Netra t 1120 / 1125 User's Guide

THE NETWORK IS THE COMPUTER™

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303-4900 USA
650 960-1300 Fax 650 969-9131

Part No.: 805-6802-10
Revision A, August 1998

Contents

Figures

Tables

Preface

The *Netra t 1120/1125 User's Guide* provides information about the system administration and software operation of the Netra t 1120 (order code N04) and Netra t 1125 (order code N03).

Note – This Guide does not apply to the version of Netra t 1120 supplied as order code N02.

Note – All illustrations in this guide are of the Netra t 1125, except where the two types of system differ, in which case examples of both are shown.

Who Should Use This Guide

This guide is intended to be read by the system users and administrators of the Netra t 1120/1125 computer system.

How This Guide Is Organized

The guide is arranged as follows:

Chapter 1, "Product Overview"**,** describes the key features of the Netra t 1120/1125 computer system.

Chapter 2, “System LEDs and Controls”, provides information on the power-on and -off procedures, system LEDs and connectors.

Chapter 3, “System Start-up and Operation”, describes the procedures required to boot the Netra t 1120/1125 system.

Chapter 4, “Open Boot PROM (OBP)”, explains where changes have been made to the Open Boot Prom (OBP) for the Netra t 1120/1125 system.

Chapter 5, “System Shut-down”, describes the procedures required to shut-down the Netra t 1120/1125 system cleanly and safely.

Appendix A, “Manual Pages”, contains the manual pages for the `tsalarm`(7D) driver and the `tsctl`(1M), `tsdog`(1M), `tsmonitor`(1M), `tsstate`(1M) and `tsunlock`(1M) utilities.

Accompanying Documentation

- Netra t 1120/1125 Compliance and Safety Manual (805-6806-10)

Note – It is important that you read the Netra t 1120/1125 *Compliance and Safety Manual* before doing anything else.

- Netra t 1120/1125 Installation and Basic Maintenance Guide (805-6803-10)
- Netra t 1120/1125 Service Manual (805-6804-10)
- Netra t 1120/1125 System Reference Manual (805-6805-10)

Conventions used in this Guide

The following table shows the type changes and symbols used in this guide.

TABLE P-1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `system% You have mail.`
`AaBbCc123`	What you type, as opposed to on-screen computer output	`system% su` `Password:`
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*`.`
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide.* These are called *class* options. You *must* be root to do this.
%	UNIX C shell prompt	`system%`
$	UNIX Bourne and Korn shell prompt	`system$`
#	super-user prompt, all shells	`system#`

Symbols

The following symbols mean:

Note – A note provides information which should be considered by the reader.

Caution – Cautions identified by this Attention icon carry information about procedures or events which if not considered may cause damage to the data or hardware of your system.

Caution – Cautions identified by this Hazard icon carry information about procedures which must be followed to reduce the risk of electric shock and danger to personal health. Follow all instructions carefully.

1125 Paragraphs accompanied by this 1125 icon apply only to Netra t 1125 systems.

1120 Paragraphs accompanied by this 1120 icon apply only to Netra t 1120 systems.

CHAPTER 1

Product Overview

The Netra t 1120/1125 computer system is a one- or two-processor device that uses the family of UltraSPARC™ II processors. Housed within a rack-mounting enclosure, the Netra t 1120/1125 provides the following:

- Increased power and cooling for high performance processors;
- Extensive I/O expansion and a wide range of options;
- Modular internal design;
- High performance disk, system, memory and I/O subsystems;
- High-performance peripheral component interconnect (PCI) I/O expansion with comparable options to existing SBus options.

The Netra t 1120 is powered by a –-48V/–60Vdc supply. The Netra t 1125 is powered by a standard AC supply. This is the only difference between the systems.

FIGURE 1-2 on page 3 and FIGURE 1-4 on page 4 illustrate the front of the Netra t 1120/1125 system; FIGURE 1-3 on page 4 and FIGURE 1-4 on page 4 illustrate the rear of the Netra t 1120/1125 system. The following sections provide a brief description of the Netra t 1120/1125 I/O devices and a detailed overview of the system unit features.

System Unit Features

System unit components are housed in a rack-mounting enclosure. Overall enclosure dimensions (width x depth x height) are 431.8mm x 496.1mm x 177mm (17in x 19.53in x 7in (4U)). System unit electronics are contained on a single printed circuit board (motherboard). The motherboard contains the CPU module(s), memory, system control application-specific integrated circuits (ASICs) and I/O ASICs.

The system unit has the following features:

- Rack-mounting enclosure with DC (Netra t 1120) or AC (Netra t 1125) power supply.
- Support for modular UltraSPARC II processors with 1, 2 or 4Mbyte cache, and system operating frequencies from 300MHz to 400MHz.
- UPA coherent memory interconnect.
- Use of SIMMs, with an interleaved memory system. Each pair of SIMM slots (four rows of two pairs each) accepts 32, 64 or 128Mbyte SIMM modules. Populating with two pairs of identical capacity SIMMs enables the memory controller to interleave and overlap, providing optimal system performance. There are a total of 16 SIMM slots.
- Four PCI slots:
  - Three 33MHz, 64- or 32-bit, 5Vdc slots
  - One 66MHz or 33MHz, 64- or 32-bit, 3.3Vdc slot

  Universal PCI cards can be used in any of the four PCI slots.
- 10/100megabit per second (Mbps) Ethernet.
- 40Mb/s UltraSCSI (Fast-20).
- Two RS232/423 DB-25 serial ports (asynchronous protocols).
- Up to two UltraSCSI 18Gbyte disks.
- Up to two removable media drives (CD-ROM and/or tape).
- Parallel port.

