The UPS Howto

Table of Contents

Table of Contents

The UPS Howto

Harvey J. Stein, hjstein@bfr.co.il, Berger Financial Research, Ltd.

v2.42, 18 November 1997

This document will help you connect an uninterruptable power supply to a Linux box... if you're lucky...

6.Hardware notes

7.What to do when you're really stuck

8.Info on selected UPSs

9.How to shutdown other machines on the same UPS

1.Introduction

This HOWTO covers connecting a UPS to a computer running Linux. The idea is to connect the two in such a way that Linux can shutdown cleanly when the power goes out, and before the UPS's battery gives out.

This includes pointing out the existence of software packages which aid in establishing such communications, and detailing exactly how such communications are carried out. The latter often is unnecessary if you can find a software package that's already been configured for your UPS. Otherwise, you'll have to read on.

To a large extent this document is even more redundant than when I originally wrote it three years ago. All the basic information has always been contained in the `powerd` man page that comes with the `SysVinit` package. Whereas three years ago one could commonly find Linux distributions which didn't even include this man page, I don't believe this is the case any longer.

Furthermore, when I first wrote this Howto, there was no software other than `powerd.c` for Linux/UPS communications and control. Today there are quite afew UPS control packages available in Sunsite's UPS directory.

None the less, I'm continuing to maintain the UPS Howto. Why bother? Well,

- An additional general overview might help to understand how to connect a Linux system to a UPS, even if it's just the same information written differently.
- The HOWTO is serving as a repository for UPS specific data − there are many UPSs that haven't yet been incorporated into the general packages.
- The HOWTO contains additional details that aren't available in other documents.
- Some of the UPS software packages available in Sunsite's UPS directory seem to be quite sparsely documented. You might need to read this before you can understand how to use them.
- This thing seems to have a life of it's own now. It's clear when a Howto should be born. It's less clear when it should be put to sleep.

1.1 Contributors

I am forever indebted to those from whom I've received help, suggestions, and UPS specific data. The list includes:

- Hennus Bergman (hennus@sky.owl.nl)
- Charli (mefistos@impsat1.com.ar)
- Ciro Cattuto (Ciro Cattuto)
- Nick Christenson (npc@minotaur.jpl.nasa.gov)
- Lam Dang (angit@netcom.com)
- Markus Eiden (Markus@eiden.de)
- Dan Fandrich (dan@fch.wimsey.bc.ca)
- Ben Galliart (bgallia@orion.it.luc.edu)
- Danny ter Haar (dth@cistron.nl)
- Christian G. Holtje (docwhat@uiuc.edu)
- Raymond A. Ingles (inglesra@frc.com)
- Peter Kammer (pkammer@ics.uci.edu)
- Marek Michalkiewicz (ind43@sun1000.ci.pwr.wroc.pl)
- Jim Ockers (ockers@umr.edu)
- Evgeny Stambulchik (fnevgeny@plasma−gate.weizmann.ac.il)
- Clive A. Stubbings (cas@vjet.demon.co.uk)
- Miquel van Smoorenburg (miquels@cistron.nl)
- Slavik Terletsky (ts@polynet.lviv.ua)
- Tom Webster (webster@kaiwan.com)

Note that email addresses appearing below as excerpts from email messages can be out of date. The above is probably out of date too, but some of it's more recent than what's below.

Also, many apologies to anyone whom I've failed to note in this list. Please email me and I'll add you.

1.2 Important disclaimer

I really can't guarantee that any of this will work for you. Connecting a UPS to a computer can be a tricky business. One or the other or both might burn out, blow up, catch fire, or start World War Three. Furthermore, I only have direct experience with the Advice 1200 A UPS, and a 5kva Best Ferrups, and I didn't have to make a cable. So, BE CAREFUL. GATHER ALL INFORMATION YOU CAN ON YOUR UPS. THINK FIRST. DON'T IMPLICITLY TRUST ANYTHING YOU READ HERE OR ANYWHERE ELSE.

On the other hand, I managed to get everything working with my UPSs, without much information from the manufacturer, and without blowing anything up, so it is possible.

1.3 Other documents

This document does not cover the general features and capabilities of UPSs. For that type of information, you might turn to The UPS FAQ. It can also be found at ftp://rtfm.mit.edu/pub/usenet−by−hierarchy/comp/answers/UPS−faq. It is maintained by Nick Christenson (npc@minotaur.jpl.nasa.gov), but seems to have last been updated in 1995. In email to him, he'd like that you put `UPS` or `UPS FAQ` or something along these lines in the Subject line of the message.

There're also more and more UPS manufactures sprouting up on the net. Some of them actually supply useful information on their web sites. A convenient list of UPS manufacturers' web sites is available at The UPS Directory. Said site also has a UPS FAQ.

2.Important note on obsolete information

I've just discovered that some of the documentation below is obsolete. In particular, the `init` daemon that comes with the latest sysvinit package is more sophisticated than I've portrayed it to be. Although it seems that the current version is backward compatible with what's written here, it looks like it has some undocumented features which are **very important** for UPS support.

The control mechanism outlined below only allows `powerd` to give `init` one of two messages, namely `powerfail` or `powerok`. `init` runs one command when it receives `powerfail`, and another when it receives `powerok`. This leads to complicated `powerd` logic for dealing with low battery signals and other sorts of special situations.

Newer versions of `init` (as of version 2.58, it seems) are more sophisticated. These versions can be signaled to run one of `three` scripts. Thus, `init` can have a `powerfail` script for announcing a power outage, a `powerfailnow` script for doing an immediate shutdown, and a `powerok` script for halting any pending shutdowns. This is much cleaner than the gyrations one would have to go through with the mechanisms detailed below.

Although most of the discussion here assumes the old `init` communication method, I just added two new sections where the authors uses the new communcation method. These are sections Trust Energy Protector

400/650 and APC Smart−UPS 700. The former is especially detailed. Both include a `powerd.c` which signals `init` to do an immediate shutdown when a low battery signal is received, as well as the relevant `/etc/inittab` lines to make this work. Other than this, all I can tell you is to look at the source code for `init`.

Also, for all I know, many of the software packages listed below also use this newer communication method.

3.Smart and dumb UPSs.

UPSs fall into two categories, which I'll call ``smart'' and ``dumb''. The difference between the two is the amount of information one can get from the UPS and the amount of control one can exert over the UPS.

Dumb UPS

Connects to the computer via serial port.
Uses modem control lines to communicate with the computer.
Can signal whether or not the power is out.
Typically can signal whether or not the battery is low.
The computer can usually signal the UPS to turn itself off.

Smart UPS

Connects to the computer via serial port.
Communicates with the computer via normal data transfer through the serial port.
Typically has some sort of command language that the computer can use to get various pieces of information from the UPS, to set various operating parameters for the UPS, and to control the UPS (such as turning it off).

Usually smart UPSs can be operated in dumb mode. This is useful because as far as I know, the company which manufactures the most popular smart UPS (namely APC) will only disclose the communication protocol for their UPSs to people who sign a non−disclosure agreement.

As far as I know, the only smart UPS available which is easy to communicate with under Linux are those made by Best. Furthermore, BEST fully documents the smart mode (and the dumb mode) of their UPSs. BEST also supplies source code for programs which can communicate with their UPSs.

All the packages listed in section Software will communicate with a UPS in dumb mode. This is all you really need. The ones specifically for the APC UPSs make various claims as to being usable in smart mode, but I don't know exactly what they permit. A full implementation would give you a pop−up window with all sorts of fun gauges displaying various statistics for the UPS, such as load, internal temperature, fault history, input voltage, output voltage, etc. It seems like the `smupsd−0.9−1.i386.rpm` package (section Software) approaches this. I'm not sure about the others.

The rest of this document is pretty much confined to configuring your system to work with a dumb UPS. The general idea is about the same with a smart UPS, but the details of how `powerd` would need to work and what kind of cable you need are different for a smart UPS.

4.Software

Basically, all you need is a working `powerd` binary, usually found in `/sbin/powerd`. This is usually part of the `SysVinit` package. As far as I know, all current Linux distributions include a recent version of `SysVinit`. Very old versions didn't include `powerd`.

The only problem you might have is that your cable might not match how `powerd` is set up, in which case you'll have to either rewire your cable, or pick up a copy of `powerd.c` and modify it to work with your cable. Or, for that matter, you can always pick up one of the following packages, most of which allow you to configure them to match your cable.

As mentioned, an alternative to using the `powerd` that comes with the `SysVinit` package would be to use one of the UPS packages now available. There are many packages currently available to aid in setting up computer/ups communications. None of this was available when I first wrote this Howto, which is why I had to write it. In fact, there's a good chance that you might be able to use one of these software packages, and avoid this Howto entirely!

As of 15 March 1997 or so, Sunsite's UPS directory had quite a few packages available. Other sites seem to have UPS control packages available too. Here's what I've found to date (all but two from sunsite):

Enhanced_APC_BackUPS.tar.gz

A package for controlling APC Smart UPSs. Seems to basically follow the BUPS Howto (included here), but also seems to have some low battery warning support.

Enhanced_APC_UPSD−v1.4.tar.gz

The `.lsm` file says that it's formerly the above package, but it actually includes the above package as a `.tar.gz` file inside of this `tar.gz` file! The documentation is spotty. It seems to support APC UPSs in both smart mode and dumb mode, but I can't be sure.

apcd−0.5.tar.gz

Another package for controlling APC Smart UPSs. Seems to include some sort of master/slave support (i.e. − one machine signals others to shut down when the power goes out). Seems to use the UPS in smart mode, as opposed to via modem signal line toggling.

smupsd−0.9−1.i386.rpm

smupsd−0.9−1.src.rpm

The author (David E. Myers, dem@netsco.com) writes:
smupsd monitors an APC Smart−UPS[TM] under Red Hat[TM] Linux. Should power fail, smupsd will power down the system and the UPS in an orderly fashion.
smupsd has the following features:
Shuts down the system and the UPS based on either remaining UPS battery charge or elapsed time since power failure.
UPS parameters can be monitored live from any host with the graphical monitor program upsmon, written in JavaTM].
UPS parameters can be logged to a file for analysis and reporting.
When additional systems share the same UPS, instances of smupsd running on these systems can read UPS parameters from the one running on the system serially connected to the UPS (master/slave).
Network access from remote hosts can be controlled via the `/etc/hosts.allow` file.

genpower−1.0.1.tgz

A general UPS handling package. Includes configurations for many UPSs − two TrippLite configurations, and three APC configurations. Includes good documentation. A best buy.

powerd−2.0.tar.gz

A replacement for the `powerd` that comes with the `SysVinit` package. As opposed to comments included in the documentation it doesn't seem to have been merged into the `SysVinit` package as of version 2.62. Its advantages are that it can act as a server for other `powerds` running on other machines (for when you have a network of machines hanging off a single UPS), and it can be configured by config file − the source code doesn't have to be edited and recompiled.

upsd−1.0.tgz

Another replacement for `powerd`. Seems to be quite comparable in features to powerd−2.0.tar.gz.

checkups.tar

This package is for controlling Best UPSs. It's direct from Best's web site. Includes binaries for lots of `unix` flavors, but more importantly, it includes source code, so you can try it out under Linux, and if it doesn't work, you can try to fix it. The source code includes both ``basic checkups'' which controls the UPS in dumb mode, and ``advanced checkups'' which is a little more sophisticated − it will signal a shutdown when the UPS says it has X minutes of power remaining instead of just shutting down X minutes after the power goes out. The advanced checkups program also will shut down when the UPS registers various alarms such as High Ambient Temperature, Near Low Battery, Low AC Out, or User Test Alarm.

bestups−0.9.tar.gz

A package that might very well be on sunsite by the time you read this. It's a pair of communications module which works with Best Ferrups UPSs. It operates the UPS in smart mode. It inter−operates well with powerd−2.0 − useful if you have a big Best Ferrups UPS keeping up all the machines on a network.
NOTE − This package has yet to be uploaded to Sunsite. I keep begging the author to finish

and upload it, but he has yet to find the time.

LanSafe III

Deltec Electronics (and Exide) sell a software package called LanSafe III. They have a Linux version. It comes with their UPSs. They also say that it works with other UPSs (on the dumb level).

apcupsd−2.8.tar.gz

The author (Andre Hedrick, hedrick@astro.dyer.vanderbilt.edu) writes:
apcupsd−2.1.tar.gz replaces Enhanced−APC−UPSD.tar.gz
It is a very complete package for APC UPSs. There is support for the entire range of UPSs in their product line. I have now added smart mode signaling to the package and support with APC's own cables or a custom cable if you don't have an APC cable that is supported to date.

smartups−1.1.tgz

From the LSM:
A powerd and an X11 graphing utility which shows you the voltages, frequencies, load percentage and battery level in realtime. The protocol that the "Safeware" software uses, and "Tripplite" UPSs are supported. Source + ELF binaries.

ups.tar.gz

From the LSM:
Program to interact with battery backups (Powerbox UPS).

usvd−2.0.0.tgz

From the LSM:
usvd is a daemon that monitors the state of an uninterrupted power supply and reacts upon state changes (line fail, line back, battery low situations). You can write your own scripts that are called in these cases. It does *not* require SYSVINIT.

Note that I've only glanced at these packages. I haven't used them. We were just about to start using bestups−0.9.tar.gz in conjunction with powerd−2.0.tar.gz, but we never quite got around to it.

5.Do it yourself guide

This discussion is specifically tailored for dumb UPS control. However, most of the process is about the same for dumb UPSs and smart UPSs. The biggest difference is in the details of how the UPS monitoring daemon (typically `powerd`) communicates with the UPS.

Before doing anything, I suggest the following algorithm:

- Skim this document.
- Download and investigate all packages which seem specifically tailored to your UPS.
- Download and investigate the more generic packages. Note that some of the more generic packages are actually more powerful, better documented, and easier to use than their more specific counterparts.
- If you still can't get things working, or if points are still unclear, read this document more carefully, and hack away...

5.1 What you need to do (summary)

- Plug the computer into the UPS.
- Connect the computer's serial port to the UPS with a special cable.
- Run `powerd` (or some sort of equivalent) on the computer.
- Setup your `init` to do something reasonable on powerfail and powerok events (like start a `shutdown` and kill any currently running `shutdowns`, respectively, for example).

5.2 How it's supposed to work

UPS's job

When the power goes out, the UPS continues to power the computer and signals that the power went out by throwing a relay or turning on an opticoupler on it's control port.

Cable's job

The cable is designed so that when the UPS throws said relay, this causes a particular serial port control line (typically `DCD`) to go high.

Powerd's job

The `powerd` daemon monitors the serial port. Keeps raised/lowered whatever serial port control lines the UPS needs to have raised/lowered (typically, `DTR` must be kept high and whatever line shuts off the UPS must be kept low). When `powerd` sees the UPS control line go high, it writes `FAIL` to `/etc/powerstatus` and sends the `init` process a `SIGPWR` signal. (Older versions of `powerd` and `initd` wrote to `/etc/powerfail`.) When the control line goes low again, it writes `OK` to `/etc/powerstatus` and sends `init` a `SIGPWR` signal.

