The UPS Howto

Harvey J. Stein, hjstein@math.huji.ac.il, Berger Financial Research, Ltd. v1.3, 16 April 1995

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

Contents

1 Introduction

This HOWTO covers connecting a UPS to a PC running Linux. The idea is to connect the two in such a way that Linux can shutdown cleanly when the power goes out. To a large extent this document is reduntant, because all the basic info is contained in the powerd man page that comes with the SysVinit package. None the less, there seems to periodically be alot of discussion on the net regarding connecting Linux PCs to UPSs (and the versions of Linux that I installed didn't come with a powerd man page). I figured having a HOWTO would be a good idea because:

- A second source of information might help to understand how to connect Linux to a UPS, even if it's just the same information written differently.
- The HOWTO can serve as a repository for UPS specific data.
- The HOWTO contains additional details that aren't in the powerd man page.

None the less, this does not replace the powerd man page. Hopefully, after reading both, people will be able to deal with UPSs.

1.1 Contributors

I am forever indebted to those from whom I've received help, suggestions, and UPS specific data. The list includes, in order of appearance in this document:

- Miquel van Smoorenburg (miquels@cistron.nl.mugnet.org)
- Danny ter Haar (danny@caution.cistron.nl.mugnet.org)
- Hennus Bergman (hennus@sky.nl.mugnet.org)
- Tom Webster (webster@kaiwan.com)
- Marek Michalkiewicz (ind43@sun1000.ci.pwr.wroc.pl)

- Christian G. Holtje (docwhat@uiuc.edu)
- Ben Galliart (bgallia@orion.it.luc.edu)
- Lam Dang (angit@netcom.com)

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 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.

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

1.3 Other documents

This document does not cover the general features and capabilities of UPSs. For this type of information, you might turn to the UPS.faq, which is available via anonymous FTP from navigator.jpl.nasa.gov (128.149.23.82) in `pub/doc/faq/UPS.faq`, and can probably be found somewhere on rtfm.mit.edu. It is maintained by Nick Christenson (npc@minotaur.jpl.nasa.gov). 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.

2 What you need to do (summary)

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

3 How it's supposed to work

UPS's job

When the power goes out, the UPS continues to power the PC 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

Powerd 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/powerfail` and sends the initd process a `SIGPWR` signal. When the control line goes low again, it writes `OK` to `/etc/powerfail` and sends initd a `SIGPWR` signal.

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

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

4 Where to get the appropriate software

When I last looked, the package to pick up was `/pub/Linux/system/Daemons/SysVinit-2.50.tgz` from sunsite.unc.edu or a mirror. It includes a copy of powerd.c, shutdown.c, an initd that understands what to do with `SIGPWR` and can handle powerfail and powerokwait entries in the inittab file.

Of course, by now, I'm sure much newer versions are available. Whatever you get might differ from the description here, but will probably adhere to the overall philosophy.

5 How to set things up

The following presupposes that you have a cable that works properly with powerd.c. If you're not sure that your cable works (or how it works), see section 8 for information on dealing with poorly described cables and reconfiguring `powerd.c`. Sections 9 and 10 will also be useful.

If you need to make a cable, see section 7 for the overall details, and the subsection of section 11 that refers to your UPS. The latter might also include information on manufacturer supplied cables. You'll probably want to read all of section 11 because each section has afew 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 initd 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 modem is connected, such as `/dev/cua1`.

- Connect PC's serial port to UPS's serial port. DO NOT PLUG PC 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 PC shuts down properly in the proper amount of time.

– **The Dangerous Part.** After everything seems to be proper, powerdown the PC 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 PC shuts down cleanly before the battery on the UPS gives out. This is dangerous because if the power goes out before the PC 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 PC that's protected by a UPS and will shutdown cleanly when the power goes out!

6 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 shutdown procedure so that if it's shutting down in a powerfail situation, then it turns off the UPS after doing everything necessary.

7 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 11.3

```
>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'
```

8 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 hack powerd.c appropriately.