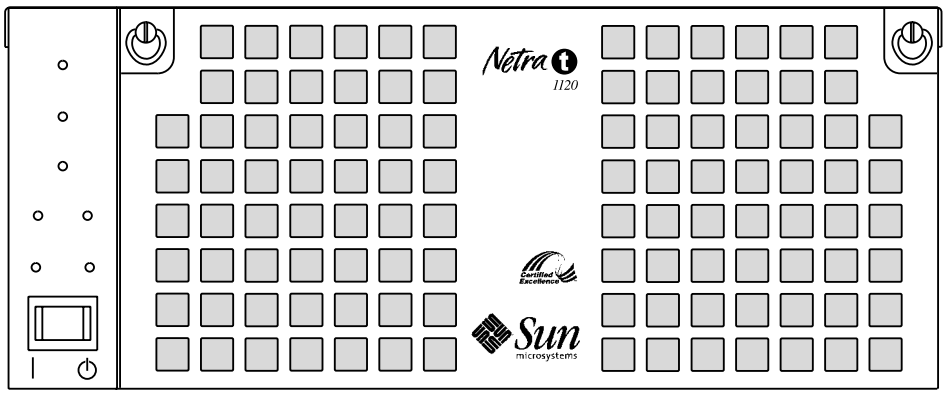


FIGURE 1-1 Netra t 1120 System Unit Front View

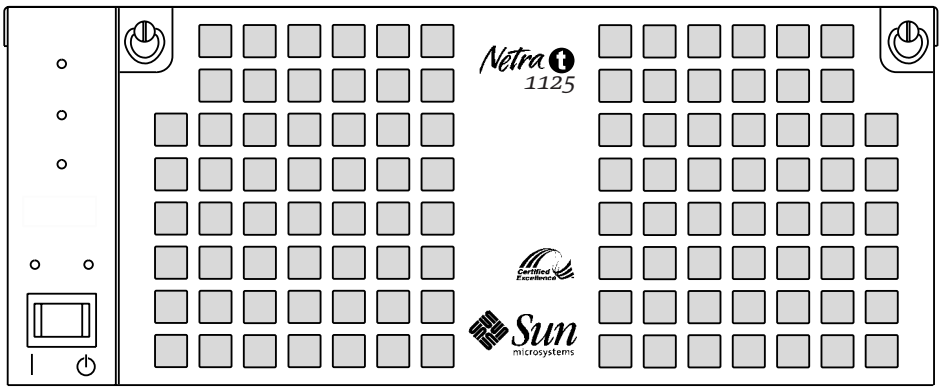


FIGURE 1-2 Netra t 1125 System Unit Front View

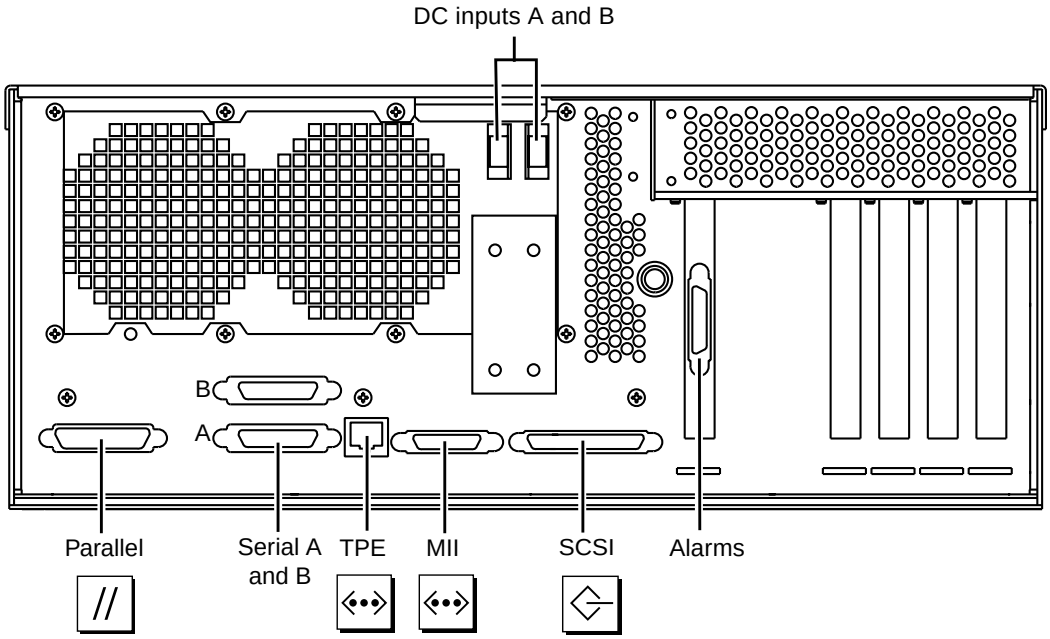


FIGURE 1-3 Netra t 1120 System Unit Rear View

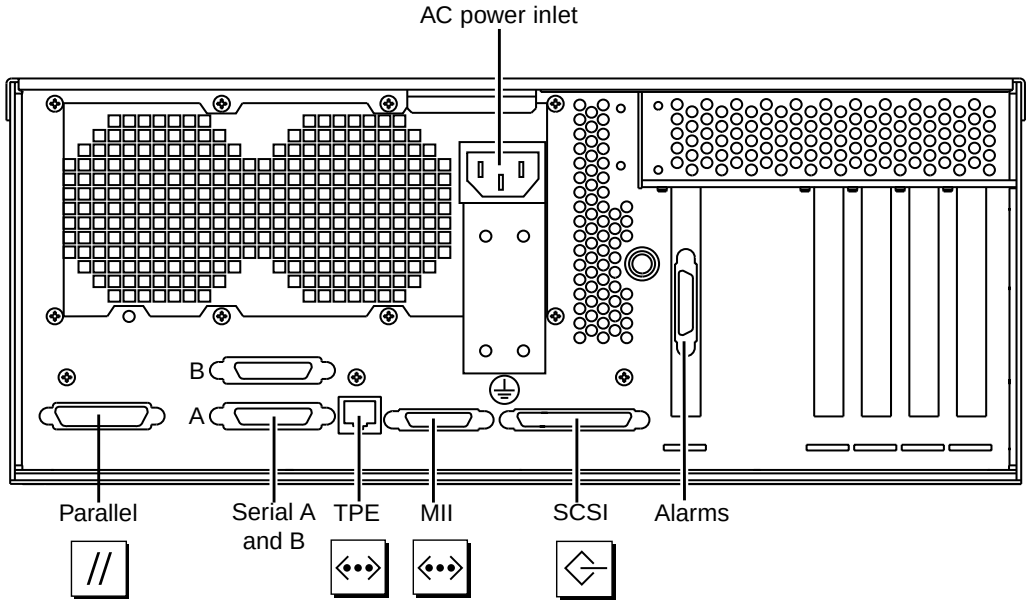


FIGURE 1-4 Netra t 1125 System Unit Rear View

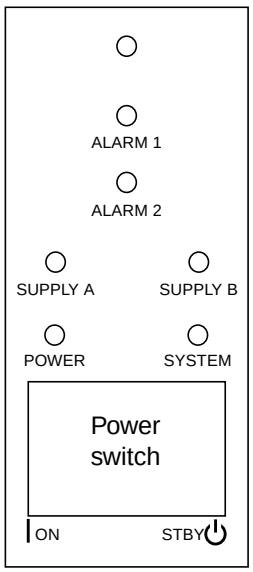
CHAPTER **2**

System LEDs and Controls

This chapter describes the system LEDs and the procedure for powering the system on and off.

System LEDs

Netra t 1120



POWER – *Green*
Illuminated at all times when the system is On.

SUPPLY A – *Green*
Illuminated whenever DC input A is present and the system is powered on.

SUPPLY B – *Green*
Illuminated whenever DC input B is present and the system is powered on.

SYSTEM – *Green*
Off (or reset) during power up procedures and is illuminated whenever UNIX is running and the alarms driver is installed. This LED is reset by a hardware Watchdog timeout, or whenever the user-defined Alarm3 is asserted.

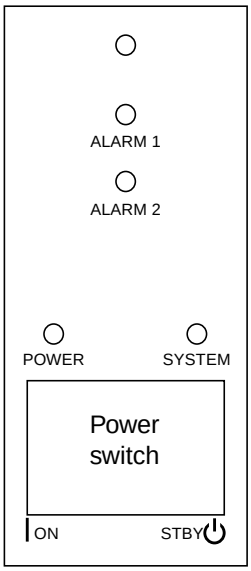
ALARM1 – *Amber*
Illuminated whenever the user-defined Alarm 1 is asserted.

ALARM2 – *Amber*
Illuminated whenever the user-defined Alarm 2 is asserted.

SPARE – *Amber*
For future enhancement.

FIGURE 2-1 Netra t 1120 System LEDs

Netra t 1125



POWER – *Green*
Illuminated at all times when the system is On.

