Linux Drivers for Baycom Modems

Thomas M. Sailer, HB9JNX/AE4WA, <sailer@ife.ee.ethz.ch>

The drivers for the baycom modems have been split into separate drivers as they did not share any code, and the driver and device names have changed.

This document describes the Linux Kernel Drivers for simple Baycom style amateur radio modems.

The following drivers are available:

baycom_ser_fdx:
: This driver supports the SER12 modems either full or half duplex. Its baud rate may be changed via the `baud` module parameter, therefore it supports just about every bit bang modem on a serial port. Its devices are called bcsf0 through bcsf3. This is the recommended driver for SER12 type modems, however if you have a broken UART clone that does not have working delta status bits, you may try baycom_ser_hdx.

baycom_ser_hdx:
: This is an alternative driver for SER12 type modems. It only supports half duplex, and only 1200 baud. Its devices are called bcsh0 through bcsh3. Use this driver only if baycom_ser_fdx does not work with your UART.

baycom_par:
: This driver supports the par96 and picpar modems. Its devices are called bcp0 through bcp3.

baycom_epp:
: This driver supports the EPP modem. Its devices are called bce0 through bce3. This driver is work-in-progress.

The following modems are supported:

ser12	This is a very simple 1200 baud AFSK modem. The modem consists only of a modulator/demodulator chip, usually a TI TCM3105. The computer is responsible for regenerating the receiver bit clock, as well as for handling the HDLC protocol. The modem connects to a serial port, hence the name. Since the serial port is not used as an async serial port, the kernel driver for serial ports cannot be used, and this driver only supports standard serial hardware (8250, 16450, 16550)
par96	This is a modem for 9600 baud FSK compatible to the G3RUH standard. The modem does all the filtering and regenerates the receiver clock. Data is transferred from and to the PC via a shift register. The shift register is filled with 16 bits and an interrupt is signalled. The PC then empties the shift register in a burst. This modem connects to the parallel port, hence the name. The modem leaves the implementation of the HDLC protocol and the scrambler polynomial to the PC.
picpar	This is a redesign of the par96 modem by Henning Rech, DF9IC. The modem is protocol compatible to par96, but uses only three low power ICs and can therefore be fed from the parallel port and does not require an additional power supply. Furthermore, it incorporates a carrier detect circuitry.
EPP	This is a high-speed modem adaptor that connects to an enhanced parallel port. Its target audience is users working over a high speed hub (76.8kbit/s).
eppfpga	This is a redesign of the EPP adaptor.

All of the above modems only support half duplex communications. However, the driver supports the KISS (see below) fullduplex command. It then simply starts to send as soon as there's a packet to transmit and does not care about DCD, i.e. it starts to send even if there's someone else on the channel. This command is required by some implementations of the DAMA channel access protocol.

The Interface of the drivers

Unlike previous drivers, these drivers are no longer character devices, but they are now true kernel network interfaces. Installation is therefore simple. Once installed, four interfaces named bc{sf,sh,p,e}[0-3] are available. sethdlc from the ax25 utilities may be used to set driver states etc. Users of userland AX.25 stacks may use the net2kiss utility (also available in the ax25 utilities package) to convert packets of a network interface to a KISS stream on a pseudo tty. There's also a patch available from me for WAMPES which allows attaching a kernel network interface directly.

Configuring the driver

Every time a driver is inserted into the kernel, it has to know which modems it should access at which ports. This can be done with the setbaycom utility. If you are only using one modem, you can also configure the driver from the insmod command line (or by means of an option line in `/etc/modprobe.d/*.conf`).

Examples:

```
modprobe baycom_ser_fdx mode="ser12*" iobase=0x3f8 irq=4
sethdlc -i bcsf0 -p mode "ser12*" io 0x3f8 irq 4
```

Both lines configure the first port to drive a ser12 modem at the first serial port (COM1 under DOS). The * in the mode parameter instructs the driver to use the software DCD algorithm (see below):

```
insmod baycom_par mode="picpar" iobase=0x378
sethdlc -i bcp0 -p mode "picpar" io 0x378
```

Both lines configure the first port to drive a picpar modem at the first parallel port (LPT1 under DOS). (Note: picpar implies hardware DCD, par96 implies software DCD).

The channel access parameters can be set with sethdlc -a or kissparms. Note that both utilities interpret the values slightly differently.

Hardware DCD versus Software DCD

To avoid collisions on the air, the driver must know when the channel is busy. This is the task of the DCD circuitry/software. The driver may either utilise a software DCD algorithm (options=1) or use a DCD signal from the hardware (options=0).

ser12	if software DCD is utilised, the radio's squelch should always be open. It is highly recommended to use the software DCD algorithm, as it is much faster than most hardware squelch circuitry. The disadvantage is a slightly higher load on the system.
par96	the software DCD algorithm for this type of modem is rather poor. The modem simply does not provide enough information to implement a reasonable DCD algorithm in software. Therefore, if your radio feeds the DCD input of the PAR96 modem, the use of the hardware DCD circuitry is recommended.
picpar	the picpar modem features a builtin DCD hardware, which is highly recommended.

Compatibility with the rest of the Linux kernel

The serial driver and the baycom serial drivers compete for the same hardware resources. Of course only one driver can access a given interface at a time. The serial driver grabs all interfaces it can find at startup time. Therefore the baycom drivers subsequently won't be able to access a serial port. You might therefore find it necessary to release a port owned by the serial driver with 'setserial /dev/ttyS# uart none', where # is the number of the interface. The baycom drivers do not reserve any ports at startup, unless one is specified on the 'insmod' command line. Another method to solve the problem is to compile all drivers as modules and leave it to kmod to load the correct driver depending on the application.

The parallel port drivers (baycom_par, baycom_epp) now use the parport subsystem to arbitrate the ports between different client drivers.

vy 73s de

Tom Sailer, sailer@ife.ee.ethz.ch

hb9jnx @ hb9w.ampr.org