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);
}
```

9 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 <GSRGAAO@TECHNION.BITNET> "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.

```
   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
```

DB-25 Pin #	DB-9 Pin #	Name	EIA	CCITT	DTE-DCE	Description
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

Table 1: Pin Assignment for the Serial Port (RS-232C), 25-pin and 9-pin

10 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.

11 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.

11.1 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" UPSes in the "dumb" mode as outlined above.

Tripp Lite: One person reported that Tripp lite 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.

11.2 Advice 1200 A

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

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!)

11.3 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
```

11.4 TrippLite BC750LAN (Standby UPS)

```
From: Tom Webster <webster@kaiwan.com>
To: hjstein@MATH.HUJI.AC.IL (Harvey J. Stein)
Subject: Re: Help - Powerd & UPS - Help
Date: Mon, 8 Aug 1994 12:26:09 -0700 (PDT)
```

```
Harvey,

First off, let me say that I enjoyed reading your HOWTO.  It is about
what I'd hoped my document might grow into.  I wrote my pseudo-HOWTO
late on night because I kept seeing the "Can I hook a UPS up to
Linux...." message, about once a month on c.o.l.*.  Mine deals
specifically with hooking one vendor's model of UPS (Tripplite's
BCxxx/LAN series) to a Linux box and making powerd work with it.

It is in need of some upkeep, things have been a little hectic.  Now
that I have posted it three or four time in response to questions, I'm
finally getting some feedback (which catches these errors).  The
problems that I know it has so far are:

The proper version of SysVinit is 2.4, not 2.04.

There is some argument about whether one or more resistors are needed
in my cable.  The only place I really see that it might be needed is
in the inverter shutoff, to make sure that I don't send too much
voltage to the UPS.  For the sensor circuit I don't see why a DTE
device can't stand to have its signal looped back to it.  All that I'm
doing is connecting a line that is held high to the Carrier Detect
line.

I should be a little clearer about how the UPS acts when it goes into
powerfail mode.  The Tripplites provide both an open and a closed
circuit on powerfail, two different pins and a common negative pin.
Thus all I have to do is wire a pin that it held high to the carrier
detect line and route this through the UPS's open on powerfail
circuit, to cause the carrier detect to drop.  I think that this
confuses some people who read Miquel van Smoorenburg's description of
a UPS that only provides only a closed circuit on powerfail, and
requires a much more complex cable.

Well that is all I can think of for now.  I'm planning on seeing if my
cable will still operate if I insert resistors into all of the
circuits.  If it does then I'll make the changes to my document,
should find out this weekend.


Tom

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

Of Linux and Uninterruptible Power Supplies
[or How to connect a TrippLite BCXXXLAN UPS]
```

```
by Tom Webster <webster@kaiwan.com>
   05/20/94 (Version 1.0)

1.0 Introduction
I struggled through connecting a TrippLite BC750LAN (Standby UPS) to
my Linux box about six months ago.  Since then I've seen several
requests for information on this subject, so I'm putting it in a
relatively stable format so I can just send this out when the question
reappears.

1.1 The Results
When the power fails in my apartment, several things happen:

   1.  The UPS switches its inverter on and the computer starts
       drawing off of the UPS.  The warning beeper also starts going
       off.  If the power comes back on, the UPS shuts the inverter
       down and switches back to line power.  Nothing else happens,
       other than the beeper turns off.

   2.  If the power is off for ~15-30 seconds, the system will send
       a message to the users (via wall) and initiate a shutdown
       (to halt) in five minutes.  If the power comes back on, the
       shutdown is canceled and a message stating this is sent
       to the users (via wall) stating that the shutdown has been
       canceled.

   3.  While the system is shutting itself down, its dying act is to
       shut off the inverter on the UPS, killing power to the system.
       This is done after the disks are unmounted, and is done to
       prevent the halted system from draining power from the UPS.

   4.  When the line power comes back on, the system restarts normally.

The BC750 has enough juice to keep my system going for quite some