SYSTEM – *Green*
Off (or reset) during power up procedures and is illuminated whenever UNIX is running and the alarms driver is installed. This LED is reset by a hardware Watchdog timeout, or whenever the user-defined Alarm3 is asserted.

ALARM1 – *Amber*
TIlluminated whenever the user-defined Alarm 1 is asserted.

ALARM2 – *Amber*
Illuminated whenever the user-defined Alarm 2 is asserted.

SPARE – *Amber*
For future enhancement.

FIGURE 2-2 Netra t 1125 System LEDs

System ON/STBY Switch

The system switch of the Netra t 1120/1125 is a rocker, momentary switch which functions as a standby device only, controlling logic circuits which enable power module output.

See the Netra t 1120/1125 *Installation and Basic Maintenance Guide* (Part No. 805-6041-05) for further information.

To Power On the System

1. **Turn on power to all connected peripherals.**

Note – Peripheral power is activated prior to system power so the system can recognize the peripherals when it is activated.

---

2. **Apply power to the system inlet.**
3. **Momentarily set the front panel ON/STBY system switch to the ON | position (**FIGURE 2-3**) and hold it until the system starts to power up.**

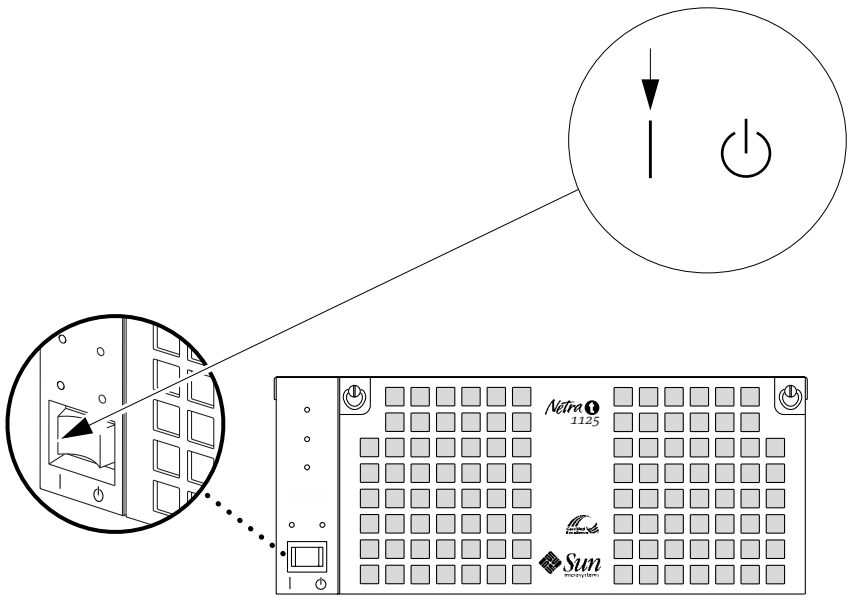


FIGURE 2-3 System Power-On (Front Panel)

To Power Off the System

Caution – Prior to turning off system power, exit from the operating system. Failure to do so may result in data loss.