Init's job (aside from everything else it does)

When it receives a `SIGPWR`, it looks at `/etc/powerstatus`. If it contains `FAIL` it runs the `powerfail` entry from `/etc/inittab`. If it contains `OK` it runs the `powerokwait` entry from `inittab`.

5.3 How to set things up

The following presupposes that you have a cable that works properly with `powerd`. If you're not sure that your cable works (or how it works), see section Reverse–engineering cables and hacking powerd.c for information on dealing with poorly described cables and reconfiguring `powerd.c`. Sections Serial port pin assignments and Ioctl to RS232 correspondence will also be useful.

If you need to make a cable, see section How to make a cable for the overall details, and the subsection of section Info on selected UPSs that refers to your UPS. The latter might also include information on manufacturer supplied cables. You may want to at least skim all of section Info on selected UPSs because each section has a few additional generally helpful details.

- Edit `/etc/inittab`. Put in something like this:

  ```
  # What to do when power fails (Halt system & drain battery :):
  pf::powerfail:/etc/powerfailscript +5

  # If power is back before shutdown, cancel the running shutdown.
  pg:0123456:powerokwait:/etc/powerokscript
  ```

- Write scripts `/etc/powerfailscript` and `/etc/powerokscript` to shutdown in 5 minutes (or whatever's appropriate) and kill any existing `shutdown`, respectively. Depending on the version of `shutdown` that you're using, this will be either so trivial that you'll dispense with the scripts, or be a 1 line `bash` script, something along the lines of:

  ```
  kill `ps -aux | grep "shutdown" | grep -v grep | awk '{print $2}'`
  ```

  and you'll keep the scripts. (In case it doesn't come out right, the first single quote on the above line is a backquote, the second and third are single quotes, and the last is also a backquote.)
- Tell `init` to re–process the `inittab` file with the command:

  ```
  telinit q
  ```

- Edit rc.local so that `powerd` gets run upon startup. The syntax is:

  ```
  powerd <line>
  ```

  Replace `<line>` with the serial port that the UPS is connected, such as `/dev/cua1`.
- Connect computer's serial port to UPS's serial port. DO NOT PLUG THE COMPUTER INTO UPS YET.
- Plug a light into the UPS.
- Turn on the UPS and the light.
- Run `powerd`.
- Test the setup:
  - ♦ Yank the UPS's plug.
    - ◊ Check that the light stays on.
    - ◊ Check that `/etc/powerfailscript` runs.
    - ◊ Check that `shutdown` is running.
  - ♦ Plug the UPS back in.

    - ◊ Check that the light stays on.
    - ◊ Check that `/etc/powerokscript` runs.
    - ◊ Check that `/etc/powerfailscript` is not running.
    - ◊ Check that `shutdown` is no longer running.
  - ♦ Yank the UPS's plug again. Leave it out and make sure that the computer shuts down properly in the proper amount of time.
  - ♦ **The Dangerous Part.** After everything seems to be proper, power down the computer and plug it into the UPS. Run a script that sync's the hard disk every second or so. Simultaneously run a second script that keeps doing a find over your entire hard disk. The first is to make this a little safer and the second is to help draw lots of power. Now, pull the plug on the UPS, check again that `shutdown` is running and wait. Make sure that the computer shuts down cleanly before the battery on the UPS gives out. This is dangerous because if the power goes out before the computer shuts down, you can end up with a corrupt file system, and maybe even lose all your files. You'll probably want to do a full backup before this test, and set the shutdown time extremely short to begin with.

Congratulations! You now have a Linux computer that's protected by a UPS and will shutdown cleanly when the power goes out!

5.4 User Enhancements

- Hack `powerd.c` to monitor the line indicating that the batteries are low. When the batteries get low, do an **immediate** shutdown.
- Modify the shutdown procedure so that if it's shutting down in a `powerfail` situation, then it turns off the UPS after doing everything necessary.

6.Hardware notes

6.1 How to make a cable

This section is just from messages I've seen on the net. I haven't done it so I can't write from experience. If anyone has, please write this section for me :). See also the message about the GPS1000 contained in section GPS1000 from ACCODATA, not to mention all the UPS specific data in section Info on selected UPSs.

```
>From miquels@caution.cistron.nl.mugnet.org Wed Jul 21 14:26:33 1993
Newsgroups: comp.os.linux
Subject: Re: UPS interface for Linux?
From: miquels@caution.cistron.nl.mugnet.org (Miquel van Smoorenburg)
Date: Sat, 17 Jul 93 18:03:37
Distribution: world
```

```
Organization: Cistron Electronics.

In article <1993Jul15.184450.5193@excaliber.uucp>
joel@rac1.wam.umd.edu (Joel M. Hoffman) writes:
>I'm in the process of buying a UPS (Uninteruptable Power Supply), and
>notice that some of them have interfaces for LAN's to signal the LAN
>when the power fails.
>
>Is there such an interface for Linux?
>
>Thanks.
>
>-Joel
>(joel@wam.umd.edu)
>

When I worked on the last versioon of SysVinit (Now version 2.4),
I temporarily had a UPS on my computer, so I added support for it.
You might have seen that in the latest <signal.h> header files there
is a #define SIGPWR 30 now :-). Anyway, I did not have such a special
interface but the output of most UPS's is just a relais that makes or breaks
on power interrupt. I thought up a simple way to connect this to the
DCD line of the serial port. In the SysVinit package there is a daemon
called 'powerd' that keeps an eye on that serial line and sends SIGPWR
to init when the status changes, so that init can do something (such as
bringing the system down within 5 minutes). How to connect the UPS to
the serial line is described in the source "powerd.c", but I will
draw it here for explanation:

                     +------------------------o  DTR
                     |
                   +---+
                   |   | resistor
                   |   | 10 kilo-Ohm
                   |   |
                   +---+                             To serial port.
                     |
       +-----o-------+------------------------o  DCD
       |             |
       o  UPS        |
     \    relais     |
      \              |
       |             |
       +-----o-------+------------------------o  GND

Nice drawing eh?

Hope this helps.
SysVinit can be found on sunsite (and tsx-11 probably) as
SysVinit2.4.tar.z

Mike.

--

Miquel van Smoorenburg, <miquels@cistron.nl.mugnet.org>
Ibmio.com: cannot open CONFIG.SYS: file handle broke off.


>From danny@caution.cistron.nl.mugnet.org Wed Jul 21 14:27:04 1993
Newsgroups: comp.os.linux
Subject: Re: UPS interface for Linux?
```

```
From: danny@caution.cistron.nl.mugnet.org (Danny ter Haar)
Date: Mon, 19 Jul 93 11:02:14
Distribution: world
Organization: Cistron Electronics.

In article <9307174330@caution.cistron.nl.mugnet.org>
miquels@caution.cistron.nl.mugnet.org (Miquel van Smoorenburg) writes:
>How to connect the UPS to the serial line is described in the source
>"powerd.c", but I will draw it here for explanation:

The drawing wasn't really clear, please use this one in stead !
>
>                     +------------------------o  DTR
>                     |
>                   +---+
>                   |   | resistor
>                   |   | 10 kilo-Ohm
>                   |   |
>                   +---+                                To serial port.
>                     |
>       +-----o-------+------------------------o  DCD
>       |
>       o  UPS
>     \    relais
>      \
>       |
>       +-----o--------------------------------o  GND
>

The DTR is kept high, when the UPS's power input is gone it
will close the relais . The computer is monitoring
the DCD input port to go LOW . When this happens it will start a
shutdown sequence...

_____
Danny

--
<=====================================================================>
Danny ter Haar  <dannyth@hacktic.nl> or <danny@cistron.nl.mugnet.org>
Robins law #103: 'a couple of lightyears can't part good friends'
```

6.2 Reverse−engineering cables and hacking powerd.c

Try to get documentation for the cables that your UPS seller supplies. In particular find out:

- What lines need to be kept high.
- What line(s) turn off the UPS.
- What lines the UPS toggles to indicate that:
    - Power is out.
    - Battery is low.

You then need to either hack `powerd.c` appropriately, or use one of the above configurable packages (see the packages `genpower-1.0.1.tgz`, `powerd-2.0.tar.gz`, or `upsd-1.0.tgz` described in section Software). If you use one of the packages, follow the instructions there. If you want to hack `powerd.c`, keep reading.

If you have trouble getting the above information, or just want to check it (a *good* idea) the following program might help. It's a hacked version of `powerd.c`. It allows you to set the necessary port flags from the command line and then monitors the port, displaying the control lines every second. I used it as ``upscheck /dev/cua1 2'' (for example) to set the 2nd bit (`DTR`) and to clear the other bits. The number base 2 indicates which bits to set, so for example to set bits 1, 2 and 3, (and clear the others) use 7. See the code for details.

Here's the (untested) upscheck.c program. It's untested because I edited the version I originally used to make it clearer, and can't test the new version at the moment.

```
/*
 * upscheck     Check how UPS & computer communicate.
 *
 * Usage:       upscheck <device> <bits to set>
 *              For example, upscheck /dev/cua4 4 to set bit 3 &
 *              monitor /dev/cua4.
 *
 * Author:      Harvey J. Stein <hjstein@math.huji.ac.il>
 *              (but really just a minor modification of Miquel van
 *              Smoorenburg's <miquels@drinkel.nl.mugnet.org> powerd.c
 *
 * Version:     1.0 19940802
 *
 */
#include <sys/types.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>

/* Main program. */
int main(int argc, char **argv)
{
  int fd;

/*  These TIOCM_* parameters are defined in <linux/termios.h>, which  */
/*  is indirectly included here.                                       */
  int dtr_bit = TIOCM_DTR;
  int rts_bit = TIOCM_RTS;
  int set_bits;
  int flags;
  int status, oldstat = -1;
  int count = 0;
  int pc;

  if (argc < 2) {
        fprintf(stderr, "Usage: upscheck <device> <bits-to-set>\n");
        exit(1);
  }
```

```
  /* Open monitor device. */
  if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
    fprintf(stderr, "upscheck: %s: %s\n", argv[1], sys_errlist[errno]);
    exit(1);}

  /* Get the bits to set from the command line. */
  sscanf(argv[2], "%d", &set_bits);

  while (1) {
    /* Set the command line specified bits (& only the command line */
    /* specified bits).                                             */
    ioctl(fd, TIOCMSET, &set_bits);
    fprintf(stderr, "Setting %o.\n", set_bits);

    sleep(1);

    /* Get the current line bits */
    ioctl(fd, TIOCMGET, &flags);
    fprintf(stderr, "Flags are %o.\n", flags);

/*  Fiddle here by changing TIOCM_CTS to some other TIOCM until    */
/*  this program detects that the power goes out when you yank     */
/*  the plug on the UPS.  Then you'll know how to modify powerd.c. */
    if (flags & TIOCM_CTS)
      {
        pc = 0 ;
        fprintf(stderr, "power is up.\n");
      }
    else
      {
        pc = pc + 1 ;
        fprintf(stderr, "power is down.\n");
      }
    }

  close(fd);
}
```

6.3 Serial port pin assignments

The previous section presupposes knowledge of the correspondence between terminal signals and serial port pins. Here's a reference for that correspondence, taken from David Tal's ``Frequently Used Cables and Connectors'' document. I'm including a diagram illustrating the connectors, and a table listing the correspondence between pin numbers and terminal line signals.

If you need a general reference for cable wiring, connectors, etc, then David Tal's would be a good one, but I can't seem to locate this document on the net any more. But I've found a good replacement. It's The Hardware Book.

Other useful sites:

- Yost Serial Device Wiring Standard which contains interesting information on how to use RJ−45 jacks and eight wire cables for all serial port connections.
- Stokely Consulting for general Unix info, and in particular their Unix Serial Port Resources.

- Unix Workstation System Administration Education Certification which contains RS−232: Connectors and Cabling

Incidentally, it seems that the Linuxdoc−sgml package still doesn't format tables very well in the `html` output. If you want to be able to read the following table, you're probably going to have to look at either the `DVI` version or the plain text version of this document.

DB−25	DB−9	Name	EIA	CCITT	DTE−DCE	Description
Pin #	Pin #					
1		FG	AA	101	−−−	Frame Ground/Chassis GND
2	3	TD	BA	103	−−−>	Transmitted Data, TxD
3	2	RD	BB	104	<−−−	Received Data, RxD
4	7	RTS	CA	105	−−−>	Request To Send
5	8	CTS	CB	106	<−−−	Clear To Send
6	6	DSR	CC	107	<−−−	Data Set Ready
7	5	SG	AB	102	−−−−	Signal Ground, GND
8	1	DCD	CF	109	<−−−	Data Carrier Detect
9		−−	−−	−	−	Positive DC test voltage
10		−−	−−	−	−	Negative DC test voltage
11		QM	−−	−	<−−−	Equalizer mode
12		SDCD	SCF	122	<−−−	Secondary Data Carrier Detect
13		SCTS	SCB	121	<−−−	Secondary Clear To Send
14		STD	SBA	118	−−−>	Secondary Transmitted Data
15		TC	DB	114	<−−−	Transmitter (signal) Clock
16		SRD	SBB	119	<−−−	Secondary

						Receiver Clock
17		RC	DD	115	−−−>	Receiver (signal) Clock
18		DCR	−−	−	<−−−	Divided Clock Receiver
19		SRTS	SCA	120	−−−>	Secondary Request To Send
20	4	DTR	CD	108.2	−−−>	Data Terminal Ready
21		SQ	CG	110	<−−−	Signal Quality Detect
22	9	RI	CE	125	<−−−	Ring Indicator
23		−−	CH	111	−−−>	Data rate selector
24		−−	CI	112	<−−−	Data rate selector
25		TC	DA	113	<−−−	Transmitted Clock
Pin Assignment for the Serial Port (RS−232C), 25−pin and 9−pin						

```
    1                         13         1         5
  _______________________________     _______________
  \  . . . . . . . . . . . . .  /     \  . . . . .  /    RS232-connectors
   \  . . . . . . . . . . . .  /       \  . . . .  /     seen from outside
    ---------------------------         -----------      of computer.
    14                      25          6       9

DTE : Data Terminal Equipment (i.e. computer)
DCE : Data Communications Equipment (i.e. modem)
RxD : Data received; 1 is transmitted "low", 0 as "high"
TxD : Data sent; 1 is transmitted "low", 0 as "high"
DTR : DTE announces that it is powered up and ready to communicate
DSR : DCE announces that it is ready to communicate; low=modem hangup
RTS : DTE asks DCE for permission to send data
CTS : DCE agrees on RTS
RI  : DCE signals the DTE that an establishment of a connection is attempted
DCD : DCE announces that a connection is established
```

6.4 Ioctl to RS232 correspondence

Since you also might need to modify `powerd.c` to raise and lower the correct lines, you might also need the numeric values of different terminal signals. The can be found in `/usr/include/linux/termios.h`, but are reproduced here for reference. Since they could change, you're best off confirming these values against said file.

```
/* modem lines */
#define TIOCM_LE        0x001
#define TIOCM_DTR       0x002
#define TIOCM_RTS       0x004
#define TIOCM_ST        0x008
#define TIOCM_SR        0x010
#define TIOCM_CTS       0x020
#define TIOCM_CAR       0x040
#define TIOCM_RNG       0x080
#define TIOCM_DSR       0x100
#define TIOCM_CD        TIOCM_CAR
#define TIOCM_RI        TIOCM_RNG
```

Note that the 3rd column is in Hex.

