Netra™ 240 サーバー
管理マニュアル

Sun Microsystems, Inc.
www.sun.com

Part No. 817-5014-11
2004 年 7 月, Revision A

コメントの送付: http://www.sun.com/hwdocs/feedback

原典: *Netra 240 Server System Administration Guide*
Part No: 817-2700-12
Revision A

目次

図目次

表目次

はじめに

このマニュアルは、経験豊富なシステム管理者を対象としています。このマニュアルでは、Netra™ 240 サーバーの診断ツールと、さまざまなサーバー管理作業の概要について説明します。

このマニュアルの情報を活用するには、コンピュータネットワークの概念および用語に関する実務的な知識と、Solaris™ オペレーティングシステム (Solaris OS) に関する高度な知識が必要です。

お読みになる前に

このマニュアルには、サーバーの設置およびラックへの搭載に関する手順は記載されていません。これらの手順については、『Netra 240 サーバー設置マニュアル』(Part No. 817-4997) を参照してください。

このマニュアルで説明する手順を実行する前に、『Important Safety Information for Sun Hardware Systems』(Part No. 816-7190) を必ずお読みください。

UNIX コマンド

このマニュアルには、システムの停止、システムの起動、およびデバイスの構成などの基本的な UNIX® コマンドと操作手順に関する説明はありません。これらについては、以下を参照してください。

- ご使用のシステムに付属のソフトウェアマニュアル
- 下記にある Solaris™ オペレーティング環境のマニュアル

  `http://docs.sun.com`

シェルプロンプトについて

シェル	プロンプト
UNIX の C シェル	**マシン名**%
UNIX の Bourne シェルと Korn シェル	$
スーパーユーザー (シェルの種類を問わない)	#

書体と記号について

書体または記号	意味	例
AaBbCc123	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例。	.login ファイルを編集します。 ls -a を実行します。 % You have mail.
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して表します。	% **su** Password:
AaBbCc123 または**ゴシック**	コマンド行の可変部分。実際の名前や値と置き換えてください。	rm *filename* と入力します。 rm **ファイル名** と入力します。
『 』	参照する書名を示します。	『Solaris ユーザーマニュアル』
「 」	参照する章、節、または、強調する語を示します。	第 6 章「データの管理」を参照。 この操作ができるのは「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅をこえる場合に、継続を示します。	% **grep '^#define ** **XV_VERSION_STRING'**

関連マニュアル

用途	タイトル	Part No.
設置の概要	『Netra 240 Server Quick Start Guide』(英語版)	817-3904
製品の最新情報	『Netra 240 Server Release Notes』(英語版)	817-3142
安全のための注意事項	『Important Safety Information for Sun Hardware Systems』(マルチリンガル版)	816-7190
	『Netra 240 Server Safety and Compliance Manual』(マルチリンガル版)	817-5018
マニュアル Web サイトの URL	『Sun Netra 240 Server Product Documentation』(英語版)	817-2697
設置	『Netra 240 サーバー設置マニュアル』	817-4997
Lights-Out Management	『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』	817-5006
保守	『Netra 240 Server Service Manual』(英語版)	817-2699

Sun のオンラインマニュアル

各言語対応版を含むサンの各種マニュアルは、次の URL から表示または印刷、購入できます。

`http://www.sun.com/documentation`

サン以外の Web サイト

このマニュアルで説明するサン以外の Web サイトの利用については、サンは責任を負いません。このようなサイトやリソース上、またはこれらを経由して利用できるコンテンツ、広告、製品、またはその他の資料についても、サンは保証しておらず、法的責任を負いません。また、このようなサイトやリソース上、またはこれらを経由して利用できるコンテンツ、商品、サービスの使用や依存に関連して発生した実際の損害や損失、またはその申し立てについても、サンは一切の責任を負いません。

Sun の技術サポート

このマニュアルに記載されていない技術的な問い合わせについては、次の URL にアクセスしてください。

`http://www.sun.com/service/contacting`

コメントをお寄せください

弊社では、マニュアルの改善に努力しており、お客様からのコメントおよびご忠告をお受けしております。コメントは下記よりお送りください。

`http://www.sun.com/hwdocs/feedback`

コメントには下記のタイトルと Part No. を記載してください。

『Netra 240 サーバー管理マニュアル』, Part No. 817-5014-11

第1章

障害追跡ツール

この章では、Netra 240 サーバーで使用できる診断ツールについて説明します。この章は、次の節で構成されます。

診断ツールの概要

サンは、Netra 240 サーバーで使用できる、さまざまな診断ツールを提供しています。次の表に、診断ツールの概要を示します。

表 1-1　障害追跡ツールの概要

診断ツール	種類	説明	アクセスおよび実行の条件	遠隔機能
ALOM	ハードウェアおよびソフトウェア	環境条件の監視、基本的な障害特定の実行、およびコンソールへの遠隔アクセスを提供	スタンバイ電力で動作可能で、オペレーティングシステムは不要	遠隔アクセス用に設計されている
LED	ハードウェア	システム全体および特定の部品の状態を表示	システムのシャーシから使用できる。電力が供給されていれば使用可能。	ローカル、ALOM による表示も可能
電源投入時自己診断 (POST)	ファームウェア	システムの主要な部品をテスト	起動時に自動的に実行。オペレーティングシステムの非動作時でも使用可能。	ローカル、ALOM による表示も可能
OpenBoot コマンド	ファームウェア	システムのさまざまな情報を表示	オペレーティングシステムの非動作時でも使用可能。	ローカル、ALOM によるアクセスも可能
OpenBoot 診断	ファームウェア	周辺装置および I/O デバイスを中心に、システム部品をテスト	自動または対話式に実行。オペレーティングシステムの非動作時でも使用可能。	ローカル、ALOM による表示も可能
Solaris ソフトウェアコマンド	ソフトウェア	システムのさまざまな情報を表示	オペレーティングシステムが必要。	ローカル、ALOM によるアクセスも可能
SunVTS™ ソフトウェア	ソフトウェア	テストを並行して実行して、システムの動作テストおよび負荷テストを行う	オペレーティングシステムが必要。オプションのパッケージ。	ネットワークを介した表示および制御が可能

システムプロンプト

Netra 240 サーバーのデフォルトのサーバープロンプトを、次に示します。

- ok ― OpenBoot PROM プロンプト
- sc> ― Advanced Lights Out Manager (ALOM) プロンプト
- # ― Solaris ソフトウェアのスーパーユーザー (Bourne および Korn シェル) プロンプト

図 1-1 に、3 つのプロンプトの関連と、各プロンプトへの切り替え方法を示します。

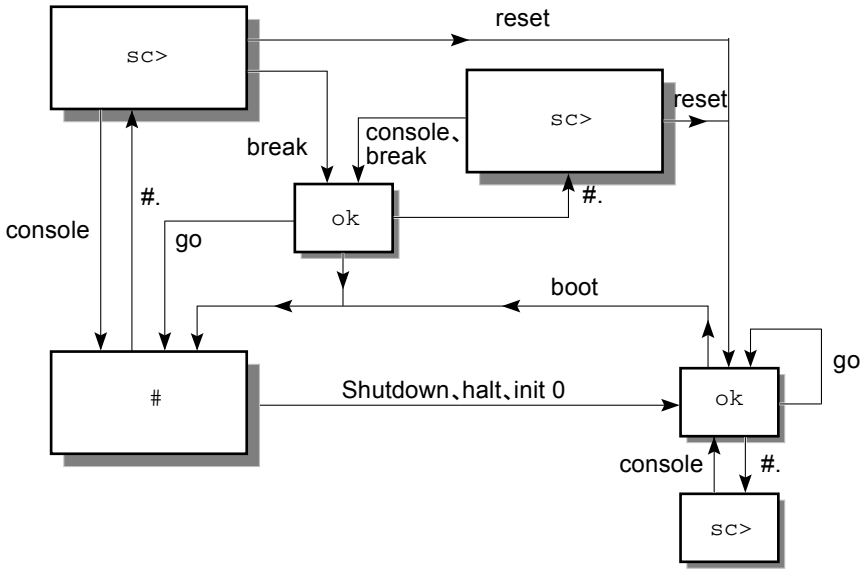

図 1-1　　システムプロンプトの流れ

図 1-1 の流れ図では、次のコマンドが使用されています。

- ALOM コマンド : console、reset、break
- エスケープシーケンス : #.
- Solaris ソフトウェアコマンド : shutdown、halt、init 0
- OpenBoot コマンド : go、boot

Advanced Lights Out Manager

Netra 240 サーバー用の Sun™ Advanced Lights Out Manager (ALOM) は、一連の LED 状態インジケータを備えています。この節では、各インジケータの状態の意味と、点灯および消灯の方法について説明します。ALOM の詳細は、第 3 章を参照してください。

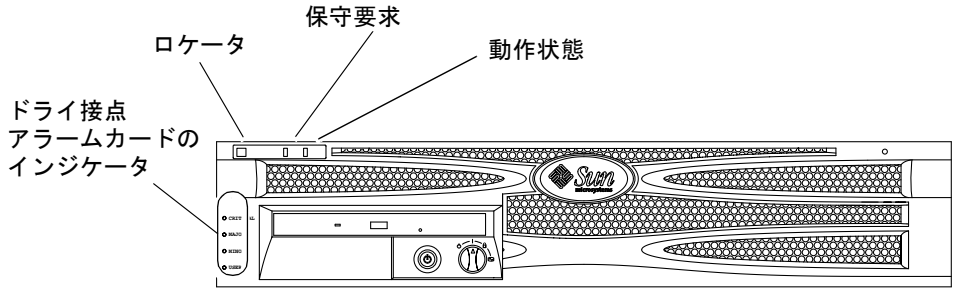

図 1-2　正面パネルのインジケータの位置

サーバー状態インジケータ

Netra 240 サーバーには、3 つの LED 状態インジケータがあります。同じ状態インジケータが、正面ベゼル (図 1-2 を参照) と背面パネルの両方に付いています。表 1-2 に、インジケータの概要を示します