1. **Where necessary, notify users that the system is going down.**
2. **Back up system files and data.**
3. **Halt the operating system.**
4. **Momentarily set the front panel ON/STBY power switch to the STBY position (**FIGURE 2-4**) until the system powers down.**

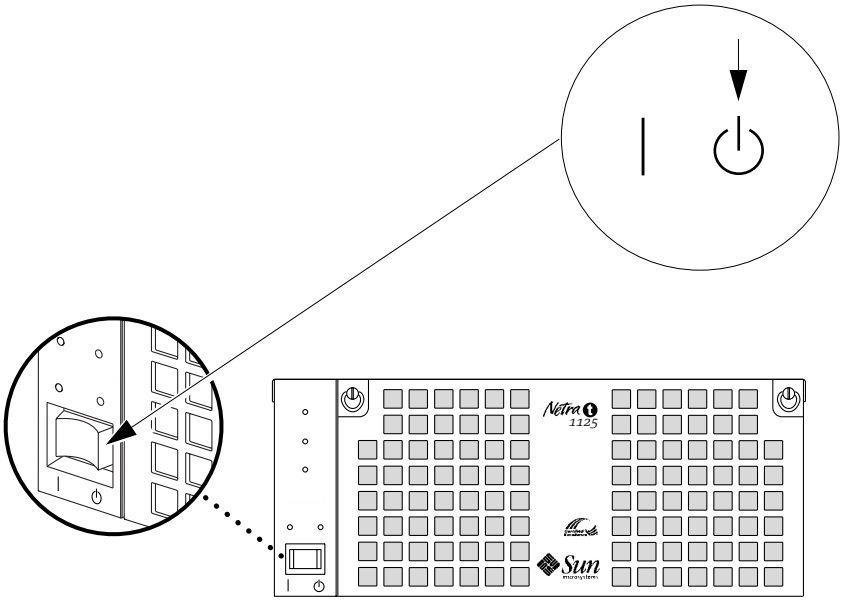


FIGURE 2-4 System Power-Off (Front Panel)

5. **Verify that the POWER LED is off.**
6. **Remove the input power connector from the power inlet**

Caution –

1120

Regardless of the position of the ON/STBY switch, where a DC power cord remains connected to the system, DC voltage is always present within the power supply.

1125

Regardless of the position of the ON/STBY switch, where an AC power cord remains connected to the system, hazardous voltage is always present within the power supply.

CHAPTER 3

System Start-up and Operation

This chapter provides information describing how to:

- Boot the Netra t 1120/1125 system.
- Modify the system Startup/Shutdown configuration parameters.

The OpenBoot firmware is part of the Netra t 1120/1125 boot PROM. Some changes may have been made and these are documented in Chapter 4, "Open Boot PROM (OBP)". Refer also to the *Solaris OpenBoot Command Reference* manual for a description of the OpenBoot environment (OBP), which is part of the boot PROM in a Netra t 1120/1125.

System Boot Procedure

In a Netra t 1120/1125 system, the boot procedure is carried out by the Sun OpenBoot PROM (OBP).

1. **Turn on the power and wait system tests are performed.**
2. **When the `ok` prompt is displayed, type `boot` and press Return.**
3. **When a login prompt is displayed, the system has booted.**

Note – When first installed, the super-user password is Return. Contact your support organization if you experience any problems.

System Start-Up

The standard *Solaris System User's Guide* contains a general explanation of how to start up and shut down a computer system. With that information and this section, you can start up and shut down the software on a Netra t 1120/1125. See also `boot`(1M), `kernel`(1M) and `kadb`(1M) for further information.

We recommend that you do not interrupt the boot sequence after powering up or rebooting a Netra t 1120/1125. When you boot the system from interactive mode, some of its automatic error-recovery features are disabled; error recovery is then dependent upon the user.

A Netra t 1120/1125 interactive mode booting sequence can be stopped to allow you to boot the system using a non-default UNIX kernel. To boot the system using a non-default UNIX kernel, use the `boot -a` option as documented in `boot`(1M).

If you initiate a boot sequence with the commands:

`shutdown -i6`

or,

`reboot`

you cannot interrupt this operation. See Chapter 5, "System Shut-down" for further information.

If the command `boot -s` (reboot into single-user mode) has been given, the message:

```
Type Ctrl-d to proceed with normal startup,
(or give root password for System Maintenance Mode)
```

appears when the boot process is complete. Type Ctrl-D to continue with normal startup, defaulting to multi-user mode. Alternatively, enter the root password for system maintenance mode.

CHAPTER 4

Open Boot PROM (OBP)

This chapter describes where changes have been made to the Open Boot PROM.

NVRAM Configuration Parameters

TABLE 4-1 lists the NVRAM configuration parameters supported by a Netra t 1120/1125 system.

TABLE 4-1 NVRAM Configuration Parameters

Parameter	Typical Default	Description
`auto-boot`	true	If true, boot automatically after power on or reset.
`boot-device`	disk	Device from which to boot.
`boot-file`	empty string	File to boot (an empty string lets secondary booter choose the default).
`diag-device`	net	Diagnostic boot source device.
`diag-file`	empty string	File from which to boot in diagnostic mode.
`diag-switch`	false	If true, run in diagnostic mode.
`fcode-debug?`	false	If true, include name fields for plug-in device FCodes.
`hardware-revision`	no default	System version information.
`input-device`	ttya	Power-on device (usually keyboard, ttya or ttyb).