7.What to do when you're really stuck

Here's a novel solution to UPS control for when the UPS and the computer just aren't on speaking terms. I must say that every time I read this, I'm struck by how clever a solution it is.

```
From: " Raymond A. Ingles" <inglesra@frc.com>
To: hjstein@math.huji.ac.il
Subject: UPS HOWTO tip
Date: Mon, 24 Feb 1997 11:48:32 -0500 (EST)


 I don't know if others would find this useful, but I thought I might
pass this along for possible inclusion in the HOWTO. Thanks for
maintaining a HOWTO that I found very useful!

-----------------

 My fiancee bought me a UPS as a present, a Tripp-Lite 400, I believe. It
was very welcome and seems to operate as expected, but unfortunately
doesn't have a serial interface to let the computer know the line power
has failed. It's apparently intended for home or office use where the
computer will not be left unattended.

 This, of course, was unacceptable and I began working on a line monitor,
planning on opening up the case and figuring out how to add the hardware
```

```
that the manufacturer had left out. Then I realized that there was a
quicker and simpler and cheaper (if somewhat less functional) way.

 I had an old 2400 baud modem that I wasn't using, and hooked it up to an
unused serial port on my computer. I then plugged the modem into a surge
supressor plugged into the wall power. I set up powerd with the options
as follows:

-----
serialline    /dev/ttyS1
monitor     DCD
failwhen    low
-----

 Now, when the wall power fails (or, since that hasn't happened lately,
when I pull the surge supressor from the wall to test this setup) the modem
fails but the UPS starts supplying power to the computer. When powerd
notices the modem has dropped DCD, it triggers the powerfail sequence.

 Obviously, this has some limitations. You can't tell from the modem when
the battery is low and so on. You can only tell that the wall power has
failed. Still, it's certainly cheap and I hate to see functioning
computer equipment lie unused. These days you should be able to get a
2400 baud modem for very nearly free.

 I'd still suggest getting a real UPS with full communication capability.
But if you're stuck with a less-functional one, this may at least make it
useful.

 Sincerely,

 Ray Ingles               (810) 377-7735                inglesra@frc.com

 "Anybody who has ever seen a photograph showing the kind of damage that
a trout traveling that fast can inflict on the human skull knows that
such photographs are very valuable. I paid $20 for mine." - Dave Barry
```

8.Info on selected UPSs

This section contains UPS specific information. What I'd like is to have the UPS control port information (what each pin does and needs to have done), information on the manufacturer supplied cable (what it connects where), and a hacked version of powerd.c which works with the UPS. What I currently have is fairly complete descriptions of setting up each UPS. I'd try to distill out the relevant information, but since I can't test each UPS, it's hard to decide exactly what's relevant. Furthermore, each UPS seems to have some additional quirks that are nicely described by the authors of each section. So for now I'm leaving everything in. Makes for a hefty Howto.

Please send me your experiences for inclusion here.

8.1 General Experiences.

I've been saving peoples comments, but haven't gotten permission yet to include them here. Here's a general summary of what I've heard from people.

APC: Won't release info on their smart mode without your signature on a non−disclosure agreement. Thus, people are forced to run their smart UPSs in the dumb mode as outlined above. Various people have had varying amounts of success reverse engineering

Best: Helpful and friendly. Supply source code and documentation both for dumb modes and smart modes.

TrippLite: One person reported that TrippLite won't release info either.

Upsonic: One person reported that Upsonic has discussed technical details over the phone, answered questions via fax and are generally helpful.

8.2 Advice 1200 A

UPS from Advice Electronics, Tel Aviv Israel (they stick their own name on the things).

I don't recommend them. Our experiences with them have been very bad. We've twice had a 17" monitor fry when the power failed. We've had computers spontaneously reboot when the power failed.

None the less, for completeness, here's he UPS Control Port's pin specifications.

- 2 − Power Fail.
- 5 − Battery Low.
- 6 − Shut Down UPS.
- 4 − Common ground for pin 2, 5, 6.

They also gave me the following picture which didn't help me, but may help you if you want to build a cable yourself:

```
2 ----------+
            |
            \
             \|
              |--------------
             /|
           \/      <--- The "\/" here indicates the type of
           |            this transister.  I forget what
           |            denotes what, but this one points
        +-----+         away from the center line.
       /  /  /
```

```
5 ----------+
            |
            \
             \|
              |--------------
             /|
           \/
           |
           |
        +-----+
       /   /   /


              +-------------
              |
              /
     10K    |/
6 --\/\/\/--|
            |\
              \/
              |
              |
           +-----+
          /   /   /


4 ----------+
            |
            |
         +-----+
        /   /   /
```

Cable supplied

They first gave me a cable that was part of a DOS UPS control package called RUPS. I used this for testing. When I was satisfied, they gave me a cable they use for Netware servers connected to UPSs. It functioned identically. Here are the details:

- DTR − Powers cable (make powerd.c keep it high).
- CTS − Power out (stays high and goes low when power goes out).
- DSR − Battery low (stays high. Goes low when battery does).
- RTS − Turns off UPS (keep it low. Set it high to turn off UPS).

(The powerd.c that comes with SysVinit set or left RTS high, causing the UPS to shut off immediately when powerd was started up!)

8.3 Trust Energy Protector 400/650

This section is good for more than just the Trust Energy Protector. It illustrates how to work with the new features of `init`.

How to use a Trust Energy Protector 400/650 under Linux

by Ciro Cattuto

Version 1.0 − 31 March 1997

The computer to UPS connection

The Trust Energy Protector 400/650 is equipped with a remote signal port. Using a properly designed cable, it is possible to connect the UPS port to the serial port of a computer, thus making it aware of power failure events.

The UPS signal port

These are the pin assignments for the DB−9 signal port of the Trust Energy Protector 400/650, as described in the user's manual:

pin 2

The relay will close when input power fails.

pin 4

Common for pins 2 and 5.

pin 5

The relay will close when the battery inside the Trust Energy Protector 400/650 has less than 1.5 minutes of backup time left.

pin 6

The user may send a high level signal (+5V − +12V) for over 1ms to turn off the Trust Energy Protector 400/650. However this option can only be activated when the input power fails.

pin 7

Common for pin 6.

The Cable

This is the cable I used to connect the UPS to the serial port of my computer:

```
computer side (DB-25)              UPS side (DB-9)
===================================================

 6 DSR --+             [R] = 10 kilo-Ohm resistor
         |
20 DTR --+----+
         |    |
        [R]  [R]                         +--- 7
         |    |                          |
 8 DCD --+----|-----------        ---------|--- 2
              |                          |
 7 GND -------|-----------        ---------+--- 4
              |              ....
 5 CTS -------+-----------        ------------- 5

 2 TX  -------------------        ------------- 6

===================================================
```

In the case of a DB−9 serial port, the pins 6,20,8,7,5,2 are mapped to pins 6,4,1,5,8,3.

How the cable works

The computer raises `DTR` and checks whether `DSR` is high, to ensure that the cable is connected to the computer. While the power is good, `DCD` and `CTS` are both high (because of the pull−up resistors).

When the power fails, the relay between pins 2 and 4 of the UPS port closes, and `DCD` becomes low, signalling the failure condition.

Similarly, when the UPS batteries are getting low, the relay between pins 5 and 4 closes, thus lowering `CTS`.

During a power failure the computer is able to turn off the UPS by raising `TX` for over 1ms. This can be easily accomplished sending a `0xFF` byte to the serial port, at a low baud rate.

The powerd daemon

To make use of the information available at the serial port we need to run a program which monitors the port, decodes the signals and sends the appropriate messages to the operating system, i.e. to the `init` process. The `init` process can execute scripts and programs designed to handle (gracefully!) the power failure event.

Compiling powerd

In Appendix A you'll find the source code of `powerd`, the daemon I use to monitor the Trust Energy Protector 400/650. To compile it you will need the source code of the sysvinit package (I used the code from sysvinit−2.60). Just overwrite the original `powerd.c` and compile it.

How powerd works

As soon as `powerd` starts it opens the serial device connected to the UPS and forces `DTR` high. It then forks a daemon and exits, leaving the daemon running. The `powerd` daemon can be in one of three states:

State 0 − POWER IS GOOD

In this state `powerd` reads the serial port every `T0_SLEEP` seconds (see the `#define` lines at the beginning of the code). If `DCD` drops, `powerd` switches to state 1. If `CTS` drops `powerd` switches to state 2 (this shouldn't happen without `DCD` dropping before, but I decided to stay on the safe side).

State 1 − POWER FAILURE

A power failure was detected. `DCD` is low and `powerd` reads the UPS port every `T1_SLEEP` seconds. If `DCD` becomes high, it switches to state 0. If `CTS` drops, it switches to state 2.

State 2 − POWER CRITICAL

UPS batteries are low. The `powerd` daemon will remain in this state.

Each time `powerd` changes state, it notifies the init process, so that the appropriate action can be taken. These events are logged using the system logging facility.

If `DSR` is low there must be something wrong with the cable. Powerd keeps monitoring the `DSR` line, and every two minutes sends a warning message to the system logging facility.

Running powerd

The `powerd` daemon should be launched from the system initialization scripts, during system startup. I added the following lines to my `/etc/rc.d/rc.local` script:

```
# Add support for the UPS
echo "Starting powerd daemon..."
rm −f /etc/turnUPSoff
stty −crtscts speed 75 < /dev/cua3 > /dev/null
```

```
if [ -x /usr/sbin/powerd ]; then
        /usr/sbin/powerd /dev/cua3
fi
```

First we remove (if present) the file `/etc/turnUPSoff`. This file is used by the system shutdown script (`/etc/rc.d/rc.0`, in my case) to decide whether we want to turn the UPS off. See later in this document for more information.

Then we disable hardware flow control on the serial device connected to the UPS, and set its baud rate to 75. Now we're confident that the TX signal will stay high for a time long enough to turn the UPS off, if we send a character to the serial port (again, see later).

Finally we launch the `powerd` daemon, specifying the serial port to monitor. Notice that we're not going to read characters from the serial device, so don't worry if you have interrupt conflicts − they'll do no harm.

The inittab file and the shutdown scripts

The `powerd` process is now running, and it will send signals to `init` whenever a power failure occurs. Now we have to configure the system so that it can react in a useful way when those signals are received.

Modifying inittab

Add the following lines near the beginning of your /etc/inittab file:

```
# What to do when power fails (delayed shutdown).
pf::powerfail:/etc/powerfail_script

# If power is back before shutdown, cancel the running shutdown.
pg::powerokwait:/etc/powerokay_script

# If UPS batteries are getting low, do an immediate shutdown.
pc::powerfailnow:/etc/powerfailnow_script
```

The scripts

The scripts `powerfail_script`, `powerokay_script` and `powerfailnow_script` are executed when `init` receives the corresponding signal. They have the responsibility of shutting down the system in a clean way or cancelling a running shutdown in case power comes back. These are the scripts I'm currently using:

`/etc/powerfail_script`:

```
#!/bin/sh
/bin/sync
/usr/bin/sleep 10m
kill -9 `ps auxw | grep "shutdown" | grep -v grep | awk '{print $2}'`
> /etc/turnUPSoff
/sbin/shutdown -t30 -h +3 "POWER FAILURE"
```

My Trust Energy Protector 400 powers only the computer, so I have quite a long backup time. Since power failures only last for some minutes in my zone, the system responds to a blackout in the following way: it waits for 10 minutes (usually the power comes back before) and then halts the system, allowing the users to close their applications and leave the machine. Before issuing the shutdown command, I make sure that there are no running shutdowns. I also create the file `/etc/turnUPSoff`, so that the system will turn off the UPS.

`/etc/powerokay_script`:

```
#!/bin/sh
kill `ps auxw | grep "powerfail_script" | grep -v grep | awk '{print $2}'`
kill -9 `ps auxw | grep "shutdown" | grep -v grep | awk '{print $2}'`
rm -f /etc/turnUPSoff
```

If power comes back, we kills the running powerfail_script and any running shutdown. We also remove `/etc/turnUPSoff`.

`/etc/powerfailnow_script`:

```
#!/bin/sh
kill -9 `ps auxw | grep "shutdown" | grep -v grep | awk '{print $2}'`
> /etc/turnUPSoff
/sbin/shutdown -h now "UPS batteries low. IMMEDIATE SHUTDOWN."
```

If batteries are getting low, we make sure that there are no running shutdowns, create the `/etc/turnUPSoff` file and then shutdown the system immediately.

The system shutdown script

When system shutdown is complete, we can turn off the UPS raising the `TX` signal of the serial port for over 1ms. The serial device is already properly configured (see the `stty` command in the `rc.local` script). If the file `/etc/turnUPSoff` is present, we send the byte `0xff` (all '1' bits) to the serial port.

To do this, add the following lines near the bottom of your system shutdown script (`/etc/rc.d/rc.0`, in my case). The proper place depends on the way your system is configured, but it should be okay to insert the lines before the echo command which prints the "System is halted" message.

```
# Is this a powerfail situation?
if [ -f /etc/turnUPSoff ]; then
  echo "Turning off UPS. Bye."
  sleep 5
  echo -e "\377" > /dev/cua3
  exit 1
fi
```

General remarks

This document contains things I learned while trying to configure *my* Linux system to use the Trust Energy Protector 400. Some informations (the path of the system inizialization scripts, for example) may be specific to my system, and you probably will need some customization. However, I hope this document will be a useful trace for those trying to use a Trust Energy Protector 400/650 under Linux. If you experience difficulties, look for general information in the rest of this UPS−Howto. Good luck!

Feedback

I would greatly appreciate receiving feedback about this document, so that I can polish it and correct possible mistakes (I know the English is not very good, but I'm Italian after all!). Direct any comments/suggestions/critics to the following e−mail address:

ciro@stud.unipg.it

If you have problems using Trust Energy Protector 400/650 under Linux, feel free to contact me. I'll try to help you.

Legal Issues

I have no relation at all with Trust Networking Products.