表 1-2　サーバー状態インジケータ (正面および背面)

インジケータ	LED の色	LED の状態	意味
動作状態	緑色	点灯	サーバーに電源が入っていて、Solaris OS が動作しています。
		消灯	電源が入っていないか、Solaris OS が動作していません。
保守要求	黄色	点灯	サーバーの障害が検出されました。保守作業員による調査が必要です。
		消灯	サーバーの障害は検出されていません。
ロケータ	白色	点灯	`setlocator` コマンドを使用すると連続点灯し、ラック内のほかのサーバーと区別できます。

ロケータ LED の点灯と消灯は、システムコンソール、または ALOM のコマンド行インタフェース (CLI) によって制御できます。

▼ ロケータ LED の状態を表示する

- **次のいずれかを実行します。**
  - スーパーユーザーで、次のように入力します。

```
# /usr/sbin/locator
```

  - ALOM のコマンド行インタフェースで、次のように入力します。

```
sc> showlocator
```

▼ ロケータ LED を点灯させる

- **次のいずれかを実行します。**
  - スーパーユーザーで、次のように入力します。

```
# /usr/sbin/locator -n
```

  - ALOM のコマンド行インタフェースで、次のように入力します。

```
sc> setlocator on
```

▼ ロケータ LED を消灯させる

- **次のいずれかを実行します。**
  - スーパーユーザーで、次のように入力します。

```
# /usr/sbin/locator -f
```

  - ALOM のコマンド行インタフェースで、次のように入力します。

```
sc> setlocator off
```

アラーム状態インジケータ

ドライ接点アラームカードには、ALOM によってサポートされる 4 つの LED 状態インジケータがあります。これらのインジケータは、正面ベゼルに縦に並んで付いています (図 1-2 を参照)。表 1-3 に、アラームインジケータおよびドライ接点アラームの状態を示します。アラームインジケータの詳細は、『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) を参照してください。アラームインジケータを制御する API の詳細は、付録 A を参照してください。

表 1-3 アラームインジケータおよびドライ接点アラームの状態

インジケータおよびリレーのラベル	インジケータの色	アプリケーションまたはサーバーの状態	状態または動作	システムインジケータの状態	アラームインジケータの状態	リレーの NC[iv] 状態	リレーの NO[v] 状態	説明
Critical (高) (Alarm0)	赤	サーバーの状態 (電源が入っているかどうか、Solaris OS が動作しているかどうか)	電力の供給なし	消灯	消灯	閉	開	デフォルトの状態
			システム電源はオフ	消灯	消灯[iii]	閉	開	入力電源には接続
			システム電源はオンだが、Solaris OS のロードは未完了	消灯	消灯[iii]	閉	開	一時的な状態
			Solaris OS を正常にロード済み	点灯	消灯	開	閉	通常の動作状態
			ウォッチドッグのタイムアウト	消灯	点灯	閉	開	一時的な状態 ; Solaris OS を再起動
			ユーザーによる Solaris OS の停止[i]	消灯	消灯[iii]	閉	開	一時的な状態
			電力供給の停止	消灯	消灯	閉	開	デフォルトの状態
			ユーザーによるシステム電源の切断	消灯	消灯[iii]	閉	開	一時的な状態
		アプリケーションの状態	ユーザーによる Critical (高) アラームの設定[ii]	—	点灯	閉	開	重要度の高い障害を検出
			ユーザーによる Critical (高) アラームの解除[ii]	—	消灯	開	閉	重要度の高い障害が解決

表 1-3　アラームインジケータおよびドライ接点アラームの状態 (続き)

インジケータおよびリレーのラベル	インジケータの色	アプリケーションまたはサーバーの状態	状態または動作	システムインジケータの状態	アラームインジケータの状態	リレーの NC[iv] 状態	リレーの NO[v] 状態	説明
Major (中) (Alarm1)	赤	アプリケーションの状態	ユーザーによる Major (中) アラームの設定[ii]	—	点灯	開	閉	重要度が中程度の障害を検出
			ユーザーによる Major (中) アラームの解除[ii]	—	消灯	閉	開	重要度が中程度の障害が解決
Minor (低) (Alarm2)	オレンジ色	アプリケーションの状態	ユーザーによる Minor (低) アラームの設定[ii]	—	点灯	開	閉	重要度の低い障害を検出
			ユーザーによる Minor (低) アラームの解除[ii]	—	消灯	閉	開	重要度の低い障害が解決
User (ユーザー) (Alarm3)	オレンジ色	アプリケーションの状態	ユーザーによる User (ユーザー) アラームの設定[ii]	—	点灯	開	閉	ユーザーの障害を検出
			ユーザーによる User (ユーザー) アラームの解除[ii]	—	消灯	閉	開	ユーザーの障害が解決

i　ユーザーは、init0、init6 などのコマンドを使用してシステムを停止できます。ただし、システムの電源は停止できません。

ii　ユーザーは、障害状態の判定に基づき、Solaris プラットフォームのアラーム API または ALOM CLI を使用して、アラームを設定できます。アラーム API の詳細は付録 A を、ALOM CLI の詳細は『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) を参照してください。

iii このアラームインジケータ状態の実装は、変更される可能性があります。

iv NC 状態とは、常閉 (Normally Closed) の状態です。これは、リレー接点のデフォルトのモードが常閉状態であることを示します。

v NO 状態とは、常開 (Normally Open) の状態です。これは、リレー接点のデフォルトのモードが常開状態であることを示します。

ユーザーがアラームを設定すると、必ずコンソール上にメッセージが表示されます。たとえば、Critical (高) アラームを設定すると、コンソール上に次のメッセージが表示されます。

```
SC Alert: CRITICAL ALARM is set
```

一部の状況では、Critical (高) アラームを設定しても、関連付けられたアラームインジケータが点灯しないことに注意してください。この実装は、将来のリリースで変更される可能性があります (表 1-3 の脚注 [iii] を参照)。

電源投入時自己診断

電源投入時自己診断 (POST) は、システムの一部に障害が発生しているかどうかを検出するのに役立つファームウェアプログラムです。POST は、CPU モジュール、マザーボード、メモリー、一部のオンボード I/O デバイスなどの、システムの中核になる部品を検証します。また、ハードウェア障害の原因の判定に役立つメッセージを生成します。POST は、システムが起動できない状態でも実行できます。

POST は、マザーボードの OpenBoot PROM に格納されているプログラムで、ほとんどのシステム障害を検出します。OpenBoot ソフトウェアをプログラムして POST を実行することができます。それには `diag-switch?` および `diag-level` フラグという 2 つの環境変数を設定します。この 2 つの変数は、システム構成カードに保存されます。

次の設定をすべて適用すると、システムに電力が供給されたとき、または自動システムリセットのあとに、POST が自動的に実行されます。

- `diag-switch?` を `true` に設定 (デフォルトは `false`)
- `diag-level` を `min`、`max`、または `menus` のいずれかに設定 (デフォルトは `min`)
- `post-trigger` をリセットのクラスと一致させる (デフォルトは `power-on-reset`)

`diag-level` を `min` または `max` に設定すると、POST は、それぞれ簡易テストまたは拡張テストを実行します。

`diag-level` を `menus` に設定すると、電源投入時に実行されるすべてのテストのメニューが表示されます。

POST の診断結果およびエラーメッセージは、コンソール上に表示されます。

POST 診断の制御

POST 診断および起動プロセスのさまざまな動作は、OpenBoot 構成変数の設定によって制御します。OpenBoot 構成変数の変更は、システムの再起動後に有効になります。表 1-4 に、もっとも重要で有用な OpenBoot 構成変数を示します。OpenBoot 構成変数の変更方法については、27 ページの「OpenBoot 構成変数を参照および設定する」を参照してください。

表 1-4 OpenBoot 構成変数

OpenBoot 構成変数	説明およびキーワード
auto-boot	オペレーティングシステムを自動的に起動するかどうかを指定します。デフォルト値は、true です。 • true − ファームウェアテストが終了すると、オペレーティングシステムが自動的に起動します。 • false − boot と入力するまで、システムは ok プロンプトを表示します。
diag-level	実行する診断のレベルまたは種類を指定します。デフォルト値は、min です。 • off − テストを実行しません。 • min − 基本テストだけを実行します。 • max − 装置の種類ごとに、より詳細なテストを実行します。 • menus − POST レベルでメニューからテストを行う場合は、各テストを個別に実行できます。
diag-script	OpenBoot 診断によってテストする装置を指定します。デフォルト値は、none です。 • none − 装置のテストを行いません。 • normal − 自己診断機能がある (センタープレーン上の) オンボード装置のテストを行います。 • all − 自己診断機能があるすべての装置のテストを行います。
diag-switch?	システムを診断モードにするかどうかを切り替えます。デフォルト値は、false です。 • true − 診断モード : POST および OpenBoot 診断テストを実行します。 • false − デフォルトモード : POST または OpenBoot 診断テストを実行しません。

表 1-4　OpenBoot 構成変数 (続き)