TABLE 4-1 NVRAM Configuration Parameters *(Continued)*

Parameter	Typical Default	Description
keyboard-click?	false	If true, enable keyboard click.
last-hardware-update	no default	System update information.
local-mac-address?	false	If true, network drivers use their own MAC address, not the system's.
mfg-switch?	false	If true, repeat system self-tests until interrupted with a Break command.
nvramrc	empty	Contents of NVRAMRC.
oem-banner	empty string	Custom OEM banner (enabled by oem-banner? true).
oem-banner?	false	If true, use custom OEM banner.
oem-logo	no default	Byte array custom OEM logo (enabled by oem-logo? true).
oem logo?	false	If true, use custom OEM logo (else, use Sun logo).
output device	ttya	Power-on output device (usually screen, ttya, or ttyb).
security-#badlogins	no default	Number of incorrect security password attempts.
security-mode	none	Firmware security level (options: none, command, or full).
security-password	no default	Firmware security password (never displayed). *Do not set this directly.*
selftest-#megs	1	Megabytes of RAM to test. Ignored if diag-switch? is true.
testarea	0	One-byte scratch field, available for read / write test.
ttya-mode	9600,8,n,1,–	TTYA (baud rate, #bits, parity, #stop, handshake).
ttyb-mode	9600,8,n,1,–	TTYB (baud rate, #bits, parity, #stop, handshake).
ttya-ignore-cd	true	If true, operating system ignores carrier-detect on TTYA.
ttyb-ignore-cd	true	If true, operating system ignores carrier-detect on TTYB.

TABLE 4-1 NVRAM Configuration Parameters *(Continued)*

Parameter	Typical Default	Description
`ttya-rts-dtr-off`	false	If true, operating system does not assert TTYB and RTS and DTR.
`ttyb-rts-dtr-off`	false	If true, operating system does not assert TTYB and RTS and DTR.
`use-nvramrc?`	false	If true, execute commands in NVRAMRC during system start-up.

Emergency Procedures

As the system does not support a keyboard, the emergency keyboard command Stop-A is not implemented on a Netra t 1120/1125 system. Use the Break command from a terminal to perform the abort function.

Running Diagnostics

The following information, as documented in the *Solaris OpenBoot Command Reference* manual, is Sun machine-specific and therefore applies to a Netra t 1120/1125 system.

- Running Diagnostics
- Testing the SCSI Bus
- Testing Memory
- Testing the Ethernet Controller
- Testing the Clock
- Monitoring the Network
- Preserving Data After a System Crash
- SCSI Problems—probe SCSI
- System boots from the wrong device
- System will not boot from Ethernet.

New Devices in the OBP Device Tree

The following device has been added to the OBP device tree:

- `/pci@1f,4000/ebus@1/SUNW,tsalarm@14,200000`
  This is the alarm device. It has an FCode PROM which allows it to identify itself to the system during probing. The FCode PROM defines the name of the device and the `reg` property which describes the address space used.

CHAPTER **5**

System Shut-down

The `shutdown` or `init` command is executed by the super user to change the operational state of the machine. By default, it brings the system to a state where only the console has access to the operating system. The `shutdown` command sends a warning message and a final message before it begins shutdown activities. Refer to `shutdown`(1M) and `init`(1M) for further information.

You can also use the following commands:

TABLE 5-1 Commands for Shutting the System Down

Command	Effect
`shutdown -i0` or **`init 0`**	Halt the operating system and enter the Open Boot PROM (OBP).
`shutdown -i5` or **`init 5`**	Halt the operating system and power the system down.
`shutdown -i6`	Return to interactive mode and immediately reboot from the default kernel/unix (the `reboot` command has the same effect).
`init s` or **`init S`**	Halt the operating system and enter maintenance mode.

After shutting down the system move the ON/STBY switch at the front of the system cabinet to the STBY ⏻ position.

To start the system again and begin the automatic booting sequence, you must momentarily set the ON/STBY switch to the ON | position.

If you have an emergency requiring an immediate system power down, switch off the system with the ON/STBY switch. This does not shut down the operating system cleanly, and is likely to corrupt your file system.

APPENDIX **A**

Manual Pages

This chapter contains the manual pages for:

- `tsalarm`(7D)
- `tsctl`(1M)
- `tsdog`(1M)
- `tsmonitor`(1M)
- `tsstate`(1M)
- `tsunlock`(1M)

NAME

tsalarm – Alarm device driver.

SYNOPSIS

tsalarm@bus_address:port_name

DESCRIPTION

The **tsalarm** driver is a Multi-threaded, non-STREAMS driver which is responsible for managing the Netra t 1120/1125 alarm card. It provides the interface through which the main system fans, watchdog heartbeat and power inputs can be monitored and by which the alarm relays and watchdog/reset functions can be controlled.

HARDWARE INTERFACE

The alarm module is a little-endian Ebus slave device that provides two hardware ports to facilitate monitoring and control activities:

1. monitor/control port.

2. watchdog timeout port.

The monitor port reads as a byte whose bits represent the fans, alarm3 output, power and the watchdog heartbeat. The fans provide a square wave oscillating at a frequency proportional to the fan speed. This signal is seen directly in the fan monitor bits.

The monitor port bits are defined as follows:

Bit 7
Heartbeat SupplyA SupplyB Alarm3_State Fan4 Fan3 Fan2 Fan1

Supply A and **Supply B** are only valid for DC products. The control port is a byte whose bits represent alarms 1, 2 and 3, the watchdog function and the reset function. The alarms are user-definable. Alarm3 can also be driven by the watchdog.

The watchdog timeout port is a byte the value of which indicates the current timeout period (in seconds) for the watchdog. The most significant byte (MSB) is always zero, hence the maximum timeout value for the watchdog is 127.

CONFIGURATION

The **tsalarm** driver schedules a monitor callback routine that samples the monitor port at a frequency fast enough to see all the fan oscillations. This callback routine is responsible for notifying applications of any state changes among the monitored components. The sampling information passed to the monitor callback routine is initialized in accordance with the information provided in the configuration file **tsalarm.conf**. The items in this file are user configurable and are presented below:

monvalidmask – This item is a byte whose bits represent the components which are to be monitored by the monitor callback routine. It is intended to configure the software for different hardware implementations.

fanminperiod – This is the time (in mS) that represents the fastest fan rotation speed (i.e. it is the inverse of the maximum fan speed).

fan [*x*] **maxperiod (where** *x* is in the range 1 to 4) – this is the *per fan* time (in ms) that represents the slowest acceptable fan rotation speed.

