd - Test


   w - Wr/ Rd SCC Reg 12 Test
   x - Xmit Char Test
   i - Internal Loopback test
   e - External Loopback Test


 Cmd= >
D'l [2 7 2 1 9 7 0 : 1 v '                                                    D
```

```
 Mouse/ Keyboard Tests Menu.
```

8.4.4.4. Memory Menu

The Memory Menu will contain a total of six *local* options. These options are listed next.

1. Option "a" will execute the address test on a range of memory. Specifically, write-read-compare cycles will be performed on long words. The datum which will be written to each memory "cell" will be its own address. Option "Address" will accept a maximum of two arguments. Argument "Low" will specify the first address to be tested. By default, the value of "Low" will be 0x2000 and "Low" will expect a base 16 number. The "High" argument will indicate the final address which is to be tested. The default high address will be the highest memory address available and "High" will also expect a base 16 number.

2. If option "c" is chosen, write-read-compare cycles will be performed on a range of addresses with a specified pattern. Only long words will be involved in the testing.
3. Option "r" will read the long words within a range of addresses and compare the observed values with an expected value. The check test will accept a maximum of three optional arguments. Arguments "Low" and "High" will indicate the range of addresses which will be involved during the test. They were covered above in the discussion corresponding to the address test. In addition, the "Pattern" argument will specify which pattern is expected throughout the range of addresses. The observed values will be compared against this expected value. By default, the value of "Pattern" will be 0xaaaaaaaa and argument "Pattern" will expect a base 16 number. The constant pattern test will accept a maximum of three optional arguments. Arguments "Low" and "High" will indicate the range of addresses which will be involved during the test. See option "Address" above for more detail. The "Pattern" argument will specify which pattern is written throughout the range of addresses. The default value of "Pattern" will be 0xaaaaaaaa and argument "Pattern" will expect a base 16 number.
4. Selecting option "s" will read the specifed range of address to observe any parity errors.
5. Selecting option "w" will simply write a pattern to a range range of addresses. The same three arguments which will be available for the constant pattern test will be applicable here. Refer to option "c" for more detail.

```
D'l |2518u 0'

 Memory Tests:  (Enter 'q' to return to Test Menu)

 Enter Cmd [low addr . 0x2000 [hi addr < 0xXXXXXX] [hex pattern]

 Cmd - Test

   a - Address Test
   c - Wr/Rd Pattern Test
   s - Scan Memory Test
   w - Write Pattern Test

