[SPARC OBP cheatsheet](http://tryamka.blogspot.com/2013/11/sparc-obp-cheatsheet.html)

**Boot PROM Basics**

**Boot PROM(programmable read only memory):**  
It is a firmware (also known as the monitor program) provides:  
1. basic hardware testing & initialization before booting.   
2. contains a user interface that provide access to many important functions.  
3. enables the system to boot from wide range of devices.  
  
It controls the system operation before the kernel becomes available. It provides a user interface and firmware utility commands known as FORTH command set. These commands include the boot commands, the diagnostic commands & the commands for modifying the default configuration.  
  
**Command to determine the version of the Open Boot PROM on the system:**# /usr/platform/'uname -m'/sbin/prtdiag -v   
(output omitted)  
System PROM revisions:  
----------------------  
OBP 4.16.4 2004/12/18 05:21 Sun Blade 1500 (Silver)  
OBDIAG 4.16.4.2004/12/18 05:21  
  
# prtconf -v  
OBP 4.16.4 2004/12/18 05:21  
  
**Open Boot Architectures Standards:**  
It is based on IEEE standard #1275, according to which the open boot architecture should provide capabilities of several system tasks including:  
1. Testing and initializing system hardware  
2. Determining the system's hardware configuration   
3. Enabling the use of third-party devices booting the OS  
4. Providing an interactive interface for configuration, testing  and debugging  
  
**Boot PROM chip:**  
It is available in Sun SPARC system.  
It is located on the same board as the CPU.  
  
**FPROM(Flash PROM):**  
It is a re-programmable boot PROM used by Ultra workstations.   
It enables to load new boot program data into PROM using software.  
  
**System configuration Information:**  
Each Sun system has another important element known as System Configuration Information.  
This information includes the Ethernet or MAC address, the system host identification number(ID), and the user configurable parameters.  
  
The user configurable parameters in System Information is called **NVRAM (Non-Volatile Random Access) Variables** or **EEPROM (Electronically Erasable PROM) parameters**.  
  
Using these parameters we can control :  
1. POST(Power on self Test)  
2. Specify the default boot device  
3. perform other configuration settings  
  
**Note**:*Depending on the system these system configuration information is stored in NVRAM chip, a SEEPROM(Serially Electronically Erasable PROM) or a System Configuration Card(SCC).*  
  
The older systems used NVRAM chip which is located on the main system board and is removable. It contains Lithium Battery to provide the battery backup for configuration information. The battery also provides the system's time of day(TOD) function.  
  
New systems uses a non-removable SEEPROM chip to store the system configuration information. The chip is located on the main board and doesn't requires battery.  
  
In addition to NVRAM and SEEPROM chip, some systems uses a removable SCC(System Configuration Card) to store system configuration information. An SCC is inserted into the SCC reader.  
  
**Working of Boot PROM Firmware:**  
The Boot PROM firmware booting proceeds in following stages:  
1. When a system is turned on, It initiates low-level POST. The low level post code is stored in system's boot PROM. The POST code tests the most elementary functions of the system.  
  
2. After the low level post completes successfully, the Boot PROM firmware takes control. It probes memory and CPU.  
  
3. Next, Boot PROM probes bus devices and interprets their drivers to build a device tree.   
  
4. After the device tree is built, the boot PROM firmware installs the console.  
  
5. The Boot PROM displays the banner once the system initialization is complete.   
  
**Note:***The system determines how to Boot the the OS by checking the parameter stored in the Boot PROM and NVRAM.*  
**Stop key sequences:**  
It can be used to enable various diagnostics mode. The Stop Key sequences affect the OpenBoot PROM and help to define how POST runs when the system is powered on.  
  
**Using Stop Key Sequences:**  
When the system is powered on use :  
1. STOP+D to switch the boot PROM to the diagnostic mode. In this mode the variable "*diag-switch?*" is set true.  
2. STOP+N to set NVRAM parameters to the default value. You can release the key when the LED starts flashing on the key board.  
  