IOCTLS

The interface provided by the **tsalarm** driver comprises ioctls that enable applications to manipulate the alarm module and monitor the system components. Read, write and mmap system calls are not supported by the driver. The alarm module is accessed via two device nodes: i) **/dev/tsalarm:mon** for monitoring activity and ii) **/dev/tsalarm:ctl** for controlling the alarm relays and watchdog / reset functions.

The following ioctls are supported by the **/dev/tsalarm:mon** device:

TSIOCNBMON

To obtain a copy of the hardware monitor port component status bits without blocking.

For the power, fans, and heartbeat, a **1** bit indicates the corresponding component is **OK**. For the alarm3 output, however, a **1** bit indicates that alarm3 is on.

The argument is a pointer to an integer which will hold the copy of the status of the monitored components.

TSIOCWTMON

To block until a change occurs to the hardware monitor port status bits.

The argument is a pointer to an integer which holds the last remembered status of the monitored components. This integer is compared with the current status of the components and, if different, is updated with the new value. If there is no difference (i.e. no status change has occurred), the calling process is blocked until it is i) subsequently notified of a component state change, ii) interrupted by a signal (in which case an error (**EINTR**) is returned) or iii) awoken by the monitor callback routine because it cannot reschedule itself. In this case an error **ENXIO** is returned.

TSIOCGETMASK

To retrieve the bit mask representing those components which will be monitored by the monitor callback routine.

The argument is a pointer to an integer which will be set to the value of the bitmask. This integer may then be used to tell an application which components are being monitored.

It is assumed that applications will not check the component status by polling the **/dev/tsalarm:mon** device. Instead, an application could use the **TSIOCGETMASK** ioctl to tell it which components are being monitored, then use the **TSIOCNBMON** ioctl to retrieve an initial copy of the status of the components, and then repeatedly block awaiting any changes to any component(s) that it is interested in by using the **TSIOCWTMON** ioctl.

The following ioctls are supported by the **/dev/tsalarm:ctl** device:

TSIOCALCTL

To turn an alarm **on** or **off**. Alarm3 is special since it can be set or reset via the alarm watchdog as well as by applications.

The argument is a pointer to the **ts_aldata_t** structure. This structure is described below. The **alarm_no** member of the structure is an integer which indicates the alarm to apply the command to. This **alarm_state** member of the structure indicates the state to which the alarm is set (where 0 == off). An error (**EINVAL**) is returned if either an invalid **alarm_no** or invalid **alarm_state** is provided.

TSIOCALSTATE

To get the state of the alarms.

The argument is a pointer to the **ts_aldata_t** structure. This structure is described below. The **alarm_no** member is an integer which indicates the alarm to apply the command to. The **alarm_state** member is used to hold the current state of the alarm. It is filled in by the driver. A zero indicates that the alarm is off. An error (**EINVAL**) is returned if an invalid **alarm_no** is provided.

TSIOCDOGSTATE

To get the state of the watchdog and reset functions and retrieve the current timeout period for the watchdog.

The argument is a pointer to the **ts_dogstate_t** structure. This structure is described below. The structure members are used to hold the current states of the reset circuitry, watchdog circuitry and the current watchdog timeout period in seconds. Note that this is not the time remaining before the watchdog is triggered.

TSIOCDOGCTL

To enable or disable the alarm module watchdog and reset functions.

The argument is a pointer to the **ts_dogctl_t** structure. This structure is described below. The **reset_enable** member is used to enable or disable the reset function of the alarm. The **dog_enable** member is used to enable or disable the alarm watchdog function of the alarm module. An error **(**EINVAL**)** is returned if the watchdog is disabled but the reset is enabled.

TSIOCDOGTIME

To set the timeout period for the alarm module watchdog.

The argument is a pointer to an unsigned integer. This integer holds the new timeout period for the watchdog in seconds. If the watchdog function is enabled, it is immediately reset so that the new timeout can take effect. An error (**EINVAL**) is returned if the timeout period is zero or is longer than 127 seconds.

NOTE - This ioctl is not intended for general purpose use. Setting the watchdog timeout to too low a value may cause the system to receive a hardware reset if the watchdog and reset functions are enabled. The default timeout has been determined to give a good compromise between early fault detection and avoidance of false positives.

TSIOCDOGPAT

To reset the alarm module watchdog.

This ioctl requires no arguments. If the watchdog is enabled, it must be used at regular intervals that are less than the watchdog timeout.

NOTE – This ioctl also has the side effect of clearing alarm3 if it was previously set as a result of the watchdog triggering.

TSIOCUNLOCK

This ioctl requires no arguments. It is used to enable the **tsalarm** driver to be unloaded. It is assumed that this ioctl will only be used to enable configuration changes to take effect (e.g changing the *fanminperiod* property to allow for the different types of fan fitted in the system).

The structures and definitions for the values are defined below:

Values for the alarms

These values are defined in <tsalarm_io.h>

```
#define ALARM_NUM_1     1/* number of first alarm */
#define ALARM_NUM_2     2/* number of second alarm */
#define ALARM_NUM_3     3/* number of third alarm */
```

Alarm Data Structure

This structure is defined in <tsalarm_io.h>

```
typedef struct {
        int alarm_no;           /* alarm to apply command to */
        int alarm_state;        /* state of alarm (0 == off) */
} ts_aldata_t;
```

Watchdog/Reset State Data Structure

This structure is defined in <tsalarm_io.h>

```
typedef struct {
        int reset_enable;       /* reset enabled iff non-zero */
        int dog_enable;         /* watchdog enabled iff non-zero */
        uint_t dog_timeout;     /* current watchdog timeout */
} ts_dogstate_t;
```

Watchdog/Reset Control Data Structure

This structure is defined in <tsalarm_io.h>

```
typedef struct {
        int reset_enable;       /* enable reset if non-zero */
        int dog_enable;         /* enable watchdog if non-zero */
} ts_dogctl_t;
```

ERRORS

An open() will fail if:

ENXIO The driver is not installed in the system.

EINVAL
The device minor being opened is the wrong type.

An ioctl() will fail if:

EFAULT
A bad user-space address was specified.

