

# **Synchronous PPP and Cisco HDLC Programming Guide**

**Alan Cox**

**[alan@redhat.com](mailto:alan@redhat.com)**

# **Synchronous PPP and Cisco HDLC Programming Guide**

by Alan Cox

Copyright © 2000 Alan Cox

This documentation 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.

This program is distributed in the hope that it will be useful, but 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.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

For more details see the file COPYING in the source distribution of Linux.

# Table of Contents

|  |          |
|--|----------|
| <b>1. Introduction.....</b>                | <b>1</b> |
| <b>2. Known Bugs And Assumptions .....</b> | <b>3</b> |
| <b>3. Public Functions Provided .....</b>  | <b>5</b> |
| sppp_close.....                            | 5        |
| sppp_open .....                            | 5        |
| sppp_reopen .....                          | 6        |
| sppp_do_ioctl .....                        | 7        |
| sppp_attach .....                          | 8        |
| sppp_detach.....                           | 9        |



# Chapter 1. Introduction

The syncppp drivers in Linux provide a fairly complete implementation of Cisco HDLC and a minimal implementation of PPP. The longer term goal is to switch the PPP layer to the generic PPP interface that is new in Linux 2.3.x. The API should remain unchanged when this is done, but support will then be available for IPX, compression and other PPP features



# Chapter 2. Known Bugs And Assumptions

PPP is minimal

The current PPP implementation is very basic, although sufficient for most wan usages.

Cisco HDLC Quirks

Currently we do not end all packets with the correct Cisco multicast or unicast flags. Nothing appears to mind too much but this should be corrected.





# Chapter 3. Public Functions Provided

## sppp\_close

**LINUX**

Kernel Hackers Manual July 2008

### Name

`sppp_close` — close down a synchronous PPP or Cisco HDLC link

### Synopsis

```
int sppp_close (struct net_device * dev);
```

### Arguments

*dev*

The network device to drop the link of

### Description

This drops the logical interface to the channel. It is not done politely as we assume we will also be dropping DTR. Any timeouts are killed.

## sppp\_open

**LINUX**

## Name

sppp\_open — open a synchronous PPP or Cisco HDLC link

## Synopsis

```
int sppp_open (struct net_device * dev);
```

## Arguments

*dev*

Network device to activate

## Description

Close down any existing synchronous session and commence from scratch. In the PPP case this means negotiating LCP/IPCPC and friends, while for Cisco HDLC we simply need to start sending keepalives

# sppp\_reopen

## LINUX

## Name

sppp\_reopen — notify of physical link loss

## Synopsis

```
int sppp_reopen (struct net_device * dev);
```

## Arguments

*dev*

Device that lost the link

## Description

This function informs the synchronous protocol code that the underlying link died (for example a carrier drop on X.21)

We increment the magic numbers to ensure that if the other end failed to notice we will correctly start a new session. It happens do to the nature of telco circuits is that you can lose carrier on one end only.

Having done this we go back to negotiating. This function may be called from an interrupt context.

## sppp\_do\_ioctl

### LINUX

Kernel Hackers Manual July 2008

## Name

sppp\_do\_ioctl — Ioctl handler for ppp/hdlc

## Synopsis

```
int sppp_do_ioctl (struct net_device * dev, struct ifreq *  
ifr, int cmd);
```

## Arguments

*dev*

Device subject to ioctl

*ifr*

Interface request block from the user

*cmd*

Command that is being issued

## Description

This function handles the ioctls that may be issued by the user to control the settings of a PPP/HDLC link. It does both busy and security checks. This function is intended to be wrapped by callers who wish to add additional ioctl calls of their own.

## sppp\_attach

### LINUX

Kernel Hackers Manual July 2008

## Name

`sppp_attach` — attach synchronous PPP/HDLC to a device

## Synopsis

```
void sppp_attach (struct ppp_device * pd);
```

## Arguments

*pd*

PPP device to initialise

## Description

This initialises the PPP/HDLC support on an interface. At the time of calling the `dev` element must point to the network device that this interface is attached to. The interface should not yet be registered.

## sppp\_detach

### LINUX

Kernel Hackers Manual July 2008

## Name

`sppp_detach` — release PPP resources from a device

## Synopsis

```
void sppp_detach (struct net_device * dev);
```

## **Arguments**

*dev*

Network device to release

## **Description**

Stop and free up any PPP/HDLC resources used by this interface. This must be called before the device is freed.