 Cmd= >
D'l [2518u 0:1v'                                          D'l 0 |28
Memory Tests Menu.
```

8.4.4.5. Serial Ports Menu

The Serial Ports Menu will contain a total of five *local* options. For the explanation of each test reader is referd to **Keyboard/Mouse** tests.

D'l |2662u 0'

Serial Ports Tests: (Enter 'q' to return to Test Menu)

Enter port cmd: Cmd [port(A or B)] Baud rate(decimal #)] [hex byte pattern]

Cmd - Test

w - Wr/ Rd SCC Reg 12 Test
x - Xmit Char Test
i - Internal Loopback test
e - External Loopback Test

Cmd= >
D'l [2662u 0'v' D'l 0 |4

Serial Ports Tests Menu.

8.4.4.6. Video Menu

The Video Menu will contain a total of five *local* options. Each of these six options is covered below.

For the explanations of each local option please refer to **Memory Menu**.

The only difference is in address domain which will be the frame buffer.
D'l |2533u 0'

Video Tests: (Enter 'q' to return to Test Menu)

Enter Cmd [low addr > 0xXXXXXX [hi addr < 0xXXXXXX] [hex pattern]

a - Address Test
c - Wr/ Rd/ Compare Pattern Test
s - Scan Memory Test
w - Write Pattern Test

Cmd= >
D'l [2533u 0'v' D'l 0 |2877

Video Tests Menu.

8.4.4.7. Keyboard Test

Selecting option "Keyboard" will invoke the Keyboard input test which determines whether or not keyboard characters are correctly transmitted to the CPU. After invoking this test, (1) the ASCII code corresponding to a character and (2) the character itself will appear on the screen as you press keys on the keyboard. At any moment by typing an < ESC> character it will return to the main menu.

8.4.4.8. Options Menu

The Options Menu will enable users to specify the sequence of testing . The default is to do the test once and return to the menu.

1. The **f** option will allow the testing in an endless loop.
2. The **h** option will allow the testing in an endless loop except that if an error occurs the testing halts.
3. The **l** option will run the test once but enters a scope loop upon the occurrence of an error.
4. The **n** option will run the test for ever and the error occurance has no effect on it.

D'l |1665u 0'

Test Options: (Enter 'q' to return to Test Menu)

Cmd - Option

f - Loop forever
h - loop forever with halt on error
l - Loop once with loop on error
n - Loop forever with error messages inhibited
< cr> - Loop once

Cmd= >

D'l [1665u-1v' **D'l 0 |2115u-1v'**
Options menu.

9. Summary

In conclusion, this document presented the design specifications for Sun-2 firmware. This Firmware while has many new advanced features maintains its compatibility to the previous Sun-2 firmware releases.

Each of the four areas of responsibility associated with Sun-2 firmware, (power-up tests, initialization, boot sequences and monitor tools) were covered.

10. Sunromvec Layout

Sun ROM vector table is the first thing in the PROM. It provides the RESET vector which starts everything on power-up, as well as assorted information about where to find things in the ROMs and in low memory. There's a header file sunromvec.h associated with this vector table which defines the entire interface between the PROM Monitor and programs (or kernels) that run under it. This vector table is the only knowledge the outside world has of this PROM. It is referenced by hardware (Reset SSP, PC), and software. Once located, no entry can be relocated unless you change the world that needs it. The easiest way to reference elements of this vector is to say *romp-> xxx, such as (*romp-> v_putchar) (c).

According to sunromvec.h, the ROM vector table layout is discussed here in terms of C structure. The structure's name is sunromvec, and we define romp as a pointer to this structure. Each element of the structure is 32 bit long. It may be a pointer to a function, a pointer to a variable, or a variable. The following is the description of each element in the structure.

char *v_initsp

Initial SSP for hardware RESET.

void (*v_startmon)()

Initial PC for hardware RESET.

int *v_diagberr

Bus error handler for diagnostics.

struct bootparam **v_bootparam

Information for boot parameters.

unsigned *v_memorysize

Totol physical memory in bytes.

unsigned char (*v_getchar)()

Get character from current input source.

void (*v_putchar)()

Put character to current output sink.

int (*v_mayget)()

Maybe get character from current input source, or return -1.

int (*v_mayput)();

Maybe put character to current output sink, or return -1.

unsigned char *v_echo

Should getchar echo its input?

unsigned char *v_insource

Input source selector.

unsigned char *v_outsink

Output sink selector.

int (*v_getkey)()

Get next translated key if one exists.

void (*v_initgetkey)()

Initialize before first getkey.

unsigned int *v_translation

Up/down keyboard translation selector.

unsigned char *v_keybid

Up/down keyboard ID byte.

int *v_screen_x

V2: R/O value of current X position on screen.

int *v_screen_y

V2: R/O value of current Y position on screen.

struct keybuf *v_keybuf

Up/down keycode buffer.

char *v_mon_id

New location of monitor revision information.

void (*v_fwritechar)()

Write a character to FB "terminal".

int *v_fbaddr

Address of frame buffer.

char **v_font

Address of current font definition.

void (*v_fwritestr)()

Write a string to FB terminal - faster.

void (*v_boot_me)()

Boot with the specified parameter (like "b" command.)

unsigned char *v_linebuf

The line input buffer.

unsigned char **v_lineptr

Current pointer into line input buffer.

int *v_linesize

Total length of line in line buffer.

void (*v_getline)()

Fill line buffer from current input source.

unsigned char (*v_getone)()

Get next character from line buffer.

unsigned char (*v_peekchar)()

Peek at next character without reading it.

int *v_fbthere

Is frame buffer physically there? (1 = yes)

int (*v_getnum)()

Get next numerics and translate to binary.

int (*v_printf)()

Print a null-terminated string, which is similar to "kernel printf".

void (*v_printhex)()

Print N digits of a longword in hex.

unsigned char *v_leds

RAM copy of LED register value.

void (*v_set_leds)()

Sets LED register and RAM copy to argument value.

void (*v_nmi)()

Address for level 7 vector.

void (*v_abortent)()

Monitor entry point from keyboard abort.

int *v_nmiclock

Refresh routines's millisecond count.

int *v_fbtype

Which type of frame buffer do we have at runtime?

unsigned v_romvec_version

Version number of sunromvec.

struct globram *v_gp

Pointer to global data structure.

struct zscc_device *v_keybzscc

Address of keyboard in use.

int *v_keyrinit

Millisecond to wait before repeating a held key.

unsigned char *v_keyrtick

Millisecond to wait between repetitions.

unsigned *v_memoryavail

V1: Main memory usable size.

long *v_resetaddr

Vector address for watchdog resets.

long *v_resetmap

Page map entry for watchdog resets.

void (*v_exit_to_mon)()

Exit-to-monitor entry point.

unsigned char **v_memorybitmap

Pointer to pointer to memory bit map or 0.

void (*v_setcxsegmap)()

Routine to set segment map in any context.

void (v_vector_cmd)()**

V2: Handler for 'v' and low 'g' commands.

int dummy1z

Reserved (0).

int dummy2z

Reserved (0).

int dummy3z

Reserved (0).

int dummy4z

Reserved (0).

11. References

Sun-4 Firmware Design Document, Revision B,February 13, 1986, Tom Kraus and Bob Harris.

Sun-3 CPU Board Boot PROM Design Document, Revision H, September 13, 1985, Bob Harris and Mike Polivick.

Sun-2 Processor Architecture, Version .5, May 22, 1984