The information contained in this document comes "as is". Use it at your own risk. I can't be held responsible for any damage or loss of data resulting from the use of the code and information given here.

Ciro Cattuto

−−−−−

Appendix A – Source code for the powerd daemon

`powerd.c`:

```
/*
 * powerd       Catch power failure signals from
 *              a Trust Energy Protector 400/650
 *              and notify init
 *
 * Usage:       powerd /dev/cua3 (or any other serial device)
 *
 * Author:      Ciro Cattuto <ciro@stud.unipg.it>
 *
 * Version 1.0 - 31 March 1997
 *
 * This code is heavily based on the original powerd.c code
 * by Miquel van Smoorenburg <miquels@drinkel.ow.org>.
 *
 * This program is free software; you can redistribute it and/or
 * modify it under the terms of the GNU General Public License
 * as published by the Free Software Foundation; either version
 * 2 of the License, or (at your option) any later version.
 *
 */

/* state 0 - power is good */
#define T0_SLEEP        10      /* interval between port reads, in seconds */
#define T0_DCD          3       /* number of seconds DCD has to be high
                                   to cause an action                      */
#define T0_CTS          3       /* number of seconds CTS has to be high
                                   to cause an action                      */
/* state 1 - power is failing */
#define T1_SLEEP        2       /* interval between ports reads            */
#define T1_DCD          3       /* same as T0_DCD                          */
#define T1_CTS          3       /* same as T0_CTS                          */

#define DSR_SLEEP       2
#define DSR_TRIES       60

/* Use the new way of communicating with init. */
#define NEWINIT

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>
#include <syslog.h>
#include <string.h>
#include "paths.h"
#ifdef NEWINIT
#include "initreq.h"
#endif

#ifndef SIGPWR
```

```
#  define SIGPWR SIGUSR1
#endif

#ifdef NEWINIT
void alrm_handler()
{
}
#endif

/* Tell init that the power has gone (1), is back (0),
   or the UPS batteries are low (2). */
void powerfail(int event)
{
  int fd;
#ifdef NEWINIT
  struct init_request req;

  /* Fill out the request struct. */
  memset(&req, 0, sizeof(req));
  req.magic = INIT_MAGIC;
  switch (event)
        {
        case 0:
                req.cmd = INIT_CMD_POWEROK;
                break;
        case 1:
                req.cmd = INIT_CMD_POWERFAIL;
                break;
        case 2:
        default:
                req.cmd = INIT_CMD_POWERFAILNOW;
        }

  /* Open the fifo (with timeout) */
  signal(SIGALRM, alrm_handler);
  alarm(3);
  if ((fd = open(INIT_FIFO, O_WRONLY)) >= 0
                && write(fd, &req, sizeof(req)) == sizeof(req)) {
        close(fd);
        return;
  }
  /* Fall through to the old method.. */
#endif

  /* Create an info file for init. */
  unlink(PWRSTAT);
  if ((fd = open(PWRSTAT, O_CREAT|O_WRONLY, 0644)) >= 0) {
  switch (event)
        {
        case 0:
                write(fd, "OK\n", 3);
                break;

        case 1:
                write(fd, "FAIL\n", 5);
                break;

        case 2:
        default:
                write(fd, "LOW\n", 4);
                break;
        }
```

```
  close(fd);
  }

  kill(1, SIGPWR);
}

/* Main program. */
int main(int argc, char *argv[])
{
  int fd;
  int dtr_bit = TIOCM_DTR;
  int flags;
  int DCD, CTS;
  int status = -1;
  int DCD_count = 0, CTS_count = 0;
  int tries;

  if (argc < 2) {
        fprintf(stderr, "Usage: powerd <device>\n");
        exit(1);
  }

  /* Start syslog. */
  openlog("powerd", LOG_CONS|LOG_PERROR, LOG_DAEMON);

  /* Open monitor device. */
  if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
        syslog(LOG_ERR, "%s: %s", argv[1], sys_errlist[errno]);
        closelog();
        exit(1);
  }

  /* Line is opened, so DTR is high. Force it anyway to be sure. */
  ioctl(fd, TIOCMBIS, &dtr_bit);

  /* Daemonize. */
  switch(fork()) {
        case 0: /* Child */
                closelog();
                setsid();
                break;
        case -1: /* Error */
                syslog(LOG_ERR, "can't fork.");
                closelog();
                exit(1);
        default: /* Parent */
                closelog();
                exit(0);
  }

  /* Restart syslog. */
  openlog("powerd", LOG_CONS, LOG_DAEMON);

  /* Now sample the DCD line. */
  while(1) {
        /* Get the status. */
        ioctl(fd, TIOCMGET, &flags);

        /* Check the connection: DSR should be high. */
        tries = 0;
        while((flags & TIOCM_DSR) == 0) {
                /* Keep on trying, and warn every two minutes. */
```

```
        if ((tries % DSR_TRIES) == 0)
            syslog(LOG_ALERT, "UPS connection error");
        sleep(DSR_SLEEP);
        tries++;
        ioctl(fd, TIOCMGET, &flags);
}
if (tries > 0)
        syslog(LOG_ALERT, "UPS connection OK");

/* Calculate present status. */
DCD = flags & TIOCM_CAR;
CTS = flags & TIOCM_CTS;

if (status == -1)
        {
        status = (DCD != 0) ? 0 : 1;
        if (DCD == 0)
                {
                syslog(LOG_ALERT, "Power Failure. UPS active.");
                powerfail(1);
                }
        }

switch (status)
        {
        case 0:
                if ((DCD != 0) && (CTS != 0))
                        {
                        DCD_count = 0;
                        CTS_count = 0;
                        sleep(T0_SLEEP);
                        continue;
                        }
                if (DCD == 0)
                        DCD_count++;
                if (CTS == 0)
                        CTS_count++;
                if ((DCD_count < T0_DCD) && (CTS_count < T0_CTS))
                        {
                        sleep(1);
                        continue;
                        }
                if (CTS_count == T0_CTS)
                        {
                        status = 2;
                        syslog(LOG_ALERT, "UPS batteries low!");
                        break;
                        }
                status = 1;
                DCD_count = 0;
                syslog(LOG_ALERT, "Power Failure. UPS active.");
                break;

        case 1:
                if ((DCD == 0) && (CTS != 0))
                        {
                        DCD_count = 0;
                        CTS_count = 0;
                        sleep(T1_SLEEP);
                        continue;
                        }
                if (DCD != 0)
```

```
                        DCD_count++;
                if (CTS == 0)
                        CTS_count++;
                if ((DCD_count < T1_DCD) && (CTS_count < T1_CTS))
                        {
                        sleep(1);
                        continue;
                        }
                if (CTS_count == T1_CTS)
                        {
                        status = 2;
                        syslog(LOG_ALERT, "UPS batteries low!");
                        break;
                        }
                status = 0;
                DCD_count = 0;
                CTS_count = 0;
                syslog(LOG_ALERT, "Power okay.");
                break;

        case 2:
                sleep(1);
                continue;

        default:
                break;
        }

    powerfail(status);
  }
  /* Never happens */
  return(0);
}
```

8.4 Trust UPS 400−A

I received a submission about the Trust UPS 400−A. I don't know if it's the same as the Trust Energy Protector 400, so I'm including the submission.

```
From: "Marcel Ammerlaan" <marcel@ch.twi.tudelft.nl>
To: hjstein@math.huji.ac.il
Subject: UPS addition
Date: Wed, 16 Jul 1997 01:17:11 +100

Hello Harvey,

I've got an addition to your UPS Howto. I've got a
"Trust UPS 400-A" which isn't listed. This product doesn't seem
to be manufactured anymore by it's producer (www.trust.box.nl).
But that doesn't mean it's not available anymore, I've got mine
really cheap just a month ago. Also this company just relabels
products so maybe there are others that have got the same UPS.

I have included a picture of the UPS in case anybody got such a beast
```

```
under another label.

The cable was easily constructed based on the original powerd cable
and the documentation from trust.
It clearly describes which pins of the D-shell connector of the UPS
carry which signal.

It extends the original design with 2 extra functions:
1) Battery low indication
2) Power down UPS

The cable I created looks like (see the other attachement).

This cable has been tested with powergend by Tom Webster and did work
completely (although your milage may vary).

Type:               "pleur"
Cable Power:        {TIOCM_DTR,0}
Inverter Kill:      {TIOCM_RTS,1}
Inverter Kill Time: 5
Power Check:        {TIOCM_CTS,0}
Battery Check:      {TIOCM_CAR,0}
Cable Check:        {TIOCM_RI,0}

Although (just as the powerd cable) the cable check function isn't
used because the UPS doesn't seem to support it.

Well that's about it I guess. If you need more information about the
UPS the cable or the software feel free to contact me.

And remember, everything described here works for me but I don't
guarantee it will for you.

Marcel Ammerlaan
CEO Pleursoft (explains the cablename doesn't it :-)
The Netherlands

<RSA implemented in 3 lines of perl deleted by the editor ;)>

Marcel Ammerlaan    | <m.j.ammerlaan@twi.tudelft.nl>
Paardenmarkt 78     | Just another nerd on the loose
2611 PD Delft       |
The Netherlands     |
```

8.5 Sustainer S−40a

Information on the Sustainer S−40a.

```
From: fnevgeny@plasma-gate.weizmann.ac.il (Evgeny Stambulchik)
To: hjstein@math.huji.ac.il, hjstein@math.huji.ac.il, hjstein@math.huji.ac.il,
        hjstein@math.huji.ac.il
Subject: UPS-HowTo add-ons
Date: Sun, 10 Sep 1995 13:09:50 +0300 (IST)
```

```
Hi Harvey,

This is an addition to your UPS-HowTo. I'm using Sustainer S-40a UPS for a few
months with unipower package (now it's called genpower) and home-made cable
constructed as follows (I've sent all this stuff to Tom Webster, author of the
package, too, and it should appear in the next version):

         UPS SIDE                                    LINUX SIDE


               2 POWER FAIL                              1(8)
   +-----------o-------------------------+----------------o  DCD
   |                                     |
   o                                     |
    /                                    |
   /                                     |
   |           4 COMMON                  |               5(7)
   +-----------o------+------------------|----------------o  GND
   |                  |                  |
   \                  |                  |
    \                 |                  |
   o                  |                  |
   |           5 BATT | LOW              |               8(5)
   +-----------o------|------------------|--------+-------o  CTS
                      |                  |        |
                      |                +-+-+    +-+-+
                      |                |   |    |   |
                      |    3 resistors |   |    |   |
                      |                |   |    |   |
                      |    by 10 kOhm  |   |    |   |
                      |                +-+-+    +-+-+
                      |                  |        |      4(20)
                      |                  +--------+-------o  DTR
                      |
|              6 SHUT | DOWN              +-------+      7(4)
+-+       +----o------|-------------------+       +-------o  RTS
  \       |           |                   +-------+
   \|    -+-          |
    | <- \ /          |
   /|    -+-          |
  /       |    7      |
  |       +----o------+
--+--
 ---
  -


NOTE!!!: Shutdown pins in the tech info supplied with UPS (4 and 6) are given
incorrectly! The valid ones are 6 and 7, as shown above.
Note2: Pin numbers on the PC side in the brackets are for 25-pin connector,
outside - for 9-pin one.


Here's the unipowerd.h file I used:


/************************************************************************/
/* File Name            : unipowerd.h                                   */
/* Program Name         : unipowerd                 Version: 1.0.0     */
/* Author               : Tom Webster <webster@kaiwan.com>              */
/* Created              : 1994/04/20                                    */
/* Last Modified By     : Tom Webster               Date: 1995/04/09   */
```

```
/* Last Modified By     : Evgeny Stambulchik (for Sustainer UPS)        */
/*                                                                      */
/* Compiler (created)   : GCC 2.5.8                                     */
/* Compiler (env)       : Linux 1.0.9                                   */
/* ANSI C Compatable    : No                                            */
/* POSIX Compatable     : Yes?                                          */
/*                                                                      */
/* Purpose              : Header file for unipowerd.                    */
/*                      : Contains the configuration information for    */
/*                      : unipowerd.  Edit this file as indicated       */
/*                      : below to activate features and to customize   */
/*                      : unipowerd for your UPS.                       */
/*                                                                      */
/* Copyright            : GNU Copyleft                                  */
/************************************************************************/

/* The following are the RS232 control lines      */
/*                                                */
/*                                            D D */
/*                                            T C */
/* Macro           English                    E E */
/* ---------------------------------------------- */
/* TIOCM_DTR       DTR - Data Terminal Ready  --> */
/* TIOCM_RTS       RTS - Ready to send        --> */
/* TIOCM_CTS       CTS - Clear To Send        <-- */
/* TIOCM_CAR       DCD - Data Carrier Detect  <-- */
/* TIOCM_RNG       RI  - Ring Indicator       <-- */
/* TIOCM_DSR       DSR - Data Signal Ready    <-- */

#define HIGH            (1)
#define LOW             0
#define PWRSTAT         "/etc/powerstatus"
#define UPSSTAT         "/etc/upsstatus"

/* CABLEPOWER is the line which provides power to */
/* the cable for normal monitoring activities.    */
#define CABLEPOWER      TIOCM_DTR

#define POWERBIT        TIOCM_CAR
#define POWEROK         HIGH

/* Define CABLECHECK as 1 to check for low battery */
/* Define CABLECHECK as 0 value to skip            */
#define CABLECHECK      0
#define CABLEBIT        TIOCM_RNG
#define CABLEOK         HIGH

/* Define BATTCHECK as 1 to check for low battery  */
/* Define BATTCHECK as 0 value to skip.            */
#define BATTCHECK       1
#define BATTBIT         TIOCM_CTS
#define BATTOK          HIGH

/* Define INVERTERKILL as 1 to hndle killing the inverter */
/* Define INVERTERKILL as 0 value to skip.                */
/* INVERTERBIT is the line which will kill the inverter   */
/*    while the UPS is in powerfail mode.                 */
/* INVERTERTIME is the time in seconds to hold the line   */
/*    defiined by INVERTERBIT high to kill the inverter.  */
#define INVERTERKILL    1
#define INVERTERBIT     TIOCM_RTS
#define INVERTERTIME    5
```

```
/************************************************************************/
/* End of File unipowerd.h                                              */
/************************************************************************/

I'm aware that current name of the package is genpower. I haven't try it yet as
see no reason to switch to the new version meantime; the former seems to work
very stable. Nevertheless, here is the add-on for genpower-1.0.1's genpowerd.h
file (hopefully, I "translated" unipowerd.h correctly):
Add-on for genpower-1.0.1's genpowerd.h file:

/* Evgeny's Sustainer S-40A */
 {"sustainer",   {TIOCM_DTR,0}, {TIOCM_RTS,1},  5, {TIOCM_CAR,0}, {TIOCM_CTS,0},
{0,0}}


Evgeny
```

8.6 Systel

Another Israeli company. I never ended up purchasing a UPS from them, but they were very good about getting me detailed documentation on their communication port. It should be easy enough to control their UPS. Their phone number is 972−8−409−019 (972−8−407−216 for fax).

