OpenBoot™ PROM Enhancements for Diagnostic Operation

Note – One of the OpenBoot™ PROM enhancements, a new standard (default) configuration, will increase the boot time of your system during a power cycle or after an error reset event. Note that there is no increase in boot time after a reset initiated by user commands from OpenBoot (`reset-all` or `boot`) or from Solaris™ (`reboot`, `shutdown`, or `init`). Before you power on your new system for the first time, see “Reference for Estimating System Boot Time (to the ok Prompt)” on page 12 of this document for information about the increased system boot time.

Sun Microsystems, Inc.
www.sun.com

Part No. 817-6957-10
August 2004, Revision A

Submit comments about this document at: `http://www.sun.com/hwdocs/feedback`

Contents

OpenBoot PROM Enhancements for Diagnostic Operation

This document describes the diagnostic operation enhancements provided by OpenBoot™ PROM Version 4.15 and later and presents information about how to use the resulting new operational features. Note that the behavior of certain operational features on your system might differ from the behavior described in this document. Check your system's Product Notes for information about differences that apply to your system.

This document is intended for system administrators who are experienced with setting and modifying OpenBoot configuration variables.

This document covers the following tasks:

It also includes the following sections:

What's New in Diagnostic Operation

The following features are the diagnostic operation enhancements:

- New and redefined configuration variables simplify diagnostic controls and allow you to customize a "normal mode" of diagnostic operation for your environment. See "About the New and Redefined Configuration Variables" on page 2.
- New standard (default) configuration enables and runs diagnostics and enables Automatic System Recovery (ASR) capabilities at power-on and after error reset events. See "About the New Standard (Default) Configuration" on page 3.
- Service mode establishes a Sun prescribed methodology for isolating and diagnosing problems. See "About Service Mode" on page 6.
- The `post` command executes the power-on self-test (POST) and provides options that enable you to specify the level of diagnostic testing and verbosity of diagnostic output. See "About the `post` Command" on page 10.

About the New and Redefined Configuration Variables

New and redefined configuration variables simplify diagnostic operation and provide you with more control over the amount of diagnostic output. The following list summarizes the configuration variable changes. See TABLE 1 for complete descriptions of the variables.

- New variables:
  - `service-mode?` places the system into service mode.
  - `diag-trigger` replaces and consolidates the functions of `post-trigger` and `obdiag-trigger`.
  - `verbosity` controls the amount and detail of firmware output.
- Redefined variable:
  - `diag-switch?` parameter has modified behaviors for controlling diagnostic execution in normal mode on Sun UltraSPARC™ based volume servers. Behavior of the `diag-switch?` parameter is unchanged on Sun workstations.
- Default value changes:
  - `auto-boot-on-error?` – New default value is `true`.
  - `diag-level` – New default value is `max`.
  - `error-reset-recovery` – New default value is `sync`.

About the New Standard (Default) Configuration

The new standard (default) configuration runs diagnostic tests and enables full ASR capabilities during power-on and after the occurrence of an error reset (RED State Exception Reset, CPU Watchdog Reset, System Watchdog Reset, Software-Instruction Reset, or Hardware Fatal Reset). This is a change from the previous default configuration, which did not run diagnostic tests. When you power on your system for the first time, the change will be visible to you through the increased boot time and the display of approximately two screens of diagnostic output produced by POST and OpenBoot Diagnostics.

Note – The standard (default) configuration does not increase system boot time after a reset that is initiated by user commands from OpenBoot (`reset-all` or `boot`) or from Solaris (`reboot`, `shutdown`, or `init`).

The visible changes are due to the default settings of two configuration variables, `diag-level` (`max`) and `verbosity` (`normal`):

- `diag-level` (`max`) specifies maximum diagnostic testing, including extensive memory testing, which increases system boot time. See “Reference for Estimating System Boot Time (to the ok Prompt)” on page 12 for more information about the increased boot time.
- `verbosity` (`normal`) specifies that diagnostic messages and information will be displayed, which usually produces approximately two screens of output. See “Reference for Sample Outputs” on page 14 for diagnostic output samples of `verbosity` settings `min` and `normal`.