OpenBoot 構成変数	説明およびキーワード
post-trigger obdiag-trigger	これらの 2 つの変数を使用して、POST (または OpenBoot 診断テスト) を実行するきっかけとなるリセットイベントのクラスを指定します。これらの変数には、1 つのキーワードを指定するか、スペースで区切られたキーワードを 3 つまで組み合わせて指定できます。詳細は、27 ページの「OpenBoot 構成変数を参照および設定する」を参照してください。 • error-reset − 回復不能なハードウェアのエラー状態によって発生したリセットです。通常、ハードウェアの問題によってシステムの状態データが破壊された場合に、エラーリセットが発生します。エラーリセットには、CPU およびシステムウォッチドッグリセット、重大なエラー、いくつかの CPU リセットイベントなどがあります (デフォルト)。 • power-on-reset − オン/スタンバイボタンを押すことによって発生するリセットです (デフォルト)。 • user-reset − ユーザーまたはオペレーティングシステムによって発生するリセットです。 • all-resets − すべての種類のシステムリセットです。 • none − POST (または OpenBoot 診断テスト) は実行されません。
input-device	入力するコンソールを選択します。デフォルトは ttya です。 • ttya − 組み込まれた SERIAL MGT ポートから入力します。 • ttyb − 組み込まれた汎用シリアルポート (10101) から入力します。 • keyboard − グラフィックス端末の一部として接続されたキーボードから入力します。
output-device	診断およびその他のコンソール出力の表示先を選択します。デフォルトは ttya です。 • ttya − 組み込まれた SERIAL MGT ポートに出力します。 • ttyb − 組み込まれた汎用シリアルポート (10101) に出力します。 • screen − グラフィックス端末の一部として接続された画面に出力します。[i]

i POST メッセージは、グラフィックス端末には表示できません。output-device が screen に設定されている場合でも、ttya に出力されます。

注 – この表の変数は、POST 診断だけでなく、OpenBoot 診断テストにも影響します。

POST 診断が終了すると、POST によって実行された各テストの状態が OpenBoot ファームウェアに報告されます。そのあと、制御は OpenBoot ファームウェアのコードに戻ります。

POST 診断では障害が検出されないのにサーバーを起動できない場合は、OpenBoot 診断テストを実行します。

▼ POST 診断を開始する

1. ok プロンプトを表示します。
2. 次のように入力します。

```
ok setenv diag-switch? true
```

3. 次のように入力します。

```
ok setenv diag-level value
```

value には、必要な診断情報の量に応じて min、max、または menus を指定します。

4. 次のように入力します。

```
ok reset-all
```

post-trigger に user-reset を設定していると、システムによって POST 診断が実行されます。状態およびエラーメッセージがコンソールウィンドウに表示されます。POST がエラーを検出した場合には、障害の詳細を説明するエラーメッセージが表示されます。

5. POST の実行が終了したら、次のように入力して、diag-switch? の値を false に戻します。

```
ok setenv diag-switch? false
```

diag-switch? を false に戻すことで、起動時間を最小限に抑えます。

OpenBoot コマンド

OpenBoot コマンドは、ok プロンプトから実行します。診断に役立つ情報を提供する OpenBoot コマンドは、次のとおりです。

- probe-scsi および probe-scsi-all
- probe-ide
- show-devs

probe-scsi および probe-scsi-all コマンド

probe-scsi および probe-scsi-all コマンドは、SCSI 装置の問題を診断します。

注意 – halt コマンドまたは Stop-A キーシーケンスを使用して ok プロンプトを表示した場合に、probe-scsi または probe-scsi-all コマンドを使用すると、システムがハングアップすることがあります。

probe-scsi コマンドは、オンボードの SCSI コントローラに接続されたすべての SCSI 装置との通信を行います。probe-scsi-all コマンドは、PCI スロットに取り付けられているすべてのホストアダプタに接続している装置にもアクセスします。

probe-scsi および probe-scsi-all コマンドは、接続されて動作している SCSI 装置の、ループ ID、ホストアダプタ、論理ユニット番号、一意の WWN (World Wide Name)、および装置の説明 (タイプ、メーカー名など) を表示します。

次に、probe-scsi コマンドの出力例を示します。

コード例 1-1 probe-scsi コマンドの出力

```
{1} ok probe-scsi
Target 0
  Unit 0   Disk     SEAGATE ST373307LSUN72G 0207
Target 1
  Unit 0   Disk     SEAGATE ST336607LSUN36G 0207
{1} ok
```

次に、probe-scsi-all コマンドの出力例を示します。

コード例 1-2 probe-scsi-all コマンドの出力

```
{1} ok probe-scsi-all
/pci@1c,600000/scsi@2,1

/pci@1c,600000/scsi@2
Target 0
  Unit 0   Disk     SEAGATE ST373307LSUN72G 0207
Target 1
  Unit 0   Disk     SEAGATE ST336607LSUN36G 0207

{1} ok
```

probe-ide コマンド

probe-ide コマンドは、IDE (Integrated Drive Electronics) バスに接続されているすべての IDE 装置との通信を行います。IDE バスは、DVD ドライブなどの媒体装置に使用する内部システムバスです。

注意 – halt コマンドまたは Stop-A キーシーケンスを使用して ok プロンプトを表示した場合に、probe-ide コマンドを使用すると、システムがハングアップすることがあります。

次に、probe-ide コマンドの出力例を示します。

コード例 1-3　　probe-ide コマンドの出力

```
{1} ok probe-ide
Device 0  ( Primary Master )
          Not Present

  Device 1  ( Primary Slave )
          Not Present

  Device 2  ( Secondary Master )
          Not Present

  Device 3  ( Secondary Slave )
          Not Present

{1} ok
```

show-devs コマンド

show-devs コマンドは、ファームウェアデバイスツリー内の各装置のハードウェアデバイスパスを一覧で表示します。コード例 1-4 に、出力例の一部を示します。

コード例 1-4　　show-devs コマンドの出力

```
/pci@1d,700000
/pci@1c,600000
/pci@1e,600000
/pci@1f,700000
/memory-controller@1,0
/SUNW,UltraSPARC-IIIi@1,0
/memory-controller@0,0
/SUNW,UltraSPARC-IIIi@0,0
/virtual-memory
/memory@m0,0
/aliases
/options
/openprom
/chosen
/packages
/pci@1d,700000/network@2,1
/pci@1d,700000/network@2
/pci@1c,600000/scsi@2,1
/pci@1c,600000/scsi@2
/pci@1c,600000/scsi@2,1/tape
/pci@1c,600000/scsi@2,1/disk
/pci@1c,600000/scsi@2/tape
/pci@1c,600000/scsi@2/disk
/pci@1e,600000/ide@d
/pci@1e,600000/usb@a
/pci@1e,600000/pmu@6
/pci@1e,600000/isa@7
/pci@1e,600000/ide@d/cdrom
/pci@1e,600000/ide@d/disk.........
```

▼ OpenBoot コマンドを実行する

1. **システムを停止して、ok プロンプトを表示します。**
   システムを停止する前にユーザーに通知します。
2. **コンソールのプロンプトで、適切なコマンドを入力します。**

OpenBoot 診断

POST 診断と同様に、OpenBoot 診断のコードはファームウェアベースで、Boot PROM に格納されています。

▼ OpenBoot 診断を開始する

1. 次のように入力します。

```
ok setenv diag-switch? true
ok setenv auto-boot? false
ok reset-all
```

2. 次のように入力します。

```
ok obdiag
```

このコマンドを実行すると、OpenBoot 診断のメニューが表示されます。

```
ok obdiag
_____________________________________________________________________________
|                                o b d i a g                                  |
|_____________________________ _____________________________ _____________________________|
|                             |                             |                             |
|  1 i2c@0,320                |  2 ide@d                    |  3 network@2                |
|  4 network@2,1              |  5 rtc@0,70                 |  6 scsi@2                   |
|  7 scsi@2,1                 |  8 serial@0,2e8             |  9 serial@0,3f8             |
| 10 usb@a                    | 11 usb@b                    | 12 flashprom@2,0            |
|_____________________________|_____________________________|_____________________________|
|   Commands: test test-all except help what printenvs setenv versions exit   |
|_____________________________________________________________________________|
```

注 – サーバーに PCI カードを取り付けていると、obdiag メニューに追加のテストが表示されます。

3. 次のように入力します。

```
obdiag> test n
```

n には、実行するテストに対応する番号を指定します。

テストの概要を確認することもできます。obdiag> プロンプトで、次のように入力します。

```
obdiag> help
```

OpenBoot 診断テストの制御

POST の制御に使用する OpenBoot 構成変数 (表 1-4 を参照) は、そのほとんどが OpenBoot 診断テストにも影響します。

- OpenBoot 診断テストのレベルは、diag-level 変数によって制御します。
- テストの実行方法は、test-args 変数によってカスタマイズします。

デフォルトでは、test-args には空の文字列が設定されています。test-args を変更するには、表 1-5 に示すキーワードを 1 つ以上指定します。

表 1-5　OpenBoot 構成変数 test-args のキーワード

キーワード	説明
bist	外部装置および周辺装置で組み込み型自己診断 (BIST) を起動
debug	すべてのデバッグメッセージを表示
iopath	バスおよびインターコネクトの完全性を検証
loopback	外部装置へのループバックパスをテスト
media	外部装置および周辺装置のメディアへのアクセスを検証
restore	前のテストが失敗した場合、装置の元の状態への復元を試行
silent	各テストの状態は表示せず、エラーだけを表示
subtests	メインテストと、そこから呼び出された各サブテストを表示

表 1-5　OpenBoot 構成変数 test-args のキーワード (続き)

キーワード	説明
verbose	すべてのテストの詳細な状態メッセージを表示
callers=*n*	エラー発生時に、*N* 個の呼び出し元のバックトレースを表示 callers=0 ― エラー発生前の、すべての呼び出し元のバックトレースを表示
errors=*n*	エラーが *N* 回発生するまでテストの実行を継続 errors=0 ― テストを終了せずに、すべてのエラーレポートを表示