**Abort Sequences:**  
STOP+A puts the system into command entry mode for the OpenBoot PROM & interrupts any running program. When the OK prompt is displayed, the system is ready to accept OpenBoot PROM commandds.  
  
**Disabling the Abort Sequences:**  
1. Edit /etc/default/kbd and comment out the statement "*KEYBOARD\_ABORT=disable*".  
2. Run the command: #kbd -i  
Once the abort sequence is disabled, it can only be used during the boot process.

**Commonly used Open Boot Prompt (OBP) commands**  
  
**ok>banner**: It displays the system information such as the model name, the boot PROM version, the memory, the Ethernet addresses, and the host identification number (ID).  
  
**ok>boot:** It is used to boot the system  
It can be used with follwoing options:  
-s : for single user mode. Here only root user is allowed to log in.  
cdrom -s : for booting into single user mode using cdrom  
-a: To boot the system in interactive mdoe  
-r: To perform reconfiguration boot. This is used to detect and create entry for a newly attached device.  
-v: To display the detailed information on the console during the boot process.  
  
**ok>help:** It is used to list the main help categories of OpenBoot firmware. the help command can be used with specific keyword to get the corresponding help. For example:  
ok> help boot   
ok> help diag  
  
**ok>printenv**: To display the all the NVRAM parameters. This command displays the default and current values of parameter. It can be used with single parameters to display the corresponding value.  
e.g. printenv auto-boot? : This command displays the value of auto-boot variable.  
e.g. printenv oem-banner? : This command displays the status of variable oem-banner.  
e.g. printenv oem-banner : This command displays customized OEM banner information.  
e.g. printenv oem-logo? : This displays the status of the variable oem-logo.  
e.g. printenv oem-logo : This displays the oem-logo.  
e.g. printenv boot-device : It displays the default boot device.setenv : It is use to assign the value to the environment variable.  
e.g. setenv auto-boot?  false : This command sets the value of variable auto-boot to false.  
e.g. setenv oem-banner? true : This command sets the value of variable oem-banner to true. By default its value is false.  
e.g. setenv oem-banner <customized message> : This command sets the customized message for the OEM banner.  
e.g. setenv oem-log? true : It sets value of oem-logo? to true/false.  
e.g. setenv oem-logo <logo name> : It sets customized logo name.  
e.g. setenv boot-device cdrom/disk/net : It sets the default boot device.Emergency Open  
  
**ok>setenv:** It is used for setting NVRAM parameters.   
e.g. setenv autoboot? false: This command sets the autoboot? parameter to false.  
  
**ok>reset-all:** It functions similar to power cycle, and rclears all buffers & registers, and execute a powered off/power on command.  
  
**ok>set-defaults:** It is used to reset all parameter values to factory defalut. To restore a particular parameter to its default setting use set-default command followed by parameter name.  
e.g. set-default auto-boot?  
**Note**: *The set-default command can only be used with those parameters for which the default value is defined.*  
  
**The probe commands** are used to display all the peripheral devices connected to the system.  
ok> probe-ide : It displays all the disks & CD-ROMS attached to the on-board IDE Controller.  
ok> probe-scsi : It displays all peripheral devices connected to the primary on-board SCSI controller.  
ok> probe-scsi-all : It displays all peripheral devices connected to the primary on-board SCSI controller & additional SBUS or PCI SCSI controllers.  
  
**ok>sifting <OpenBoot PROM command>:** Shifting command with an OpenBoot PROM command as an parameter displays the the syntax of OpenBoot PROM command.  
  
**ok>.registers:** It displays the content of the OBP registers.  
  
**To ensure the system is not hung when probe command is used :**  
1. set the parameter auto-boot? to false.  
ok> setenv auto-boot? false  
2.Use reset-all command to clear all the buffers & registers.  
3. Confirm all the values of OBP registers are set to zero using .registers command.  
Now we are ready to use any probe command without any problem.  
  