After initial power-on, you can customize the standard (default) configuration by setting the configuration variables to define a “normal mode” of operation that is appropriate for your production environment. TABLE 1 lists and describes the defaults and keywords of the OpenBoot configuration variables that control diagnostic testing and ASR capabilities. These are the variables you will set to define your normal mode of operation.

Note – The standard (default) configuration is recommended for improved fault isolation and system restoration, and for increased system availability.

TABLE 1 OpenBoot Configuration Variables That Control Diagnostic Testing and Automatic System Recovery

OpenBoot Configuration Variable	Description and Keywords
`auto-boot?`	Determines whether the system automatically boots. Default is `true`. • `true` – System automatically boots after initialization, provided no firmware-based (diagnostics or OpenBoot) errors are detected. • `false` – System remains at the `ok` prompt until you type `boot`.
`auto-boot-on-error?`	Determines whether the system attempts a degraded boot after a nonfatal error. Default is `true`. • `true` – System automatically boots after a nonfatal error if the variable `auto-boot?` is also set to `true`. • `false` – System remains at the `ok` prompt.
`boot-device`	Specifies the name of the default boot device, which is also the normal mode boot device.
`boot-file`	Specifies the default boot arguments, which are also the normal mode boot arguments.
`diag-device`	Specifies the name of the boot device that is used when `diag-switch?` is `true`.
`diag-file`	Specifies the boot arguments that are used when `diag-switch?` is `true`.
`diag-level`	Specifies the level or type of diagnostics that are executed. Default is `max`. • `off` – No testing. • `min` – Basic tests are run. • `max` – More extensive tests might be run, depending on the device. Memory is extensively checked.
`diag-out-console`	Redirects system console output to the system controller. • `true` – Redirects output to the system controller. • `false` – Restores output to the local console. **Note:** See your system documentation for information about redirecting system console output to the system controller. (Not all systems are equipped with a system controller.)
`diag-passes`	Specifies the number of consecutive executions of OpenBoot Diagnostics self-tests that are run from the OpenBoot Diagnostics (`obdiag`) menu. Default is `1`. **Note:** `diag-passes` applies only to systems with firmware that contains OpenBoot Diagnostics and has no effect outside the OpenBoot Diagnostics menu.

TABLE 1 OpenBoot Configuration Variables That Control Diagnostic Testing and Automatic System Recovery *(Continued)*

OpenBoot Configuration Variable	Description and Keywords
`diag-script`	Determines which devices are tested by OpenBoot Diagnostics. Default is `normal`. • `none` – OpenBoot Diagnostics do not run. • `normal` – Tests all devices that are expected to be present in the system's baseline configuration for which self-tests exist. • `all` – Tests all devices that have self-tests.
`diag-switch?`	Controls diagnostic execution in normal mode. Default is `false`. *For servers:* • `true` – Diagnostics are *only* executed on power-on reset events, but the level of test coverage, verbosity, and output is determined by user-defined settings. • `false` – Diagnostics are executed upon next system reset, but only for those class of reset events specified by the OpenBoot configuration variable `diag-trigger`. The level of test coverage, verbosity, and output is determined by user-defined settings. *For workstations:* • `true` – Diagnostics are *only* executed on power-on reset events, but the level of test coverage, verbosity, and output is determined by user-defined settings. • `false` – Diagnostics are disabled.
`diag-trigger`	Specifies the class of reset event that causes diagnostics to run automatically. Default setting is `power-on-reset error-reset`. • `none` – Diagnostic tests are not executed. • `error-reset` – Reset that is caused by certain hardware error events such as RED State Exception Reset, Watchdog Resets, Software-Instruction Reset, or Hardware Fatal Reset. • `power-on-reset` – Reset that is caused by power cycling the system. • `user-reset` – Reset that is initiated by an operating system panic or by user-initiated commands from OpenBoot (`reset-all` or `boot`) or from Solaris (`reboot`, `shutdown`, or `init`). • `all-resets` – Any kind of system reset. **Note:** Both POST and OpenBoot Diagnostics run at the specified reset event if the variable `diag-script` is set to `normal` or `all`. If `diag-script` is set to `none`, only POST runs.
`error-reset-recovery`	Specifies recovery action after an error reset. Default is `sync`. • `none` – No recovery action. • `boot` – System attempts to boot. • `sync` – Firmware attempts to execute a Solaris `sync` callback routine.