EINVAL
A non-existent control command was requested or invalid parameters were supplied.

EINTR A thread awaiting a component state change was interrupted.

ENXIO The driver is not installed in the system or the monitor callback routine could not be scheduled.

FILES

/usr/kernel/drv/tsalarm # device driver module.
/usr/kernel/drv/tsalarm.conf # configuration file.
/etc/devlinks.tab # devlinks table.
/dev/tsalarm:mon # alarm monitor device.
/dev/tsalarm:ctl # alarm control device.

SEE ALSO

tsdog(1M), **tsctl**(1M), **tsstate**(1M).

NAME

tsctl – control alarms and watchdog

SYNOPSIS

tsctl status

tsctl *object*=**on | off...**

DESCRIPTION

The Netra t 1120/1125 alarm card supports two user alarms and a watchdog alarm which can also function as a third user alarm. **tsctl**(1M) can be used to control these alarms.

The watchdog is a hardware timer which (if enabled) will set alarm3 unless it is regularly reset, or *patted*, by the software. The watchdog also has the ability to invoke a hardware reset, thus rebooting the system. The hardware reset can only be enabled if the watchdog is also enabled. **tsctl** does not pat the watchdog itself, therefore it is primarily used to disable rather than enable the watchdog and the hardware reset.

Although **tsctl** can be used directly from the command line, it is intended for use in shell scripts. The format of its arguments and output (i.e. *object=value* pairs) is intended to facilitate this. *object* can be **alarm1**, **alarm2**, **alarm3**, **watchdog**, or **hwreset**. *value* can be either **on** or **off**.

The command **tsctl status** outputs the status of all alarms on **stdout**.

The alarms and the watchdog can be turned **on** or **off**:

tsctl alarm1=on

tsctl watchdog=off

As the watchdog shares its output with alarm3, setting alarm3 to **off** with **tsctl** will not be sufficient to turn off the alarm3 output, as the watchdog can also have turned it on. The alarm3 output can be examined with the **tsstate**(1M) utility.

The watchdog hardware reset can also be enabled:

tsctl watchdog=on hwreset=on

You cannot enable the hardware reset if the watchdog is disabled; **tsctl** will produce an error if this is attempted.

Warning: Enabling the watchdog and the hardware reset function will cause a hardware reset as soon as the watchdog timeout expires, unless it is regularly *patted*. A hardware reset will cause loss of all user data which has not been flushed to disk, and can cause corruption of mounted file systems.

EXAMPLE

Display the status of all alarms:

```
# tsctl status
alarm1=off
alarm2=off
alarm3=off
watchdog=off
hwreset=off
```

Assign the status of each alarm to a (Bourne or Korn) shell variable:

```
# eval tsctl status
# echo $alarm1
off
```

Disable the hardware reset, leaving the watchdog as before:

```
# tsctl hwreset=off
```

RETURN VALUE

On error, a message is output on **stderr**, and **tsctl**(1M) exits with an exit code as follows:

Message	Exit code	Meaning
Unrecognized argument	1	An argument could not be parsed.
Inappropriate argument	2	Arguments were syntactically correct but were invalid.
Can't open device	3	Failed to open alarm device. (Preceded by name of device and error.)
Inappropriate command	4	ioctl failed with EINVAL.
Command interrupted	5	ioctl failed with EINTR.
Command failed	6	ioctl failed for other reason.

SEE ALSO

tsalarm(7D) **tsdog**(1M) **tsstate**(1M), **tsmonitor**(1M) **sh**(1),

NAME

tsdog – enable the hardware watchdog and pat it forever

SYNOPSIS

tsdog [**dogtimeout=***mS*] [**patinterval=***mS*] [**reset=enabled**]

DESCRIPTION

tsdog(1M) is a simple watchdog process that can be used to activate alarm3, and optionally issue a hardware reset, if the system stops running. It must be run with root user id.

Warning: A hardware reset can cause loss of user data and file system corruption; it is equivalent to pressing the front-panel power switch.

To ensure that, under heavy load, **tsdog** is given an opportunity to run frequently, it should be started as an RT process, using **priocntl**(1).

To avoid **tsdog** being accidentally killed with the result that the watchdog expires, possibly causing a reset, **tsdog** ignores most signals. However, it traps **SIGHUP** and exits gracefully, disabling the watchdog first. This is the recommended way of switching off **tsdog** and the watchdog.

The timeout (in milliseconds) for the watchdog can be set by the **dogtimeout** argument in the range 1 to 127 seconds (i.e. 1000 to 127000 milliseconds). Note that the resolution of the hardware timer is 1 second, so the timeout specified will be rounded up to the next whole second (e.g. 1001 will become 2000). If **dogtimeout** is not specified, the current watchdog timeout will be used.

The **patinterval** argument specifies how long to wait, in milliseconds, between patting the watchdog. If it is not given, it defaults to 1/4 of the watchdog timeout. Attempts to set it to the same or longer than the watchdog timeout will result in an error message, and the watchdog will not be enabled. Unlike **dogtimeout**, the resolution is in milliseconds.

The hardware reset function can be enabled by specifying the **reset=enabled** argument. This causes a hardware reset of the system if the watchdog timer expires before it is patted. This will result in loss of any data not already flushed to disk, and corruption of any mounted file system is likely. Care should be taken to ensure the watchdog will not trigger under heavy, but acceptable, load. It is strongly recommended that **tsdog** is started as a real-time process using **priocntl**(1) when the hardware reset function is enabled, to ensure it is scheduled when necessary.

To ensure that only one instance of **tsdog** is running at any one time, it creates a lock file in the **/tmp** directory and writes its PID into it.

