Generic HDLC layer

Krzysztof Halasa <khc@pm.waw.pl>

Generic HDLC layer currently supports:

1. Frame Relay (ANSI, CCITT, Cisco and no LMI)
   - Normal (routed) and Ethernet-bridged (Ethernet device emulation) interfaces can share a single PVC.
   - ARP support (no InARP support in the kernel - there is an experimental InARP user-space daemon available on: http://www.kernel.org/pub/linux/utils/net/hdlc/).
2. raw HDLC - either IP (IPv4) interface or Ethernet device emulation
3. Cisco HDLC
4. PPP
5. X.25 (uses X.25 routines).

Generic HDLC is a protocol driver only - it needs a low-level driver for your particular hardware.

Ethernet device emulation (using HDLC or Frame-Relay PVC) is compatible with IEEE 802.1Q (VLANs) and 802.1D (Ethernet bridging).

Make sure the hdlc.o and the hardware driver are loaded. It should create a number of "hdlc" (hdlc0 etc) network devices, one for each WAN port. You'll need the "sethdlc" utility, get it from:

> http://www.kernel.org/pub/linux/utils/net/hdlc/

Compile sethdlc.c utility:

```
gcc -O2 -Wall -o sethdlc sethdlc.c
```

Make sure you're using a correct version of sethdlc for your kernel.

Use sethdlc to set physical interface, clock rate, HDLC mode used, and add any required PVCs if using Frame Relay. Usually you want something like:

```
sethdlc hdlc0 clock int rate 128000
sethdlc hdlc0 cisco interval 10 timeout 25
```

or:

```
sethdlc hdlc0 rs232 clock ext
sethdlc hdlc0 fr lmi ansi
sethdlc hdlc0 create 99
ifconfig hdlc0 up
ifconfig pvc0 localIP pointopoint remoteIP
```

In Frame Relay mode, ifconfig master hdlc device up (without assigning any IP address to it) before using pvc devices.

Setting interface:

- v35 | rs232 | x21 | t1 | e1
  - sets physical interface for a given port if the card has software-selectable interfaces

  loopback
  - activate hardware loopback (for testing only)
- clock ext
  - both RX clock and TX clock external
- clock int
  - both RX clock and TX clock internal
- clock txint
  - RX clock external, TX clock internal
- clock txfromrx
  - RX clock external, TX clock derived from RX clock
- rate
  - sets clock rate in bps (for "int" or "txint" clock only)

Setting protocol:

- hdlc - sets raw HDLC (IP-only) mode

  nrz / nrzi / fm-mark / fm-space / manchester - sets transmission code

  no-parity / crc16 / crc16-pr0 (CRC16 with preset zeros) / crc32-itu

  crc16-itu (CRC16 with ITU-T polynomial) / crc16-itu-pr0 - sets parity

- hdlc-eth - Ethernet device emulation using HDLC. Parity and encoding as above.
- cisco - sets Cisco HDLC mode (IP, IPv6 and IPX supported)

  interval - time in seconds between keepalive packets

  timeout - time in seconds after last received keepalive packet before
  we assume the link is down
- ppp - sets synchronous PPP mode
- x25 - sets X.25 mode
- fr - Frame Relay mode

  lmi ansi / ccitt / cisco / none - LMI (link management) type

  dce - Frame Relay DCE (network) side LMI instead of default DTE (user).

  It has nothing to do with clocks!

  - t391 - link integrity verification polling timer (in seconds) - user
  - t392 - polling verification timer (in seconds) - network
  - n391 - full status polling counter - user
  - n392 - error threshold - both user and network
  - n393 - monitored events count - both user and network

Frame-Relay only:

- create n | delete n - adds / deletes PVC interface with DLCI #n. Newly created interface will be named pvc0, pvc1 etc.
- create ether n | delete ether n - adds a device for Ethernet-bridged frames. The device will be named pvceth0, pvceth1 etc.

Board-specific issues

n2.o and c101.o need parameters to work:

```
insmod n2 hw=io,irq,ram,ports[:io,irq,...]
```

example:

```
insmod n2 hw=0x300,10,0xD0000,01
```

or:

```
insmod c101 hw=irq,ram[:irq,...]
```

example:

```
insmod c101 hw=9,0xdc000
```

If built into the kernel, these drivers need kernel (command line) parameters:

```
n2.hw=io,irq,ram,ports:...
```

or:

```
c101.hw=irq,ram:...
```

If you have a problem with N2, C101 or PLX200SYN card, you can issue the "private" command to see port's packet descriptor rings (in kernel logs):

```
sethdlc hdlc0 private
```

The hardware driver has to be build with #define DEBUG_RINGS. Attaching this info to bug reports would be helpful. Anyway, let me know if you have problems using this.

For patches and other info look at: <http://www.kernel.org/pub/linux/utils/net/hdlc/>.