OpenBoot 診断テストをカスタマイズする場合は、次の例のように、キーワードをコンマで区切って test-args に設定します。

```
ok setenv test-args debug,loopback,media
```

test および test-all コマンド

ok プロンプトから直接 OpenBoot 診断テストを実行することもできます。これを行うには、test コマンドに続けて、テストする装置 (または装置一式) のハードウェアパスをフルパス名で入力します。次に、例を示します。

```
ok test /pci@x,y/SUNW,qlc@2
```

test-args を次のように指定すると、個々のテストをカスタマイズできます。

```
ok test /usb@1,3:test-args={verbose,debug}
```

この指定は現在のテストにだけ影響し、OpenBoot 構成変数 test-args の値は変更されません。

test-all コマンドを使用すると、デバイスツリー内のすべての装置をテストできます。

```
ok test-all
```

test-all の引数にパスを指定すると、指定した装置とそこに接続された装置だけがテストされます。次に、USB バスと USB バスに接続された自己診断機能があるすべての装置をテストする場合の入力例を示します。

```
ok test-all /pci@9,700000/usb@1,3
```

OpenBoot 診断のエラーメッセージ

OpenBoot 診断のエラー結果は、表形式で報告されます。この表には、問題の概略および問題によって影響を受けるハードウェア装置、不合格になったサブテスト名、その他の診断情報が含まれます。コード例 1-5 に、OpenBoot 診断のエラーメッセージの例を示します。

コード例 1-5　　OpenBoot 診断のエラーメッセージ

```
Testing /pci@1e,600000/isa@7/flashprom@2,0

   ERROR   : FLASHPROM CRC-32 is incorrect
   SUMMARY : Obs=0x729f6392 Exp=0x3d6cdf53 XOR=0x4ff3bcc1 Addr=0xfeebbffc
   DEVICE  : /pci@1e,600000/isa@7/flashprom@2,0
   SUBTEST : selftest:crc-subtest
   MACHINE : Netra 240
   SERIAL# : 52965531
   DATE    : 03/05/2003 01:33:59  GMT
   CONTR0LS: diag-level=max test-args=

Error: /pci@1e,600000/isa@7/flashprom@2,0 selftest failed, return code = 1
Selftest at /pci@1e,600000/isa@7/flashprom@2,0 (errors=1) .............
failed
Pass:1 (of 1) Errors:1 (of 1) Tests Failed:1 Elapsed Time: 0:0:0:27
```

オペレーティングシステムの診断ツール

OpenBoot 診断テストに合格すると、システムは Solaris OS を起動します。サーバーがマルチユーザーモードで起動すると、ソフトウェアベースの診断ツールおよび SunVTS ソフトウェアを使用できるようになります。これらのツールによって、サーバーの監視、動作テスト、および障害の特定を行うことができます。

注 – OpenBoot 構成変数 `auto-boot?` を `false` に設定した場合は、ファームウェアベースのテストのあとに、オペレーティングシステムは起動されません。

前述のツールのほか、エラーメッセージとシステムメッセージのログファイルおよび Solaris のソフトウェア情報コマンドを参照することもできます。

エラーメッセージおよびシステムメッセージのログファイル

エラーメッセージおよびその他のシステムメッセージは、`/var/adm/messages` ファイルに記録されます。オペレーティングシステム、環境制御サブシステム、さまざまなソフトウェアアプリケーションなどが発信元となって、このファイルにメッセージを記録します。

Solaris ソフトウェアのシステム情報コマンド

次の Solaris ソフトウェアのシステム情報コマンドは、Netra 240 サーバーの状態を評価するために使用できるデータを表示します。

- `prtconf`
- `prtdiag`
- `prtfru`
- `psrinfo`
- `showrev`

次に、これらのコマンドを実行することによって表示される情報について説明します。コマンドの使用方法の詳細は、適切なマニュアルページを参照してください。

`prtconf` コマンド

`prtconf` コマンドは、Solaris ソフトウェアのデバイスツリーを表示します。このデバイスツリーには、OpenBoot ファームウェアによってプローブされたすべての装置に加えて、オペレーティングシステムのソフトウェアだけが認識している個々のディスクなどの追加装置も含まれます。`prtconf` の出力には、システムメモリーの合計も表示されます。コード例 1-6 に、`prtconf` 出力の一部を示します。

コード例 1-6　　prtconf コマンドの出力

```
# prtconf

System Configuration:  Sun Microsystems  sun4u
Memory size: 5120 Megabytes
System Peripherals (Software Nodes):

SUNW,Netra-240
    packages (driver not attached)
        SUNW,builtin-drivers (driver not attached)
        deblocker (driver not attached)
        disk-label (driver not attached)
        terminal-emulator (driver not attached)
        dropins (driver not attached)
        kbd-translator (driver not attached)
        obp-tftp (driver not attached)
        SUNW,i2c-ram-device (driver not attached)
        SUNW,fru-device (driver not attached)
        ufs-file-system (driver not attached)
    chosen (driver not attached)
    openprom (driver not attached)
        client-services (driver not attached)
    options, instance #0
    aliases (driver not attached)
    memory (driver not attached)
    virtual-memory (driver not attached)
    SUNW,UltraSPARC-IIIi (driver not attached)
    memory-controller, instance #0
    SUNW,UltraSPARC-IIIi (driver not attached)
    memory-controller, instance #1
    pci, instance #0........
```

prtconf コマンドに -p オプションを付けて実行すると、OpenBoot の show-devs コマンドと同様な出力が生成されます。この出力には、システムのファームウェアによって編集された装置だけの一覧が表示されます。

prtdiag コマンド

prtdiag コマンドは、システム部品の状態を要約した診断情報の表を表示します。prtdiag コマンドの表示形式は、システムで動作している Solaris OS のバージョンによって異なります。次のコード例は、Solaris ソフトウェアが動作している正常な Netra 240 サーバーで prtdiag を実行した場合の出力の一部を示しています。

コード例 1-7　　prtdiag コマンドの出力

```
# prtdiag
System Configuration: Sun Microsystems  sun4u Netra 240
System clock frequency: 160 MHZ
Memory size: 2GB
==================================== CPUs ====================================
                     E$          CPU     CPU       Temperature          Fan
       CPU  Freq     Size        Impl.   Mask     Die    Ambient   Speed   Unit
       ---  --------  ----------  ------  ----  --------  --------  -----   ----
     MB/P0  1280 MHz  1MB         US-IIIi   2.3     -     -
     MB/P1  1280 MHz  1MB         US-IIIi   2.3     -     -
================================ IO Devices ================================
     Bus   Freq
Brd  Type  MHz   Slot        Name                          Model
---  ----  ----  ----------  ----------------------------  --------------------
 0   pci    66            2  network-pci14e4,1648.108e.16+
 0   pci    66            2  network-pci14e4,1648.108e.16+
 0   pci    66            2  scsi-pci1000,21.1000.1000.1 +
 0   pci    66            2  scsi-pci1000,21.1000.1000.1 +
 0   pci    66            2  network-pci14e4,1648.108e.16+
 0   pci    66            2  network-pci14e4,1648.108e.16+
 0   pci    33            7  isa/serial-su16550 (serial)
 0   pci    33            7  isa/serial-su16550 (serial)
 0   pci    33            7  isa/rmc-comm-rmc_comm (seria+
 0   pci    33           13  ide-pci10b9,5229.c4 (ide)
============================ Memory Configuration ============================
Segment Table:
------------------------------------------------------------------------
Base Address       Size       Interleave Factor  Contains
------------------------------------------------------------------------
0x0                1GB                1          GroupID 0
0x1000000000       1GB                1          GroupID 0

Memory Module Groups:
------------------------------------------------
ControllerID   GroupID  Labels
------------------------------------------------
0              0        MB/P0/B0/D0,MB/P0/B0/D1
Memory Module Groups:
------------------------------------------------
ControllerID   GroupID  Labels
------------------------------------------------
1              0        MB/P1/B0/D0,MB/P1/B0/D1
```

prtdiag に冗長オプション (-v) を指定すると、コード例 1-7 に示す情報のほかに、正面パネルの状態、ディスクの状態、ファンの状態、電源装置、ハードウェアのバージョン、およびシステムの温度も報告されます (コード例 1-8 を参照)。

コード例 1-8　　prtdiag コマンドの冗長出力

```
--------------------------------------------------------------
Location    Sensor      Temperature  Lo LoWarn HiWarn  Hi Status
--------------------------------------------------------------
MB          T_ENC           22C     -7C   -5C   55C   58C   okay
MB/P0       T_CORE          57C     -       -    110C  115C okay
MB/P1       T_CORE          54C     -       -    110C  115C okay
PS0         FF_OT           -       -       -    -       -  okay
PS1         FF_OT           -       -       -    -       -  okay
```

適正温度を超えた状態が発生すると、prtdiag は、「Status」列にエラーを表示します (コード例 1-9)。

コード例 1-9　　高温状態を示す prtdiag コマンドの出力

```
--------------------------------------------------------------
Location    Sensor      Temperature  Lo LoWarn HiWarn  Hi Status
--------------------------------------------------------------
MB          T_ENC           22C     -7C    -5C    55C   58C okay
MB/P0       T_CORE         118C      -       -    110C  115C failed
MB/P1       T_CORE         112C      -       -    110C  115C warning
PS0         FF_OT           -       -       -    -       -  okay
PS1         FF_OT           -       -       -    -       -  okay
```