8.7 Deltec Power, Fiskars Power Systems and Exide.

Fiskars is a Finnish holding company. They used to own Deltec Power. In March of 1996 Fiskars sold Deltec Power to Exide Electronics Group. At that time, Deltec Power was one of the world's largest makers of UPSs.

Under Fiskars, Deltec used to make the PowerServers 10, 20, 30, and 40. The Deltec Power home page mentions other UPSs.

Exide now bundles UPS control software with their UPSs that works under Linux. They also sell the software separately. They say that their software works with other UPSs too.

I'd like to hear from people using their software.

Here's the advertisement they emailed me:

Exide Electronics announces LanSafe III UPS Power Management Software for Linux.

LanSafe III is a UPS Power Management application. It provides automatic orderly shutdown functionality incase of an extended power failure that should outlast the UPS battery run time.

LanSafe III enables broadcast messages and e−mail to be sent according to user defined power condition changes. The shutdown procedure can also be customized.

LanSafe III works together with the vast majority of all Exide Electronics UPS models. It goes even one step further by supporting basic shutdown functionality also with other manufacturers UPSs.

LanSafe III for Linux runs on Intel based Linux systems. Both character based and X11/Motif based user interfaces are provided.

LanSafe III supports all the major OS platforms: Linux, IBM AIX, HP UX, Digital UNIX, SCO UNIX, Solaris, SunOS, AT&T UNIX, all Windows platforms, OS/2, Novell and Macintosh among others.

LanSafe III is bundled with the following Exide Electronics UPSs: OneUPS Plus, NetUPS, PowerWare Prestige, PowerWare Profile, PowerWare Plus 5xx.

It also ships with FPS Power Systems UPSs: PowerRite Plus, PowerRite Max, PowerWorks A30, PowerWorks A40, Series 9000 and Series 10000.

It is also possible to purchase a separate software license to use with a previous UPS model or an other manufactures UPS. Regular licenses are S$149, with site licenses also available.

For details please visit our Web sites at www.exide.com, www.fiskarsUPS.com and www.deltecpower.com.

Incidentally, when I tried to connect to www.fiskarsUPS.com, it prompted me for a username and password.

8.8 Beaver model UB500 UPS

dan@fch.wimsey.bc.ca (Dan Fandrich) writes:

I seem to have gotten my old Beaver model UB500 UPS working with genpower. The interface uses RS−232 compatible voltage levels, so installing it is a snap. There is a DE−9 female connector on the back which plugs directly into a 9−pin PC serial port using a plain 9−pin video monitor extension cable.

The DIP switches allow quite versatile pinouts. To emulate genpower's apc1−nt type of UPS, they must be set as follows:

1	on	(CTS = power fail)
2	off	(CTS = low battery)
3	off	(DSR = power fail)
4	off	(DSR = low battery)
5	off	(CD = power fail)
6	on	(CD = low battery)
7	off	(RI = power fail)
8	off	(RI = low battery)

9	on	(DTR = inverter off)
10	off	(RTS = inverter off)
DIP switch SW601 for Beaver model UB500 UPS.		

The switches form groups of adjacent pairs for each output pin. They are mutually exclusive−−don't try to turn on both switch 5 and 6 simultaneously, for example, or you'll be shorting the low battery and power fail signals.

That's all there is to it. Feel free to add this do your documentation.

8.9 Sendom

Documentation on using the Sendom UPS.

```
From: charli <mefistos@impsat1.com.ar>
To: hjstein@math.huji.ac.il
Subject: ups howto contribution
Date: Wed, 13 Nov 1996 19:07:41 -0200

hjstein@math.huji.ac.il


sir:

i connected a sendom ups with the help of your UPS-howto and man powerd
and discovered something useful. perhaps this thing extends to some
other ups.
im using slackware 3.0 distribution. i has the soft configuration in
/etc/inittab already done. its only needed to add the /rc.local powerd
/cuaX


i used the man powerd diagram:
        9pin    25pin

DTR     4       20      ---------
                         |      >
DSR     6       6       --      < 10k
                                >
DCD     1       8       -------------------------
                                                relais
GND     5       7       -------------------------


the fact is that the sendom ups dont use relais but some electronic
solid state device, and it works one way BUT NO THE OTHER. so if you
make the cable and doesnt work, first try inverting the cable in the
ups "relais"
```

```
i hope this can be useful, if you want to include this somewhere, feel
free to correct my english. please aknowledge this mail even with an
empty
mail so i know it arrived
end
```

8.10 Best

Information on Best UPSs is available on at the Best Power website. Their website includes the `checkups.tar` (section Software) package for communicating with Best UPSs, both in smart mode and in dumb mode, and it includes source code, so you can compile it under Linux.

Best Fortress − Using Best's software

Linux Best Power UPS Mini−HOWTO by Michael Stutz (stutz@dsl.org, and http://dsl.org/m/) v1.0, 14 Aug 97

0.0 Disclaimer

Copyright 1997 by Michael Stutz; this information is free; it may be redistributed and/or modified under the terms of the GNU General Public License, either Version 2 of the License, or (at your preference) any later version, and as long as this sentence remains; this information comes WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE; see the GNU General Public License for more details.

1.0 Introduction

Best Power are the makers of high quality UPS products, with their Fortress line in particular being well−suited for typical Linux users. Although their products are not currently priced as low as some (such as APC), Best Power provide source code for their UPS software and have been very respondent to queries from Linux users. Furthermore, their hardware seems to be highly regarded, making Best Power a winning choice for Linux users.

This document describes the installation of a Best Power Fortress UPS (model used was a 1996−model 660a with accompanying Best Power CD−ROM) to a Linux box.

2.0 Installation

2.1 Hardware

Install the hardware as indicated in the instructions. The Best Power ``Fortress" series comes with an RS−232 cable that should attach to a spare serial port on the back of the computer.

2.2 Software

This is where it differs from the manual, which does not currently have Linux−specific instructions. The accompanying Fortress CD−ROM, however, does come with source code for the UPS software, so getting it up and running on a Linux system is a trivial task.

To do this, follow these steps, and use the manual as a reference to get an overall feel for how the software works. I took the liberty of making a few changes in this HOWTO from the way the Fortress software is set up on other UNIX systems that I feel are better suited for a Linux system. For example, I eliminated the need for an `/etc/best` directory and put the executables in `/usr/local/sbin`, which I feel is a more appropriate place.

- First, create the "upsdown" script that is executed during a power outage. This one will halt the system:

```
cat > /etc/upsdown << EOF
#!/bin/sh
shutdown −h now < /dev/console &
EOF
```

- Now, make directories for the documentation and the source code:

```
mkdir /usr/doc/best
mkdir /usr/local/src/best
```

- Mount the CD−ROM, and untar the unix/checkups.tar file into the /tmp directory or somewhere similar:

```
cd /tmp
tar /cdrom/unix/checkups.tar
```

- Change into the etc/best/advanced directory that should have been extracted from the checkups tarball.
- Copy the documentation and UPS script files to their proper place in the sytem:

```
cp README /usr/doc/best
cp manual.txt /usr/doc/best
cp bestsend /etc
cp source/*.c /usr/local/src/best
```

- Clean up the /tmp mess and compile the software:

```
cd /usr/local/src/best
rm −R /tmp/etc
```

```
gcc -o checkups checkups.c
gcc -o mftalk mftalk.c
mv checkups /usr/local/sbin
mv mftalk /usr/local/sbin
```

- Test the UPS. Replace ttySx with the serial port of your choice. If you have a good connection, you should see a row of characters print across the screen:

```
mftalk /dev/ttySx
```

- Make the checkups program run at startup for testing. This can be done in several different ways (described in the manual). The way I did it is by adding this line to /etc/inittab:

```
ups:234:once:/usr/local/sbin/checkups -c500 /dev/ttyS1
```

- Test it. Do this by taking out power to UPS by pulling out the fuse connected to the UPS, and wait a couple of minutes. It print a warning messages and then halt the system after a few mintues.
- If that works, take out the "−c500" from the line in your inittab (which basically means shut down the system right away instead of when the UPS power runs out), and you're good to go!

3.0 Conclusions

I welcome suggestions for improving this document or the techniques described herein. As of this writing, Best Power seemed interested in including this or other information in their documentation to help Linux users with their product, so this is definitely a company to support. Let them know how you feel at sales@bestpower.com and support@bestpower.com.

Best Fortress LI−950

Some comments on the Best Fortress.

```
From lnz@dandelion.com Wed May 31 19:53:09 1995
Newsgroups: comp.os.linux.hardware
Subject: Re: UPS for use with Linux?
From: Leonard N. Zubkoff <lnz@dandelion.com>
Date: 25 May 1995 16:27:55 -0700
Organization: Dandelion Digital
NNTP-Posting-Host: dandelion.com
NNTP-Posting-User: root
In-reply-to: nautix@community.net's message of 23 May 1995 09:41:40 -0700

In article <3pt384$sic@odin.community.net> nautix@community.net writes:

  Ditto what Craig says.  APC was very uncooperative, but I have only
  good things to say about Best.  I use their Fortress LI 660 model;
  660 VA, lots of status features on the front, etc.  The CheckUPS
  software costs extra and needs some hacking to fit into my
  FSSTND-ish file system (the directories and file names are hard-coded
  to fit into SunOS 4.1.x).  I'd be happy to send you my diffs, if
```

```
  you want them.  (I love it when a vendor ships the source as
  a normal business practice!)

  The CheckUPS software is limited to doing automagic shutdowns, though.
  The UPS can give lots of status information; CheckUPS only asks for
  ``If the power has failed, how much battery time remains?''

  Best follows up on their customer satisfaction cards, too.
  I indicated that I was dissappointed that CheckUPS didn't do more
  status reporting (like input voltage, output voltage, percent load,
  etc.), which is available from the UPS.  I asked for a the
  spec on the interface lingo; they said ``sure'' and had it to me in
  2 days, free.  A full-featured UPS status checker is on my back burner.
  Does anyone see a demand for such a utility?

Let me add yet another recommendation for Best Power.  I just purchased a
Fortress LI-950, though I declined on the CheckUPS software.  Unlike some
other brands, a simple three wire cable is all that's needed to connect the
Fortress to a serial port -- no need for pull-up circuitry in the cabling.
A few minutes hacking and I had program to act as both a shutdown monitor
daemon, and to kill the inverter output when the system is shutdown while
on battery power.

I may eventually want to use the smarter serial communication mode rather
than the simple contact mode, so I asked Best technical support for the
documentation, and it arrived today, a week after I called them.  Once I
peruse the documentation I'll decide if a smarter interface is really
worthwhile, especially since at some point I'll need to shut down two
networked machines sharing the UPS.

                Leonard
```

Best Ferrups

In addition to the doumentation and softare on Best's web site, you could also use the `bestups-0.9.tar.gz` (section Software) package. We've just started testing it with our 5kva FERRUPS.

The basic idea is that there are two modules. One which fields information requests on a network port, relays those requests to the UPS, and returns the results. The second module talks to the first, interprets the results, and responds with either OK or FAIL.

This is sufficient to allow the `powerd-2.0.tar.gz` package (section Software) to do the rest of the work.

The details can be gotten from the `bestups-0.9.tar.gz` package (section Software).

Incidentally, our 5kva Ferrups has performed flawlessly in keeping our 10 computers and 30 screens humming.

8.11 GPS1000 from ACCODATA

```
>From hennus@sky.nl.mugnet.org Thu Mar 10 15:10:22 1994
Newsgroups: comp.os.linux.help
Subject: Re: auto-shutdown with UPS
From: hennus@sky.nl.mugnet.org (Hennus Bergman)
Date: Tue, 1 Mar 1994 22:17:45 GMT
Distribution: world
Organization: The Organization For Removal Of On-Screen Logos

In article <CRAFFERT.94Feb28125452@nostril.lehman.com>,
Colin Owen Rafferty <craffert@nostril.lehman.com> wrote:
>I am about to buy an Uninterruptable Power Supply for my machine, and
>I would like to get one that has the "auto-shutdown" feature.
>
I just got one of those real cheap :-)
It's a GPS1000 by ACCODATA. Anybody know how good the output
signal of these things is? [Don't have a scope myself :-(]

>I assume that these each have some kind of serial connection that
>tells the system information about it.
>
I took it apart to find out how it worked. There were three optocouplers
(two output, one input) connected to a 9 pin connector at the back.
One turns on when the power fails, and goes off again when the power
returns. While the power is off, you can use the `input' to shut the
battery off. [It releases the power-relay.] The third one is some kind
of feedback to tell that it did accepted the `shut-down command'.
I think the interface for my UPS was designed to be connected to TTL-level
signals, but with some resistors it could be connected to serial port.
It's wired in such a way that using a RS-232 port you cannot use both
output optocouplers; but the shutdown feedback is not necessary anyway,
just use the important one. ;-)
[Note that it is possible to blow the transistor part in optocouplers
with RS-232 levels if you wire it the wrong way round ;-)]

I was hoping I would be able to connect it to my unused game port,
but that doesn't have an output, does it?
I'll probably end up getting an extra printer port for this.

Not all UPS' use optocouplers, some use simple relays, which are
less critical to connect, but of course not as `nice'.

>Has anyone written a package that watches the UPS and does a shutdown
>(or something) when the power is off?
SysVinit-2.4 (and probably 2.5 as well) has a `powerd' daemon that
continually watches a serial port for presence of the CD (Carrier
Detect) line and signals init when it drops. Init then activates
shutdown with a time delay. If the power returns within a few minutes
the shutdown is cancelled. Very Nice.
The only problem I had with it is that it doesn't actually tell the
UPS to turn off when the shutdown is complete. It just sits there with
a root prompt. I'll probably write a small program to shut it down
>from /etc/brc. RSN.

>    Colin Rafferty, Lehman Brothers <craffert@lehman.com>

Hennus Bergman
```

8.12 TrippLite BC750LAN (Standby UPS)

Tom Webster (webster@kaiwan.com, the author of the genpower package) sent me information on the TrippLite BC750LAN. If you have one of these, your probably best off using his package.

But, for completeness, here's his cable wiring diagram (done by trial and error, and without documentation):

```
  UPS               System
 DB-25              DB-25
   1 <--------------->  1       Ground

   2 <--------------->  4       Power Fail
   8 <--------------->  8       Sensing Circuit

   3 <--------------->  2       Inverter Shutdown
  20 <---------------> 22       Circuit
```

8.13 APC

If the above plethora of APC packages doesn't get you running, maybe the following sections will help.

APC Back−UPS

There seems to be some controversy as to the accuracy of the information here on APC Back−UPSs. So, please be careful. I'm prefacing this section with one message of caution I received. It might not make a lot of sense before the rest of this section is read, but this way, at least you're more likely to see it. And again, since I don't have any APC UPS units, I can't verify the accuracy of either of these messages.