TABLE 1 OpenBoot Configuration Variables That Control Diagnostic Testing and Automatic System Recovery *(Continued)*

OpenBoot Configuration Variable	Description and Keywords
`service-mode?`	Controls whether the system is in service mode. Default is `false`. • `true` – Service mode. Diagnostics are executed at Sun-specified levels, overriding but preserving user settings. • `false` – Normal mode, unless overridden by the panel keyswitch. Diagnostics execution depends entirely on the settings of `diag-switch?` and other user-defined OpenBoot configuration variables. **Note:** If the panel keyswitch is in the Diagnostics position, the system will boot in service mode even if the `service-mode?` variable is `false`.
`test-args`	Customizes OpenBoot Diagnostics tests. Allows a text string of reserved keywords (separated by commas) to be specified in the following ways: • As an argument to the `test` command at the `ok` prompt. • As an OpenBoot variable to the `setenv` command at the `ok` or `obdiag` prompt. **Note:** The variable `test-args` applies only to systems with firmware that contains OpenBoot Diagnostics. See your system documentation for a list of keywords.
`verbosity`	Controls the amount and detail of OpenBoot, POST, and OpenBoot Diagnostics output. Default is `normal`. • `none` – Only error and fatal messages are displayed on the system console. Banner is not displayed. **Note:** Problems in systems with `verbosity` set to `none` might be deemed not diagnosable, rendering the system unserviceable by Sun. • `min` – Notice, error, warning, and fatal messages are displayed on the system console. Transitional states and banner are also displayed. • `normal` – Summary progress and operational messages are displayed on the system console in addition to the messages displayed by the `min` setting. The work-in-progress indicator shows the status and progress of the boot sequence. • `max` – Detailed progress and operational messages are displayed on the system console in addition to the messages displayed by the `min` and `normal` settings.

About Service Mode

Service mode is an operational mode defined by Sun that facilitates fault isolation and recovery of systems that appear to be nonfunctional. When initiated, service mode overrides the settings of key OpenBoot configuration variables.

Note that service mode does not change your stored settings. After initialization (at the ok prompt), all OpenBoot PROM configuration variables revert to the user-defined settings. In this way, you or your service provider can quickly invoke a known and maximum level of diagnostics and still preserve your normal mode settings.

TABLE 2 lists the OpenBoot configuration variables that are affected by service mode and the overrides that are applied when you select service mode.

TABLE 2 Service Mode Overrides

OpenBoot Configuration Variable	Service Mode Override
auto-boot?	false
diag-level	max
diag-trigger	power-on-reset error-reset user-reset
input-device	Factory default
output-device	Factory default
verbosity	max
The following apply only to systems with firmware that contains OpenBoot Diagnostics:	
diag-script	normal
test-args	subtests,verbose

About Initiating Service Mode

The enhancements provide two mechanisms for specifying service mode:

- service-mode? configuration variable – When set to true, initiates service mode. (Service mode should be used only by authorized Sun service providers.)

Note – The diag-switch? configuration variable should remain at the default setting (false) for normal operation. To specify diagnostic testing for your operating environment, see "How to Initiate Normal Mode" on page 11.

- Panel keyswitch – When set to the Diagnostics position, initiates service mode.

Note – Not all systems are equipped with a panel keyswitch.

For instructions, see "How to Initiate Service Mode" on page 11.

About Overriding Service Mode Settings

When the system is in service mode, three commands can override service mode settings. TABLE 3 describes the effect of each command.