  
**ok>.speed:** It displays the speed of the processor.  
**ok>.enet-addr:** It displays the MAC address of the NIC  
**ok>.version:** It displays the release and version information of PROM chip.  
**ok> show-disks:** It displays all the connected disks/CD-ROM  
**ok> page :** To clear the screen  
**ok> watch-net:** It displays the NIC status.ok> test-all : It is nothing but performing POST i.e. self testing all the connected devices.  
**ok>sync:** It manually attempts to flush memory and synchronize the file system.  
**ok>test:**It is used to perform self test on the device specified.  
  
  
**Device Tree:**  
It is used to organize the devices attached to the system.   
It is built by the OpenBoot Firmware by using the information collected at the POST.  
  
**Node of the device tree:**  
1. The top most node of the device tree is the root device node.  
2. Bus nexus node follows the root device node.  
3. A leaf node(acts as a controller for the an attached device) is connected to the bus nexus node.  
**Examples:**  
1. The disk device path of an Ultra workstation with a PCI IDE Bus:  
/pci@1f,0/pci@,1/ide@3/dad@0,0  
  
/ -> Root device  
pci@1f,0/pci@,1/ide@3 -> Bus devices & controllers  
dad@ -> Device type(IDE disk)  
0 -> IDE Target address  
0 -> Disk number (LUN logical Unit Number)  
   
2. The disk device path of an Ultra workstation with a PCI SCSI Bus:  
/pci@1f,0/pci@,1/SUNW,isptwo@4/sd@3,0  
  
/ -> Root device  
pci@1f,0/pci@,1/SUNW,isptwo@4 -> Bus devices & controllers  
sd -> Device type(SCSI Device)  
3 -> SCSI Target address  
0 -> Disk number (LUN logical Unit Number)  
 **ok> show-devs:** Displays the list of all the devices in the OpenBoot device tree.  
  
**ok>devalias:** It is used to display the list of defined device aliases on a system.  
  
**Device aliases** provides shot names for longer physical device paths. The alias names are stored under NVRAMRC(contains registes to store the parameters). It is part of NVRAM.  
  
**Creating an alias name for device in Solaris**  
1. Use the show-disks command to list all the disks connected. Select and copy the location of the disk for which the alias need to be created. The partial path provided in show-disks command is completed by entering right targer & disk values.  
  
2. Use the following command to create the alias :  
nvalias <alias name> <physical path>  
The physical path is the location copied in step 1. The alias name can be anything of user choice.  
  
ok> devalias boot-device : It displays current boot devices alias for the system.  
ok> nvunalias <alias name>: It removes device alias name.  
  
**The /usr/sbin/eeprom command:**  
It is used to display & change the NVRAM parameters while Solaris OS is running.  
Note: It can be only used by root user.  
e.g. #eeprom -> list all the NVRAM parameters.  
e.g. #eeprom boot-device -> It lists the value of parameter boot-device  
e.g. #eeprom boot-device=disk2 -> Changes the boot-device parameter  
e.g. #eeprom auto-boot?=true -> Sets the parameter auto-boot? parameter to true  
e.g. #eeprom auto-boot? -> It lists the value of auto-boot? parameter  
  
**Interrupting an Unresponsive System:**  
1. Kill the unresponsive process & then try to reboot unresponsive system gracefully.  
2. If the above step fails, press STOP+A.  
3. use sync command at Open Boot prompt. This command creates panic situation in the system & synchronizes the file systems. Additionally, it creates a crash dump of memory and reboots system.

**GRUB (Grand Unified Loader for x86 systems only):**  
1. It loads the boot archive(contains kernel modules & configuration files) into the system's memory.  
2. It has been implemented on x86 systems that are running the Solaris OS.  
  
**Some Important Terms:**  
1. **Boot Archive:** Collection of important system file required to boot the Solaris OS. The system maintains two boot archive:  
2. **Primary boot archive:** It is used to boot Solaris OS on a system.  
3. **Secondary boot archive:** Failsafe Archive is used for system recovery in case of failure of primary boot archive. It is referred as Solaris  failsafe in the GRUB menu.  
4. **Boot loader:** First software program executed after the system is powered on.  
5. **GRUB edit Menu:** Submenu of the GRUB menu.  
  