A message of caution

```
From ind43@sun1000.ci.pwr.wroc.pl Sun Oct  9 11:00:42 1994
Newsgroups: comp.os.linux.admin
Subject: BUPS-HOWTO warning
From: ind43@sun1000.ci.pwr.wroc.pl (Marek Michalkiewicz)
Date: 6 Oct 1994 18:38:15 GMT
Organization: Technical Univeristy of Wroclaw
NNTP-Posting-Host: ci3ux.ci.pwr.wroc.pl
X-Newsreader: TIN [version 1.2 PL2]

If you want to connect the APC Back-UPS to your Linux box, this might
be of interest to you.

There is a good BUPS-HOWTO which describes how to do this. But it has
one "bug".
```

```
The RTS serial port signal is used to shut down the UPS. The UPS will
shut down only if it operates from its battery. The manual says that
the shutdown signal must be high for at least 0.5s. But few milliseconds
is enough, at least for my APC Back-UPS 600.

Using RTS to shut down the UPS can be dangerous, because the RTS goes
high when the serial device is opened. The backupsd program then turns
RTS off, but it is on (high) for a moment. This kills the power when
backupsd is first started and there is a power failure at this time.
This can happen for example when the UPS is shut down, unattended,
and the power comes back for a while.

Either start backupsd before mounting any filesystems for read-write,
or (better) use TX (pin 3) instead of RTS (pin 7) to shut down the
UPS (pin numbers are for 9-pin plug). Use ioctl(fd, TCSBRKP, 10);
to make TX high for one second, for example. Using TX should be safe.
Maybe I will post the diffs if time permits...

-- Marek Michalkiewicz
ind43@ci3ux.ci.pwr.wroc.pl
```

BUPS−HOWTO

Luminated Software Group Presents

HOWTO use Back−UPS (by APC) (to keep your linux box from frying)

Version: 1.01 BETA

Document by: Christian G. Holtje <docwhat@uiuc.edu> Cabling info and help: Ben Galliart <bgallia@orion.it.luc.edu>

** Warning! I, nor any of us who have written or helped with this document, make and guarantees or claims for this text/source/hints. If anything is damaged, we take NO RESPONSIBILITY! This works to the BEST OF OUR KNOWLEDGE, but we may have made mistakes. So be careful! **

Al right, you just bought (or are going to buy) a Back−UPS from APC. (Other brands might be able to use this info, with little or no modification, but we don't know) You've looked at the price of the Power−Chute software/cabling, and just are not sure it's worth the price. Well, I made my own cable, and my own software and am using it to automatically shut off the power to my linux box when a power failure hits. Guess what? You can too!

*** The Cable ***

This was the hardest part to figure out (I know little about hardware, so Ben did the most work for this). To build one, you need to buy from your local radio shack (or other part supplier) this stuff:

```
1 9-Position Male D-Subminature Connector (solder-type)
        [Radio Shack cat. no. 276-1537c]
1 9-Position Female D-Subminature Connector (solder-type)
        [Radio Shack cat. no. 276-1538c]
2 casings for the above plugs (usually sold separately)
Some stranded wire (wire made of strands, not solid wire)
```

You also need, but may be able to borrow:

```
1 soldering iron
solder
```

Okay...this is how you connect it up!

These diagrams are looking into the REVERSE SIDE (the side where you solder the wire onto the plugs) The letters G, R, and B represent the colors of the wires I used, and help to distinguish one line from the next. (NOTE: I'm use standard rs−232 (as near as we can tell) numbering. The APC book uses different numbers. Ignore them! Use ours...I already changed the numbers for you!)

```
   ---------------------     Male Side! (This goes into the UPS)
    \  B   R  *  *  * /
      \  *  *  *  G  /
        ------------


   ---------------------     Female Side! (This goes into your COM port)
    \  R   *  *  *  G /
      \  *  B  *  *  /
        ------------
```

For those who like the numbers better:

```
        Male            Female
---------------------------------------
        1               7               Black
        2               1               Red
        9               5               Green
```

−−−−−−−−−Aside: What the rs−232 pins are for!−−−−−−−−−−− Since we had to dig this info up anyway:

>From the REAR (the soldering side) the pins are numbered so:

```
          ---------------------
           \  1   2  3  4  5 /
             \  6  7  8  9  /
               ------------
```

The pins mean:

```
                Number  Name                    Abbr. (Sometimes written with D prefix)
                1       Carrier Detect          CD
                2       Receive Data            RD
                3       Transmit Data           TD(?)
                4       Data Terminal Ready     DTR
                5       Signal Ground           Gnd
                6       Data Set Ready          DSR
                7       Request to Send         RTS(?)
                8       Clear to Send           CS
                9       Ring Indicator          RI
```

What we did is connect the UPS's RS−232 Line Fail Output to the CD, the UPS's chassis to Gnd, and the UPS's RS−232 Shut Down Input to RTS. Easy now that we told you, no?

I have no idea if the software below will work, if you purchase the cable from APC. It might, and it might not.

*** The Software ***

Okay, I use the SysVInit package by Miquel van Smoorenburg for Linux. (see end for file locations, credits, email addresses, etc.) I don't know what would have to be changed to use someone elses init, but I know this code (following) will work with Miquel's stuff. Just so I give credit where credit's due. I looked at Miquel's code to figure out how ioctl()'s worked. If I didn't have that example, I'd have been in trouble. I also used the powerfail() routine (verbatim, I think), since it must interact with his init, I thought that he should know best. The .c file is at the end of this document, and just needs to be clipped off. To clip the file, edit away and extra '.sigs' and junk. This document should end on the line /* End of File */.....cut the rest.

This program can either be run as a daemon to check the status of the UPS and report it to init, or it can be run to send the kill−power command to the UPS. The power will only be killed if there is a power problem, and the UPS is running off the battery. Once the power is restored, it turns back on.

To run as a daemon, just type: backupsd /dev/backups

/dev/backups is a link to /dev/cua0 at the moment (COM 1, for you DOSers). The niceness of the link is that I can just re−link the device if I change to com 2 or 3.

Then, if the power dies init will run the commands for the powerwait. An example (This is from my `/etc/inittab`):

```
#Here are the actions for powerfailure.
pf::powerwait:/etc/rc.d/rc.power start
po::powerokwait:/etc/rc.d/rc.power stop
```

The powerwait will run, if the power goes down, and powerokwait will run if the power comes back up.

Here is my entire rc.power:

```
#! /bin/sh
#
# rc.power      This file is executed by init when there is a powerfailure.
#
# Version:      @(#)/etc/rc.d/rc.power   1.50    1994-08-10
#
# Author:       Christian Holtje, <docwhat@uiuc.edu>
#

  # Set the path.
  PATH=/sbin:/etc:/bin:/usr/bin:/sbin/dangerous

  # Find out how we were called.
  case "$1" in
        start)
                echo "Warning there is Power problems."  | wall
                # Save current Run Level
                ps | gawk '{ if (($5 == "init") && ($1 == "1")) print $6 }' \
                         | cut -f2 -d[ | cut -f1 -d] \
                         > /tmp/run.level.power
                /sbin/shutdown -h +1m
                ;;
        stop)
                echo "Power is back up.  Attempting to halt shutdown." | wall
                shutdown -c
                ;;
        *)
                echo "Usage:  $0 [start|stop]"
                exit 1
                ;;
  esac
```

Pretty nifty, no? Actually, there is a problem here...I haven't had time to figure it out...If there is a 'sh' wizard out there....

There is one little detail left, that is having the UPS turn off the power if it was halted with the power out. This is accomplished by adding this line into the end of your halt script:

```
/sbin/backupsd /dev/backups killpower
```

This will only kill the power if there is no power being supplied to your UPS.

*** Testing the stuff ***

This is just a short section saying this:

BE CAREFUL!

I recommend backing up your linux partitions, syncing several times before testing and just being careful in general. Of course, I'm just recommending this. I wasn't careful at all, and had to clean my partition several times testing my config. But it works. :)

*** Where to Get It ***

Miquel van Smoorenburg's SysVInit can be gotten at:

`sunsite.unc.edu:/pub/Linux/system/Daemons/SysVinit-2.50.tgz`

and a fix for some bash shells is right next−door as:

`sunsite.unc.edu:/pub/Linux/system/Daemons/SysVinit-2.50.patch1`

As to getting this HOWTO, you can email me. docwhat@uiuc.edu with the subject saying 'request' and the keyword 'backups' in body of the letter. (I may automate this, and other stuff)

*** Credit Where Credit's Due Dept. ***

Thanks to Miquel van Smoorenburg <miquels@drinkel.nl.mugnet.org> for his wonderful SysVInit package and his powerd.c which helped me very much.

Christian Holtje <docwhat@uiuc.edu> Documentation backupsd.c (what wasn't Miquel's) rc.power

Ben Galliart <bgallia@orion.it.luc.edu> The cable Information for the RS−232 standard Lousy Jokes (none quoted here)

```
/*  backupsd.c -- Simple Daemon to catch power failure signals from a
 *                Back-UPS (from APC).
 *
 *  Parts of the code are from Miquel van Smoorenburg's powerd.c
 *  Other parts are original from Christian Holtje <docwhat@uiuc.edu>
 *  I believe that it is okay to say that this is Public Domain, just
 *  give credit, where credit is due.
 *
 *  Disclaimer:  We make NO claims to this software, and take no
 *               resposibility for it's use/misuse.
```

```
 */

#include <sys/types.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>

/* This is the file needed by SysVInit */
#define PWRSTAT         "/etc/powerstatus"

void powerfail(int fail);

/* Main program. */
int main(int argc, char **argv)
{
  int fd;
  int killpwr_bit = TIOCM_RTS;
  int flags;
  int status, oldstat = -1;
  int count = 0;

  if (argc < 2) {
        fprintf(stderr, "Usage: %s <device> [killpower]\n", argv[0]);
        exit(1);
  }

  /* Open the the device */
  if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
        fprintf(stderr, "%s: %s: %s\n", argv[0], argv[1], sys_errlist[errno]);
        exit(1);
  }

  if ( argc >= 3  && (strcmp(argv[2], "killpower")==0) )
      {
          /* Let's kill the power! */
          fprintf(stderr, "%s: Attempting to kill the power!\n",argv[0] );
          ioctl(fd, TIOCMBIS, &killpwr_bit);
          /* Hmmm..... If you have a power outtage, you won't make it! */
          exit(0);
      }
  else
      /* Since we don't want to kill the power, clear the RTS. (killpwr_bit) */
      ioctl(fd, TIOCMBIC, &killpwr_bit);

/* Become a daemon. */
  switch(fork()) {
  case 0: /* I am the child. */
              setsid();
              break;
  case -1: /* Failed to become daemon. */
              fprintf(stderr, "%s: can't fork.\n", argv[0]);
              exit(1);
  default: /* I am the parent. */
              exit(0);
  }


  /* Now sample the DCD line. */
```

```
  while(1) {
      ioctl(fd, TIOCMGET, &flags);
      status = (flags & TIOCM_CD);
      /* Did DCD jumps to high? Then the power has failed. */
      if (oldstat == 0 && status != 0) {
          count++;
          if (count > 3) powerfail(0);
          else { sleep(1); continue; }
      }
      /* Did DCD go down again? Then the power is back. */
      if (oldstat > 0 && status == 0) {
          count++;
          if (count > 3) powerfail(1);
          else { sleep(1); continue; }
      }
      /* Reset count, remember status and sleep 2 seconds. */
      count = 0;
      oldstat = status;
      sleep(2);
  }
  /* Error! (shouldn't happen) */
  return(1);
}


/* Tell init the power has either gone or is back. */
void powerfail(ok)
int ok;
{
  int fd;

  /* Create an info file needed by init to shutdown/cancel shutdown */
  unlink(PWRSTAT);
  if ((fd = open(PWRSTAT, O_CREAT|O_WRONLY, 0644)) >= 0) {
        if (ok)
                write(fd, "OK\n", 3);
        else
                write(fd, "FAIL\n", 5);
        close(fd);
  }
  kill(1, SIGPWR);
}

/* End of File */
```