同様に、特定の部品に障害がある場合、prtdiag は、該当する「Status」列に障害を表示します (コード例 1-10)。

コード例 1-10　　障害を示す prtdiag コマンドの出力

```
Fan Speeds:
----------------------------------------
Location        Sensor      Status   Speed
----------------------------------------
MB/P0/F0        RS          failed   0 rpm
MB/P0/F1        RS          okay     3994 rpm
F2              RS          okay     2896 rpm
PS0             FF_FAN      okay
F3              RS          okay     2576 rpm
PS1             FF_FAN      okay
--------------------------------
```

prtfru コマンド

Netra 240 サーバーは、システムのすべての現場交換可能ユニット (FRU) の階層構造のリストと、各種 FRU の固有の情報を保持しています。

prtfru コマンドは、この階層リストと、多くの FRU 上の SEEPROM (Serial Electrically-Erasable Programmable Read-Only Memory) に記録されているデータを表示します。コード例 1-11 に、prtfru コマンドに -l オプションを指定した場合に生成される FRU の階層リストの一部を示します。

コード例 1-11 prtfru -l コマンドの出力

```
# prtfru -l
/frutree
/frutree/chassis (fru)
/frutree/chassis/MB?Label=MB
/frutree/chassis/MB?Label=MB/system-board (container)
/frutree/chassis/MB?Label=MB/system-board/SC?Label=SC
/frutree/chassis/MB?Label=MB/system-board/SC?Label=SC/sc (fru)
/frutree/chassis/MB?Label=MB/system-board/BAT?Label=BAT
/frutree/chassis/MB?Label=MB/system-board/BAT?Label=BAT/battery (fru)
/frutree/chassis/MB?Label=MB/system-board/P0?Label=P0
/frutree/chassis/MB?Label=MB/system-board/P0?Label=P0/cpu (fru)
/frutree/chassis/MB?Label=MB/system-board/P0?Label=P0/cpu/F0?Label=F0
/frutree/chassis/MB?Label=MB/system-board/P0?Label=P0/cpu/F0?Label=F0/fan-unit
(fru)
/frutree/chassis/MB?Label=MB/system-board/P0?Label=P0/cpu/F1?Label=F1
/frutree/chassis/MB?Label=MB/system-board/P0?Label=P0/cpu/F1?Label=F1/fan-unit
(fru).........
```

コード例 1-12 に、prtfru コマンドに -c オプションを指定した場合に生成される SEEPROM データの一部を示します。この出力には、コンテナとそのデータのみが表示されます。FRU の階層ツリーは出力されません。

コード例 1-12　　prtfru -c コマンドの出力

```
# prtfru -c
/frutree/chassis/MB?Label=MB/system-board (container)
   SEGMENT: SD
      /ManR
      /ManR/UNIX_Timestamp32: Mon Dec  2 19:47:38 PST 2002
      /ManR/Fru_Description: FRUID,INSTR,M'BD,2X1.28GHZ,CPU
      /ManR/Manufacture_Loc: Hsinchu,Taiwan
      /ManR/Sun_Part_No: 3753120
      /ManR/Sun_Serial_No: 000615
      /ManR/Vendor_Name: Mitac International
      /ManR/Initial_HW_Dash_Level: 02
      /ManR/Initial_HW_Rev_Level: 0E
      /ManR/Fru_Shortname: MOTHERBOARD
      /SpecPartNo: 885-0076-11
/frutree/chassis/MB?Label=MB/system-board/P0?Label=
P0/cpu/B0?Label=B0/bank/D0?La
bel=D0/mem-module (container)
/frutree/chassis/MB?Label=MB/system-board/P0?Label=
P0/cpu/B0?Label=B0/bank/D1?La
bel=D1/mem-module (container).........
```

prtfru コマンドが表示するデータは、FRU の種類によって異なります。一般的に、次の情報が含まれます。

- FRU の説明
- メーカーの名前と所在地
- パーツ番号およびシリアル番号
- ハードウェアのバージョン

psrinfo コマンド

psrinfo コマンドは、各 CPU がオンラインになった日付と時刻を表示します。冗長 (-v) オプションを指定すると、クロックスピードを含む CPU の追加情報が表示されます。コード例 1-13 に、-v オプションを指定した psrinfo コマンドの出力例を示します。

コード例 1-13　　psrinfo -v コマンドの出力

```
# psrinfo -v
Status of processor 0 as of: 07/28/2003 14:43:29
  Processor has been on-line since 07/21/2003 18:43:37.
  The sparcv9 processor operates at 1280 MHz,
        and has a sparcv9 floating point processor.
Status of processor 1 as of: 07/28/2003 14:43:29
  Processor has been on-line since 07/21/2003 18:43:36.
  The sparcv9 processor operates at 1280 MHz,
        and has a sparcv9 floating point processor
```

showrev コマンド

showrev コマンドは、現在のハードウェアおよびソフトウェアのバージョン情報を表示します。コード例 1-14 に、showrev コマンドの出力例を示します。

コード例 1-14　　showrev コマンドの出力

```
# showrev
Hostname: vsp78-36
Hostid: 8328c87b
Release: 5.8
Kernel architecture: sun4u
Application architecture: sparc
Hardware provider: Sun_Microsystems
Domain: vsplab.SFBay.Sun.COM
Kernel version: SunOS 5.8 Generic 108528-18 November 2002
```

showrev コマンドに -p オプションを指定すると、インストールされているパッチが表示されます。コード例 1-15 に、showrev コマンドに -p オプションを指定した場合の出力例の一部を示します。

コード例 1-15　　showrev -p コマンドの出力

```
Patch: 109729-01 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsu
Patch: 109783-01 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsu
Patch: 109807-01 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsu
Patch: 109809-01 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsu
Patch: 110905-01 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsu
Patch: 110910-01 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsu
Patch: 110914-01 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsu
Patch: 108964-04 Obsoletes:  Requires:  Incompatibles:  Packages: SUNWcsr
```

▼ Solaris プラットフォームのシステム情報コマンドを実行する

- **コマンドプロンプトで、表示するシステム情報に応じたコマンドを入力します。**

  詳細は、19 ページの「Solaris ソフトウェアのシステム情報コマンド」を参照してください。表 1-6 に、コマンドの概要を示します。

表 1-6　Solaris プラットフォームの情報表示コマンド

コマンド	表示される情報	入力内容	備考
`prtconf`	システムの構成情報	`/usr/sbin/prtconf`	—
`prtdiag`	診断および構成情報	`/usr/platform/sun4u/sbin/prtdiag`	詳細情報を表示するには、`-v` オプションを使用します。
`prtfru`	FRU の階層および SEEPROM メモリーの内容	`/usr/sbin/prtfru`	階層を表示するには、`-l` オプションを使用します。SEEPROM データを表示するには、`-c` オプションを使用します。
`psrinfo`	各 CPU がオンラインになった日付および時刻、プロセッサのクロックスピード	`/usr/sbin/psrinfo`	クロックスピードおよびその他のデータを表示するには、`-v` オプションを使用します。
`showrev`	ハードウェアおよびソフトウェアのバージョン情報	`/usr/bin/showrev`	ソフトウェアのパッチを表示するには、`-p` オプションを使用します。

最新の診断テストの結果

最新の POST および OpenBoot 診断テスト結果の概要は、電源を再投入したあとも残っています。

▼ 最新のテスト結果を参照する

1. `ok` プロンプトを表示します。

2. 次のいずれかを実行します。

   ■ 最新の POST の結果の概要を表示するには、次のように入力します。

```
ok show-post-results
```

   ■ 最新の OpenBoot 診断テストの結果の概要を表示するには、次のように入力します。

```
ok show-obdiag-results
```

このコマンドを実行すると、そのシステムのハードウェア部品の一覧と、各部品に対する POST または OpenBoot 診断テストの結果 (合格または不合格) が表示されます。

OpenBoot 構成変数

IDPROM に格納されているスイッチおよび診断構成変数は、POST および OpenBoot 診断テストの実施方法および実施時期を決定します。この節では、OpenBoot 構成変数の表示および変更方法について説明します。重要な OpenBoot 構成変数の一覧は、表 1-4 を参照してください。

OpenBoot 構成変数の変更は、次の再起動時に有効になります。

▼ OpenBoot 構成変数を参照および設定する

● **サーバーを停止して、ok プロンプトを表示します。**

   ■ すべての OpenBoot 構成変数の現在の設定を表示するには、printenv コマンドを使用します。

   次に、このコマンドの出力例の一部を示します。

```
ok printenv
Variable Name         Value                          Default Value

diag-level            min                             min
diag-switch?          false                           false
```

- OpenBoot 構成変数を設定または変更するには、setenv コマンドを使用します。

```
ok setenv diag-level max
diag-level =          max
```

- 複数のキーワードを指定できる OpenBoot 構成変数を設定する場合は、キーワードをスペースで区切って指定します。

watch-net および watch-net-all コマンドを使用したネットワーク接続の確認

watch-net 診断テストは、プライマリネットワークインタフェースの Ethernet パケットを監視します。watch-net-all 診断テストは、プライマリネットワークインタフェースと、システムボードに接続されたすべての追加ネットワークインタフェースの Ethernet パケットを監視します。システムが受信した正常なパケットは、ピリオド (.) で示されます。フレーミングエラー、巡回冗長検査 (CRC) エラーなどのエラーは X で示されて、そのエラーの説明も表示されます。

- **watch-net 診断テストを開始するには、ok プロンプトで watch-net コマンドを入力します (コード例 1-16)。**

コード例 1-16　　watch-net 診断の出力メッセージ

```
{0} ok watch-net
Internal loopback test -- succeeded.
Link is -- up
Looking for Ethernet Packets.
'.' is a Good Packet. 'X' is a Bad Packet.
Type any key to stop.............................
```