TABLE 3 Scenarios for Overriding Service Mode Settings

Command	Issued From	What It Does
`post`	`ok` prompt	OpenBoot firmware forces a one-time execution of normal mode diagnostics. • For information about normal mode, see "About Normal Mode" on page 8. • For information about `post` command options, see "About the `post` Command" on page 10.
`bootmode diag`	system controller	OpenBoot firmware overrides service mode settings and forces a one-time execution of normal mode diagnostics.[1]
`bootmode skip_diag`	system controller	OpenBoot firmware suppresses service mode and bypasses all firmware diagnostics.[1]

1 – If the system is not reset within 10 minutes of issuing the `bootmode` system controller command, the command is cleared.

Note – Not all systems are equipped with a system controller.

About Normal Mode

Normal mode is the customized operational mode that you define for your environment. To define normal mode, set the values of the OpenBoot configuration variables that control diagnostic testing. See TABLE 1 for the list of variables that control diagnostic testing.

Note – The standard (default) configuration is recommended for improved fault isolation and system restoration, and for increased system availability.

When you are deciding whether to enable diagnostic testing in your normal environment, remember that you always should run diagnostics to troubleshoot an existing problem or after the following events:

- Initial system installation
- New hardware installation and replacement of defective hardware
- Hardware configuration modification
- Hardware relocation
- Firmware upgrade
- Power interruption or failure
- Hardware errors
- Severe or inexplicable software problems

About Initiating Normal Mode

If you define normal mode for your environment, you can specify normal mode by either of the following methods:

- Panel keyswitch and `service-mode?` variable – When you set the panel keyswitch to the Normal or Locked position *and* the OpenBoot configuration variable `service-mode?` to `false`, this specifies normal mode.
- System controller `bootmode diag` command – When you issue this command, it specifies normal mode with the configuration values defined by you—with the following exceptions:
  - If you defined `diag-level` = `off`, `bootmode diag` specifies diagnostics at `diag-level` = `min`.
  - If you defined `verbosity` = `none`, `bootmode diag` specifies diagnostics at `verbosity` = `min`.

Note – The next reset cycle must occur within 10 minutes of issuing the `bootmode diag` command or the `bootmode` command is cleared and normal mode is not initiated.

For instructions, see “How to Initiate Normal Mode” on page 11.

About the post Command

The post command enables you to easily invoke POST diagnostics and to control the level of testing and the amount of output. When you issue the post command, OpenBoot firmware performs the following actions:

- Initiates a user reset
- Triggers a one-time execution of POST at the test level and verbosity that you specify
- Clears old test results
- Displays and logs the new test results

Note – The post command overrides service mode settings and pending system controller bootmode diag and bootmode skip_diag commands.

The syntax for the post command is:

```
post [level [verbosity]]
```

where:

- level = min or max
- verbosity = min, normal, or max

The level and verbosity options provide the same functions as the OpenBoot configuration variables diag-level and verbosity. To determine which settings you should use for the post command options, see TABLE 1 for descriptions of the keywords for diag-level and verbosity.

You can specify settings for:

- Both level and verbosity
- level only (If you specify a verbosity setting, you must also specify a level setting.)
- Neither level nor verbosity

If you specify a setting for level only, the post command uses the normal mode value for verbosity with the following exception:

- If the normal mode value of verbosity = none, post uses verbosity = min.

If you specify settings for neither `level` nor `verbosity`, the `post` command uses the normal mode values you specified for the configuration variables, `diag-level` and `verbosity`, with two exceptions:

- If the normal mode value of `diag-level` = `off`, `post` uses `level` = `min`.
- If the normal mode value of `verbosity` = `none`, `post` uses `verbosity` = `min`.

How to Initiate Service Mode

For background information, see “About Service Mode” on page 6.

What to Do

1. **Do one of the following:**
   - Set the `service-mode?` variable. At the `ok` prompt, type:

```
ok setenv service-mode? true
```

   - Turn the panel keyswitch to the Diagnostics position.

   For service mode to take effect, you must reset the system.