EXAMPLES

Pat the watchdog with a time-sharing process at intervals of 15 seconds, with a watchdog timeout of 1 minute:

tsdog patinterval=15000 dogtimeout=60000

Pat the watchdog with a real-time process at intervals of 10 seconds, with a watchdog timeout of 30 seconds:

priocntl –e –c RT tsdog patinterval=10000 dogtimeout=30000

Pat the watchdog with a real-time process with a watchdog timeout of 40 seconds, and a (defaulted) pat interval of 10 seconds, with the hardware reset function enabled:

priocntl –e –c RT tsdog dogtimeout=40000 reset=enabled

FILES

/tmp/.tsdog.lock Lock file containing PID of **tsdog** process.
/dev/tsalarm:ctl Watchdog / Alarm device.

RETURN VALUES

On error, a message is output on **stderr**, and **tsdog** exits with an exit code as follows:

Message	Exit code	Meaning
tsdog must be run as root	1	Program was not invoked with UID 0.
Unrecognized argument	1	An argument could not be parsed.
Inappropriate argument	2	Arguments were syntactically correct but were invalid.
Can't open device	3	Failed to open alarm device. (Preceded by name of device and error.)
Inappropriate command	4	ioctl failed with EINVAL.
Command interrupted	5	ioctl failed with EINTR.
A copy of tsdog is already running	6	The lock file exists, suggesting there is already a tsdog running.
Command failed	6	ioctl failed for other reason.

SEE ALSO

tsctl(1M), **tsstate**(1M), **sh**(1), **priocntl**(1), **tsalarm**(7D)

NAME

tsmonitor – monitors fans, power and watchdog, sets alarm1 accordingly

SYNOPSIS

tsmonitor &

DESCRIPTION

tsmonitor(1M) is a simple monitor process that can be used with the Netra t 1120/1125 alarm/monitor card to monitor fans, power and the watchdog clock. **tsmonitor** also sets an alarm when any failure occurs. It can be used as an example for developing new monitor scripts.

tsmonitor should be invoked as a background process with root user id. It uses **tsstate**(1M) to find the current state of the monitored objects, and sets alarm1 if any are reported faulty. It then loops forever, calling **tsstate** wait to block until the state changes and updating alarm1 accordingly. It ignores **SIGHUP**, **SIGINT**, and **SIGQUIT**, but can be killed with **SIGTERM**.

SEE ALSO

tsctl(1M), **tsstate**(1M), **sh**(1), **tsalarm**(7D)

NAME

tsstate – get state of fans, power, watchdog clock, and alarm3 output

SYNOPSIS

tsstate

tsstate wait [**alarm3=on | off**] [*object*=**ok | faulty**] ...

DESCRIPTION

The Netra t 1120/1125 alarm card provides a monitoring function for up to four fans, two power supply inputs, and the watchdog clock circuitry. These, and also the output of alarm3 (which can be driven by the watchdog), can be examined using the **tsstate**(1M) utility.

Although **tsstate** can be used directly from the command line, it is intended for use in shell scripts. The format of its arguments and output (i.e. *object=value* pairs) is intended to facilitate this. *object* can be **fan1**, **fan2**, **fan3**, **fan4**, **supplya**, **supplyb** or **watchdog**. *value* is either **ok** or **faulty**. *object* can also be **alarm3** for which the values are **on** or **off**.

The first form of the command **tsstate** outputs the status of all monitored objects, including alarm3.

It is anticipated that **tsstate** will be used in a monitoring script, such as **tsmonitor**(1M), which sets an alarm whenever a monitored object becomes faulty. To avoid continuous polling, **tsstate** can be instructed to wait until the state of the *object* changes from a specified state. The default state is all **ok**, including **alarm3=off**. The command:

tsstate wait

will return as soon as one or more objects become **faulty** or **alarm3** is turned **on** (either by the user or the watchdog).

If wait is followed by the current state of one or more objects, **tsstate** will return only when the state becomes different from the specified state:

tsstate wait fan1=faulty

will not return until either **fan1** recovers, or another *object* becomes **faulty**. As the default is **ok**, specifying *object*=**ok** is equivalent to omitting the *object* altogether. There is no way to specify *don't-care* conditions.

Only objects which are configured in the **tsalarm**(1M) driver **conf** file are supported by **tsstate**(1M). Unconfigured objects are silently ignored.

EXAMPLE

Display the status of all monitored objects:

```
# tsstate
fan1=ok
fan2=ok
```

```
supplya=ok
supplyb=faulty
alarm3=off
watchdog=ok
```

As there is a fault, this is equivalent to:

```
# tsstate wait
```

Display the status when either a new fault develops, or **supplyb** recovers:

```
# tsstate wait supplyb=faulty
fan1=ok
fan2=faulty
supplya=ok
supplyb=faulty
alarm3=off
watchdog=ok
```

See **/usrs/bin/tsmonitor** for an example of using **tsstate** in a monitor script.

RETURN VALUE

On error, a message is output on **stderr**, and **tsstate** exits with an exit code as follows:

Message	Exit code	Meaning
Unrecognized argument	1	An argument could not be parsed.
Inappropriate argument	2	Arguments were syntactically correct but were invalid.
Can't open device	3	Failed to open alarm device. (Preceded by name of device and error.)
Inappropriate command	4	ioctl failed with EINVAL.
Command interrupted	5	ioctl failed with EINTR.
Command failed	6	ioctl failed for other reason.

SEE ALSO

tsctl(1M), **tsdog**(1M), **tsmonitor**(1M), **sh**(1), **tsalarm**(7D)

NAME

tsunlock – instruct **tsalarm** driver to comply with detach requests

SYNOPSIS

tsunlock

DESCRIPTION

The Netra t 1120/1125 alarm/monitor driver normally rejects requests to detach itself, so that the ability to control the alarms and watchdog remains available. However, in order to upgrade the driver without re-booting, the driver must detach. The **tsunlock** utility instructs the **tsalarm** driver to compky with future detach requests.

There is no equivalent utility to reverse the effect of **tsunlock**, as the default state of the driver when first loaded is always to refuse to comply with detach requests, so reloading the driver will achieve the same effect. It is anticipated that the driver will usually be unlocked shortly before it is unloaded and a new version reloaded.

SEE ALSO

tsalarm(7D), **modload**(1M), **modunload**(1M)

Index

S

T

U