- **watch-net-all 診断テストを開始するには、ok プロンプトで watch-net-all を入力します (コード例 1-17)。**

コード例 1-17　　watch-net-all 診断の出力メッセージ

```
{0} ok watch-net-all
/pci@1f,0/pci@1,1/network@c,1
Internal loopback test -- succeeded.
Link is -- up
Looking for Ethernet Packets.
'.' is a Good Packet. 'X' is a Bad Packet.
Type any key to stop.
```

自動システム回復

注 – 自動システム回復 (Automatic System Recovery : ASR) は、Netra 240 サーバーがサポートするもう 1 つの機能である自動サーバー再起動 (Automatic Server Restart) とは異なります。自動サーバー再起動の詳細は、第 3 章を参照してください。

ASR には、自己診断機能と自動構成機能があり、障害が発生したハードウェア部品を検出して構成から解除します。ASR を有効にすると、ハードウェアの重大でない問題や障害が発生したあとに、サーバーが動作を回復できるようになります。

ASR によって部品が監視されていて、その部品がなくてもサーバーが動作できる場合には、部品に問題や障害が発生してもサーバーは自動的に再起動します。こうして、ハードウェア部品の障害によってシステム全体の動作が停止すること、またはシステムで繰り返し問題が発生することを回避できます。

電源投入シーケンス中に障害が検出された場合には、障害のある部品は使用不可になります。その部品がなくてもシステムが機能できれば、起動処理は継続されます。

この縮退起動の機能をサポートするため、OpenBoot ファームウェアは IEEE 1275 に準拠したクライアントインタフェースを使用して (デバイスツリーを介して)、該当するデバイスツリーのノードに適切な状態属性を作成し、デバイスに「障害 (Failed)」または「使用不可 (Disabled)」のマークを付けます。Solaris OS は、このようにマークが付けられたサブシステムのドライバを起動しません。

障害が発生した部品が電気的に休止した状態にあれば (たとえば、不規則なバスエラーやシグナルノイズを発生させていなければ)、システムは自動的に再起動して保守呼び出しを行っている間も動作を継続できます。

「障害 (failed)」または「使用不可 (disabled)」のマークが付いたデバイスを新しいデバイスと交換すると、再起動時に、OpenBoot ファームウェアによってデバイスの状態が自動的に変更されます。

注 – ASR 機能は、使用可能に設定しないと起動されません (31 ページの「ASR を使用可能にする」を参照)。

auto-boot オプション

`auto-boot?` 設定は、リセットのたびにファームウェアが自動的にオペレーティングシステムを起動するかどうかを制御します。デフォルト設定は `true` です。

auto-boot-on-error? 設定は、サブシステムの障害が検出されたときにシステムが縮退起動を試みるかどうかを制御します。自動縮退起動を使用可能にするには、auto-boot? および auto-boot-on-error? の両方に true を設定する必要があります。

- **スイッチに値を設定するには、次のように入力します。**

```
ok setenv auto-boot? true
ok setenv auto-boot-on-error? true
```

注 – auto-boot-on-error? のデフォルト設定は false です。そのため、この設定を true に変更しないかぎり、システムは縮退起動を試みません。また、縮退起動が可能に設定されていても、重大で回復不可能なエラーがあるときは、システムは縮退起動を試みません。重大で回復不可能なエラーの例は、30 ページの「エラー処理の概要」を参照してください。

エラー処理の概要

電源投入シーケンスでのエラー処理は、次の 3 つの状況に分類されます。

- POST または OpenBoot 診断でエラーが検出されない場合、auto-boot? が true に設定されているときは、システムは起動を試みます。
- POST または OpenBoot 診断で、重大でないエラーのみが検出された場合、auto-boot? が true に、auto-boot-on-error? が true に設定されているときは、システムは起動を試みます。

注 – POST または OpenBoot 診断が、通常の起動デバイスに関する重大でないエラーを検出した場合は、OpenBoot ファームウェアは自動的に障害のあるデバイスを構成解除し、構成変数 boot-device で次に指定されている起動デバイスからの起動を試みます。

- POST または OpenBoot 診断で重大なエラーが検出された場合は、auto-boot? または auto-boot-on-error? の設定にかかわらず、システムは起動されません。重大で回復不可能なエラーには、次のものがあります。
  - すべての CPU の障害
  - すべての論理メモリーバンクの障害
  - フラッシュ RAM の巡回冗長検査 (CRC) の障害
  - 重大な現場交換可能ユニット (FRU) PROM 構成データの障害
  - 重大な特定用途向け集積回路 (ASIC) の障害

リセットシナリオ

OpenBoot の 3 つの構成変数 diag-switch?、obdiag-trigger、および post-trigger は、システムでリセットイベントが発生したときにシステムがファームウェア診断を実行する方法を制御します。

標準のシステムリセットプロトコルは、変数 diag-switch? に true が設定されていないかぎり、POST および OpenBoot 診断を完全に省略します。変数 diag-switch? のデフォルト設定は false です。ASR は障害検出をファームウェア診断に依存しているため、ASR を実行するには、diag-switch? を true に設定する必要があります。詳細は、31 ページの「ASR を使用可能にする」を参照してください。

どのリセットイベントが自動的にファームウェア診断を行うかを制御するには、obdiag-trigger および post-trigger を使用します。これらの変数の説明および使用方法については、9 ページの「POST 診断の制御」および 16 ページの「OpenBoot 診断テストの制御」を参照してください。

▼ ASR を使用可能にする

1. **システムの** ok **プロンプトで、次のように入力します。**

```
ok setenv diag-switch? true
ok setenv auto-boot? true
ok setenv auto-boot-on-error? true
```

2. **変数** obdiag-trigger **に** power-on-reset、error-reset、**または** user-reset **を設定します。**

   次のように入力します。

```
ok setenv obdiag-trigger user-reset
```

3. **次のように入力します。**

```
ok reset-all
```

また、OpenBoot 変数 auto-boot? が true (デフォルト値) に設定されていると、システムが自動的に起動されます。

注 – 正面パネルのオン/スタンバイボタンを使用してシステムの電源を再投入する方法でも、パラメタの変更を保存できます。

▼ ASR を使用不可にする

1. システムの ok プロンプトで、次のように入力します。

```
ok setenv diag-switch? false
```

2. 次のように入力します。

```
ok reset-all
```

パラメタの変更が、システムに永続的に保存されます。

注 – 正面パネルのオン/スタンバイボタンを使用してシステムの電源を再投入する方法でも、パラメタの変更を保存できます。

第2章

SunVTS ソフトウェア

この章では、SunVTS について説明します。この章は、次の節で構成されます。

SunVTS ソフトウェアの概要

Netra 240 サーバーは、SunVTS 5.1 Patch Set 5 (PS5) ソフトウェアおよびそれ以降の互換性のあるバージョンをサポートしています。

SunVTS (Sun Validation Test Suite) ソフトウェアは、ハードウェアコントローラ、装置、およびプラットフォームの構成と機能性を検証するためのオンライン診断ツールです。このツールは Solaris OS 上で動作して、次のインタフェースを提供します。

- コマンド行インタフェース (CLI)
- シリアル (`tty`) インタフェース

SunVTS ソフトウェア群は、システムおよび周辺装置の負荷テストを実行します。SunVTS ソフトウェアセッションは、ネットワークを介して表示および制御できます。遠隔マシンを使用して、SunVTS テストセッションの進行状況の表示や、テストオプションの変更、ネットワーク上のほかのマシンのすべてのテスト機能の制御ができます。

SunVTS ソフトウェアは、次の 3 つのテストモードで実行できます。

- 接続 (Connection) モード ー デバイスコントローラの存在を検証します。通常、この作業には数分しかかかりません。システム接続の「健全性の確認」のために有効な方法です。

- 機能 (Functional) モード ー SunVTS ソフトウェアは、選択した特定のサブシステムの動作テストだけを行います。このモードがデフォルトです。
- 自動構成 (Auto Config) モード ー SunVTS ソフトウェアは自動的にすべてのサブシステムを検出し、次のいずれかの方法で動作をテストします。
  - 信用 (Confidence) テスト ー すべてのサブシステムに対してテストを行い、1 回ずつ合格するとテストを終了します。一般的なシステム構成では、このテストには 1 ～ 2 時間かかります。
  - 総合 (Comprehensive) テスト ー すべてのサブシステムに対して繰り返しテストを行います。24 時間かかる場合があります。

SunVTS ソフトウェアは、多数のテストを並行して実行できるので、大量のシステム資源を消費します。実際に稼働しているシステムでこのソフトウェアを実行する場合は注意が必要です。SunVTS ソフトウェアの総合テストモードでシステムの負荷テストを行う場合は、そのシステム上では、ほかの作業を同時に行わないでください。

SunVTS ソフトウェアでテストするサーバーでは、Solaris OS が動作している必要があります。SunVTS ソフトウェアはオプションのパッケージであるため、システムにインストールされていない場合があります。詳細は、36 ページの「SunVTS ソフトウェアがインストールされているかどうかを確認する」を参照してください。

SunVTS テスト

SunVTS ソフトウェアを使用すると、遠隔で接続したサーバー上のテストセッションの監視および制御を実行できます。表 2-1 に、このツールで実行できるテストの一部を示します。

表 2-1　SunVTS ソフトウェアテスト

SunVTS ソフトウェアテスト	説明
cputest	CPU のテスト
disktest	ローカルディスクドライブのテスト
dvdtest	DVD-ROM ドライブのテスト
n240atest	アラームカードのアラームリレー、LED、および FRU ID のテスト
fputest	浮動小数点ユニットのテスト
nettest	システムボード上の Ethernet ハードウェア、およびオプションの PCI カード上のネットワーキングハードウェアのテスト
netlbtest	Ethernet アダプタがパケットの送受信を実行できることを確認するためのループバックテストの実行