More notes

```
From ockers@carnot02.maem.umr.edu Mon Jan 16 15:27:29 1995
Newsgroups: comp.os.linux.hardware
Subject: Back-UPS, backupsd, and low battery signal
From: ockers@carnot02.maem.umr.edu (Jim Ockers)
Date: 12 Jan 1995 04:22:44 GMT
Reply-To: ockers@umr.edu
Organization: the all-male wasteland of Rolla, MO
NNTP-Posting-Host: carnot02.maem.umr.edu
X-Newsreader: TIN [version 1.2 PL2]
```

```
Hello all,

I use the backupsd on my linux system and I like it a lot.  I also
run Windows NT when I have to and it has a UPS daemon too.  The pinouts
required by Windows NT are different from the ones you specify in the
program but that is easily changed since I have the source for your
program..

Anyways I was browsing through the Windows NT knowledge base (KB) and
found something interesting.  If you look in the documentation for your
Back-UPS under "computer interface port" you will see that this UPS will
send a Low Battery signal at least two minutes before the battery fails.

At least the manual for my Back-UPS 400 says that...

However they also speak some Electrical Engineering gibberish ("Outputs ...
are actually open collector outputs which must be pulled up to a common
referenced supply no greater than +40 Vdc.  The transistors are capable
of a maximum non-inductive load of 25mAdc.)

Well that means nothing to me, but what I discovered in the NT KB was
that it is possible to use the low battery signal in the same manner that
the other signals are used.  The output from pin 5 on the UPS should go
to the pin on which you are reading the LowBatt signal into the computer.
When that line goes high, the battery is running out of charge.  When
the situation is normal, that line will be low.  (Hi/Lo in a standard
RS-232 signal, just like the other lines.)

What they don't tell you in the APC manual, and they should, is that
you need to buy a 10 KOhm resistor (50 cents at Radio Shack) and connect
pins 5 and 8 on the UPS side using the resistor.  Pin 8 provides the
"common referenced supply no greater than 40vdc".  Here's how you would
make the cable (the 1st three lines are the same as the HOWTO):

       PC side                         UPS side
pin      7 <------------------------------> 1                ShutDownUPS
         1 <------------------------------> 2                LineFail
         5 <------------------------------> 4 (same as 9 )    GND
         ? your choice  <-----------------> 5                LowBatt
                                            |
                                             > 10
                                            <  KOhm
                                            |
                                            8

So then when the LowBatt line is HIGH then the computer has 2 minutes
to shut down before the battery runs out.

This is not mentioned in the Back-UPS HOWTO nor is it addressed in the
backupsd source.  However I would think that it would be a Good Thing
to have in there; especially since a power failure would not require
a shutdown unless the UPS batteries were low.  In most cases it would
mean that the backupsd could send a warning to everyone if the LineFails,
and give everyone a one (or two) minute warning when the batteries start
running down.

As far as I know this applies to all the APC Back-UPS and Smart-UPS
products.  These instructions were for a Smart-UPS 900,1250, and 2000
according to the NT KB.  However they have been tested with a Back-UPS
400 running Windows NT and everything works properly...
```

```
I'd sure like to have a backupsd that handled the LowBatt situation too.
Does anyone feel like modifying the backupsd.c source so that it will do
this too?  (I can't program in C yet...)

P.S. The APC manual says to use only pin 4 as the common and even though
in the diagram it says that pin 9 is connected to pin 4 you might want to
be sure and use pin 4 .  This differs from the instructions in the HOWTO.

P.P.S. I mailed this to the Back-UPS HOWTO authors.

--
Jim  (ockers@umr.edu)                   Ask me about Linux!
http://www.umr.edu/~ockers/ - home page




From: Peter Kammer <pkammer@liege.ICS.UCI.EDU>
To: "Harvey J. Stein" <hjstein@math.huji.ac.il>
cc: "Christian G. Holtje" <docwhat@uiuc.edu>
Subject: UPS-Howto--minor correction
Date: Mon, 07 Oct 1996 12:00:16 -0700


Mr. Stein,

        Let me first thank you for putting together and maintaining the
Linux UPS-HowTo document.  I recently attached a APC Back-UPS 400 to a
a Linux box and the document turned out to be very helpful.
        I would like to suggest a correction to the the text diagrams which
accompany the description in section 11.5.2.  The diagrams are presented as
being the rear of the plug.  This in mind, the diagram of the male is
backwards:

         ---------------------     Male Side! (This goes into the UPS)
          \  B   R  *  *  * /
            \  *  *  *  G  /
              ------------


_From the rear_, the pins on the male connector are numbered
right-to-left.  The correct diagram should be:

         ---------------------     Male Side! (This goes into the UPS)
          \  *   *  *  R  B /
            \  G  *  *  *  /
              ------------


Similarly, the numbered diagram should be labeled as for the rear of the
female plug.

         ---------------------
          \  1   2  3  4  5 /
            \  6  7  8  9  /
              ------------

The rear of the male is numbered the reverse:

         ---------------------
          \  5   4  3  2  1 /
```

```
            \  9  8  7  6  /
              ------------
```

```
This caused us some confusion until we realized our mistake.  With four
different configurations to be aware of (front, rear) x (male, female) it
is easy to get confused.  Even now, reference in hand, I keep reexamining
my diagrams.

It might also be helpful to add a reference to the APC technical document
for the Back-UPS line which is available on-line at:

   http://www.apcc.com/english/techs/techref4/224e.htm

Once we corrected our wiring, setting up the software was relatively simple
thanks to your documentation.  We used the alternative (using TD to kill
the UPS power rather than RTS) wiring scheme and ran into few problems.
Your efforts in maintaining this information are much appreciated.

------------
Peter Kammer                       Dept. of Information and Computer Science
pkammer@ics.uci.edu                University of California
http://www.ics.uci.edu/~pkammer/   Irvine, CA 92697-3425
```

APC Back−UPS Pro 650

```
From: Troy Muller <tmuller@agora.rdrop.com>
Sender: tmuller@napalm.it.wsu.edu
To: abel@netvision.net.il
Subject: APC Back-UPS Pro 650
Date: Sun, 06 Apr 1997 12:50:40 -0700

Dear Mr. Stein,

I have a Back-UPS Pro 650 from APC and finally got it working with a
standard APC cable.

I used cable number 940-0023A and Enhanced_APC_BackUPS software.  My
only grudge is the software broadcasts every 2 seconds, but hacking the
dowall.c code to sleep 10 sec before broadcasting seems to limit it to
every 10 seconds with a 2-3 message queued to be printed (ie. much more
acceptable).
```

APC Smart−UPS

Many people have APC Smart UPSs. There seem to be packages for using them in smart modes (see the afore mentioned packages `Enhanced_APC_UPSD-v1.4.tar.gz`, `apcd-0.5.tar.gz`, and `smupsd-0.7-1.i386.rpm` described in section Software). I don't know how the support in each package is. It seems that APC **still** refuses to release the protocol for the ``smart'' mode without a non−disclosure agreement, so everyone's left reverse engineering it.

The general consensus is to buy from a brand which does release the information, such as Best.

Another possibility is to run the SCO Unix version of APC's Powerchute UPS control software under Linux via the iBCS compatibility package. I'm told by Clive A. Stubbings (cas@vjet.demon.co.uk) that this works nicely after some install script adjustments. He says that the only problem is "the GUI stuff seems to have difficulty controlling non−local UPSs across the net".

If you have an APC Smart−UPS, and you can't get any of the above software to work in smart mode, you can still use it in dumb mode. The following sections detail how to do that. In particular, I've received messages from people regarding the Model 600, the Model 700, and the model 1400. You'll probably have to hack `powerd.c` as outlined in section Reverse−engineering cables and hacking powerd.c.

APC Smart−UPS, Model 600

```
From dangit@netcom.com Mon Aug 22 10:16:23 1994
Newsgroups: comp.os.linux.misc
Subject: UPS Monitoring Cable For APC
From: dangit@netcom.com (Lam Dang)
Date: Fri, 19 Aug 1994 11:56:28 GMT
Organization: NETCOM On-line Communication Services (408 261-4700 guest)
X-Newsreader: TIN [version 1.2 PL1]

[Didn't make it the first time.]

A few netters have asked about UPS monitoring cables.  This is what I
found when I made one for my APC Smart-UPS, Model 600.  A disclaimer is in
order.  This is just an experimenter's report; use it at your own risks.
Please read the User's Manual first, especially Section 6.4, Computer
Interface Port.

The cable is to run between a 9-pin female connector on the UPS and a
25-pin male connector on the PC.  Since I cut off one end of a 9-pin
cable and replaced it with a 25-pin connector, I had to be VERY
CAREFUL ABOUT PIN NUMBERS.  The 25-pin hood is big enough to contain a
voltage regulator and two resistors.  I got all the materials (listed
below) from Radio Shack for less than 10 bucks.  As required by Windows NT
Advanced Server 3.5 (Beta 2), the "interface" between the UPS connector
and the PC connector is as follows:

        UPS (9-pin)              PC (25-pin)

        1 (Shutdown)             20 (DTR)
        3 (Line Fail)             5 (CTS)
        4 (Common)                7 (GND)
        5 (Low Battery)           8 (DCD)
        9 (Chassis Ground)        1 (Chassis Ground)

This is pretty straightforward.  You can use UPS pin 6 instead of 3
(they're the inverse of each other).  The complication is in pulling up
UPS open collector pins 3 (or 6) and 5.

This APC model provides an unregulated output of 24 Vdc at UPS pin 8. The
output voltage is available all the time (at least until some time after
Low Battery has been signalled).  The supply is limited to 40 mA.  To
pull up, UPS pin 8 is input to a +5 Vdc voltage regulator.  The output of
```

```
the regulator goes into two 4.7K resistors.  The other end of one
resistor connects both UPS pin 3 (Line Fail) and PC pin 5 (CTS).  That
of the other resistor connects both UPS pin 5 (Low Battery) and PC pin 8
(DCD).  The two resistors draw about 2 mA when closed.

Test your cable without connecting it to the PC.  When the UPS is on
line, pins 5 (CTS) and 8 (DCD) at the PC end of the cable should be very
close to 5 Vdc, and applying a high to pin 20 (DTR) for 5 seconds should
have no effect.  Now pull the power plug to put the UPS on battery.  Pin
5 (CTS) should go down to zero Vdc, pin 8 (DCD) should stay the same at 5
Vdc, and applying a high to pin 20 (DTR), e.g., by shorting pins 8 and 20,
should shut down the UPS after about 15 seconds.

Keep the UPS on battery until Low Battery is lighted on its front panel.
Now pin 8 (DCD) should go down to zero Vdc too.  Wait until the UPS
battery is recharged.  Then connect your cable to the PC, disable the UPS
option switches by turning all of them ON, and run your favorite UPS
monitoring software.

For those who want to run it with Windows NT Advanced Server, the UPS
interface voltages are NEGATIVE for both power failure (using UPS pin 3)
and low battery conditions, and POSITIVE for remote shutdown.  Serial
line parameters such as baud rate don't matter.

I haven't tested my cable with Linux powerd.  When you do, please let us
know.  I run NT as often as Linux on the same PC.  I must conform to NT's
UPS scheme.  Perhaps somebody can modify powerd to work with it and post
the source code here.

List of materials:

        1 shielded D-sub connector hood (Radio Shack 276-1510)
        1 25-pin female D-sub crimp-type connector (276-1430)
        1 7805 +5Vdc voltage regulator (276-1770)
        2 4.7K resistors
        1 component perfboard (276-148)
        1 cable with at least one 9-pin male connector.

You'll need a multimeter, a soldering iron, and a couple of hours.

Hope this helps.

Regards,

--
Lam Dang
dangit@netcom.com
```

APC Smart−UPS 700

Here're some details for running an APC Smart−UPS 700 in dumb mode.

It has a clever usage of a transistor in the cable so that the UPS will turn off when the computer is turned off. And it includes a `powerd.c` which also does a fast low battery shutdown.