time, so why do I only run it off of the UPS for five minutes?  The
answer rests in a couple of 'rules-of-thumb' (your mileage may vary):

   a.  If the power browns out or blacks out, 90% of the time it will
       be out for 0-2 minutes.

   b.  If it is out for longer, it will be out for .5-3+ hours.

   c.  If it is out for a while (see b), the power will yo-yo at least
       once while the power company is working on it.  (The power
       will come up for 5-45 seconds, and then fail again.)

So, I set my system up to cover the majority of the power outages I
get, without trying to cover the really long ones.  I also keep plenty
```

```
of reserve in the UPS to handle yo-yo situations.

1.2  Disclaimer
I make no warranties or guarantees as to the suitability or sanity of
following my advice.  This is how my system is setup, and as far as I
can tell it works fine for me.  Your setup may need to be different to
fit your needs, especially if you are using different UPS hardware.

2.0  Hardware
In the case of my UPS, I thought that the RS-232 interface was
something of a misnomer.  I was expecting the UPS to send and receive
data, like talking to your modem with Hayes "AT" commands.  This was
not the case.  It seems that it is called an RS-232 interface because
it stays within the voltage and signal limits of the RS-232 spec.  To
communicate with the UPS, you need to be able to sense changes in
state on certain lines and change the state of other lines.  The fact
that these lines may have nothing in common to the lines your system
might expect to use, if it were talking to say a modem or printer, is
probably why the UPS needs special cables to allow software (including
the manufacturer's) to communicate with the UPS.

Through trial and error with a RS-232 patch panel, I was able to come
up with this cable diagram for the cable between the UPS and the
computer.  Please note that I did this without looking at the official
TrippLite cable and it may be different.

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

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

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

Once you have the cable patched together, just hook the UPS side to
the UPS and the System side to a free serial port on your Linux box.
You will probably have to mess around with 9->25 and 25->9 adapters
to get your cable to fit, but you and a good computer store should
be able to handle this.

3.0 Software
The software that I use is all available to Linux users and comes
with most distributions (SLS and Slackware at least).  This setup
has worked for me through Kernels .99.9, .99.14, and 1.00.

3.1 System V Init
```

```
This package is needed to make the whole thing work.  If you are
still using the "Simple Init" package, perhaps it is time you looked
at upgrading.  The version I am using is 2.04, and I believe that
Miquel van Smoorenburg is the author of the package.

3.2 powerd
powerd is the power daemon, by default is sits and watches for a
change in state on the DCD line and reports these changes to the
system via the signal mechanism.  The source for powerd is provided
in the System V Init package.  Compile it, move it to a binary
directory (I put it in /sbin on my system), and alter your rc.local
script to start the daemon.  The relevant part of my rc.local looks
like this:

    ----- snip -----
    # Add support for the UPS
    echo "Starting powerd daemon..."
    if [ -x /sbin/powerd ]; then
       /sbin/powerd /dev/cua4
    fi
    ----- snip -----

3.3 inittab
Your inittab needs to be modified to properly handle the signals
that powerd will send if there is a power failure.  The relevant
lines of my inittab look like this:

   ----- snip -----
   # What to do when power fails (shutdown to single user).
   pf::powerfail:/sbin/shutdown -f +5 "THE POWER IS FAILING"

   # If power is back before shutdown, cancel the running shutdown.
   pg:0123456:powerokwait:/sbin/shutdown -c "THE POWER IS BACK"

   # If power comes back in single user mode, return to multi user mode.
   ps:S:powerokwait:/sbin/init 2
   ----- snip -----

3.4 rc.0 (brc)
Depending on how your system is setup either rc.0 or the brc script
is executed immediately prior to shutdown.  These scripts take care
of things like unmounting disks and any other last minute clean-up.

The inverter shutdown circuit, is designed to signal an inverter
shutdown when data is sent out over the DTR line.  In my case,
I just cat a short file to the serial port (/etc/passwd - since I
know it will always be there).  My rc.0 is as follows, please note
that it is overly conservative, the sync can be removed and the
```

```
sleep times can probably be tightened, but it works so I haven't
messed with it.

  ----- snip -----