**Additional GRUB Terms:**  
**1. GRUB main menu:** It lists the OS installed on a system.  
**menu.lst file:** It contains the OS installed on the system. The OS displayed on the GRUB main menu is determined by menu.lst file.  
**2. Miniroot:** It is a minimal bootable root(/) file system that is present on the Solaris installation media. It is also used as failsafe boot archive.  
  
**GRUB-Based Booting:**  
1. Power on system.  
2. The BIOS intializes the CPU, the memory & the platform hardware.  
3. BIOS loads the boot loader from the configured boot device. The BIOS then gives the control of system to the boot loader.  
  
The GRUB implementation on x86 systems in the Solaris OS is compliant with the multiboot specification. This enables to :  
1. Boot x86 systems with GRUB.  
2. individually boot different OS from GRUB.  
  
**Installing OS instances:**  
1. The GRUB main menu is based on a configuration file.  
2. The GRUB menu is automatically updated if you install or upgrade the Solaris OS.  
3. If another OS is installed, the /boot/grub/menu.lst need to be modified.  
  
**GRUB Main Menu:**   
It can be used to :   
1. Select a boot entry.  
2. modify a boot entry.  
3. load an OS kernel from the command line.  
  
**Editing the GRUB Main menu:**  
1. Highlight a boot entry in GRUB Main menu.  
2. Press 'e' to display the GRUB edit menu.  
3. Select a boot entry and press 'c'.  
  
**Working of GRUB-Based Booting:**  
1. When a system is booted, GRUB loads the primary boot archive & multiboot program. The primary boot archive, called /platform/i86pc/boot\_archive, is a RAM image of the file system that contains the Solaris kernel modules & data.  
2. The GRUB transfers the primary boot archive and the multiboot program to the memory without any interpretations.  
3. System Control is transferred to the multiboot program. In this situation, GRUB is inactive & system memory is restored.   
The multiboot program is now responsible for assembling core kernel modules into memory by reading the boot archive modules and passing boot-related information to the kernel.  
  
**GRUB device naming conventions:**  
(fd0), (fd1) : First diskete, second diskette  
(nd): Network device  
(hd0,0),(hd0,1): First & second fdisk partition of the first bios disk  
(hd0,0,a),(hd0,0,b): SOLARIS/BSD slice 0 & 1 (a & b) on the first fdisk partition on the first bios disk.  
  
**Functional Component of GRUB:**  
It has three functional components:  
**1. stage 1:** It is installed on first sector of SOLARIS fdisk partition  
**2. stage 2:**It is installed in a reserved areal in SOLARIS fdisk partition.  It is the core image of GRUB.  
**3. menu.lst:** It is a file located in /boot/grub directory. It is read by GRUB stage2 functional component.  
  
**The GRUB Menu**  
1. It contains the list of all OS instances installed on the system.  
2. It contains important boot directives.  
3. It requires modification of the active GRUB menu.lst file for any change in its menu options.  
  
**Locating the GRUB Menu:**  
#bootadm list-menu  
The location for the active GRUB menus is : /boot/grub/menu.lst  
  
Edit the menu.lst file to add new OS entries & GRUB console redirection information.  
Edit the menu.lst file to modify system behavior.  
  
**GRUB Main Menu Entries:**  
On installing the Solaris OS, by default two GRUB menu entries are installed on the system:  
1. **Solaris OS entry:** It is used to boot Solaris OS on a system.  
2. **miniroot(failsafe) archieve:** Failsafe Archive is used for system recovery in case of failure of primary boot archive. It is referred as Solaris  failsafe in the GRUB menu.  
  
**Modifying menu.lst:**   
When the system boots, the GRUb menu is displayed for a specific period of time. If the user do not select during this period, the system boots automatically using the default boot entry.   
The timeout value in the menu.lst file:  
1. determines if the system will boot automatically  
2. prevents the system from booting automatically if the value specified as -1.  
  