表 2-1　SunVTS ソフトウェアテスト (続き)

SunVTS ソフトウェアテスト	説明
pmem	物理メモリーのテスト (読み取りのみ)
sutest	サーバーのオンボードシリアルポートのテスト
vmem	仮想メモリー (スワップパーティションと物理メモリーの組み合わせ) のテスト
env6test	環境装置のテスト
ssptest	ALOM ハードウェアのテスト
i2c2test	I^2C 装置の動作のテスト

SunVTS ソフトウェアとセキュリティー

SunVTS ソフトウェアのインストール時に、基本セキュリティーまたは Sun Enterprise Authentication Mechanism™ (SEAM) セキュリティーのいずれかを選択する必要があります。基本セキュリティーでは、SunVTS ソフトウェアのインストール先ディレクトリにあるローカルのセキュリティーファイルを使用して、ユーザー、グループ、およびホストに対する SunVTS ソフトウェアの使用権限を制限します。SEAM セキュリティーは、標準のネットワーク認証プロトコルである Kerberos に基づいて、セキュリティー保護されたユーザー認証およびデータの完全性、ネットワークトランザクションの機密性を提供します。

SEAM セキュリティーを使用する場合は、ネットワーク環境に SEAM のクライアントおよびサーバーソフトウェアをインストールして、Solaris ソフトウェアおよび SunVTS ソフトウェアの両方で正しく設定する必要があります。SEAM セキュリティーを使用していない場合は、SunVTS ソフトウェアのインストール時に、SEAM オプションを選択しないでください。

インストール中に間違ったセキュリティースキーマを使用可能にした場合、または選択したセキュリティースキーマを正しく設定しなかった場合には、SunVTS ソフトウェアテストを実行できません。詳細は、『SunVTS ユーザーマニュアル』および SEAM ソフトウェアに付属するマニュアルを参照してください。

▼ SunVTS ソフトウェアがインストールされているかどうかを確認する

- 次のように入力します。

```
# pkginfo -l SUNWvts
```

  - SunVTS ソフトウェアがインストールされている場合は、パッケージに関する情報が表示されます。
  - SunVTS ソフトウェアがインストールされていない場合は、次のエラーメッセージが表示されます。

```
ERROR: information for "SUNWvts" was not found
```

SunVTS ソフトウェアのインストール

デフォルトでは、Netra 240 サーバー上に SunVTS ソフトウェアはインストールされていません。Solaris OS のサプリメント CD からインストールするか、次の Web サイトから最新版をダウンロードしてください。

`http://www.sun.com/oem/products/vts/`

注 – Netra 240 サーバーは、SunVTS 5.1 Patch Set 5 (PS5) ソフトウェアおよびそれ以降の互換性のあるバージョンをサポートしています。

SunVTS ソフトウェアの使用方法の詳細は、使用している Solaris ソフトウェアのバージョンに対応する SunVTS の関連マニュアルを参照してください。また、前述の Web サイトで、SunVTS ソフトウェアの詳細およびインストール方法を確認することもできます。

SunVTS ソフトウェアのマニュアルの参照

SunVTS 関連マニュアルは、Solaris メディアキットに付属するソフトウェアサプリメント CD に収録されています。また、`http://docs.sun.com` から入手することもできます。

SunVTS ソフトウェアの関連マニュアルには、次の情報も記載されています。

- 『SunVTS ユーザーマニュアル』には、SunVTS 診断ソフトウェアのインストール方法、設定方法、および実行方法が記載されています。
- 『SunVTS リファレンスカード』には、SunVTS インタフェースの使用方法の要約が記載されています。
- 『SunVTS テストリファレンスマニュアル』には、SunVTS の各テストの詳細が記載されています。

第3章

Advanced Lights Out Manager

この章では、Sun Advanced Lights Out Manager (ALOM) ソフトウェアの概要を説明します。この章は、次の節で構成されます。

Advanced Lights Out Manager の概要

Netra 240 サーバーは、Sun Advanced Lights Out Manager がインストールされた状態で出荷されます。システムコンソールは、デフォルトで ALOM に接続されて、起動時にサーバーコンソールの情報を表示するように設定されています。

ALOM を使用すると、SERIAL MGT ポートを使用したシリアル接続、または NET MGT ポートを使用した Ethernet 接続のいずれかを介して、サーバーを監視および制御できます。Ethernet 接続の設定方法については、『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) を参照してください。

注 – 「SERIAL MGT」のラベルが付いた ALOM のシリアルポートは、サーバー管理専用のポートです。汎用シリアルポートが必要な場合は、「10101」のラベルが付いたシリアルポートを使用してください。

ALOM は、サーバーまたは ALOM に関連するハードウェア障害およびその他のイベントを、電子メールで通知するように設定できます。

ALOM 回路は、サーバーのスタンバイ電力を使用します。これは、次のことを意味します。

- ALOM は、サーバーが電源に接続された直後から、電源ケーブルを取り外すまで動作します。
- ALOM のファームウェアおよびソフトウェアは、サーバーのオペレーティングシステムがオフラインになっても動作を継続します。

表 3-1 に、ALOM の監視の対象と、各対象についてこのソフトウェアが提供する情報の一覧を示します。

表 3-1 ALOM の監視の対象

対象	提供される情報
ハードドライブ	存在の有無、状態
システムおよび CPU のファン	回転速度、状態
CPU	存在の有無、温度、温度に関する警告または障害報告
電源装置	存在の有無、状態
システム温度	周辺の温度、温度に関する警告または障害報告
サーバーの正面パネル	回転式スイッチの設定位置、LED 状態
電圧	状態、しきい値
SCSI および USB の回路遮断器	状態
ドライ接点リレーアラーム	状態

ALOM のポート

デフォルトの管理用ポートは、「SERIAL MGT」のラベルの付いたポートです。このポートのコネクタは RJ-45 で、サーバーの管理だけに使用します。このポートは、外部コンソールへの ASCII 接続だけをサポートします。サーバーをはじめて操作するときは、このポートを使用します。

もう 1 つのシリアルポートには「10101」のラベルが付いています。これは、データのシリアル転送に使用できる汎用のポートです。このポートのコネクタは DB-9 です。ピン配列については、『Netra 240 サーバー設置マニュアル』(Part No. 817-4997) を参照してください。

また、このサーバーは、10BASE-T Ethernet の管理ドメインインタフェースも 1 つ備えています。これには、「NET MGT」のラベルが付いています。このポートを使用するには、ALOM の設定を変更する必要があります。詳細は、『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) を参照してください。

admin パスワードの設定

最初の電源投入後、ALOM ソフトウェアに切り替えると、sc> プロンプトが表示されます。この時点で、ユーザーアクセス権を必要としないコマンドを実行できます(『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) を参照)。ユーザーアクセス権を必要とするコマンドを実行しようとすると、ユーザー admin のパスワードを設定するプロンプトが表示されます。

- **パスワードを指定するプロンプトが表示されたら、admin ユーザーのパスワードを設定します。**

  パスワードは、次の条件を満たす必要があります。

  - 2 文字以上の英字が含まれていること
  - 1 文字以上の数字または特殊文字が含まれていること
  - 6 ～ 8 文字であること

  パスワードを設定すると、admin ユーザーにはすべての権限が与えられ、すべての ALOM CLI コマンドを実行できるようになります。このユーザーは、あとで ALOM に切り替えるときに、管理者パスワードを使用してログインするように求められます。

ALOM の基本機能

この節では、ALOM の基本機能の一部を説明します。詳細は、『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006)、および『Netra 240 Server Release Notes』(Part No. 817-3142) を参照してください。

▼ ALOM プロンプトに切り替える

- コマンドプロンプトで、#. キーシーケンスを入力します。

```
# #.
```

注 – ALOM プロンプトに切り替えるときは、ユーザー ID admin でログインします。詳細は、41 ページの「admin パスワードの設定」を参照してください。

▼ サーバーコンソールプロンプトに切り替える

- 次のように入力します。

```
sc> console
```

サーバーコンソールには、同時に複数の ALOM ユーザーが接続できますが、コンソールに文字を入力できるユーザーは 1 人だけです。

ほかのユーザーがログインして書き込み権限を持っている場合には、console コマンドを実行したときに次のメッセージが表示されます。

```
sc> Console session already in use. [view mode]
```

▼ コンソールへの書き込み権限をほかのユーザーから取得する

- 次のように入力します。

```
sc> console -f
```

自動サーバー再起動

注 – 自動システム回復 (Automatic System Recovery : ASR) は、Netra 240 サーバーがサポートするもう 1 つの機能である自動サーバー再起動 (Automatic Server Restart) とは異なります。

自動サーバー再起動は、ALOM の機能の 1 つです。この機能は、デフォルトでは、動作中の Solaris OS を監視し、障害が発生するとファイルシステムに対して sync を実行してサーバーを再起動します。

ALOM は、ウォッチドッグプロセスをカーネルの監視だけに使用します。プロセスがハングアップしてもカーネルが動作していれば、ALOM はサーバーを再起動しません。ALOM ウォッチドッグのチェック間隔とタイムアウトのパラメタは、ユーザーからは設定できません。

カーネルがハングアップしてウォッチドッグのタイムアウトが発生すると、ALOM はこのイベントを通知および記録してから、ユーザーが設定した次の 3 つの動作のいずれかを実行します。

- xir ― これはデフォルトの動作で、ファイルシステムに対して sync を実行してサーバーを再起動します。システムがハングアップした場合は、15 分後に ALOM がハードリセットを実行します。
- Reset ― ハードリセットを実行して、迅速にシステムを回復します。ただし、ハングアップの原因を示す診断データは保存されず、重大な損傷が発生する可能性があります。
- None ― ウォッチドッグのタイムアウトが通知されたあと、システムは無期限にハング状態になります。