  #! /bin/sh
  #
  # brc       This file is executed by init(8) when the system is being
  #           shutdown (i.e. set to run at level 0).  It usually takes
  #           care of un-mounting all unneeded file systems.
  #
  # Version:  @(#)/etc/brc            2.01    02/17/93
  #
  # Authors:  Miquel van Smoorenburg, <miquels@drinkel.nl.mugnet.org>
  #           Fred N. van Kempen, <waltje@uwalt.nl.mugnet.org>
  #
  # Modified: 01/15/94 - Inverter shutdown support added.
  #           Tom Webster <webster@kaiwan.com>

  PATH=/sbin:/bin:/usr/sbin:/usr/bin
  echo Unmounting file systems.....
  sync
  umount -a
  sleep 2
  cat /etc/passwd > /dev/cua4
  sleep 5
  echo Done.
  ----- snip -----

(On my UPS the inverter is only running when the line power is off,
so there is no harm in sending the shutdown signal at every shutdown.)

4.0 Conclusion
Well that's how I hooked my TrippLite UPS up to my Linux box.
Feel free to drop me a line with the results of you attempts,
especially if you have any improvements.  :->

From: Tom Webster <webster@kaiwan.com>
To: hjstein@MATH.HUJI.AC.IL (Harvey J. Stein)
Subject: Re: Help - Powerd & UPS - Help
Date: Thu, 11 Aug 1994 12:20:50 -0700 (PDT)

Harvey,

> Like I tried to say, powerd can just run shutdown directly in the
> event of a low battery, so that init doesn't need to deal with it &
> doesn't need to be hacked - no new signal necessary.  Although this
> violates the nice separation of labor between powerd & init, it's
> easier than adding another signal.  Or do you just mean another
```

```
> command?  Something like having powerok/powerfail/powerfailnow as
> commands in inittab which execute when SIGPWR is received &
> /etc/powerfail contains OK/FAIL/LOWBATT (respectively).  This would be
> cleanest, but having powerd execute shutdown -r now is trivial to do
> now, and might as well be done - the logic will be the same regardless
> of the action that powerd takes when it senses a low battery.  For now
> it can just run shutdown, & when init gets hacked it can write LOWBATT
> to /etc/powerfail (or whatever the hell the file is called) & give
> init a SIGPWR.

I'd like to add the LOWBATT command, it would be the cleaner way to do
it.  I just need to take a look at the code and see how hard it would
be to add it.  Also need to look at my wiring, guess this may mean I've
got to run the system all the way down for a final test (once I think I
have it working.

Tom



From: Tom Webster <webster@kaiwan.com>
To: hjstein@MATH.HUJI.AC.IL (Harvey J. Stein)
Subject: Re: Help - Powerd & UPS - Help
Date: Mon, 15 Aug 1994 09:46:06 -0700 (PDT)

Harvey,

Well, I messed about all weekend taking readings with my multi-
tester and comparing it to the scant documentation that I have for
the UPS.  The only conclusions I came to were:

1.  My system has been working for about 8-9 months now.  If I was
going to fry anything, it should have happened by now.

2.  If my success is based on my serial hardware doing odd things
(I'm using an STB 4COM board), there is no way I'm going to be
able to find out on my system.  I'll have to leave that to other
poor souls on the net.

3.  As far as I can tell the only reason the 10kohm resistor was
in Miquel's diagram, was to keep the line higher than DCD, even
after the circuit had been shunted to ground.

4.  The only circuit that is expected to do anything other that be
switched by the UPS (the inverter shutdown circuit) has a 40kohm
resistor built into it (inside the UPS).  That should take care of
any worries there.
```

```
5.  Miquel's circuit will work for power fail sensing, and might
be extrapolated to include the low battery circuit.  If my circuit
proves unviable for others, it would just require more soldering
than I'd care to deal with in a cable.

In other related news, I broke down this morning and ordered a
cable from Tripplite.  I ordered the LanTastic/LAN Manager/Win NT
cable, it's just a cable (9M-9F, no software).  This will set me
back about $40 (the PC UNIX cable (w/ software) is about $140).
The motivation for doing this was three part.

(1)  Within a year I'll probably be running Win 4.0, or WinNT 3.5
(I beta'ed 3.1) and both should support UPS monitoring (I know NT
does and saw the power management icon on a Win4.0 desktop in one
of the computer mags).

(2)  My current cable isn't going to be compatible with any other
monitoring software for other OS's (except by random chance),
working to a known (and presumably soon to be common?) cable via a
hacked powerd, should widen the audience.

(3)  I can always reverse engineer the cable to see if Tripplite
is indeed building any safety into their cables.

It should be here in a week or so.  In the mean time, I'll start
looking into hacking powerd for LOWBATT.


Tom
```