2. **At the `ok` prompt, type:**

```
ok reset-all
```

How to Initiate Normal Mode

For background information, see “About Normal Mode” on page 8.

What to Do

1. **Turn the panel keyswitch to the Normal or Locked position.**

2. **At the** ok **prompt, type:**

```
ok setenv service-mode? false
```

The system will not actually enter normal mode until the next reset.

3. **Type:**

```
ok reset-all
```

Reference for Estimating System Boot Time (to the ok Prompt)

Note – The standard (default) configuration does not increase system boot time after a reset that is initiated by user commands from OpenBoot (`reset-all` or `boot`) or from Solaris (`reboot`, `shutdown`, or `init`).

The measurement of system boot time begins when you power on (or reset) the system and ends when the OpenBoot `ok` prompt appears. During the boot time period, the firmware executes diagnostics (POST and OpenBoot Diagnostics) and performs OpenBoot initialization. The time required to run OpenBoot Diagnostics and to perform OpenBoot setup, configuration, and initialization is generally similar for all systems, depending on the number of I/O cards installed when `diag-script` is set to `all`. However, at the default settings (`diag-level` = `max` and `verbosity` = `normal`), POST executes extensive memory tests, which will increase system boot time.

System boot time will vary from system-to-system, depending on the configuration of system memory and the number of CPUs:

- Because each CPU tests its associated memory and POST performs the memory tests simultaneously, memory test time will depend on the amount of memory on the most populated CPU.
- Because the competition for system resources makes CPU testing a less linear process than memory testing, CPU test time will depend on the number of CPUs.

If you need to know the approximate boot time of your new system before you power on for the first time, the following sections describe two methods you can use to estimate boot time:

- If your system configuration matches one of the three typical configurations cited in “Boot Time Estimates for Typical Configurations” on page 13, you can use the approximate boot time given for the appropriate configuration.
- If you know how the memory is configured among the CPUs, you can estimate the boot time for your specific system configuration using the method described in “Estimating Boot Time for Your System” on page 13.

Boot Time Estimates for Typical Configurations

The following are three typical configurations and the approximate boot time you can expect for each:

- Small configuration (2 CPUs and 2 Gbytes of memory) – Boot time is approximately 5 minutes.
- Medium configuration (4 CPUs and 16 Gbytes of memory) – Boot time is approximately 10 minutes.
- Large configuration (8 CPUs and 64 Gbytes of memory) – Boot time is approximately 25 minutes.

Estimating Boot Time for Your System

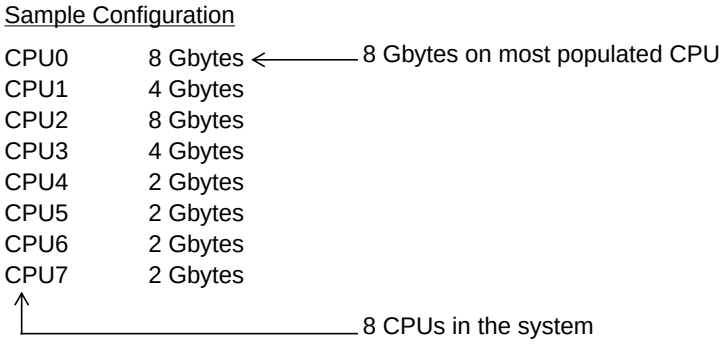
Generally, for systems configured with default settings, the times required to execute OpenBoot Diagnostics and to perform OpenBoot setup, configuration, and initialization are the same for all systems:

- 1 minute for OpenBoot Diagnostics testing might require more time for systems with a greater number of devices to be tested.
- 2 minutes for OpenBoot setup, configuration, and initialization

To estimate the time required to run POST memory tests, you need to know the amount of memory associated with the most populated CPU. To estimate the time required to run POST CPU tests, you need to know the number of CPUs. Use the following guidelines to estimate memory and CPU test times:

- 2 minutes per Gbyte of memory associated with the most populated CPU
- 1 minute per CPU

The following example shows how to estimate the system boot time of a sample configuration consisting of 8 CPUs and 32 Gbytes of system memory, with 8 Gbytes of memory on the most populated CPU.