**Modifying X86 System Boot Behavior**  
**1. eeprom command:** It assigsn a different value to a standard set of properties. These values are equivalent to the SPARC OpenBoot PROM NVRAM variables and are saved in /boot/solaris/bootenv.rc  
**2. kernel command:** It is used to modify the boot behavior of a system.  
**3. GRUB menu.lst:**   
**Note:**   
*1.The kernel command settings override the changes done by using the eeprom command. However, these changes are only effective until you boot the system again.*  
*2. GRUB menu.lst is not preferred option because entries in menu.lst file can be modified during a software upgrade & changes made are lost.*  
  
**Verifying the kernel in use:**  
After specifying the kernel to boot using the eeprom or kernel commands, verify the kernel in use by following command:  
#prtconf -v | grep /platform/i86pc/kernel  
  
**GRUB Boot Archives**  
The GRUB menu in Solaris OS uses two boot archive:  
**1. Primary boot archive:**It shadows a root(/) file system. It contains all the kernel modules, driver.conf files & some configuration files. All these configuration files are placed in /etc directory. Before mounting the root file system the kernel reads the files from the boot archive. After the root file system is mounted, the kernel removes the boot archive from the memory.  
**2. failsafe boot archieve:** It is self-sufficient and can boot without user intervention. It does not require any maintenance. By default, the failsafe boot archive is created during installation and stored in /boot/x86.minor-safe.  
  
**Default Location of primary boot archive:**/platform/i86pc/boot\_archive  
  
**Managing the primary boot archive:**  
The boot archive :  
1. needs to be rebuilt, whenever any file in the boot archive is modified.  
2. Should be build on system reboot.  
3. Can be built using bootadm command  
#bootadm update-archive -f -R /a  
Options of the bootadm command:  
-f: forces the boot archive to be updated  
-R: enables to provide an alternative root where the boot archive is located.  
-n: enables to check the archive content in an update-archive operation, without updating the content.  
  
The boot archive can be rebuild by booting the system using the failsafe archive.  
  
**Booting a system in GRUB-Based boot environment:**  
 **Booting a System to Run Level 3(Multiuser Level):**  
To boot a system functioning at run level 0 to 3:  
1. reboot the system.  
2. press the Enter key when the GRUB menu appears.  
3. log in as the root & verify that the system is running at run level 3 using :  
#who -r  
  
Booting a system to run level S (Single-User level):  
1. reboot the system  
2. type e at the GRUB menu prompt.  
3. from the command list select the "kernel /platform/i86pc/multiboot" boot entry and type e to edit the entry.  
4. add a space and -s option at the end of the "kernel /platform/i86pc/multiboot -s" to boot at run level S.  
5. Press enter to return the control to the GRUB Main Menu.  
6. Type b to boot the system to single user level.  
7. Verify the system is running at run level S:  
#who -r  
8. Bring the system back to muliuser state by using the Ctrl+D key combination.  
  
**Booting a system interactively:**  
1. reboot the system  
2. type e at the GRUB menu prompt.  
3. from the command list select the "kernel /platform/i86pc/multiboot" boot entry and type e to edit the entry.  
4. add a space and -a option at the end of the "kernel /platform/i86pc/multiboot -a" .  
5. Press enter to return the control to the GRUB Main Menu.  
6. Type b to boot the system interactively.  
  
Stopping an X86 system:  
1. init 0  
2. init 6  
3. Use reset button or power button.  
  
**Booting the failsafe archive for recovery purpose:**  
1. reboot the system.  
2. Press space bar while while GRUB menu is displayed.  
3. Select Solaris failsafe entry and press b.  
4. Type y to automatically update an out-of-date boot archive.  
5. Select the OS instance on which the read write mount can happen.  
6. Type y to mount the selected OS instance on /a.  
7. Update the primary archive using following command:  
#bootadm update-archive -f -R /a  
8. Change directory to root(/): #cd /  
9. Reboot the system.  
  
**Interrupting an unresponsive system**  
1. Kill the offending process.  
2. Try rebooting system gracefully.  
3. Reboot the system by holding down the ctrl+alt+del key sequence on the keyboard.  
4. Press the reset button.  
5. Power off the system & then power it back on.