11.5 APC Backup-UPS

There seems to be some controversy as to the accuracy of the information here on APC Back-UPSes. So, please be careful. I'm prefacing this section with one message of caution I received. It might not make alot 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 accruacy of either of these messages.

11.5.1 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
```

11.5.2 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>

This document, under one condition, is placed in Public Domain. The one condition is that credit is given where credit is due. Modify this as much as you want, just give some credit to us who worked.

** 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 */
```

11.6 APC Smart-UPS, Model 600

Many people have APC Smart UPSes. To the best of my knowledge, no one can run them in "smart" mode under Linux. This is because APC refuses to release the protocol for the "smart" mode without a non-disclosure agreement. Not very smart of them, I'd say :).

The general consensus is to buy from a brand which does release the information. I hear that Best is one such brand.

If you are stuck with an APC Smart-UPS, you can still use it, but only in a dumb mode like all the other UPSes and as outlined above.

Here's some info on how to make a cable for doing such. You'll probably have to hack powerd.c as outlined in section 8

```
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
```

12 How to shutdown other machines on the same UPS

Some people (myself included) have several Linux PCs connected to one UPS. One PC monitors the UPS and needs to get the other PCs to shut down when the power goes out. There are a number of ways to do this, all are do-it-yourself currently, and most are just hypothetical.

We assume the PCs can communicate over a network. Call the PC that monitors the UPS the master & the other PCs the slaves.

12.1 UPS status port method

Set up a port on the master which, when connected to, either sends "OK", "FAIL", or "BATLOW", the first when the power is ok, the second when the power has failed, and the third when the battery is low. Model this on port 13 (the time port) which one can telnet to & receive the local time.

Have the slaves run versions of powerd that look at this port instead of checking a serial line.

I think this is probably the best method, and intend to eventually upgrade my systems to use it.

12.2 Broadcast method

Same as section 12.1 except send an ethernet broadcast message that the power has just gone down.

This might have security implications, since it could be spoofed.

12.3 Dummy login method

Set up dummy logins on the slaves with login names `powerok` and `powerfail`, both with the same UID. Make `/etc/powerokscript` the shell of the powerok user, and make `/etc/powerfailscript` the shell of the powerfail user. On the master, have the `/etc/powerokscript` rlogin to each slave as user powerok and have the `/etc/powerfailscript` rlogin to each slave as user powerfail. Put a `.rhosts` file on each slave in the home directory of powerok and powerfail to allow root from the master to login as user powerok and powerfail to each slave.

This is the system I'm currently using. Unfortunately, there are some difficulties in getting the remote logins to execute and return without hanging. One probably wants the `/etc/powerfailscript` on the master to rsh to the slaves in background so that the `/etc/powerfailscript` doesn't hang. However, I never managed to get logging in in background to work right. I even tried arcane combinations such as having

`/etc/powerfailscript` run `foo` in background and having `foo` log into the slave. Whatever I did gave problems with the thing being stopped for tty input (or output, I can't remember).

Also, this might create security holes.