Estimation of Boot Time

POST memory test	8 Gbytes	x	2 min per Gbyte	=	16 min
POST CPU test	8 CPUs	x	1 min per CPU	=	8 min
OpenBoot Diagnostics					1 min
OpenBoot initialization					2 min
Total system boot time (to the ok prompt)					27 min

Reference for Sample Outputs

At the default setting of `verbosity` = `normal`, POST and OpenBoot Diagnostics generate less diagnostic output (about 2 pages) than was produced before the OpenBoot PROM enhancements (over 10 pages). This section includes output samples for `verbosity` settings at `min` and `normal`.

Note – The `diag-level` configuration variable also affects how much output the system generates. The following samples were produced with `diag-level` set to `max`, the default setting.

The following sample shows the firmware output after a power reset when `verbosity` is set to `min`. At this `verbosity` setting, OpenBoot firmware displays notice, error, warning, and fatal messages but does not display progress or operational messages. Transitional states and the power-on banner are also displayed. Since no error conditions were encountered, this sample shows only the POST execution message, the system's install banner, and the device self-tests conducted by OpenBoot Diagnostics.

```
Executing POST w/%o0 = 0000.0400.0101.2041

Sun Fire V890, Keyboard Present
Copyright 1998-2004 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.15.0, 4096 MB memory installed, Serial #12980804.
Ethernet address 8:0:20:c6:12:44, Host ID: 80c61244.

Running diagnostic script obdiag/normal

Testing /pci@8,600000/network@1
Testing /pci@8,600000/SUNW,qlc@2
Testing /pci@9,700000/ebus@1/i2c@1,2e
Testing /pci@9,700000/ebus@1/i2c@1,30
Testing /pci@9,700000/ebus@1/i2c@1,50002e
Testing /pci@9,700000/ebus@1/i2c@1,500030
Testing /pci@9,700000/ebus@1/bbc@1,0
Testing /pci@9,700000/ebus@1/bbc@1,500000
Testing /pci@8,700000/scsi@1
Testing /pci@9,700000/network@1,1
Testing /pci@9,700000/usb@1,3
Testing /pci@9,700000/ebus@1/gpio@1,300600
Testing /pci@9,700000/ebus@1/pmc@1,300700
Testing /pci@9,700000/ebus@1/rtc@1,300070

{7} ok
```

The following sample shows the diagnostic output after a power reset when `verbosity` is set to `normal`, the default setting. At this `verbosity` setting, the OpenBoot firmware displays summary progress or operational messages in addition to the notice, error, warning, and fatal messages; transitional states; and install banner displayed by the `min` setting. On the console, the work-in-progress indicator shows the status and progress of the boot sequence.

```
Hardware Power On

Probing core system FRUs..

Executing POST w/%o0 = 0000.0800.0101.2041
4:0>
4:0>@(#) Sun Fire V890 POST 4.15.0 2004/04/12 10:17
4:0>Copyright © 2004 Sun Microsystems, Inc. All rights reserved
  SUN PROPRIETARY/CONFIDENTIAL.
  Use is subject to license terms.
4:0>Jump from OBP->POST.
4:0>Diag level set to MIN.
4:0>
4:0>Start selftest...
4:0>CPUs present in system: 4:0 5:0 6:0 7:0
4:0>Test CPU(s)....Done
4:0>Init Scan/I2C....Done
4:0>Basic Memory Test....Done
4:0>Memory Block....Done
4:0>IO-Bridge Tests....Done
4:0>Enable Errors....Done
4:0>INFO:
4:0>    POST Passed all devices.
4:0>POST:       Return to OBP.
POST Reset

Enabling system bus....... Done
Probing Memory............ Done
Initializing CPUs......... Done
Initializing boot memory.. Done

Initializing OpenBoot
Probing system devices
Probing I/O buses
```

```
Sun Fire V890, Keyboard Present
Copyright 1998-2004 Sun Microsystems, Inc.  All rights reserved.
OpenBoot 4.15.0, 4096 MB memory installed, Serial #12980804.
Ethernet address 8:0:20:c6:12:44, Host ID: 80c61244.

Running diagnostic script obdiag/normal

Testing /pci@8,600000/network@1
Testing /pci@8,600000/SUNW,qlc@2
Testing /pci@9,700000/ebus@1/i2c@1,2e
Testing /pci@9,700000/ebus@1/i2c@1,30
Testing /pci@9,700000/ebus@1/i2c@1,50002e
Testing /pci@9,700000/ebus@1/i2c@1,500030
Testing /pci@9,700000/ebus@1/bbc@1,0
Testing /pci@9,700000/ebus@1/bbc@1,500000
Testing /pci@8,700000/scsi@1
Testing /pci@9,700000/network@1,1
Testing /pci@9,700000/usb@1,3
Testing /pci@9,700000/ebus@1/gpio@1,300600
Testing /pci@9,700000/ebus@1/pmc@1,300700
Testing /pci@9,700000/ebus@1/rtc@1,300070

{7} ok
```