詳細は、『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) の「`sys_autorestart`」を参照してください。

自動システム回復 (ASR) については、第 1 章を参照してください。

環境の監視および制御

Netra 240 サーバーおよびその部品は、環境監視サブシステム機能によって、次の問題から保護されます。

- 極端な低温および高温
- システム内の通気の不足

- 欠落した部品または誤って構成された部品がある状態での動作
- 電源装置の障害
- 内部ハードウェア障害

監視および制御機能は、ALOM のファームウェアによって処理されるため、システムが停止したり起動できない場合でも監視機能は動作を続けることができます。また、ALOM ファームウェアによってシステムを監視することでシステムが解放され、CPU およびメモリー資源をオペレーティングシステムおよびアプリケーションソフトウェアのためだけに使用できるようになります。

環境監視サブシステムは、業界標準の I^2C バスを使用します。I^2C バスは、システム全体で使用される単純な 2 線式のシリアルバスです。このバスによって、温度センサー、ファン、電源装置、状態表示 LED、および正面パネルの回転式システム制御スイッチの監視と制御が可能になります。

サーバーには、サーバーの周辺温度および 2 つの CPU のダイ温度を監視する 3 つの温度センサーが付いています。監視サブシステムは、各センサーにポーリングしてサンプリングした温度に基づいて、適正温度を超えた状態または適正温度より低い状態があれば通知して対処します。その他の I^2C 装置は、部品の有無および障害を検出します。

ハードウェアおよびソフトウェアは、格納装置内の温度が所定の安全動作範囲を超えないようにします。センサーが監視する温度が低温警告しきい値より低くなるか、高温警告しきい値を超えると、監視サブシステムソフトウェアによって、正面パネルおよび背面パネルのシステム保守要求 LED が点灯します。この温度状態が持続し、ソフトウェアによる停止が行われる高温または低温のしきい値に達すると、システムの正常な停止が開始されます。この温度状態が持続し、ハードウェアによる停止が行われる高温または低温のしきい値に達すると、システムの強制停止が開始されます。

エラーおよび警告メッセージは、システムコンソールに送信され、`/var/adm/messages` ファイルに記録されます。保守要求 LED は、障害診断のため、システムの自動停止後も点灯し続けます。

システムコンソールに送信され、`/var/adm/messages` ファイルに記録されるメッセージのタイプは、ALOM のユーザー変数 `sc_clieventlevel` および `sys_eventlevel` の設定方法によって異なります。これらの変数の設定方法については、『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) を参照してください。

表 3-2 Netra 240 サーバーの格納装置の温度のしきい値

温度のしきい値	温度	サーバーの動作
低温、ハードウェアによる停止	-11 ℃	サーバーは、システムの強制停止を開始します。
低温、ソフトウェアによる停止	-9 ℃	サーバーは、システムの正常な停止を実行します。
低温警告	-7 ℃	サーバーは、正面パネルおよび背面パネルのシステム保守要求 LED インジケータを点灯します。

表 3-2　Netra 240 サーバーの格納装置の温度のしきい値 (続き)

温度のしきい値	温度	サーバーの動作
高温警告	57 ℃	サーバーは、正面パネルおよび背面パネルのシステム保守要求 LED インジケータを点灯します。
高温、ソフトウェアによる停止	60 ℃	サーバーは、システムの正常な停止を実行します。
高温、ハードウェアによる停止	63 ℃	サーバーは、システムの強制停止を開始します。

また、監視サブシステムは、4 つのシステム送風機で発生した障害を検出するように設計されています。送風機のいずれかに障害が発生すると、監視サブシステムが障害を検出してシステムコンソールに送信するエラーメッセージを生成し、`/var/adm/messages` ファイルにそのメッセージを記録して、保守要求 LED を点灯します。

電源サブシステムも同じ方法で監視されます。監視サブシステムは、電源装置の状態をときどきポーリングして、各電源装置の出力および入力、電源装置の有無を示します。

電源装置に障害が検出されると、エラーメッセージがシステムコンソールに送信され、`/var/adm/messages` ファイルに記録されます。また、各電源装置の LED が点灯して、障害が発生したことを示します。システム保守要求 LED も点灯して、システム障害を示します。ALOM のコンソール警告は、電源装置の障害を記録します。

電源サブシステムおよびファンの回転速度の警告しきい値を確認するには、ALOM の `showenvironment` コマンドを使用します。このコマンドの使用方法については、『Sun Advanced Lights Out Manager ソフトウェアユーザーマニュアル Netra 240 サーバー』(Part No. 817-5006) を参照してください。

付録A

アラームリレー出力のアプリケーションプログラミングインタフェース

この付録では、アラームの状態を取得および設定するプログラムの例 (コード例 A-1) を示します。このアプリケーションでは、ioctl の LOMIOCALSTATE を使用して各アラームの状態を取得し、ioctl の LOMIOCALCTL を使用して各アラームを個別に設定できます。アラームインジケータの詳細は、『Netra 240 Server Service Manual』(Part No. 817-2699) を参照してください。

コード例 A-1　　アラームの状態を取得および設定するプログラムの例

```
#include <sys/types.h>
#include <string.h>
#include <stdlib.h>
#include <sys/unistd.h>
#include <fcntl.h>
#include "lom_io.h"

#define ALARM_INVALID   -1
#define LOM_DEVICE  "/dev/lom"

static void usage();
static void get_alarm(const char *alarm);
static int set_alarm(const char *alarm, const char *alarmval);
static int parse_alarm(const char *alarm);
static int lom_ioctl(int ioc, char *buf);
static char *get_alarmval(int state);
static void get_alarmvals();


main(int argc, char *argv[])
{

        if (argc < 3) {
                usage();
                if (argc == 1)
                        get_alarmvals();
                exit(1);
```

コード例 A-1　　アラームの状態を取得および設定するプログラムの例 (続き)

```
        }

        if (strcmp(argv[1], "get") == 0) {
                if (argc != 3) {
                        usage();
                        exit (1);
                }
                        get_alarm(argv[2]);
        }
        else
        if (strcmp(argv[1], "set") == 0) {
                if (argc != 4) {
                        usage();
                        exit (1);
                }
                set_alarm(argv[2], argv[3]);
        } else {
                usage();
                exit (1);
        }
}

static void
usage()
{
        printf("usage: alarm [get|set] [crit|major|minor|user] [on|off]\n");
}

static void
get_alarm(const char *alarm)
{
        ts_aldata_t     ald;
        int altype = parse_alarm(alarm);
        char *val;

        if (altype  == ALARM_INVALID) {
                usage();
                exit (1);
        }

        ald.alarm_no = altype;
        ald.alarm_state = ALARM_OFF;

        lom_ioctl(LOMIOCALSTATE, (char *)&ald);

        if ((ald.alarm_state != ALARM_OFF) &&
                        (ald.alarm_state != ALARM_ON)) {
```

コード例 A-1　　アラームの状態を取得および設定するプログラムの例 (続き)

```
                printf("Invalid value returned: %d\n", ald.alarm_state);
                exit(1);
        }

        printf("ALARM.%s = %s\n", alarm, get_alarmval(ald.alarm_state));
}

static int
set_alarm(const char *alarm, const char *alarmstate)
{
        ts_aldata_t     ald;
        int alarmval = ALARM_OFF, altype = parse_alarm(alarm);

        if (altype  == ALARM_INVALID) {
                usage();
                exit (1);
        }

        if (strcmp(alarmstate, "on") == 0)
                alarmval = ALARM_ON;
        else
        if (strcmp(alarmstate, "off") == 0)
                alarmval = ALARM_OFF;
        else {
                usage();
                exit (1);
        }

        ald.alarm_no = altype;
        ald.alarm_state = alarmval;

        if (lom_ioctl(LOMIOCALCTL, (char *)&ald) != 0) {
                printf("Setting ALARM.%s to %s failed\n", alarm, alarmstate);
                return (1);
        } else {
                printf("Setting ALARM.%s successfully set to %s\n", alarm,
alarmstate);
                return (1);
        }
}

static int
parse_alarm(const char *alarm)
{
        int altype;

        if (strcmp(alarm, "crit") == 0)
```

コード例 A-1　　アラームの状態を取得および設定するプログラムの例 (続き)

```
                altype = ALARM_CRITICAL;
        else
        if (strcmp(alarm, "major") == 0)
                altype = ALARM_MAJOR;
        else
        if (strcmp(alarm, "minor") == 0)
                altype = ALARM_MINOR;
        else
        if (strcmp(alarm, "user") == 0)
                altype = ALARM_USER;
        else {
                printf("invalid alarm value: %s\n", alarm);
                altype = ALARM_INVALID;
        }

        return (altype);

}

static int
lom_ioctl(int ioc, char *buf)
{
        int fd, ret;

        fd = open(LOM_DEVICE, O_RDWR);

        if (fd == -1) {
                printf("Error opening device: %s\n", LOM_DEVICE);
                exit (1);
        }

        ret = ioctl(fd, ioc, (void *)buf);

        close (fd);

        return (ret);
}

static char *
get_alarmval(int state)
{
        if (state == ALARM_OFF)
                return ("off");
        else
        if (state == ALARM_ON)
                return ("on");
        else
```

コード例 A-1　　アラームの状態を取得および設定するプログラムの例 (続き)

```
                return (NULL);
}
static void
get_alarmvals()
{
        get_alarm("crit");
        get_alarm("major");
        get_alarm("minor");
        get_alarm("user");

}
```

索引