Also, note that Markus' is also using `init`'s new capabilities. So we have here another illustration of how to use the new `init` to your advantage.

```
From: Markus Eiden <Markus@eiden.de>
Sender: eiden@eiden.de
To: "Harvey J. Stein" <abel@netvision.net.il>
Subject: Re: APC Smart-UPS
Date: Sun, 30 Mar 1997 16:21:05 +0200

I'm using an APC Smart-UPS 700 for my Linux box, running 2.0.21 on an
ASUS-Board.

To use some features of the UPS you need four things:


1) You have to build a RS232-cable with a small interface.
2) You need the powerd-source from the sysvinit-package (I use version 2.65
    from Miquel van Smoorenburg). Then you have to patch his powerd.
3) You have to change your /etc/inittab
4) You need a script to run some commands if the power is down or battery
   is low.


Some features:

When the power goes down, a script will start and a syslog-entry is done.

If the battery is low, an other  script will start
(which shutdown your computer of course) and a syslog-entry is done.

If you shutdown your computer and the power is down, the UPS will be shut
down too.




 1)Let's start with the cable:
================================

If you have a look at the back side of you UPS you will see a female
connector like this:



            8            1: Shutdown the UPS when the power is down and
                             pin 1 is high.
     X   X   X   X       3: Goes low on "Linefail"
   X   X   X   X   X     4: GND
                         5: Goes low on  "Low battery"
   1       3   4   5     8: +24V




On the other hand at the back side of your PC there exist a male
connector like this:


        8       6        1: DCD
     X   X   X   X       4: DTR
   X   X   X   X   X     5: GND
   5   4           1     6: DSR
                         8: CTS
```

```
You have to build the following interface between these connectors:



 PC                                                          UPS


                                         #------------------  (8)
                                         |
                                        470 Ohm
                                         |
           #-----#-----#-----#-----#-----#----- ca. 9-12V
           |     |     |     |     |     |
           47    3.3   3.3   3.3   1     470
           kOhm  kOhm  kOhm  kOhm  kOhm  Ohm
           |     |     |     |     |     |
 (8) ------------------------#     |     |
           |     |     |           |     |
 (6) ------------#------------------------------------------ (5)
           |           |           |     |
 (1) ------------------#------------------------------------ (3)
           |                       |     |
           |                      C#------------------------- (1)
           |                      -|     |
           |                    B/       |
 (4) ------#-----12kOhm---------|        |
                                 \>E     |
                                  |      |
 (5)------------------------------#-------#------------------- (4)



  -I use a "BC140" - transistor, but nearly any simple NPN-transistor
   should work  ;-)

  -The transistor works as an "inverter". If you shutdown your PC,
   AND the power ist down, then pin 4(PC) goes low and 1(UPS) goes
   high. This shuts down the UPS  for saving batteries power.



 2) The powerd-Source:
=======================

I patched the powerd-source very little (so it is really the source
of Miquel).

(a) Give an "alert" to the syslogd if 8(PC, CTS) is down (Cause
     the cable is not connected)

(b) DCD droped to zero => power has failed => call powerfail(0)
    => Give INIT_CMD_POWERFAIL to the init-process

(c) DCD comes up again => power is back => call powerfail(1)
    => Give INIT_CMD_POWEROK to the init-process
```

```
(d) DSR and DCD are dropped to zero => power has failed and
    battery is low => call powerfail(2) => Give
    INIT_CMD_POWERFAILNOW to the init-process

 Thats it.


------------------------------>8---- Schnipp ----------------------------


/*
 * powerd       Monitor the DCD line of a serial port connected to
 *              an UPS. If the power goes down, notify init.
 *              If the power comes up again, notify init again.
 *              As long as the power is OK, the DCD line should be
 *              "HIGH". When the power fails, DCD should go "LOW".
 *              Powerd keeps DTR high so that you can connect
 *              DCD and DTR with a resistor of 10 Kilo Ohm and let the
 *              UPS or some relais pull the DCD line to ground.
 *              You also need to connect DTR and DSR together. This
 *              way, powerd can check now and then if DSR is high
 *              so it knows the UPS is connected!!
 *
 * Usage:       powerd /dev/cua4 (or any other serial device).
 *
 * Author:      Miquel van Smoorenburg, <miquels@drinkel.cistron.nl>.
 *              Some minor changes by Markus Eiden, <Markus@Eiden.de>
 *              for the APC-Smart-UPS-powerd.
 *
 * Version:     1.31,  29-Feb-1996.
 *
 *              This program was originally written for my employer,
 *                      ** Cistron Electronics **
 *              who has given kind permission to release this program
 *              for general puppose.
 *
 *              Copyright 1991-1996 Cistron Electronics.
 *
 *              This program is free software; you can redistribute it and/or
 *              modify it under the terms of the GNU General Public License
 *              as published by the Free Software Foundation; either version
 *              2 of the License, or (at your option) any later version.
 *
 *              Some minor changes for the APC-powerd by Markus Eiden
 *              Markus@Eiden.de
 */

/* Use the new way of communicating with init. */
#define NEWINIT

#include <sys/types.h>
#include <sys/stat.h>
#include <sys/ioctl.h>
#include <fcntl.h>
#include <errno.h>
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <signal.h>
#include <syslog.h>
#include <string.h>
```

```
#include "paths.h"
#ifdef NEWINIT
#include "initreq.h"
#endif

#ifndef SIGPWR
#  define SIGPWR SIGUSR1
#endif

#ifdef NEWINIT
void alrm_handler()
{
}
#endif

/* Tell init the power has either gone or is back. */
void powerfail(ok)
int ok;
{
  int fd;
#ifdef NEWINIT
  struct init_request req;

  /* Fill out the request struct. */
  memset(&req, 0, sizeof(req));
  req.magic = INIT_MAGIC;


  /* INIT_CMD_* are definied in initreq.h                  *
   * Have a look at  init.c and /etc/inittab               *
   *                                                       *
   * ok=0 -> INIT_CMD_POWERFAIL      -> powerwait          *
   * ok=1 -> INIT_CMD_POWEROK        -> powerokwait        *
   * ok=2 -> INIT_CMD_POWERFAILNOW   -> powerfailnow        */


  switch (ok) {
    case 0 : req.cmd = INIT_CMD_POWERFAIL;
             /* Linefail -> warning */
             break;
    case 1 : req.cmd = INIT_CMD_POWEROK;
             /* Power is back -> cancel warning */
             break;
    case 2 : req.cmd = INIT_CMD_POWERFAILNOW;
             /* Linefail and LowBatt -> reboot */
             break;
             }

  /* Open the fifo (with timeout) */
  signal(SIGALRM, alrm_handler);
  alarm(3);
  if ((fd = open(INIT_FIFO, O_WRONLY)) >= 0
                && write(fd, &req, sizeof(req)) == sizeof(req)) {
        close(fd);
        return;
  }
  /* Fall through to the old method.. */
#endif

  /* Create an info file for init. */
  unlink(PWRSTAT);
  if ((fd = open(PWRSTAT, O_CREAT|O_WRONLY, 0644)) >= 0) {
```

```
        if (ok)
                write(fd, "OK\n", 3);
        else
                write(fd, "FAIL\n", 5);
        close(fd);
  }
  kill(1, SIGPWR);
}

/* Main program. */
int main(int argc, char **argv)
{
  int fd;
  int dtr_bit = TIOCM_DTR;
  int flags;
  int status, oldstat = -1;
  int count = 0;
  int tries = 0;
  int powerfailed = 0;
  int rebootnow   = 0;

  if (argc < 2) {
        fprintf(stderr, "Usage: powerd <device>\n");
        exit(1);
  }

  /* Start syslog. */
  openlog("powerd", LOG_CONS|LOG_PERROR, LOG_DAEMON);

  /* Open monitor device. */
  if ((fd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
        syslog(LOG_ERR, "%s: %s", argv[1], sys_errlist[errno]);
        closelog();
        exit(1);
  }

  /* Line is opened, so DTR is high. Force it anyway to be sure. */

  /* USE: low Batt -> Reboot -> DTR goes low                    *
   * transistor is open -> shutdown-pin  of the UPS goes        *
   * high -> UPS goes down after about 20s. If there is a       *
   * linefail and the computer is off, the                      *
   * UPS goes down. If the power is back, the                   *
   * UPS goes on, the computer boots, and powerd                *
   * is startet.                                                */

 /* Verwendung: Die UPS meldet low Batt -> Reboot -> DTR geht   *
   * auf Low -> Transistor oeffnet -> Shutdown der UPS geht auf  *
   * High -> UPS schaltet sich nach circa 20s aus. Bei jedem     *
   * Linefail und ausgeschaltetem Computer, schaltet sich die    *
   * UPS aus. Kommt der Strom zurueck, dann schaltet sich die    *
   * UPS selbstaendig ein, der Computer bootet, und der powerd   *
   * wird gestartet.                                             */

  ioctl(fd, TIOCMBIS, &dtr_bit);




  /* Daemonize. */
  switch(fork()) {
        case 0: /* Child */
```

```
                closelog();
                setsid();
                break;
        case -1: /* Error */
                syslog(LOG_ERR, "can't fork.");
                closelog();
                exit(1);
        default: /* Parent */
                closelog();
                exit(0);
  }

  /* Restart syslog. */
  openlog("powerd", LOG_CONS, LOG_DAEMON);

   syslog(LOG_INFO, "APCpowerd started...");

  /* Now sample the DCD line. */
  while(1) {
        /* Get the status. */
        ioctl(fd, TIOCMGET, &flags);

        /* Check the connection: CTS should be high. */
        tries = 0;
        /* TIOCM_*- have a look at  .../ams/termios.h */
        while((flags & TIOCM_CTS) == 0) {
                /* Keep on trying, and warn every two minutes. */
                if ((tries % 60) == 0)
                    syslog(LOG_ALERT, "UPS connection error");
                sleep(2);
                tries++;
                ioctl(fd, TIOCMGET, &flags);
        }
        if (tries > 0)
                syslog(LOG_ALERT, "UPS connection OK");

        /* Calculate present status. */
        status = (flags & TIOCM_CAR);

        /* Did DCD drop to zero? Then the power has failed. */
        if (oldstat != 0 && status == 0) {
                count++;
                if (count > 3) {
                        powerfailed = 1;
                        powerfail(0);
                        }
                else {
                        sleep(1);
                        continue;
                }
        }
        /* Did DCD come up again? Then the power is back. */
        if (oldstat == 0 && status > 0) {
                count++;
                if (count > 3) {
                        powerfailed = 0;

                        /* eigentlich unnoetig: */
                        rebootnow = 0;

                        powerfail(1);
                        }
```

```
                else {
                        sleep(1);
                        continue;
                }
        }

        /* Low battery and Linefail ? */
        if (rebootnow==0)
        if (powerfailed==1)
        if ((flags & TIOCM_DSR) == 0)
        {
           rebootnow=1;
           powerfail(2);

        }

        /* Reset count, remember status and sleep 2 seconds. */
        count = 0;
        oldstat = status;
        sleep(2);
  }
  /* Never happens */
  return(0);
}


------------------- schnap ------------ 8<----------------------------

 3) Change your inittab:
=========================


Init gets the INIT_CMDs and will start a script:

pf::powerwait:/sbin/init.d/powerfail    start
pn::powerfailnow:/sbin/init.d/powerfail now
po::powerokwait:/sbin/init.d/powerfail  stop

(Which means for example: if the power has failed (powerwait) start the
script /sbin/init.d/powerfail with the parameter "start".)



4) The powerfail-Script
========================



------------------ 8< ------- schnipp ---------------

#! /bin/sh
# Copyright (c) 1997 Markus Eiden, Markus@Eiden.de
#

case "$1" in
    start)
        echo "THE POWER IS DOWN!" | wall
        logger "Powerfail"
        ;;
    now)
        echo "BATTERY IS LOW! Shutdown in 1 minute" | wall
        logger "Battery is low, shutdown in 1 minute"
```

```
        sync
        /sbin/shutdown -r -t 5 +1
        ;;
    stop)
        echo "THE POWER IS BACK!!" | wall
        logger "Power is back"

        /sbin/shutdown -c >/dev/null 2>/dev/null

        ;;
  *)
        echo "Usage: $0 {start|now|stop}"
        exit 1
        ;;
esac

exit 0

--------------------- >8 ----------- schnapp -------------

Well, that should be easy ;-)



You are ready now, but be careful: It works for me, but I really can't
guarantee that any of this will work for you.

Some advice at the end:  If /sbin/init.d/powerfail shuts down your
PC  then DTR goes down, so the shutdown pin (UPS) goes high.
>From that time it takes about 20 or 30 seconds for the UPS to shut down.
It is your job to prevent your Linux-box from booting within these 20
seconds (in particular to mount the filesystem). On my system it was no
problem. There are four easy ways to prevent the PC from the fast booting:

 1) The BIOS should do some routines (Like searching the number of
     tracks of your floppydisk if you have one)

 2) If you have LILO installed, tell him to wait.

 3) You do nothing (like I did)

 4) You buy some more memory so that counting the memory takes 30 seconds.
    That should be about 1024 Megabytes ;-).


Markus Eiden

Markus@Eiden.de


--
-----------------------------------------------------------------------
StR Dipl.-Ing. Markus Eiden \\://                       Markus@eiden.de
Am alten Sportplatz 3       (o -)        http://www.rp.schule.de/eiden/
D-67599 Gundheim      ---ooO-(_)-Ooo---             NIC-HDL: ME256-RIPE
```

APC Smart−UPS 1400

Another day, another APC. This is for the Smart−UPS 1400, in dumb mode.

```
From: "Slavik Terletsky" <ts@polynet.lviv.ua>
To: hjstein@math.huji.ac.il
Subject: my contribution to UPS HOWTO
Date: Mon, 27 Jan 1997 21:10:16 +0000

Hello
I just hacked ups daemon, if you want, you may enclose it
in your UPS HOWTO document (i used some info from).
Please replay.
--

UPS daemon for FreeBSD (2.1.5 - tested).
Interacts with APC Smart-UPS 1400.

Connection scheme:

UPS (pin, signal name)          PC (pin, signal name)
----------------------          ---------------------
1 Shutdown              >----------->   4 Data Terminal Ready
2 Line Failed           >----------->   8 Clear To Send
4 Common                >----------->   5 Ground
5 Battery Low           >---------+-->   1 Data Carrier Detector
8 Battery (+24V)        >--|10K|-+


UPSD DESCRIPTION

usage: upsd <device> [wait [script]]

device  - device name upsd interacts thru (e.g. /dev/cuaa1)
wait    - time (secs) to wait before running script, (default value 0 sec).
script  - system shutdown script (default /etc/rc.shutdown).

Actions:
upsd logs all the changes of UPS status (power {up,down}, battery {low,ok}).
When "power down" and "battery low" upsd activates UPS SHUTDOWN signal,
waits for a <wait> seconds, and then runs system shutdown script - <script>.

Script sample:

#!/bin/sh
# Script is executed when system is going down.

PATH=/sbin:/bin:/usr/sbin:/usr/bin

echo "System is going DOWN right NOW" | wall

reboot


Upsd source:
/* UPS daemon
 * Copyright 1997 Slavik Terletsky. All rights reserved.
```

```
 * Author: Slavik Terletsky <ts@polynet.lviv.ua>
 * System: FreeBSD
 */
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <syslog.h>
#include <unistd.h>
#include <varargs.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/uio.h>
#include <sys/types.h>
#include <sys/ioctl.h>
#include <sys/ttycom.h>

int status;
int wait = 0;
FILE *fd;
char *scr = "/etc/rc.shutdown";
char *idf = "/var/run/upsd.pid";

void upsterm();
void upsdown(int);

int main(int argc, char *argv[]) {
 int pd;
 int zero = 0;
 char d5, d6, d7;
 char low = 0;
 char pow = 1;

 /* check arguments */
 switch(argc) {
 case  4:
 scr = argv[3];
 case  3:
 wait = atoi(argv[2]);
 case  2:
 break;
 default:
 fprintf(stderr, "usage: %s <device> [wait [script]]\n", argv[0]);
 exit(1);
 }

 /* check if script exists */
 if(!(fd = fopen(scr, "r"))) {
 fprintf(stderr, "fopen: %s: %s\n", scr, sys_errlist[errno]);
 exit(1);
 }
 fclose(fd);

 /* check if upsd is already running */
 if(fd = fopen(idf, "r")) {
 fprintf(stderr, "fopen: %s: File already exists\n", idf);
 exit(1);
 }

 /* become a daemon */
 switch(fork()) {
 case -1:       /* error */
 fprintf(stderr, "fork: %s\n", sys_errlist[errno]);
```

```
exit(1);
case  0:       /* child */
break;
default:       /* parent */
exit(0);
}

/* save the pid */
if(!(fd = fopen(idf, "w"))) {
fprintf(stderr, "fopen: %s: %s\n", idf, sys_errlist[errno]);
exit(1);
}
fprintf(fd, "%d\n", (int)getpid());
fclose(fd);

/* open monitor device */
if((pd = open(argv[1], O_RDWR | O_NDELAY)) < 0) {
fprintf(stderr, "open: %s: %s\n", argv[1], sys_errlist[errno]);
exit(1);
}

/* daemon is alive */
openlog("upsd", LOG_PID, LOG_DAEMON);
syslog(LOG_INFO, "daemon started");

/* signal reaction */
(void)signal(SIGTERM, upsterm);

/* monitor device */
while(1) {
/* clear bits */
if(ioctl(pd, TIOCMSET, &zero) < 0) {
 fprintf(stderr, "ioctl: %s\n", sys_errlist[errno]);
 exit(1);
}

/* get device status */
if(ioctl(pd, TIOCMGET, &status) < 0) {
 fprintf(stderr, "ioctl: %s\n", sys_errlist[errno]);
 exit(1);
}

/* determin status */
d5 = status & 0x20;
d6 = status & 0x40;
d7 = status & 0x80;

/* power up */
if(!(d7 + d5)) {
 if(!pow) {
  syslog(LOG_CRIT, "power up");
  pow = 1;
 }
/* power down */
} else {
 if(pow) {
  syslog(LOG_CRIT, "power down");
  pow = 0;
 }
}

/* battery low */
```

```
 if(!d6 && !low) {
  syslog(LOG_ALERT, "battery low");
  low = 1;

  /* down ups */
  if(!pow) {
   upsdown(pd);
  }
 }

 /* battery ok */
 if(d6 && low) {
  syslog(LOG_CRIT, "battery ok");
  low = 0;
 }

 sleep(1);
 }

 /* not reached */
 return 0;

}

void upsterm() {
 /* log termination message */
 syslog(LOG_INFO, "daemon terminated");

 /* remove pid file */
 unlink(idf);

 exit(0);
}

void upsdown(int pd) {
 /* log shutdown message */
 syslog(LOG_ALERT, "system is going down");

 /* remove pid file */
 unlink(idf);

 /* save our filesystem */
 system("/bin/sync");
 system("/bin/sync");
 system("/bin/sync");

 /* shutdown ups */
 status = TIOCM_DTR;
 if(ioctl(pd, TIOCMSET, &status) < 0) {
 fprintf(stderr, "ioctl: %s\n", sys_errlist[errno]);
 exit(1);
 }

 /* wait and then run script */
 sleep(wait);
 system(scr);
}
# Slavik Terletsky      # University "Lvivska Poytechnika" #
# Network Administrator # mailto:ts@polynet.lviv.ua        #
```

9.How to shutdown other machines on the same UPS

Some people (myself included) have several computers running Linux connected to one UPS. One computer monitors the UPS and needs to get the other computers to shut down when the power goes out.

We assume the computers can communicate over a network. Call the computer that monitors the UPS the master and the other computers the slaves.

In the old days this required some fancy programming.

These days, the best thing to do is to pick up either the `powerd-2.0.tar.gz` package or the `upsd-1.0.tgz` package (see section Software), and follow the instructions. Both are able to run on the slaves in a mode where they connect over the network to a `powerd` or `upsd` process running on the master to query the status of the UPS. Some of the APC specific packages seem to have this ability too.

Note, though, that if your network is insecure, you might want to add a little security to this, lest someone spoof the slave `powerd` processes into thinking that the power is out.

Another possibility is to go for SNMP (Simple Network Management Protocol). Detailing the use of SNMP is beyond the scope of this document, not to mention currently being beyond me.