Reference for Determining Diagnostic Mode

The flowchart in FIGURE 1 summarizes graphically how various system controller and OpenBoot variables affect whether a system boots in normal or service mode, as well as whether any overrides occur.

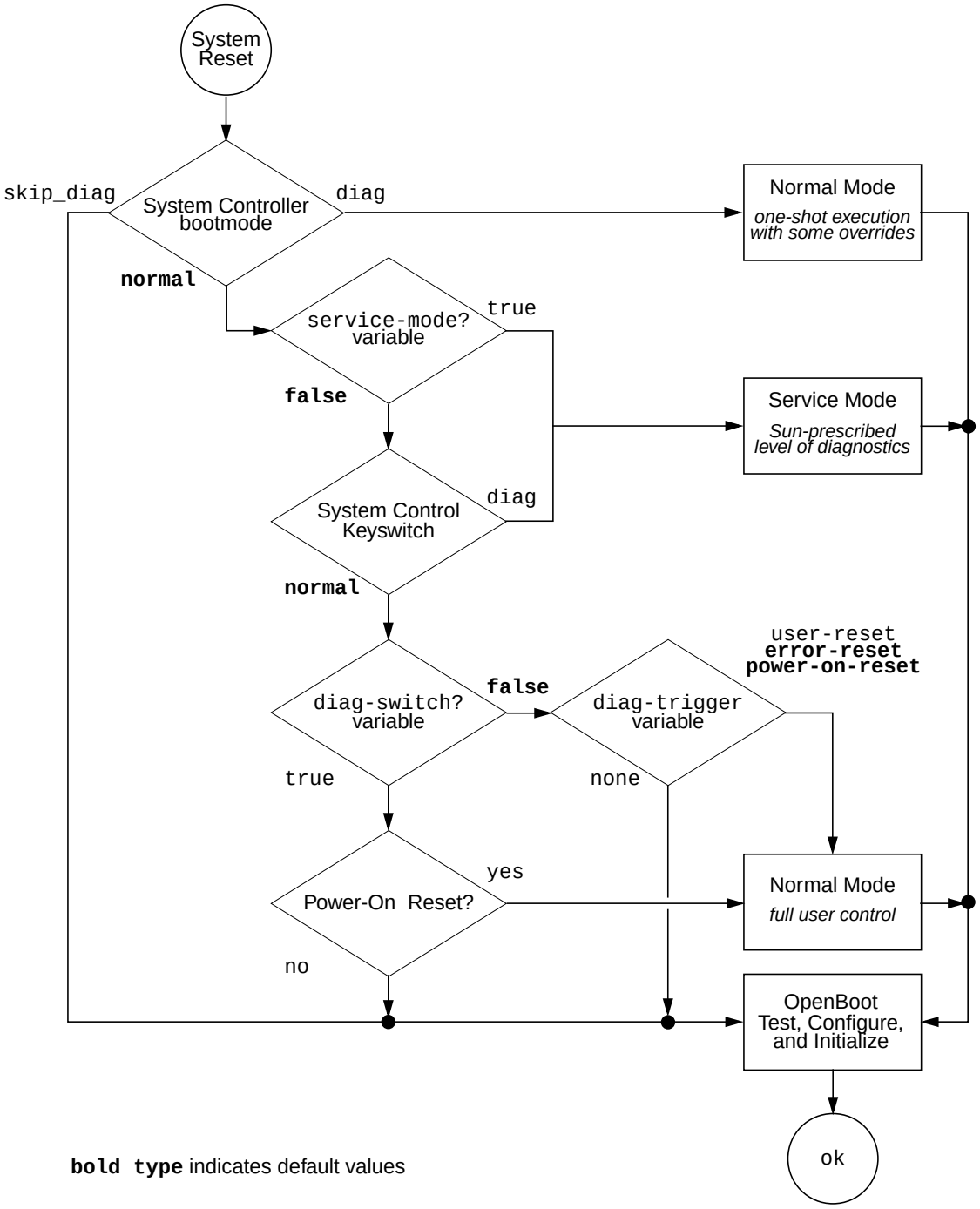


FIGURE 1 Diagnostic Mode Flowchart

Quick Reference for Diagnostic Operation

TABLE 4 summarizes the effects of the following user actions on diagnostic operation:

- Turn the panel keyswitch to the Diagnostics position
- Set `service-mode?` to `true`
- Turn the panel keyswitch to the Normal or the Locked position
- Issue the `bootmode` commands, `bootmode diag` or `bootmode skip_diag`
- Issue the `post` command

TABLE 4 Summary of Diagnostic Operation

User Action	Sets Configuration Variables	And Initiates
	Service Mode	
Turn keyswitch to Diagnostics position or Set `service-mode?` to `true`	**Note:** Service mode overrides the settings of the following configuration variables without changing your stored settings: • `auto-boot?` = `false` • `diag-level` = `max` • `diag-trigger` = `power-on-reset error-reset user reset` • `input-device` = Factory default • `output-device` = Factory default • `verbosity` = `max` The following apply only to systems with firmware that contains OpenBoot Diagnostics: • `diag-script` = `normal` • `test-args` = `subtests,verbose`	Service mode (defined by Sun)
	Normal Mode	
Turn keyswitch to Normal or Locked position and set `service-mode?` to `false`	• `auto-boot?` = user-defined setting • `auto-boot-on-error?` = user-defined setting • `diag-level` = user-defined setting • `verbosity` = user-defined setting • `diag-script` = user-defined setting • `diag-trigger` = user-defined setting • `input-device` = user-defined setting • `output-device` = user-defined setting	Normal mode (user-defined)
	`bootmode` Commands	
Issue `bootmode diag` command	Overrides service mode settings and uses normal mode settings with the following exceptions: • `diag-level` = `min` if normal mode value = `off` • `verbosity` = `min` if normal mode value = `none`	Normal mode diagnostics with the exceptions in the preceding column.
Issue `bootmode skip_diag` command		OpenBoot initialization without running diagnostics

TABLE 4 Summary of Diagnostic Operation *(Continued)*

User Action	Sets Configuration Variables	And Initiates
post Command **Note:** If the value of `diag-script` = `normal` or `all`, OpenBoot Diagnostics also run.		
Issue `post` command		POST diagnostics
Specify both `level` and `verbosity`	`level` and `verbosity` = user-defined values	
Specify neither `level` nor `verbosity`	`level` and `verbosity` = normal mode values with the following exceptions: • `level` = `min` if normal mode value of `diag-level` = `none` • `verbosity` = `min` if normal mode value of `verbosity` = `none`	
Specify `level` only	`level` = user-defined value `verbosity` = normal mode value for `verbosity` (**Exception:** `verbosity` = `min` if normal mode value of `verbosity` = `none`)	